PyAlink 产品版
组件文档

版本：v1.5.5-20220711

目录

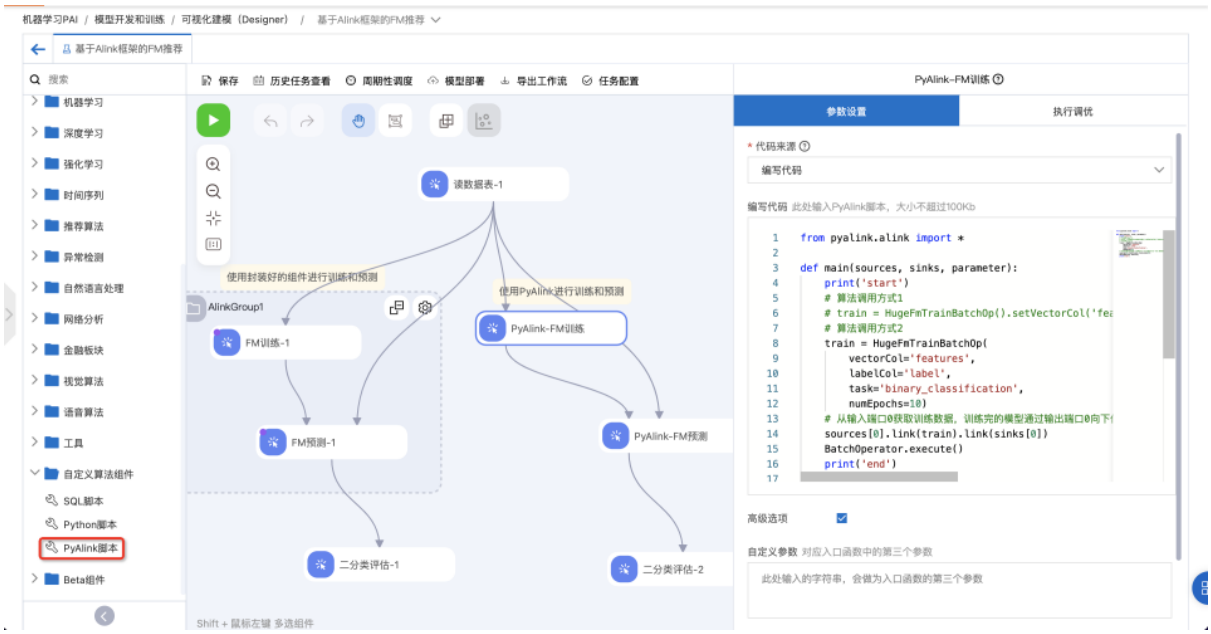
目录

简介

Alink 是阿里巴巴计算平台事业部 PAI (Platform of Artificial Intelligence) 团队基于 Flink 计算引擎研发的批流一体的机器学习算法平台，该平台提供了丰富的算法组件库和便捷的操作框架。开发者可以一键搭建覆盖数据处理、特征工程、模型训练、模型预测的算法模型开发全流程。

PyAlink 是 Alink 提供的简单易用的 Python 接口，在阿里云 PAI 产品的可视化建模 Designer 中提供了 PyAlink 组件，方便开发者轻松做代码编写来构建算法模型，并一键提交至 DLC、MaxCompute 或全托管 Flink 集群中进行分布式计算。

使用入口：



PAI-Designer快速入门：https://help.aliyun.com/document_detail/69213.html

PyAlink组件使用文档：https://help.aliyun.com/document_detail/424831.html

AK文件读入 (AkSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.AkSourceBatchOp

Python 类名：AkSourceBatchOp

功能介绍

从文件系统读Ak文件。Ak文件格式是Alink 自定义的一种文件格式，能够将数据的Schema保留输出的文件格式。

分区选择

Export2FileSinkStreamOp组件能将数据分区保存，AkSourceBatchOp可以选择分区读取。 分区目录名格式为"分区名=值"，例如： month=06/day=17;month=06/day=18。 Alink将遍历目录下的分区名和分区值，构造分区表：

month	day
06	17
06	18

使用SQL语句查找分区，例如：AkSourceBatchOp.setPartitions("day = '17'")，分区选择语法参考《Flink SQL 内置函数》，分区值为String类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
partitions	分区名	1)单级、单个分区示例：ds=20190729；2)多级分区之间用" / "分隔，例如：ds=20190729/dt=12；3)多个分区之间用","分隔，例如：ds=20190729,ds=20190730	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]])


batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')

filePath = "/tmp/test_alink_file_sink";

# write file to local disk
batchData.link(AkSinkBatchOp()\
                .setFilePath(FilePath(filePath))\
                .setOverwriteSink(True)\
                .setNumFiles(1))
BatchOperator.execute()

# read ak file and print
AkSourceBatchOp().setFilePath(FilePath(filePath)).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.common.io.filesystem.FilePath;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.AkSinkBatchOp;
import com.alibaba.alink.operator.batch.source.AkSourceBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AkSourceBatchOpTest {
```

```
    @Test
    public void testAkSourceBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        String filePath = "/tmp/test_alink_file_sink";
        batchData.link(new AkSinkBatchOp()
            .setFilePath(new FilePath(filePath))
            .setOverwriteSink(true)
            .setNumFiles(1));
        BatchOperator.execute();
        new AkSourceBatchOp().setFilePath(new FilePath(filePath)).print();
    }
}
```

运行结果

f0	f1	label
4	3	2
1	2	1
2	1	1
3	2	1
4	3	2
2	4	1
2	2	1

Catalog读入 (CatalogSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.CatalogSourceBatchOp

Python 类名：CatalogSourceBatchOp

功能介绍

Catalog描述了数据库的属性和数据库的位置, 支持Mysql, Derby, Sqlite, Hive.

在使用时，需要先下载插件，详情请看https://www.yuque.com/pinshu/alink_guide/czg4cx

定义分成三步：

第一步，定义Catalog

数据库	Java 接口
Derby	DerbyCatalog(String catalogName, String defaultDatabase, String derbyVersion, String derbyPath)
MySql	MySqlCatalog(String catalogName, String defaultDatabase, String mysqlVersion,String mysqlUrl, String port, String userName, String password)
Sqlite	SqliteCatalog(String catalogName, String defaultDatabase, String sqliteVersion, String dbUrl)
Hive	HiveCatalog(String catalogName, String defaultDatabase, String hiveVersion, String hiveConfDir) HiveCatalog(String catalogName, String defaultDatabase, String hiveVersion, FilePath hiveConfDir) HiveCatalog(String catalogName, String defaultDatabase, String hiveVersion, String hiveConfDir,String kerberosPrincipal, String kerberosKeytab)

示例：

```
    derby = DerbyCatalog("derby_test_catalog", DERBY_SCHEMA, "10.6.1.0",
derbyFolder+'/'+DERBY_DB)
```

各插件提供的版本：

```
    Hive: 2.3.4
    MySQL: 5.1.27
    Derby: 10.6.1.0
    SQLite: 3.19.3
    odps: 0.36.4-public
```

第二步， 定义CatalogObject

```
dbName = "sqlite_db"
tableName = "table"

# 第一个参数是Catalog, 第二个参数是DB/Project
catalogObject = CatalogObject(derby, ObjectPath(dbName, tableName))
```

第三步，定义Source和Sink

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
catalogObject	catalog object	catalog object	String	✓		

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

Derby

```
derbyFolder = "*"
DERBY_SCHEMA = "derby_schema"
DERBY_DB = "derby_db"
derby = DerbyCatalog("derby_test_catalog", DERBY_SCHEMA, "10.6.1.0",
derbyFolder+'/'+DERBY_DB)

catalogObject = CatalogObject(derby, ObjectPath("test_catalog_source_sink",
"test_catalog_source_sink3"))

catalogSinkBatchOp = CatalogSinkBatchOp()\
    .setCatalogObject(catalogObject)


source.link(catalogSinkBatchOp)

BatchOperator.execute()

catalogSourceBatchOp = CatalogSourceBatchOp()\
    .setCatalogObject(catalogObject)

catalogSourceBatchOp.print()
```

Java 代码

```
String derbyFolder = "*";
String DERBY_SCHEMA = "derby_schema";
String DERBY_DB = "derby_db";
DerbyCatalog derby = new DerbyCatalog("derby_test_catalog", DERBY_SCHEMA,
"10.6.1.0",
    derbyFolder + '/' + DERBY_DB);

CatalogObject catalogObject = new CatalogObject(derby,
    new ObjectPath("test_catalog_source_sink", "test_catalog_source_sink3"));

catalogSinkBatchOp catalogSinkStreamOp = new catalogSinkBatchOp()
    .setCatalogObject(catalogObject);

source.link(catalogSinkStreamOp);

StreamOperator.execute();

CatalogSourceBatchOp catalogSourceStreamOp = new CatalogSourceBatchOp()
    .setCatalogObject(catalogObject);

catalogSourceStreamOp.print();

StreamOperator.execute();
```

Sqlite

Python 代码

```
sqliteFolder = "*"
SQLITE_SCHEMA = "sqlite_schema"
SQLITE_DB = "sqlite_db"
sqlite = SqliteCatalog("sqlite_test_catalog", None, "3.19.3",
[sqliteFolder+'/'+SQLITE_DB])

catalogObject = CatalogObject(sqlite, ObjectPath(SQLITE_DB,
"test_catalog_source_sink3"))

catalogSinkBatchOp = CatalogSinkBatchOp()\
    .setCatalogObject(catalogObject)


source.link(catalogSinkBatchOp)

BatchOperator.execute()

catalogSourceBatchOp = CatalogSourceBatchOp()\
    .setCatalogObject(catalogObject)
```

```
catalogSourceBatchOp.print()
```

Java代码

```
String sqliteFolder = "*";
String SQLITE_SCHEMA = "sqlite_schema";
String SQLITE_DB = "sqlite_db";
SqliteCatalog sqlite = new SqliteCatalog("sqlite_test_catalog", null, "3.19.3",
sqliteFolder + '/' +
    SQLITE_DB);

CatalogObject catalogObject = new CatalogObject(sqlite, new
ObjectPath(SQLITE_DB,
    "test_catalog_source_sink3"));

CatalogSinkBatchOp catalogSinkStreamOp = CatalogSinkBatchOp()
    .setCatalogObject(catalogObject);

source.link(catalogSinkBatchOp);

StreamOperator.execute();

CatalogSourceBatchOp catalogSourceStreamOp = new CatalogSourceBatchOp()
    .setCatalogObject(catalogObject);

catalogSourceStreamOp.print();

StreamOperator.execute();
```

CSV文件读入 (CsvSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.CsvSourceBatchOp

Python 类名：CsvSourceBatchOp

功能介绍

读CSV文件。支持从本地、hdfs、http读取

分区选择

分区目录名格式为"分区名=值"，例如： city=beijing/month=06/day=17;city=hangzhou/month=06/day=18。 Alink将遍历目录下的分区名和分区值，构造分区表：

city	month	day
beijing	06	17
hangzhou	06	18

使用SQL语句查找分区，例如：CsvSourceBatchOp.setPartitions("city = 'beijing'")，分区选择语法参考《Flink SQL 内置函数》，分区值为String类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
filePath	文件路径	文件路径	String	✓	
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓	
fieldDelimiter	字段分隔符	字段分隔符	String		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"

ignoreFirstLine	是否忽略第一行数据	是否忽略第一行数据	Boolean		
lenient	是否容错	若为true，当解析失败时丢弃该数据；若为false，解析失败是抛异常	Boolean		
partitions	分区名	1)单级、单个分区示例：ds=20190729；2)多级分区之间用" / "分隔，例如：ds=20190729/dt=12； 3)多个分区之间用","分隔，例如：ds=20190729,ds=20190730	String		
quoteChar	引号字符	引号字符	Character		
rowDelimiter	行分隔符	行分隔符	String		
skipBlankLine	是否忽略空行	是否忽略空行	Boolean		

支持的字段类型包括：

字段类型	描述	值域	Flink类型	Java类
VARCHAR/STRING	可变长度字符串	最大容量为4mb	Types.STRING	java.lang.Stri
BOOLEAN	逻辑值	值：TRUE，FALSE，UNKNOWN	Types.BOOLEAN	java.lang.Boo
TINYINT	微整型，1字节整数	范围是-128到127	Types.BYTE	java.lang.Byt
SMALLINT	短整型，2字节整数	范围为-32768至32767	Types.SHORT	java.lang.Sho

INT	整型，4字节整数	范围是-2147483648到2147483647	Types.INT	java.lang.Inte
BIGINT/LONG	长整型，8字节整数	范围是-9223372036854775808至9223372036854775807	Types.LONG	java.lang.Lon
FLOAT	4字节浮点数	6位数字精度	Types.FLOAT	java.lang.Flo
DOUBLE	8字节浮点数	15位十进制精度	Types.DOUBLE	java.lang.Dou
DECIMAL	小数类型	示例：123.45是DECIMAL(5,2)值	Types.DECIMAL	java.math.Big
DATE	日期	示例：'1969-07-20'	Types.SQL_DATE	java.sql.Date
TIME	时间	示例：'20:17:40'	Types.SQL_TIME	java.sql.Time
TIMESTAMP	时间戳，日期和时间	示例：'1969-07-20 20:17:40'	Types.SQL_TIMESTAMP	java.sql.Time

关于分隔符的说明：

Web前端支持用户输入如下转义字符和unicode字符作为分隔符：

输入分隔符	含义
\t	制表符(tab键)
\n	换行符
\b	退格符
\r	回车
\f	换页
\\	反斜线字符
'	单引号

"	双引号
\ddd	1到3位八进制数所代表的任意字符，例如\001表示'ctrl + A', \40表示空格符
\udddd	1到4位十六进制数所代表unicode字符，例如\u0001表示'ctrl + A', \u0020表示空格符

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

filePath = 'https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/iris.csv'
schema = 'sepal_length double, sepal_width double, petal_length double,
petal_width double, category string'
csvSource = CsvSourceBatchOp()\
    .setFilePath(filePath)\
    .setSchemaStr(schema)\
    .setFieldDelimiter(",")
BatchOperator.collectToDataframe(csvSource)
```

Java 代码

```
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class CsvSourceBatchOpTest {

    @Test
    public void testCsvSourceBatchOp() throws Exception {
        String filePath = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv";
        String schema
            = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string";
        CsvSourceBatchOp csvSource = new CsvSourceBatchOp()
            .setFilePath(filePath)
            .setSchemaStr(schema)
            .setFieldDelimiter(",");
        csvSource.print();
    }
}
```

运行结果

sepal_length	sepal_width	petal_length	petal_width	category
5.0000	3.2000	1.2000	0.2000	Iris-setosa
6.6000	3.0000	4.4000	1.4000	Iris-versicolor
5.4000	3.9000	1.3000	0.4000	Iris-setosa
5.0000	2.3000	3.3000	1.0000	Iris-versicolor
5.5000	3.5000	1.3000	0.2000	Iris-setosa
6.2000	3.4000	5.4000	2.3000	Iris-virginica

CSV读入 (CsvSourceForDesignerBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.CsvSourceForDesignerBatchOp

Python 类名：CsvSourceForDesignerBatchOp

功能介绍

读CSV文件。支持从本地、hdfs、http读取

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
fileSource	文件来源	文件来源	String		"OSS", "OTHERS"	"OSS"
filePath	文件路径	文件路径	String			
ossFilePath	文件路径	文件路径	String			
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
fieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
ignoreFirstLine	是否忽略第一行数据	是否忽略第一行数据	Boolean			false

lenient	是否容错	若为true，当解析失败时丢弃该数据；若为false，解析失败是抛异常	Boolean			false
quoteString	引号字符	引号字符	String			"\""
rowDelimiter	行分隔符	行分隔符	String			"\n"
skipBlankLine	是否忽略空行	是否忽略空行	Boolean			true

支持的字段类型包括：

字段类型	描述	值域	Flink类型	Java类
VARCHAR/STRING	可变长度字符串	最大容量为4mb	Types.STRING	java.lang.Stri
BOOLEAN	逻辑值	值：TRUE，FALSE，UNKNOWN	Types.BOOLEAN	java.lang.Boo
TINYINT	微整型，1字节整数	范围是-128到127	Types.BYTE	java.lang.Byt
SMALLINT	短整型，2字节整数	范围为-32768至32767	Types.SHORT	java.lang.Sho
INT	整型，4字节整数	范围是-2147483648到2147483647	Types.INT	java.lang.Inte
BIGINT/LONG	长整型，8字节整数	范围是-9223372036854775808至9223372036854775807	Types.LONG	java.lang.Lon

FLOAT	4字节浮点数	6位数字精度	Types.FLOAT	java.lang.Flo
DOUBLE	8字节浮点数	15位十进制精度	Types.DOUBLE	java.lang.Dou
DECIMAL	小数类型	示例：123.45是DECIMAL(5,2)值	Types.DECIMAL	java.math.Bi
DATE	日期	示例：'1969-07-20'	Types.SQL_DATE	java.sql.Date
TIME	时间	示例：'20:17:40'	Types.SQL_TIME	java.sql.Time
TIMESTAMP	时间戳，日期和时间	示例：'1969-07-20 20:17:40'	Types.SQL_TIMESTAMP	java.sql.Time

关于分隔符的说明：

Web前端支持用户输入如下转义字符和unicode字符作为分隔符：

输入分隔符	含义
\t	制表符(tab键)
\n	换行符
\b	退格符
\r	回车
\f	换页
\\	反斜线字符
'	单引号
"	双引号
\ddd	1到3位八进制数所代表的任意字符，例如\001表示'ctrl + A', \40表示空格符
\udddd	1到4位十六进制数所代表unicode字符，例如\u0001表示'ctrl + A', \u0020表示空格符

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

filePath = 'https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/iris.csv'
schema = 'sepal_length double, sepal_width double, petal_length double,
petal_width double, category string'
csvSource = CsvSourceBatchOp()\
    .setFilePath(filePath)\
    .setSchemaStr(schema)\
    .setFieldDelimiter(",")
BatchOperator.collectToDataframe(csvSource)
```

Java 代码

```
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class CsvSourceBatchOpTest {

    @Test
    public void testCsvSourceBatchOp() throws Exception {
        String filePath = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv";
        String schema
            = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string";
        CsvSourceBatchOp csvSource = new CsvSourceBatchOp()
            .setFilePath(filePath)
            .setSchemaStr(schema)
            .setFieldDelimiter(",");
        csvSource.print();
    }
}
```

运行结果

sepal_length	sepal_width	petal_length	petal_width	category
5.0000	3.2000	1.2000	0.2000	Iris-setosa
6.6000	3.0000	4.4000	1.4000	Iris-versicolor
5.4000	3.9000	1.3000	0.4000	Iris-setosa

5.0000	2.3000	3.3000	1.0000	Iris-versicolor
5.5000	3.5000	1.3000	0.2000	Iris-setosa
6.2000	3.4000	5.4000	2.3000	Iris-virginica

CSV读入 (CsvSourceForWebBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.CsvSourceForWebBatchOp

Python 类名：CsvSourceForWebBatchOp

功能介绍

读CSV文件。支持从本地、hdfs、http读取

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
filePath	文件路径	文件路径	String	✓	
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓	
fieldDelimiter	字段分隔符	字段分隔符	String		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"
ignoreFirstLine	是否忽略第一行数据	是否忽略第一行数据	Boolean		
lenient	是否容错	若为true，当解析失败时丢弃该数据；若为false，解析失败是抛异常	Boolean		
partitions	分区名	1)单级、单个分区示例：ds=20190729；2)多级分区之间用" / "分隔，例如：ds=20190729/dt=12； 3)多个分区之间用","分隔，例如：ds=20190729,ds=20190730	String		

quoteChar	引号字符	引号字符	Character		
quoteString	引号字符	引号字符	String		
rowDelimiter	行分隔符	行分隔符	String		
skipBlankLine	是否忽略空行	是否忽略空行	Boolean		

支持的字段类型包括：

字段类型	描述	值域	Flink类型	Java类
VARCHAR/STRING	可变长度字符串	最大容量为4mb	Types.STRING	java.lang.Stri
BOOLEAN	逻辑值	值：TRUE，FALSE，UNKNOWN	Types.BOOLEAN	java.lang.Boo
TINYINT	微整型，1字节整数	范围是-128到127	Types.BYTE	java.lang.Byt
SMALLINT	短整型，2字节整数	范围为-32768至32767	Types.SHORT	java.lang.Sho
INT	整型，4字节整数	范围是-2147483648到2147483647	Types.INT	java.lang.Inte
BIGINT/LONG	长整型，8字节整数	范围是-9223372036854775808至9223372036854775807	Types.LONG	java.lang.Lon

FLOAT	4字节浮点数	6位数字精度	Types.FLOAT	java.lang.Flo
DOUBLE	8字节浮点数	15位十进制精度	Types.DOUBLE	java.lang.Dou
DECIMAL	小数类型	示例：123.45是DECIMAL(5,2)值	Types.DECIMAL	java.math.Big
DATE	日期	示例：'1969-07-20'	Types.SQL_DATE	java.sql.Date
TIME	时间	示例：'20:17:40'	Types.SQL_TIME	java.sql.Time
TIMESTAMP	时间戳，日期和时间	示例：'1969-07-20 20:17:40'	Types.SQL_TIMESTAMP	java.sql.Time

关于分隔符的说明：

Web前端支持用户输入如下转义字符和unicode字符作为分隔符：

输入分隔符	含义
\t	制表符(tab键)
\n	换行符
\b	退格符
\r	回车
\f	换页
\\	反斜线字符
'	单引号
"	双引号
\ddd	1到3位八进制数所代表的任意字符，例如\001表示'ctrl + A', \40表示空格符
\udddd	1到4位十六进制数所代表unicode字符，例如\u0001表示'ctrl + A', \u0020表示空格符

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

URL = "https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/iris.csv"
SCHEMA_STR = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string";

batch_data = CsvSourceBatchOp().setFilePath(URL).setSchemaStr(SCHEMA_STR)
batch_data.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class CsvSourceBatchOpTest {

    @Test
    public void testCsvSourceBatchOp() throws Exception {
        String url = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv";
        String schemaStr = "sepal_length double, sepal_width double,
petal_length double, petal_width double, category string";

        new CsvSourceBatchOp().setFilePath(url).setSchemaStr(schemaStr)
            .print();
    }
}
```

LibSvm文件读入 (LibSvmSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.LibSvmSourceBatchOp

Python 类名：LibSvmSourceBatchOp

功能介绍

读LibSVM文件。支持从本地、hdfs、oss、http等读取。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
partitions	分区名	1)单级、单个分区示例：ds=20190729；2)多级分区之间用" / "分隔，例如：ds=20190729/dt=12； 3)多个分区之间用","分隔，例如：ds=20190729,ds=20190730	String			null
startIndex	起始索引	起始索引	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ['1:2.0 2:1.0 4:0.5', 1.5],
    ['1:2.0 2:1.0 4:0.5', 1.7],
```

```
    ['1:2.0 2:1.0 4:0.5', 3.6]
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 string, f2
double')

filepath = '/tmp/abc.svm'

sink =
LibSvmSinkBatchOp().setFilePath(filepath).setLabelCol("f2").setVectorCol("f1").
setOverwriteSink(True)
batch_data = batch_data.link(sink)

BatchOperator.execute()

batch_data = LibSvmSourceBatchOp().setFilePath(filepath)
batch_data.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.LibSvmSinkBatchOp;
import com.alibaba.alink.operator.batch.source.LibSvmSourceBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LibSvmSourceBatchOpTest {
    @Test
    public void testLibSvmSourceBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("1:2.0 2:1.0 4:0.5", 1.5),
            Row.of("1:2.0 2:1.0 4:0.5", 1.7),
            Row.of("1:2.0 2:1.0 4:0.5", 3.6)
        );
        BatchOperator <?> batch_data = new MemSourceBatchOp(df_data, "f1
string, f2  double");
        String filepath = "/tmp/abc.svm";
        BatchOperator <?> sink = new
LibSvmSinkBatchOp().setFilePath(filepath).setLabelCol("f2").setVectorCol("f1")
            .setOverwriteSink(true);
        batch_data = batch_data.link(sink);
        BatchOperator.execute();
```

```
        batch_data = new LibSvmSourceBatchOp().setFilePath(filepath);
        batch_data.print();
    }
}
```

运行结果

label	features
1.7000	1:2.0 2:1.0 4:0.5
1.5000	1:2.0 2:1.0 4:0.5
3.6000	1:2.0 2:1.0 4:0.5

内存数据读入 (MemSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.MemSourceBatchOp

Python 类名：MemSourceBatchOp

功能介绍

从内存中读取数据生成表。

MemSourceBatchOp支持多个构造函数

构造函数	参数	示例
`MemSourceBatchOp(Object[] vals, String colName)`	数据只有一列，列类型从数据判断	`MemSourceBatchOp(new Object[]{1.0, 2.0}, "f0")`
`MemSourceBatchOp(Object[][] vals, String[] colNames)`	colNames是列名列表，列类型从数据判断	`MemSourceBatchOp(new Object[][]{ {1.0, 2.0}, {3.0, 4.0} }, new String[]{"f0", "f1"})`
`MemSourceBatchOp(List<Row> rows, TableSchema schema)`	schema	`MemSourceBatchOp source = new MemSourceBatchOp(df, new TableSchema(new String[]{"f1", "f2"}, new TypeInformation[]{Types.STRING, Types.DOUBLE}))`
`MemSourceBatchOp(List<Row> rows, String schemaStr)`	schemaStr格式是col1 string, f1 int...	`MemSourceBatchOp(df, "f1 string, f2 double")`
`MemSourceBatchOp(Row[] rows, String[] colNames)`	colNames是列名列表，列类型从数据判断	`MemSourceBatchOp(rows, new String[]{"f0", "f1"})`
`MemSourceBatchOp(List<Row> rows, String[] colNames)`	colNames是列名列表，列类型从数据判断	`MemSourceBatchOp(rows, new String[]{"f0", "f1"})`

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

无，仅在Java中使用

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MemSourceBatchOpTest {

    @Test
    public void testMemSourceBatchOp() throws Exception {
        List<Row> df = Arrays.asList(
            Row.of("1:2.0 2:1.0 4:0.5", 1.5),
            Row.of("1:2.0 2:1.0 4:0.5", 1.7),
            Row.of("1:2.0 2:1.0 4:0.5", 3.6)
        );
        BatchOperator<?> batchData = new MemSourceBatchOp(df, "f1 string, f2
double");
        batchData.print();
    }
}
```

运行结果

f1	f2
1:2.0 2:1.0 4:0.5	1.5000
1:2.0 2:1.0 4:0.5	1.7000
1:2.0 2:1.0 4:0.5	3.6000

MySQL读入 (MySqlSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.MySqlSourceBatchOp

Python 类名：MySqlSourceBatchOp

功能介绍

写Mysql表

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
dbName	数据库名字	数据库名字	String	✓		
inputTableName	输入表名字	输入表名字	String	✓		
ip	IP地址	IP地址	String	✓		
password	密码	密码	String	✓		
port	端口	端口	String	✓		
username	用户名	用户名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
data = MySqlSourceBatchOp()\
     .setDbName("alink")\
     .setUsername("alink")\
     .setPassword(passwd)\
     .setIp("***")\
```

```
        .setPort("20001")\
        .setInputTableName("test")
```

数值队列数据源 (NumSeqSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.NumSeqSourceBatchOp

Python 类名：NumSeqSourceBatchOp

功能介绍

生成连续整数的表

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
from	开始	开始	Long	✓		
to	截止	截止	Long	✓		
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

无，仅在Java中使用

Java 代码

```
import com.alibaba.alink.operator.batch.source.NumSeqSourceBatchOp;
import org.junit.Test;

public class NumSeqSourceBatchOpTest {
    @Test
    public void testMemSourceBatchOp() throws Exception {
        NumSeqSourceBatchOp batchData = new NumSeqSourceBatchOp()
            .setFrom(0)
            .setTo(1);
        batchData.print();
    }
}
```

运行结果

num
0
1

Oracle读入 (OracleSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.OracleSourceBatchOp

Python 类名：OracleSourceBatchOp

功能介绍

从 Oracle 数据库中读取数据，当前支持 11g 版本

URL 说明

格式类似于如下：

"jdbc:oracle:thin:@//host:port/service"

注意事项

1. 报错 `ORA-01882: timezone region not found`

   原因是因为 Oracle 服务的时区和服务器的时区不一致造成的，解决办法可以通过在 JVM 上配置 `-Doracle.jdbc.timezoneAsRegion=false` 忽略时区来绕过， 如果是在 python 里边，可以通过 `useLocalEnv(1, config={"jvm_startup_options":["-Doracle.jdbc.timezoneAsRegion=false"]})` 来设置

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
inputTableName	输入表名字	输入表名字	String	✓		
password	密码	密码	String	✓		
url	URL地址	URL地址	String	✓		
username	用户名	用户名	String	✓		
fetchSize	fetchSize	fetchSize	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

from pyalink.alink import *
useLocalEnv(1)

OracleSourceBatchOp()\
    .setUrl(URL)\
    .setUsername(USER_NAME)\
    .setPassword(PASSWORD)\
    .setInputTableName("iris")\
    .print()
```

Java 代码

```
    @Test
    public void test2() throws Exception {
        OracleSourceBatchOp oracleSourceBatchOp = new OracleSourceBatchOp()
            .setUrl(URL)
            .setUsername(USER_NAME)
            .setPassword(PASSWORD)
            .setInputTableName("iris");

        System.out.println(oracleSourceBatchOp.getSchema().toString());

        oracleSourceBatchOp.print();
    }
```

运行结果

root |-- sepal_length: DOUBLE |-- sepal_width: DOUBLE |-- petal_length: DOUBLE |-- petal_width: DOUBLE |-- category: STRING

sepal_length	sepal_width	petal_length	petal_width	category
4.6000	3.1000	1.5000	0.2000	Iris-setosa
4.8000	3.4000	1.9000	0.2000	Iris-setosa
6.1000	2.9000	4.7000	1.4000	Iris-versicolor
7.7000	3.0000	6.1000	2.3000	Iris-virginica
5.8000	2.7000	3.9000	1.2000	Iris-versicolor
5.6000	2.9000	3.6000	1.3000	Iris-versicolor
5.0000	3.0000	1.6000	0.2000	Iris-setosa
5.0000	3.6000	1.4000	0.2000	Iris-setosa

6.0000	2.7000	5.1000	1.6000	Iris-versicolor
6.3000	3.4000	5.6000	2.4000	Iris-virginica

..... 6.2000|3.4000|5.4000|2.3000|Iris-virginica 5.6000|2.7000|4.2000|1.3000|Iris-versicolor
6.8000|2.8000|4.8000|1.4000|Iris-versicolor 5.1000|3.5000|1.4000|0.3000|Iris-setosa
4.9000|3.1000|1.5000|0.1000|Iris-setosa 6.7000|3.0000|5.0000|1.7000|Iris-versicolor
5.9000|3.0000|5.1000|1.8000|Iris-virginica 5.7000|3.0000|4.2000|1.2000|Iris-versicolor
4.4000|3.0000|1.3000|0.2000|Iris-setosa 5.7000|3.8000|1.7000|0.3000|Iris-setosa

parquet文件读入 (ParquetSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.ParquetSourceBatchOp

Python 类名：ParquetSourceBatchOp

功能介绍

读parquet文件数据。支持从本地、hdfs、http读取，可以递归读取目录下全部文件，如果是分区目录，可以对分区进行选择。

分区选择

分区目录名格式为"分区名=值"，例如： city=beijing/month=06/day=17;city=hangzhou/month=06/day=18。 Alink将遍历目录下的分区名和分区值，构造分区表：

city	month	day
beijing	06	17
hangzhou	06	18

使用SQL语句查找分区，例如：ParquetSourceBatchOp.setPartitions("city = 'beijing'")，分区选择语法参考《Flink SQL 内置函数》，分区值为String类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
partitions	分区名	1)单级、单个分区示例：ds=20190729；2)多级分区之间用" / "分隔，例如：ds=20190729/dt=12；3)多个分区之间用","分隔，例如：ds=20190729,ds=20190730	String			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
filePath = 'https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/iris.parquet'
parquetSource = ParquetSourceBatchOp()\
    .setFilePath(filePath)
parquetSource.print()
```

Java 代码

```
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.CsvSourceStreamOp;
import org.junit.Test;

public class ParquetSourceStreamOpTest {
    @Test
    public void testParquetSourceStreamOp() throws Exception {
        PluginDownloader downloader =
AlinkGlobalConfiguration.getPluginDownloader();
        downloader.downloadPlugin("parquet");
        String parquetName = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.parquet";
        ParquetSourceBatchOp source = new ParquetSourceBatchOp()
            .setFilePath(new FilePath(parquetName, new
HttpFileReadOnlyFileSystem()));
        source.print();

        //选择分区示例
        new ParquetSourceBatchOp()
            .setFilePath(filepath)
            .setPartitions("year>'9'")
            .print();

        new ParquetSourceBatchOp()
            .setFilePath(filepath)
            .setPartitions("city = 'beijing'")
            .print();

        new ParquetSourceBatchOp()
            .setFilePath(filepath)
            .setPartitions("year < '7' OR year > '9'")
            .print();

        new ParquetSourceBatchOp()
            .setFilePath(new FilePath(parquetName2))
            .setPartitions("year BETWEEN '9' AND '10'")
            .print();
```

```
    }
}
```

运行结果

class	f0	f1	f2	f3
Iris-setosa	5.1000	3.5000	1.4000	0.2000
Iris-versicolor	5.0000	2.0000	3.5000	1.0000
Iris-setosa	5.1000	3.7000	1.5000	0.4000
Iris-virginica	6.4000	2.8000	5.6000	2.2000
Iris-versicolor	6.0000	2.9000	4.5000	1.5000
	.....			
Iris-setosa	4.9000	3.0000	1.4000	0.2000
Iris-versicolor	5.7000	2.6000	3.5000	1.0000
Iris-setosa	4.6000	3.6000	1.0000	0.2000
Iris-versicolor	5.9000	3.0000	4.2000	1.5000

随机生成结构数据源 (RandomTableSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.RandomTableSourceBatchOp

Python 类名：RandomTableSourceBatchOp

功能介绍

随机生成批式的表数据。 支持5种随机数据生成方法：uniform, uniform_open, gauss, weight_set 和 poisson

使用方法

setOutputColConfs(列配置信息参数)的含义和编写方法如下

参数示例	生成方法
uniform(1,2,nullper=0.1)	1到2闭区间上服从均匀分布，有10%的数据为null的随机数序列
uniform_open(1,2)	1到2开区间上服从均匀分布的随机数序列
weight_set(1.0,1.0,5.0,2.0)	由1和5组成的随机序列，生成概率为1：2
gauss(0,1)	满足均值为0、方差为1的正态分布的随机数序列
poisson(0.5)	满足lambda为0.5的泊松分布的随机序列

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
numCols	输出表列数目	输出表中列的数目，整型	Integer	✓		
numRows	输出表行数目	输出表中行的数目，整型	Long	✓		
idCol	id 列名	列名，若列名非空，表示输出表中包含一个整形序列id列，否则无该列	String			null
outputColConfs	列配置信息	表示每一列的数据分布配置信息	String			null

outputCols	输出列名数组	字符串数组，当参数不设置时，算法自动生成	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

# 生成测试数据，该数据符合高斯分布
data = RandomTableSourceBatchOp() \
                .setNumCols(4) \
                .setNumRows(10) \
                .setIdCol("id") \
                .setOutputCols(["group_id", "f0", "f1", "f2"]) \

.setOutputColConfs("group_id:weight_set(111.0,1.0,222.0,1.0);f0:gauss(0,2);f1:g
auss(0,2);f2:gauss(0,2)")
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.RandomTableSourceBatchOp;
import org.junit.Test;

public class RandomTableSourceBatchOpTest {
    @Test
    public void testRandomTableSourceBatchOp() throws Exception {
        BatchOperator <?> data = new RandomTableSourceBatchOp()
            .setNumCols(4)
            .setNumRows(10L)
            .setIdCol("id")
            .setOutputCols("group_id", "f0", "f1", "f2")

.setOutputColConfs("group_id:weight_set(111.0,1.0,222.0,1.0);f0:gauss(0,2);f1:g
auss(0,2);f2:gauss(0,2)");
    }
}
```

运行结果

group_id	f0	f1	f2
0	-0.9964239045050353	0.49718679973825497	0.1792735119342329
1	-0.6095874515728233	-0.4127806660140688	3.0630804909945755
2	2.213734518739278	-0.8455555927440792	-1.600352103528522
3	-2.815758921864138	0.1660690040206056	2.5530930456104337
4	0.4511700080470712	-0.3189014028331945	1.074516449728338
5	-0.04526610697025353	0.9372115152797922	0.8801699948291315
6	1.399940708626108	0.094717796828264	1.8070419026410982
7	-1.9583513162921702	2.8640034727735295	0.8341853784130754
8	-1.507498072024416	-0.1315003650747711	-3.695551497151364
9	-0.5509621008083586	-0.5880629518273883	1.5237202683647566

随机生成向量数据源 (RandomVectorSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.RandomVectorSourceBatchOp

Python 类名：RandomVectorSourceBatchOp

功能介绍

生成随机张量表

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
numRows	输出表行数目	输出表中行的数目，整型	Integer	✓		
size	张量size	整型数组，张量的size	Integer[]	✓		
sparsity	稀疏度	非零元素在所有张量数据中的占比	Double	✓		
idCol	id 列名	列名，若列名非空，表示输出表中包含一个整形序列id列，否则无该列	String			"alink_id"
outputCol	输出列名	输出随机生成的数据存储列名	String			"tensor"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

RandomVectorSourceBatchOp().setNumRows(5).setSize([2]).setSparsity(1.0).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.source.RandomVectorSourceBatchOp;
import org.junit.Test;

public class RandomVectorSourceBatchOpTest {
    @Test
    public void testRandomVectorSourceBatchOp() throws Exception {
        new RandomVectorSourceBatchOp().setNumRows(5).setSize(new Integer[]
{2}).setSparsity(1.0).print();
    }
}
```

运行结果

alink_id	tensor
0	$2$0:0.6374174253501083 1:0.5504370051176339
1	$2$0:0.20771484130971707
2	$2$0:0.49682259343089075 1:0.9858769332362016
3	$2$0:0.06712000939049956 1:0.768156984078079
4	$2$1:0.9186071189908658

TFRecordDataset文件读入 (TFRecordDatasetSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.TFRecordDatasetSourceBatchOp

Python 类名：TFRecordDatasetSourceBatchOp

功能介绍

读取 TFRecordDataset 文件（TFRecordDataset 的介绍可以参考 TensorFlow 文档：https://www.tensorflow.org/tutorials/load_data/tfrecord ）。

使用说明

需要指定文件路径 `filePath`，可以是单个文件，也可以是包含多个 TFRecordDataset 的目录。

为了将读取的 `TFRecord` 转换为 Alink 中的格式，需要指定数据的 `schemaStr`。由于 `TFRecord` 中 `Feature` 允许的数据类型仅有 `float, int64, bytes`，因此 `schemaStr` 填写的类型有一定的限制：

- `FLOAT_TENSOR` / `DOUBLE_TENSOR`：要求 `float` 特征；
- `LONG_TENSOR` / `INT_TENSOR`：要求 `int64` 特征；
- `STRING_TENSOR`：要求 `bytes` 特征；
- `BYTE_TENSOR`：要求 `bytes` 特征；
- `DENSE_VECTOR`：要求 `float` 特征；
- `LONG` / `INT`：要求 `int64` 特征，并且只使用第一个元素；
- `FLOAT` / `DOUBLE`：要求 `float` 特征，并且只使用第一个元素；
- `STRING`：要求 `bytes` 特征，并且只使用第一个元素；
- `VARBINARY`：要求 `bytes` 特征。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		

代码示例

Python 代码

```
schemaStr = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string"
source = TFRecordDatasetSourceBatchOp() \
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/iris.tfrecord") \
    .setSchemaStr(schemaStr)
source.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.TFRecordDatasetSourceBatchOp;
import org.junit.Test;

public class TFRecordDatasetSourceBatchOpTest {
    @Test
    public void testTFRecordDatasetSourceBatchOp() throws Exception {
        String schemaStr
            = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string";
        BatchOperator <?> source = new TFRecordDatasetSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/iris.tfrecord")
            .setSchemaStr(schemaStr);
        source.print();
    }
}
```

Table数据读入 (TableSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.TableSourceBatchOp

Python 类名：TableSourceBatchOp

功能介绍

从Table中生成BatchOperator数据

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "1 1 1"],
    [2, "2 2 2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
inOp.getOutputTable()
TableSourceBatchOp(inOp.getOutputTable()).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.source.TableSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class TableSourceBatchOpTest {
    @Test
    public void testTableSourceBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "1 1 1"),
            Row.of(2, "2 2 2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, vec
string");
        inOp.getOutputTable();
        new TableSourceBatchOp(inOp.getOutputTable()).print();
    }
}
```

运行结果

id	vec
0	0 0 0
1	1 1 1
2	2 2 2

Text文件读入 (TextSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.TextSourceBatchOp

Python 类名：TextSourceBatchOp

功能介绍

按行读取文件数据。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
ignoreFirstLine	是否忽略第一行数据	是否忽略第一行数据	Boolean			false
partitions	分区名	1)单级、单个分区示例：ds=20190729；2)多级分区之间用" / "分隔，例如：ds=20190729/dt=12；3)多个分区之间用","分隔，例如：ds=20190729,ds=20190730	String			null
textCol	文本列名称	文本列名称	String			"text"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

URL = "https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/iris.csv"
data = TextSourceBatchOp().setFilePath(URL).setTextCol("text")
data.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.TextSourceBatchOp;
import org.junit.Test;

public class TextSourceBatchOpTest {
    @Test
    public void testTextSourceBatchOp() throws Exception {
        String URL = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv";
        BatchOperator <?> data = new
TextSourceBatchOp().setFilePath(URL).setTextCol("text");
        data.print();
    }
}
```

运行结果

text
6.5,2.8,4.6,1.5,Iris-versicolor
6.1,3.0,4.9,1.8,Iris-virginica
7.3,2.9,6.3,1.8,Iris-virginica
5.7,2.8,4.5,1.3,Iris-versicolor
6.4,2.8,5.6,2.1,Iris-virginica
6.7,2.5,5.8,1.8,Iris-virginica

TSV文件读入 (TsvSourceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.source.TsvSourceBatchOp

Python 类名：TsvSourceBatchOp

功能介绍

读Tsv文件，Tsv文件是以tab为分隔符。文件来源可以是本地，oss，http，hdfs等。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
ignoreFirstLine	是否忽略第一行数据	是否忽略第一行数据	Boolean			false
partitions	分区名	1)单级、单个分区示例：ds=20190729；2)多级分区之间用" / "分隔，例如：ds=20190729/dt=12； 3)多个分区之间用","分隔，例如：ds=20190729,ds=20190730	String			null
skipBlankLine	是否忽略空行	是否忽略空行	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
                ["0L", "1L", 0.6],
                ["2L", "2L", 0.8],
                ["2L", "4L", 0.6],
                ["3L", "1L", 0.6],
                ["3L", "2L", 0.3],
                ["3L", "4L", 0.4]
        ])

source = BatchOperator.fromDataframe(df, schemaStr='uid string, iid string,
label double')

filepath = "/tmp/abc.tsv"
tsvSink = TsvSinkBatchOp()\
    .setFilePath(filepath)\
    .setOverwriteSink(True)

source.link(tsvSink)

BatchOperator.execute()

tsvSource = TsvSourceBatchOp().setFilePath(filepath).setSchemaStr("f string")
tsvSource.print()
```

Java 代码

```
package javatest.com.alibaba.alink.batch.source;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.TsvSinkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.source.TsvSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TsvSourceBatchOpTest {
    @Test
    public void testTsvSourceBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
```

```
            Row.of("0L", "1L", 0.6),
            Row.of("2L", "2L", 0.8),
            Row.of("2L", "4L", 0.6),
            Row.of("3L", "1L", 0.6),
            Row.of("3L", "2L", 0.3),
            Row.of("3L", "4L", 0.4)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df, "uid string, iid
string, label double");
        String filepath = "/tmp/abc.tsv";
        BatchOperator <?> tsvSink = new TsvSinkBatchOp()
            .setFilePath(filepath)
            .setOverwriteSink(true);
        source.link(tsvSink);
        BatchOperator.execute();
        BatchOperator <?> tsvSource = new
TsvSourceBatchOp().setFilePath(filepath).setSchemaStr("f string");
        tsvSource.print();
    }
}
```

运行结果

f
0L
2L
3L
3L
2L
3L

AK文件导出 (AkSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.AkSinkBatchOp

Python 类名：AkSinkBatchOp

功能介绍

将一个批式数据，以Ak文件格式写出到文件系统。Ak文件格式是Alink 自定义的一种文件格式，能够将数据的Schema保留输出的文件格式。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
numFiles	文件数目	文件数目	Integer			1
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
partitionCols	分区列	创建分区使用的列名	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')
```

```
filePath = "/tmp/test_alink_file_sink";

# write file to local disk
batchData.link(AkSinkBatchOp()\
                .setFilePath(FilePath(filePath))\
                .setOverwriteSink(True)\
                .setNumFiles(1))
BatchOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.common.io.filesystem.FilePath;
import com.alibaba.alink.common.io.filesystem.HadoopFileSystem;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.AkSinkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AkSinkBatchOpTest {
    @Test
    public void testAkSinkBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        String filePath = "/tmp/test_alink_file_sink";
        batchData.link(new AkSinkBatchOp()
            .setFilePath(new FilePath(filePath))
            .setOverwriteSink(true)
            .setNumFiles(1));
        String hdfsFilePath = "alink_fs_test/test_alink_file_sink";
        HadoopFileSystem fs = new HadoopFileSystem("2.8.3",
"hdfs://10.101.201.169:9000");
        batchData.link(new AkSinkBatchOp()
            .setFilePath(new FilePath(hdfsFilePath, fs))
```

```
            .setOverwriteSink(true)
            .setNumFiles(1));
        BatchOperator.execute();
    }
}
```

模型流导出 (AppendModelStreamFileSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.AppendModelStreamFileSinkBatchOp

Python 类名：AppendModelStreamFileSinkBatchOp

功能介绍

将模型按照给定的时间戳，插入模型流。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
modelTime	批模型时间戳	模型时间戳。默认当前时间。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null
numFiles	文件数目	文件数目	Integer			1
numKeepModel	保存模型的数目	实时写出模型的数目上限	Integer			2147483647

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0, 1.0],
    [2.0, "B", 1, 1, 0, 2.0],
    [3.0, "C", 2, 2, 1, 3.0],
    [4.0, "D", 3, 3, 1, 4.0]
])

input = BatchOperator.fromDataframe(df, schemaStr='f0 double, f1 string, f2
int, f3 int, label int, reg_label double')

rfOp = RandomForestTrainBatchOp()\
    .setLabelCol("reg_label")\
    .setFeatureCols(["f0", "f1", "f2", "f3"])\
    .setFeatureSubsamplingRatio(0.5)\
    .setSubsamplingRatio(1.0)\
    .setNumTreesOfInfoGain(1)\
    .setNumTreesOfInfoGain(1)\
    .setNumTreesOfInfoGainRatio(1)\
    .setCategoricalCols(["f1"])

modelStream = AppendModelStreamFileSinkBatchOp()\
    .setFilePath("/tmp/random_forest_model_stream")\
    .setNumKeepModel(10)

rfOp.linkFrom(input).link(modelStream)

BatchOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.RandomForestTrainBatchOp;
import com.alibaba.alink.operator.batch.sink.AppendModelStreamFileSinkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
```

模型流导出 (AppendModelStreamFileSinkBatchOp)

```
public class AppendModelStreamFileSinkBatchOpTest {
    @Test
    public void testAppendModelStreamFileSinkBatchOp() throws Exception {
        Row[] rows = new Row[] {
            Row.of(1.0, "A", 0L, 0, 0, 1.0),
            Row.of(2.0, "B", 1L, 1, 0, 2.0),
            Row.of(3.0, "C", 2L, 2, 1, 3.0),
            Row.of(4.0, "D", 3L, 3, 1, 4.0)
        };

        String[] colNames = new String[] {"f0", "f1", "f2", "f3", "label",
"reg_label"};

        String labelColName = colNames[4];

        MemSourceBatchOp input = new MemSourceBatchOp(
            Arrays.asList(rows), new String[] {"f0", "f1", "f2", "f3", "label",
"reg_label"}
        );

        RandomForestTrainBatchOp rfOp = new RandomForestTrainBatchOp()
            .setLabelCol(labelColName)
            .setFeatureCols(colNames[0], colNames[1], colNames[2], colNames[3])
            .setFeatureSubsamplingRatio(0.5)
            .setSubsamplingRatio(1.0)
            .setNumTreesOfInfoGain(1)
            .setNumTreesOfInfoGain(1)
            .setNumTreesOfInfoGainRatio(1)
            .setCategoricalCols(colNames[1]);

        rfOp.linkFrom(input).link(
            new AppendModelStreamFileSinkBatchOp()
                .setFilePath("/tmp/random_forest_model_stream")
                .setNumKeepModel(10)
        );

        BatchOperator.execute();
    }
}
```

Catalog数据表导出 (CatalogSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.CatalogSinkBatchOp

Python 类名：CatalogSinkBatchOp

功能介绍

Catalog描述了数据库的属性和数据库的位置, 支持Mysql, Derby, Sqlite, Hive.

在使用时，需要先下载插件，详情请看https://www.yuque.com/pinshu/alink_guide/czg4cx

定义分成三步：

第一步，定义Catalog

数据库	Java 接口
Derby	DerbyCatalog(String catalogName, String defaultDatabase, String derbyVersion, String derbyPath)
MySql	MySqlCatalog(String catalogName, String defaultDatabase, String mysqlVersion,String mysqlUrl, String port, String userName, String password)
Sqlite	SqliteCatalog(String catalogName, String defaultDatabase, String sqliteVersion, String dbUrl)
Hive	HiveCatalog(String catalogName, String defaultDatabase, String hiveVersion, String hiveConfDir) HiveCatalog(String catalogName, String defaultDatabase, String hiveVersion, FilePath hiveConfDir) HiveCatalog(String catalogName, String defaultDatabase, String hiveVersion, String hiveConfDir,String kerberosPrincipal, String kerberosKeytab)

示例：

```
    derby = DerbyCatalog("derby_test_catalog", DERBY_SCHEMA, "10.6.1.0",
derbyFolder+'/'+DERBY_DB)
```

各插件提供的版本：

```
    Hive: 2.3.4
    MySQL: 5.1.27
    Derby: 10.6.1.0
    SQLite: 3.19.3
    odps: 0.36.4-public
```

第二步， 定义CatalogObject

```
dbName = "sqlite_db"
tableName = "table"

# 第一个参数是Catalog, 第二个参数是DB/Project
catalogObject = CatalogObject(derby, ObjectPath(dbName, tableName))
```

第三步，定义Source和Sink

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
catalogObject	catalog object	catalog object	String	✓		

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

Derby

```
derbyFolder = "*"
DERBY_SCHEMA = "derby_schema"
DERBY_DB = "derby_db"
derby = DerbyCatalog("derby_test_catalog", DERBY_SCHEMA, "10.6.1.0",
derbyFolder+'/'+DERBY_DB)

catalogObject = CatalogObject(derby, ObjectPath("test_catalog_source_sink",
"test_catalog_source_sink3"))

catalogSinkBatchOp = CatalogSinkBatchOp()\
    .setCatalogObject(catalogObject)


source.link(catalogSinkBatchOp)

BatchOperator.execute()

catalogSourceBatchOp = CatalogSourceBatchOp()\
    .setCatalogObject(catalogObject)

catalogSourceBatchOp.print()
```

Java 代码

```
String derbyFolder = "*";
String DERBY_SCHEMA = "derby_schema";
String DERBY_DB = "derby_db";
DerbyCatalog derby = new DerbyCatalog("derby_test_catalog", DERBY_SCHEMA,
"10.6.1.0",
    derbyFolder + '/' + DERBY_DB);

CatalogObject catalogObject = new CatalogObject(derby,
    new ObjectPath("test_catalog_source_sink", "test_catalog_source_sink3"));

catalogSinkBatchOp catalogSinkStreamOp = new catalogSinkBatchOp()
    .setCatalogObject(catalogObject);

source.link(catalogSinkStreamOp);

StreamOperator.execute();

CatalogSourceBatchOp catalogSourceStreamOp = new CatalogSourceBatchOp()
    .setCatalogObject(catalogObject);

catalogSourceStreamOp.print();

StreamOperator.execute();
```

Sqlite

Python 代码

```
sqliteFolder = "*"
SQLITE_SCHEMA = "sqlite_schema"
SQLITE_DB = "sqlite_db"
sqlite = SqliteCatalog("sqlite_test_catalog", None, "3.19.3",
[sqliteFolder+'/'+SQLITE_DB])

catalogObject = CatalogObject(sqlite, ObjectPath(SQLITE_DB,
"test_catalog_source_sink3"))

catalogSinkBatchOp = CatalogSinkBatchOp()\
    .setCatalogObject(catalogObject)


source.link(catalogSinkBatchOp)

BatchOperator.execute()

catalogSourceBatchOp = CatalogSourceBatchOp()\
    .setCatalogObject(catalogObject)
```

```
catalogSourceBatchOp.print()
```

Java代码

```
String sqliteFolder = "*";
String SQLITE_SCHEMA = "sqlite_schema";
String SQLITE_DB = "sqlite_db";
SqliteCatalog sqlite = new SqliteCatalog("sqlite_test_catalog", null, "3.19.3",
sqliteFolder + '/' +
    SQLITE_DB);

CatalogObject catalogObject = new CatalogObject(sqlite, new
ObjectPath(SQLITE_DB,
    "test_catalog_source_sink3"));

CatalogSinkBatchOp catalogSinkStreamOp = CatalogSinkBatchOp()
    .setCatalogObject(catalogObject);

source.link(catalogSinkBatchOp);

StreamOperator.execute();

CatalogSourceBatchOp catalogSourceStreamOp = new CatalogSourceBatchOp()
    .setCatalogObject(catalogObject);

catalogSourceStreamOp.print();

StreamOperator.execute();
```

CSV文件导出 (CsvSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.CsvSinkBatchOp

Python 类名：CsvSinkBatchOp

功能介绍

将输入数据写出到CSV文件。支持写到本地、hdfs。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
fieldDelimiter	字段分隔符	字段分隔符	String			","
numFiles	文件数目	文件数目	Integer			1
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
partitionCols	分区列	创建分区使用的列名	String[]			null
quoteChar	引号字符	引号字符	Character			"\""
rowDelimiter	行分隔符	行分隔符	String			"\n"

代码示例

Python 代码

```
filePath = 'https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/iris.csv'
schema = 'sepal_length double, sepal_width double, petal_length double,
petal_width double, category string'
csvSource = CsvSourceBatchOp()\
    .setFilePath(filePath)\
    .setSchemaStr(schema)\
    .setFieldDelimiter(",")
csvSink = CsvSinkBatchOp()\
    .setFilePath('~/csv_test.txt')

csvSource.link(csvSink)
```

```
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.CsvSinkBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class CsvSinkBatchOpTest {

    @Test
    public void testCsvSinkBatchOp() throws Exception {
        String filePath = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv";
        String schema
            = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string";
        CsvSourceBatchOp csvSource = new CsvSourceBatchOp()
            .setFilePath(filePath)
            .setSchemaStr(schema)
            .setFieldDelimiter(",");
        CsvSinkBatchOp csvSink = new CsvSinkBatchOp()
            .setFilePath("~/csv_test.txt");

        csvSource.link(csvSink);

        BatchOperator.execute();
    }
}
```

CSV导出（For web）（CsvSinkForWebBatchOp）

Java 类名：com.alibaba.alink.operator.batch.sink.CsvSinkForWebBatchOp

Python 类名：CsvSinkForWebBatchOp

功能介绍

写出CSV格式文件，支持写出到本地文件和HDFS文件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
fieldDelimiter	字段分隔符	字段分隔符	String			","
numFiles	文件数目	文件数目	Integer			1
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
quoteChar	引号字符	引号字符	Character			"\""
quoteString	引号字符	引号字符	String			"\""
rowDelimiter	行分隔符	行分隔符	String			"\n"

<!--

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
   ['changjiang', 2000, 1.5],
    ['huanghe', 2001, 1.7],
    ['zhujiang', 2002, 3.6],
```

```
    ['changjiang', 2001, 2.4],
    ['huanghe', 2002, 2.9],
    ['zhujiang', 2003, 3.2]
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 string, f2
bigint, f3 double')

sink = CsvSinkBatchOp().setFilePath('/tmp/abc.csv').setOverwriteSink(True)
batch_data = batch_data.link(sink)

BatchOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.CsvSinkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CsvSinkForWebBatchOpTest {

    @Test
    public void testCsvSinkForWebBatchOp() throws Exception {
        List <Row> inputData = Arrays.asList(
            Row.of("changjiang", 2000, 1.5),
            Row.of("huanghe", 2001, 1.7),
            Row.of("zhujiang", 2002, 3.6),
            Row.of("changjiang", 2001, 2.4),
            Row.of("huanghe", 2002, 2.9),
            Row.of("zhujiang", 2003, 3.2)
        );
        BatchOperator <?> op = new MemSourceBatchOp(inputData, "f1 string, f2
bigint, f3 double" );

        new CsvSinkBatchOp()
            .setFilePath("/tmp/abc.csv" )
            .setOverwriteSink(true)
            .linkFrom(op);
        BatchOperator.execute();
    }
}
```

导出到HBase (HBaseSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.HBaseSinkBatchOp

Python 类名：HBaseSinkBatchOp

功能介绍

写HBase Plugin版。plugin版本为1.2.12。 写入时，指定HBase的zookeeper地址，表名称，列簇名称。指定rowkey列（可以是多列）和要写入的数据列（可以写多列）。 在使用时，需要先下载插件，详情请看 https://www.yuque.com/pinshu/alink_guide/czg4cx

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
familyName	簇值	簇值	String	✓		
pluginVersion	插件版本号	插件版本号	String	✓		
rowKeyCols	rowkey所在列	rowkey所在列	String[]	✓		
tableName	HBase表名称	HBase表名称	String	✓		
zookeeperQuorum	Zookeeper quorum	Zookeeper quorum 地址	String	✓		
timeout	HBase RPC 超时时间	HBase RPC 超时时间，单位毫秒	Integer			1000
valueCols	多数值列	多数值列	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    ["1", 10000, 10001.0, 10002],
    ["2", 20000, 20001.0, 20002]
])
```

```
data = BatchOperator.fromDataframe(df, schemaStr='userid string,red int,black
double,green int')

HBaseSinkBatchOp()\
    .setZookeeperQuorum("localhost:2181")\
    .setTableName("user")\
    .setRowKeyCols("userid")\
    .setFamilyName("color")\
    .setPluginVersion("1.2.12")\
    .setValueCols("red", "black","green")\
    .linkFrom(data)
BatchOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.HBaseSinkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HBaseTest {
    @Test
    public void testWriteStream() throws Exception {
        List <Row> datas = Arrays.asList(
            Row.of("1", 10000L, 10001.0, 10002),
            Row.of("2", 20000L, 20001.0, 20002)
        );
        BatchOperator op = new MemSourceBatchOp(datas, "userid string,red
long,black double,green int");
        HBaseSinkBatchOp hBaseSinkBatchOp = new HBaseSinkBatchOp()
            .setZookeeperQuorum("localhost:2181")
            .setTableName("user")
            .setRowKeyCols("userid")
            .setFamilyName("color")
            .setPluginVersion("1.2.12")
            .setValueCols("red", "black","green");
        hBaseSinkBatchOp.linkFrom(op);
        BatchOperator.execute();
    }
}
```

LibSvm文件导出 (LibSvmSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.LibSvmSinkBatchOp

Python 类名：LibSvmSinkBatchOp

功能介绍

写出LibSvm格式文件，支持写出到本地文件和HDFS文件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
labelCol	标签列名	输入表中的标签列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
partitionCols	分区列	创建分区使用的列名	String[]			null
startIndex	起始索引	起始索引	Integer			1

代码示例

Python 代码

```
df_data = pd.DataFrame([
    ['1:2.0 2:1.0 4:0.5', 1.5],
    ['1:2.0 2:1.0 4:0.5', 1.7],
    ['1:2.0 2:1.0 4:0.5', 3.6]
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 string, f2
double')

sink =
```

```
LibSvmSinkBatchOp().setFilePath('/tmp/abc.svm').setLabelCol("f2").setVectorCol(
"f1").setOverwriteSink(True)
batch_data = batch_data.link(sink)

BatchOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.LibSvmSinkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LibSvmSinkBatchOpTest {
    @Test
    public void testLibSvmSinkBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("1:2.0 2:1.0 4:0.5", 1.5),
            Row.of("1:2.0 2:1.0 4:0.5", 1.7),
            Row.of("1:2.0 2:1.0 4:0.5", 3.6)
        );
        BatchOperator <?> batch_data = new MemSourceBatchOp(df_data, "f1
string, f2  double");
        BatchOperator <?> sink = new
LibSvmSinkBatchOp().setFilePath("/tmp/abc.svm").setLabelCol("f2").setVectorCol(
            "f1").setOverwriteSink(true);
        batch_data.link(sink);
        BatchOperator.execute();
    }
}
```

MySQL导出 (MySqlSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.MySqlSinkBatchOp

Python 类名：MySqlSinkBatchOp

功能介绍

写Mysql表

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
dbName	数据库名字	数据库名字	String	✓		
ip	IP地址	IP地址	String	✓		
outputTableName	输出表名字	输出表名字	String	✓		
password	密码	密码	String	✓		
port	端口	端口	String	✓		
username	用户名	用户名	String	✓		

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
data = MySqlSourceBatchOp()\
     .setDbName("alink")\
     .setUsername("alink")\
     .setPassword("passwd")\
     .setIp("***")\
     .setPort("20001")\
     .setOutputTableName("test_out")
```

Oracle导出 (OracleSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.OracleSinkBatchOp

Python 类名：OracleSinkBatchOp

功能介绍

向 Oracle 数据库中写出数据，当前支持 11g 版本

URL 说明

格式类似于如下：

"jdbc:oracle:thin:@//host:port/service"

注意事项

1. 报错 `ORA-01882: timezone region not found`

   原因是因为 Oracle 服务的时区和服务器的时区不一致造成的，解决办法可以通过在 JVM 上配置 `-Doracle.jdbc.timezoneAsRegion=false` 忽略时区来绕过， 如果是在 python 里边，可以通过 `useLocalEnv(1, config={"jvm_startup_options":["-Doracle.jdbc.timezoneAsRegion=false"]})` 来设置

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputTableName	输出表名字	输出表名字	String	✓		
password	密码	密码	String	✓		
url	URL地址	URL地址	String	✓		
username	用户名	用户名	String	✓		
batchInterval	batchInterval	batchInterval	Integer			5000
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
from pyalink.alink import *
useLocalEnv(1)

CsvSourceBatchOp()\
    .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv")\
    .setSchemaStr(\
        "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string"\
    )\
    .link(
        OracleSinkBatchOp()\
            .setUrl(URL)\
            .setUsername(USER_NAME)\
            .setPassword(PASSWORD)\
            .setOverwriteSink(True)\
            .setOutputTableName("iris")\
    )

BatchOperator.execute();
```

Java 代码

```
    @Test
    public void testWrite() throws Exception {
        new CsvSourceBatchOp()
            .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv")
            .setSchemaStr(
                "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string"
            )
            .link(
                new OracleSinkBatchOp()
                    .setUrl(URL)
                    .setUsername(USER_NAME)
                    .setPassword(PASSWORD)
                    .setOverwriteSink(true)
                    .setOutputTableName("iris")
            );

        BatchOperator.execute();
    }
```

导出到Redis (RedisRowSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.RedisRowSinkBatchOp

Python 类名：RedisRowSinkBatchOp

功能介绍

将一个批式数据，按行写到Redis里，键和值可以是多列。

在使用时，需要先下载插件，详情请看https://www.yuque.com/pinshu/alink_guide/czg4cx

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
pluginVersion	插件版本号	插件版本号	String	✓		
clusterMode	集群模式	是集群模式还是单机模式	Boolean			false
databaseIndex	数据库索引号	数据库索引号	Long			
keyCols	多键值列	多键值列	String[]			null
pipelineSize	流水线大小	Redis 发送命令流水线的大小	Integer			1
redisIPs	Redis IP	Redis 集群的 IP/端口	String[]			
redisPassword	Redis 密码	Redis 服务器密码	String			
timeout	超时	关闭连接的超时时间	Integer			
valueCols	多数值列	多数值列	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
redisIP = "127.0.0.1:6379"

df = pd.DataFrame([
    ["football", 1.0],
    ["football", 2.0],
    ["football", 3.0],
    ["basketball", 4.0],
    ["basketball", 5.0],
    ["tennis", 6.0],
    ["tennis", 7.0],
    ["pingpang", 8.0],
    ["pingpang", 9.0],
    ["baseball", 10.0]])

batchData = BatchOperator.fromDataframe(df, schemaStr='id string,val double')

batchData.link(RedisRowSinkBatchOp()\
            .setRedisIPs(redisIP)\
            .setKeyCols(["id"])\
            .setValueCols(["val"])\
            .setPluginVersion("2.9.0"))

BatchOperator.execute()
```

Java 代码

```
package com.alibaba.alink.operator.batch.sink;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RedisRowSinkBatchOpTest extends AlinkTestBase {
    @Test
    public void test() throws Exception {
        String redisIP = "127.0.0.1:6379";

        List <Row> df = Arrays.asList(
            Row.of("football", 1.0),
            Row.of("football", 2.0),
            Row.of("football", 3.0),
```

```
            Row.of("basketball", 4.0),
            Row.of("basketball", 5.0),
            Row.of("tennis", 6.0),
            Row.of("tennis", 7.0),
            Row.of("pingpang", 8.0),
            Row.of("pingpang", 9.0),
            Row.of("baseball", 10.0)
        );

        BatchOperator <?> data = new MemSourceBatchOp(df, "id string,val
double");

        RedisRowSinkBatchOp sink = new RedisRowSinkBatchOp()
            .setPluginVersion("2.9.0")
            .setRedisIPs(redisIP)
            .setKeyCols("id")
            .setValueCols("val");

        data.link(sink);

        BatchOperator.execute();
    }

}
```

kv均为String的数据导出到Redis (RedisStringSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.RedisStringSinkBatchOp

Python 类名：RedisStringSinkBatchOp

功能介绍

将一个批式数据，（单列String类型键值）按行写到Redis里。

注意事项

- 与RedisRowSinkBatchOp不同写入的是序列化后的字符串，例如："A"写入Redis的结果为"A"
- 在使用时，需要先下载插件，详情请看https://www.yuque.com/pinshu/alink_guide/czg4cx

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
pluginVersion	插件版本号	插件版本号	String	✓		
clusterMode	集群模式	是集群模式还是单机模式	Boolean			false
databaseIndex	数据库索引号	数据库索引号	Long			
keyCol	单键列	单键列	String			null
pipelineSize	流水线大小	Redis 发送命令流水线的大小	Integer			1
redisIPs	Redis IP	Redis 集群的 IP/端口	String[]			
redisPassword	Redis 密码	Redis 服务器密码	String			
timeout	超时	关闭连接的超时时间	Integer			
valueCol	单值列	单值列	String			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
redisIP = "*"
redisPort = 0

df = pd.DataFrame([
    ["football", "1.0"],
    ["football", "2.0"],
    ["football", "3.0"]])

batchData = BatchOperator.fromDataframe(df, schemaStr='id string,val double')

batchData.link(RedisStringSinkBatchOp()\
            .setRedisIPs(redisIP)\
            .setKeyCol(["id"])\
            .setValueCol(["val"])\
            .setPluginVersion("2.9.0"))

BatchOperator.execute()
```

Java 代码

```
package com.alibaba.alink.operator.batch.sink;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RedisStringSinkBatchOpTest extends AlinkTestBase {
    @Test
    public void test() throws Exception {
        String redisIP = "127.0.0.1:6379";
        int redisPort = 0;

        List <Row> df = Arrays.asList(
            Row.of("football", "1.0"),
            Row.of("football", "2.0")
        );

        BatchOperator <?> data = new MemSourceBatchOp(df, "id string,val
string");
```

```
        RedisStringSinkBatchOp sink = new RedisStringSinkBatchOp()
            .setRedisIPs(redisIP)
            .setKeyCol("id")
            .setValueCol("val")
            .setPluginVersion("2.9.0");

        data.link(sink);

        BatchOperator.execute();
    }

}
```

TFRecordDataset文件导出 (TFRecordDatasetSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.TFRecordDatasetSinkBatchOp

Python 类名：TFRecordDatasetSinkBatchOp

功能介绍

写出 TFRecordDataset 文件（TFRecordDataset 的介绍可以参考 TensorFlow 文档：https://www.tensorflow.org/tutorials/load_data/tfrecord ）。

使用说明

需要指定文件路径 `filePath`，默认为单并行度写出单个文件。 如果希望并行的写出文件，那么需要设置参数 `numFiles`，得到的是一个包含多个 TFRecordDataset 文件的目录。

`TFRecord` 中， `Feature` 允许的数据类型仅有 `float, int64, bytes`。 其中数据类型为 `bytes` 时，存储的数据实际上为 `ByteString` 的列表；为 `float, int64` 时存储的数据为 `float` 或 `int64` 的列表。 数据写出时会根据在 Alink 中的类型进行转换， 其他类型请先使用类型转换组件进行转换：

- `DOUBLE, FLOAT, BIG_DEC`： 转为 `float` 特征；
- `LONG, INT, BIG_INT, SHORT`： 转为 `int64` 特征；
- `STRING`： 转为 `bytes` 特征，按 UTF8 编码对应 1 个 `ByteString`；
- `DENSE_VECTOR`： 转为 `float` 特征；
- `FLOAT_TENSOR, DOUBLE_TENSOR`： 转为 `float` 特征，数据被展平为1维；
- `INT_TENSOR, LONG_TENSOR`： 转为 `int64` 特征，数据被展平为1维；
- `BYTE_TENSOR`： 转为 `bytes` 特征，rank = 1 时对应 1 个 `ByteString`，rank = 2时对应 `ByteString` 的列表，其他 rank 不支持；
- `STRING_TENSOR`： 转为 `bytes` 特征，按 UTF8 编码对应 `ByteString` 的列表；
- `VARBINARY`： 转为 `bytes` 特征，对应 1 个 `ByteString`。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
numFiles	文件数目	文件数目	Integer			1
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

代码示例

Python 代码

```
schemaStr = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string"
source = CsvSourceBatchOp() \
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/iris.csv") \
    .setSchemaStr(schemaStr)
sink = TFRecordDatasetSinkBatchOp() \
    .setFilePath("/tmp/iris.tfrecord") \
    .setOverwriteSink(True) \
    .linkFrom(source)
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.TFRecordDatasetSinkBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class TFRecordDatasetSinkBatchOpTest {
    @Test
    public void testTFRecordDatasetSinkBatchOp() throws Exception {
        String schemaStr
            = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string";
        BatchOperator <?> source = new CsvSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/iris.csv")
            .setSchemaStr(schemaStr);
        BatchOperator <?> sink = new TFRecordDatasetSinkBatchOp()
            .setFilePath("/tmp/iris.tfrecord")
            .setOverwriteSink(true)
            .linkFrom(source);
        BatchOperator.execute();
    }
}
```

Text文件导出 (TextSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.TextSinkBatchOp

Python 类名：TextSinkBatchOp

功能介绍

按行写出到文件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
numFiles	文件数目	文件数目	Integer			1
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
partitionCols	分区列	创建分区使用的列名	String[]			null

代码示例

Python 代码

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

```
df_data = pd.DataFrame([
   ['changjiang', 2000, 1.5],
    ['huanghe', 2001, 1.7],
    ['zhujiang', 2002, 3.6],
    ['changjiang', 2001, 2.4],
    ['huanghe', 2002, 2.9],
    ['zhujiang', 2003, 3.2]
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 string, f2
bigint, f3 double')

TextSinkBatchOp().setFilePath('yourFilePath').setOverwriteSink(True).linkFrom(b
```

```
atch_data)
BatchOperator.execute()
```

Java 代码

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sink.TextSinkBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TextSinkBatchOpTest {
    @Test
    public void testTextSinkBatchOp() throws Exception {
        List <Row> inputRows = Arrays.asList(
            Row.of("changjiang", 2000, 1.5),
            Row.of("huanghe", 2001, 1.7),
            Row.of("zhujiang", 2002, 3.6),
               Row.of("changjiang", 2001, 2.4),
               Row.of("huanghe", 2002, 2.9),
              Row.of("zhujiang", 2003, 3.2)
        );

        BatchOperator <?> dataOp = new MemSourceBatchOp(inputRows, "f1 string,
f2 int, f3 double");

        new TextSinkBatchOp()
            .setFilePath("yourFilePath")
            .setOverwriteSink(true)
            .linkFrom(dataOp);
        dataOp.link(sink);
        BatchOperator.execute();
    }
}
```

运行结果

TSV文件导出 (TsvSinkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sink.TsvSinkBatchOp

Python 类名：TsvSinkBatchOp

功能介绍

写Tsv文件，Tsv文件是以tab为分隔符。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
filePath	文件路径	文件路径	String	✓		
numFiles	文件数目	文件数目	Integer			1
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
partitionCols	分区列	创建分区使用的列名	String[]			null

代码示例

Python 代码

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

```
df = pd.DataFrame([
                ["0L", "1L", 0.6],
                ["2L", "2L", 0.8],
                ["2L", "4L", 0.6],
                ["3L", "1L", 0.6],
                ["3L", "2L", 0.3],
                ["3L", "4L", 0.4]
        ])

source = BatchOperator.fromDataframe(df, schemaStr='uid string, iid string,
label double')

tsvSink = TsvSinkBatchOp().setFilePath('yourFilePath').linkFrom(source)
```

```
BatchOperator.execute()
```

Java 代码

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.TsvSinkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.source.TsvSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TsvSinkBatchOpTest {
    @Test
    public void testTsvSinkBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("0L", "1L", 0.6),
            Row.of("2L", "2L", 0.8),
            Row.of("2L", "4L", 0.6),
            Row.of("3L", "1L", 0.6),
            Row.of("3L", "2L", 0.3),
            Row.of("3L", "4L", 0.4)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df, "uid string, iid
string, label double");
        BatchOperator <?> tsvSink = new TsvSinkBatchOp()
            .setFilePath("yourFilePath")
            .setOverwriteSink(true);
        source.link(tsvSink);
        BatchOperator.execute();
    }
}
```

运行结果

Agg表查找 (AggLookupBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.AggLookupBatchOp

Python 类名：AggLookupBatchOp

功能介绍

需要查找多个值，并统计结果的总和、平均值、最大最小值或拼接查询结果时，可以使用聚合查找，该组件有两个输入，依次是模型数据表和原始数据表。模型数据有两列，依次是String类型和DenseVector类型，原始数据有任意行和列，每列都是String类型。原始数据默认使用空格作为单词的分隔符。

使用方法

在机器学习中，想要使用embedding结果时，可以用该组件进行数据处理和特征生成，例如加载训练好的词向量模型，对文本进行向量化和特征生成。在下面例子中，modelOp是一个词向量字典，inOp是三条英文句子，在该例子中，输出是拼接、简单求和、平均等方法得到的句子特征向量。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
clause	运算语句	运算语句	String	✓		
delimiter	分隔符	用来分割字符串	String			" "
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
```

```
    ["the quality of the word vectors increases"],
    ["amount of the training data increases"],
    ["the training speed is significantly improved"]
])


inOp = BatchOperator.fromDataframe(df, schemaStr='sentence string')

df2 = pd.DataFrame([
    ["the", "0.6343,0.8561,0.1249,0.4701"],
    ["training", "0.2753,0.2444,0.3699,0.6048"],
    ["of", "0.3160,0.3675,0.1649,0.4116"],
    ["increases", "1.0372,0.6092,0.1050,0.2630"],
    ["word", "0.9911,0.6338,0.4570,0.8451"],
    ["vectors", "0.8780,0.4500,0.5455,0.7495"],
    ["speed", "0.9504,0.3168,0.7484,0.6965"],
    ["significantly", "-0.0465,0.6597,0.0906,0.7137"],
    ["quality", "0.9745,0.7521,0.8874,0.5192"],
    ["is", "0.8221,0.0487,-0.0065,0.4088"],
    ["improved", "0.1910,0.0723,0.8216,0.4367"],
    ["data", "0.8985,0.0117,0.8083,0.9636"],
    ["amount", "0.9786,0.1470,0.7385,0.8856"]
])

modelOp = BatchOperator.fromDataframe(df2, schemaStr="id string, vec string")

AggLookupBatchOp() \
    .setClause("CONCAT(sentence,2) as concat, AVG(sentence) as avg,
SUM(sentence) as sum,MAX(sentence) as max,MIN(sentence) as min") \
    .setDelimiter(" ") \
    .setReservedCols([]) \
    .linkFrom(modelOp, inOp) \
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.AggLookupBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AggLookupBatchOpTest {
```

```
    @Test
    public void testAggLookupBatchOp() throws Exception {

        List <Row> df = Arrays.asList(
            Row.of("the quality of the word vectors increases"),
            Row.of("amount of the training data increases"),
            Row.of("the training speed is significantly improved")
        );

        BatchOperator <?> inOp = new MemSourceBatchOp(df, "sentence string");

        List <Row> df2 = Arrays.asList(
            Row.of("the", "0.6343,0.8561,0.1249,0.4701"),
            Row.of("training", "0.2753,0.2444,0.3699,0.6048"),
            Row.of("of", "0.3160,0.3675,0.1649,0.4116"),
            Row.of("increases", "1.0372,0.6092,0.1050,0.2630"),
            Row.of("word", "0.9911,0.6338,0.4570,0.8451"),
            Row.of("vectors", "0.8780,0.4500,0.5455,0.7495"),
            Row.of("speed", "0.9504,0.3168,0.7484,0.6965"),
            Row.of("significantly", "-0.0465,0.6597,0.0906,0.7137"),
            Row.of("quality", "0.9745,0.7521,0.8874,0.5192"),
            Row.of("is", "0.8221,0.0487,-0.0065,0.4088"),
            Row.of("improved", "0.1910,0.0723,0.8216,0.4367"),
            Row.of("data", "0.8985,0.0117,0.8083,0.9636"),
            Row.of("amount", "0.9786,0.1470,0.7385,0.8856")
        );
        BatchOperator <?> modelOp = new MemSourceBatchOp(df2, "id string, vec
string");
        AggLookupBatchOp aggLookupBatchOp = new AggLookupBatchOp()
            .setClause(
                "CONCAT(sentence,2) as concat, AVG(sentence) as avg,
SUM(sentence) as sum,MAX(sentence) as max,MIN(sentence) "
                    + "as min")
            .setDelimiter(" ")
            .linkFrom(modelOp, inOp);

        aggLookupBatchOp.select(new String[] {"e0"})
            .print();

        aggLookupBatchOp.select(new String[] {"e1", "e2", "e3", "e4"})
            .print();
    }
}
```

运行结果

concat

0.6343 0.8561 0.1249 0.4701 0.9745 0.7521 0.8874 0.5192
0.9786 0.147 0.7385 0.8856 0.316 0.3675 0.1649 0.4116
0.6343 0.8561 0.1249 0.4701 0.2753 0.2444 0.3699 0.6048

avg	sum	max	min
0.7807 0.6464 0.3442 0.5326	5.4654 4.5248 2.4096 3.7286	1.0372 0.8561 0.8874 0.8451	0.316 0.3675 0.105 0.263
0.6899 0.3726 0.3852 0.5997	4.1399 2.2359 2.3115 3.5987	1.0372 0.8561 0.8083 0.9636	0.2753 0.0117 0.105 0.263
0.4710 0.3663 0.3581 0.5550	2.8266 2.1980 2.1489 3.3306	0.9504 0.8561 0.8216 0.7137	-0.0465 0.0487 -0.0065 0.4088

添加id列 (AppendIdBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.AppendIdBatchOp

Python 类名：AppendIdBatchOp

功能介绍

将表附加ID列

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
appendType	append类型	append类型，"UNIQUE"和"DENSE"，分别为稀疏和稠密，稀疏的为非连续唯一id，稠密的为连续唯一id	String		"DENSE", "UNIQUE"	"DENSE"
idCol	ID列名	ID列名	String			"append_

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
inOp = BatchOperator.fromDataframe(df, schemaStr='f0 double,f1 string,f2 int,f3
int,label int')
AppendIdBatchOp()\
.setIdCol("append_id")\
```

```
.linkFrom(inOp)\
.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.AppendIdBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AppendIdBatchOpTest {
    @Test
    public void testAppendIdBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "f0 double,f1
string,f2 int,f3 int,label int");
        new AppendIdBatchOp()
            .setIdCol("append_id")
            .linkFrom(inOp)
            .print();
    }
}
```

运行结果

f0	f1	f2	f3	label	append_id
1.0000	A	0	0	0	0
2.0000	B	1	1	0	1
3.0000	C	2	2	1	2
4.0000	D	3	3	1	3

单列拆分多列 (FieldSplitBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.FieldSplitBatchOp

Python 类名：FieldSplitBatchOp

功能介绍

该组件将一列数据拆分为多列。可以处理csv、json、kv等格式的数据。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
colDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
dataFormat	数据格式	数据格式。json,csv,plain,kv	String		"JSON", "CSV", "PLAIN", "KV"	
fieldDelimiter	字段分隔符	字段分隔符	String			","
lengthCheck	字段检查策略	字段检查策略。skip: 字段缺失或解析异常时丢弃该数据；exception: 字段缺失或解析异常时抛出异常；pad: 缺失或解析异常的字段补null	String		"SKIP", "EXCEPTION", "PAD"	"SKIP"
reservedCols	算法保留列名	算法保留列	String[]			null

valDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([['{"id":"ID01","location":"beijing","degree":25}'],
['{"id":"ID02","location":"guangzhou","degree":35}']])
batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 string')

op = FieldSplitBatchOp().setSelectedCol('f1') \
  .setSchemaStr('id string, location string, degree bigint') \
  .setDataFormat('json').setReservedCols([])
batch_data = batch_data.link(op)

batch_data.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;

public class FieldSplitBatchOpTest extends AlinkTestBase {
    @Test
    public void testSplitField() throws Exception {
        Row[] rows = new Row[] {
            Row.of("{\"id\":\"ID01\",\"location\":\"beijing\",\"degree\":25}"),
            Row.of("
{\"id\":\"ID02\",\"location\":\"guangzhou\",\"degree\":35}")
        };

        BatchOperator data = new MemSourceBatchOp(Arrays.asList(rows), "f1
```

```
string");

        BatchOperator op = new FieldSplitBatchOp()
            .setSelectedCol("f1")
            .setDataFormat("json")
            .setSchemaStr("id string, location string, degree bigint");

        op.linkFrom(data).print();
    }
}
```

运行结果

f1	id	location	degree
{"id":"ID01","location":"beijing","degree":25}	ID01	beijing	25
{"id":"ID02","location":"guangzhou","degree":35}	ID02	guangzhou	35

前N个数 (FirstNBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.FirstNBatchOp

Python 类名：FirstNBatchOp

功能介绍

该组件输出表的前N条数据。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
size	采样个数	采样个数	Integer	✓	[1, +inf)	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
       ["0,0,0"],
       ["0.1,0.1,0.1"],
       ["0.2,0.2,0.2"],
       ["9,9,9"],
       ["9.1,9.1,9.1"],
       ["9.2,9.2,9.2"]
])

# batch source
inOp = BatchOperator.fromDataframe(df, schemaStr='Y string')

sampleOp = FirstNBatchOp()\
        .setSize(2)

inOp.link(sampleOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.FirstNBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FirstNBatchOpTest {
    @Test
    public void testFirstNBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("0,0,0"),
            Row.of("0.1,0.1,0.1"),
            Row.of("0.2,0.2,0.2"),
            Row.of("9,9,9"),
            Row.of("9.1,9.1,9.1"),
            Row.of("9.2,9.2,9.2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "Y string");
        BatchOperator <?> sampleOp = new FirstNBatchOp()
            .setSize(2);
        inOp.link(sampleOp).print();
    }
}
```

运行结果

Y
0,0,0
0.1,0.1,0.1

MTable展开 (FlattenMTableBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.FlattenMTableBatchOp

Python 类名：FlattenMTableBatchOp

功能介绍

该组件将 MTable 展开成 Table。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[M_TABLE]	
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
import numpy as np
import pandas as pd
from pyalink.alink import *
```

```
df_data = pd.DataFrame([
      ["a1", "11L", 2.2],
      ["a1", "12L", 2.0],
      ["a2", "11L", 2.0],
      ["a2", "12L", 2.0],
      ["a3", "12L", 2.0],
      ["a3", "13L", 2.0],
      ["a4", "13L", 2.0],
      ["a4", "14L", 2.0],
      ["a5", "14L", 2.0],
      ["a5", "15L", 2.0],
      ["a6", "15L", 2.0],
      ["a6", "16L", 2.0]
])

input = BatchOperator.fromDataframe(df_data, schemaStr='id string, f0 string,
f1 double')

zip = GroupByBatchOp()\
    .setGroupByPredicate("id")\
    .setSelectClause("id, mtable_agg(f0, f1) as m_table_col")

flatten = FlattenMTableBatchOp()\
    .setReservedCols(["id"])\
    .setSelectedCol("m_table_col")\
    .setSchemaStr('f0 string, f1 int')

zip.linkFrom(input).link(flatten).print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.dataproc;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.ArrayList;
import java.util.List;

/**
 * Test cases for gbdt.
 */
```

```
public class FlattenMTableTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
        List <Row> rows = new ArrayList <>();
        rows.add(Row.of("a1", "11L", 2.2));

        rows.add(Row.of("a1", "12L", 2.0));
        rows.add(Row.of("a2", "11L", 2.0));
        rows.add(Row.of("a2", "12L", 2.0));
        rows.add(Row.of("a3", "12L", 2.0));
        rows.add(Row.of("a3", "13L", 2.0));
        rows.add(Row.of("a4", "13L", 2.0));
        rows.add(Row.of("a4", "14L", 2.0));
        rows.add(Row.of("a5", "14L", 2.0));
        rows.add(Row.of("a5", "15L", 2.0));
        rows.add(Row.of("a6", "15L", 2.0));
        rows.add(Row.of("a6", "16L", 2.0));

        BatchOperator input = new MemSourceBatchOp(rows, "id string, f0 string,
f1 double");

        GroupByBatchOp zip = new GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("id, mtable_agg(f0, f1) as m_table_col");

        FlattenMTableBatchOp flatten = new FlattenMTableBatchOp()
            .setReservedCols("id")
            .setSelectedCol("m_table_col")
            .setSchemaStr("f0 string, f1 int");

        zip.linkFrom(input).link(flatten).print();
    }
}
```

运行结果

id	f0	f1
a2	11L	2
a2	12L	2
a4	13L	2
a4	14L	2
a5	14L	2

a5	15L	2
a1	11L	2
a1	12L	2
a3	12L	2
a3	13L	2
a6	15L	2
a6	16L	2

并行ID化预测 (HugeIndexerStringPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.HugeIndexerStringPredictBatchOp

Python 类名：HugeIndexerStringPredictBatchOp

功能介绍

提供字符串ID化处理功能

由StringIndexerTrainBatchOp生成词典模型，将输入数据的ID类型转化成词典模型中对应的字符串。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[LONG]	
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP"
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
        ["football", "apple"],
           ["football", "apple"],
           ["football", "apple"],
           ["basketball", "apple"],
           ["basketball", "apple"],
           ["tennis", "pair"],
           ["tennis", "pair"],
           ["pingpang", "banana"],
           ["pingpang", "banana"],
           ["baseball", "banana"]
])

data = BatchOperator.fromDataframe(df, schemaStr='f0 string, f1 string')

stringindexer = StringIndexerTrainBatchOp()\
            .setSelectedCol("f0")\
            .setSelectedCols(["f1"])\
            .setStringOrderType("alphabet_asc")

predictor = HugeStringIndexerPredictBatchOp().setSelectedCols(["f0", "f1"])\
            .setOutputCols(["f0_indexed", "f1_indexed"])

model = stringindexer.linkFrom(data)

result = predictor.linkFrom(model, data)

indexerString =
HugeIndexerStringPredictBatchOp().setSelectedCols(["f0_indexed",
"f1_indexed"])\
            .setOutputCols(["f0_source", "f1_source"])

indexerString.linkFrom(model, result).print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.dataproc;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
```

```
import java.util.List;

public class HugeIndexerStringPredictBatchOpTest {
    @Test
    public void testStringIndexerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("football", "apple"),
            Row.of("football", "apple"),
            Row.of("football", "apple"),
            Row.of("basketball", "apple"),
            Row.of("basketball", "apple"),
            Row.of("tennis", "pair"),
            Row.of("tennis", "pair"),
            Row.of("pingpang", "banana"),
            Row.of("pingpang", "banana"),
            Row.of("baseball", "banana")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 string,f1
string");

        BatchOperator <?> stringindexer = new StringIndexerTrainBatchOp()
            .setSelectedCol("f0")
            .setSelectedCols("f1")
            .setStringOrderType("alphabet_asc");
        BatchOperator <?> predictor = new
HugeStringIndexerPredictBatchOp().setSelectedCols("f0", "f1")
            .setOutputCols("f0_indexed", "f1_indexed");
        BatchOperator model = stringindexer.linkFrom(data);
        model.lazyPrint(10);
        BatchOperator result = predictor.linkFrom(model, data);
        result.lazyPrint(10);

        BatchOperator <?> indexerString = new
HugeIndexerStringPredictBatchOp().setSelectedCols("f0_indexed", "f1_indexed")
            .setOutputCols("f0_source", "f1_source");
        indexerString.linkFrom(model, result).print();
    }
}
```

运行结果

f0	f1	f0_indexed	f1_indexed	f0_source	f1_source
basketball	apple	3	0	basketball	apple
football	apple	4	0	football	apple
basketball	apple	3	0	basketball	apple

pingpang	banana	6	1	pingpang	banana
football	apple	4	0	football	apple
tennis	pair	7	5	tennis	pair
tennis	pair	7	5	tennis	pair
pingpang	banana	6	1	pingpang	banana
baseball	banana	2	1	baseball	banana
football	apple	4	0	football	apple

HugeLookup (HugeLookupBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.HugeLookupBatchOp

Python 类名：HugeLookupBatchOp

功能介绍

支持大数据量的查找功能，可实现两种数据按照某些列的合并操作，类似于数据的LEFT JOIN功能。

分别指定模型数据和输入数据中查找等值的Key列，模型数据中需要拼接到输入数据中的列名，最终输出数据数据列 + 模型数据拼接的列

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
mapKeyCols	Key列名	模型中对应的查找等值的列名	String[]			null
mapValueCols	Values列名	模型中需要拼接到样本中的列名	String[]			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

modelStreamUpdateMethod	模型更新方法	模型更新方法，可选COMPLETE（全量更新）或者INCREMENT（增量更新）	String		"COMPLETE", "INCREMENT"	"CO

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

data_df = pd.DataFrame([
    [10, 2.0],
    [1, 2.0],
    [-3, 2.0],
    [5, 1.0]
])

inOp = BatchOperator.fromDataframe(data_df, schemaStr='f0 int, f1 double')

model_df = pd.DataFrame([
    [1, "value1"],
    [2, "value2"],
    [5, "value5"]
])

modelOp = BatchOperator.fromDataframe(model_df, schemaStr="key_col int,
value_col string")

HugeLookupBatchOp()\
    .setMapKeyCols(["key_col"])\
    .setMapValueCols(["value_col"])\
    .setSelectedCols(["f0"])\
    .linkFrom(modelOp, inOp)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.HugeLookupBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HugeLookupBatchOpTest {
    @Test
    public void testHugeLookupBatchOp() throws Exception {
        List <Row> data_df = Arrays.asList(
            Row.of(10, 2.0),
            Row.of(1, 2.0),
            Row.of(-3, 2.0),
            Row.of(5, 1.0)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(data_df, "f0 int, f1
double");
        List <Row> model_df = Arrays.asList(
            Row.of(1, "value1"),
            Row.of(2, "value2"),
            Row.of(5, "value5")
        );
        BatchOperator <?> modelOp = new MemSourceBatchOp(model_df, "key_col
int, value_col string");
        new HugeLookupBatchOp()
            .setMapKeyCols("key_col")
            .setMapValueCols("value_col")
            .setSelectedCols("f0")
            .linkFrom(modelOp, inOp)
            .print();
    }
}
```

运行结果

f0	f1	value_col
10	2.0	null
1	2.0	value1
-3	2.0	null
5	1.0	value5

多列并行反ID化预测 (HugeMultiIndexerStringPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.HugeMultiIndexerStringPredictBatchOp

Python 类名：HugeMultiIndexerStringPredictBatchOp

功能介绍

提供ID转换为字符串的功能，与 HugeMultiStringIndexerPredictBatchOp 功能相反。

由 MultiStringIndexerTrainBatchOp 生成词典模型，将输入数据的ID转化成原文。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[LONG]	
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP"
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [1, "football", "apple"],
    [2, "football", "apple"],
    [3, "football", "apple"],
    [4, "basketball", "apple"],
    [5, "basketball", "apple"],
    [6, "tennis", "pair"],
    [7, "tennis", "pair"],
    [8, "pingpang", "banana"],
    [9, "pingpang", "banana"],
    [0, "baseball", "banana"]
])

data = BatchOperator.fromDataframe(df, schemaStr='id long, f0 string, f1
string')

stringindexer = MultiStringIndexerTrainBatchOp()\
    .setSelectedCols(["f0", "f1"])\
    .setStringOrderType("frequency_asc")

model = stringindexer.linkFrom(data)

predictor = HugeMultiStringIndexerPredictBatchOp()\
    .setSelectedCols(["f0", "f1"])
result = predictor.linkFrom(model, data)

stringPredictor = HugeMultiIndexerStringPredictBatchOp()\
    .setSelectedCols(["f0", "f1"])\
    .setOutputCols(["f0_source", "f1_source"])
stringPredictor.linkFrom(model, result).print();
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HugeMultiIndexerStringPredictBatchOpTest {
    @Test
```

```
    public void testHugeMultiStringIndexerPredict() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1L, "football", "apple"),
            Row.of(2L, "football", "apple"),
            Row.of(3L, "football", "apple"),
            Row.of(4L, "basketball", "apple"),
            Row.of(5L, "basketball", "apple"),
            Row.of(6L, "tennis", "pair"),
            Row.of(7L, "tennis", "pair"),
            Row.of(8L, "pingpang", "banana"),
            Row.of(9L, "pingpang", "banana"),
            Row.of(0L, "baseball", "banana")
        );
        // baseball 1
        // basketball,pair,tennis,pingpang 2
        // footbal,banana 3
        // apple 5
        BatchOperator <?> data = new MemSourceBatchOp(df, "id long,f0 string,f1
string");
        BatchOperator <?> stringindexer = new MultiStringIndexerTrainBatchOp()
            .setSelectedCols("f0", "f1")
            .setStringOrderType("frequency_asc");
        BatchOperator model = stringindexer.linkFrom(data);
        model.lazyPrint(10);

        BatchOperator <?> predictor = new
HugeMultiStringIndexerPredictBatchOp().setSelectedCols("f0", "f1");
        BatchOperator result = predictor.linkFrom(model, data);
        result.lazyPrint(10);

        BatchOperator <?> stringPredictor = new
HugeMultiIndexerStringPredictBatchOp().setSelectedCols("f0", "f1")
            .setOutputCols("f0_source", "f1_source");
        stringPredictor.linkFrom(model, result).print();
    }
}
```

运行结果

column_index	token	token_index
1	apple	2
1	pair	0
1	banana	1
-1	{"selectedCols":"[\"f0\",\"f1\"]","selectedColTypes":"[\"VARCHAR\",\"VARCHAR\"]"}	null

0	football	4
0	baseball	0
0	basketball	1
0	tennis	2
0	pingpang	3

id	f0	f1
1	4	2
2	4	2
6	2	0
7	2	0
5	1	2
3	4	2
9	3	1
0	0	1
4	1	2
8	3	1

id	f0	f1	f0_source	f1_source
5	1	2	basketball	apple
2	4	2	football	apple
6	2	0	tennis	pair
4	1	2	basketball	apple
8	3	1	pingpang	banana
3	4	2	football	apple
7	2	0	tennis	pair
9	3	1	pingpang	banana
0	0	1	baseball	banana
1	4	2	football	apple

HugeStringIndexer预测 (HugeMultiStringIndexerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.HugeMultiStringIndexerPredictBatchOp

Python 类名：HugeMultiStringIndexerPredictBatchOp

功能介绍

根据词典（由 MultiStringIndexerTrainBatchOp 组件生成）将字符串转换为ID，组件可同时处理多列数据。

由 MultiStringIndexerTrainBatchOp 生成词典模型，将输入数据的字符串转化成词典模型中的ID

对于词典模型中不存在的字符串，提供了三种处理策略，"keep"表示用最大id加1代替, "skip"表示补null， "error"表示抛异常

当词典的数据规模较大时，建议使用该组件。词典规模较小时，可以使用 MultiStringIndexerPredictBatchOp 组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP"
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1], ["b", 2], ["b", 3], ["c", 4]
])

op = BatchOperator.fromDataframe(df, schemaStr='f0 string, f1 int')

stringIndexer = MultiStringIndexerTrainBatchOp().setSelectedCols(["f1",
"f0"]).setStringOrderType("frequency_desc")
stringIndexer.linkFrom(op)

predictor =
HugeMultiStringIndexerPredictBatchOp().setSelectedCols(["f0"]).setReservedCols(
["f0", "f1"])\
    .setOutputCols(["f0_index"]).setHandleInvalid("skip");
predictor.linkFrom(stringIndexer, op).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.HugeMultiStringIndexerPredictBatchOp;
import
com.alibaba.alink.operator.batch.dataproc.MultiStringIndexerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HugeMultiStringIndexerPredictBatchOpTest {
    @Test
    public void testHugeMultiStringIndexerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1), Row.of("b", 2), Row.of("b", 3), Row.of("c", 4)
        );
        BatchOperator <?> op = new MemSourceBatchOp(df, "f0 string, f1 int");
        BatchOperator <?> stringIndexer = new
```

```
MultiStringIndexerTrainBatchOp().setSelectedCols("f1", "f0")
            .setStringOrderType("frequency_desc");
        stringIndexer.linkFrom(op);
        BatchOperator <?> predictor = new
HugeMultiStringIndexerPredictBatchOp().setSelectedCols("f0").setReservedCols(
            "f0", "f1")
            .setOutputCols("f0_index").setHandleInvalid("skip");
        predictor.linkFrom(stringIndexer, op).print();
    }
}
```

运行结果

f0	f1	f0_index
a	1	1
b	2	0
b	3	0
c	4	2

并行ID化预测 (HugeStringIndexerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.HugeStringIndexerPredictBatchOp

Python 类名：HugeStringIndexerPredictBatchOp

功能介绍

提供字符串ID化处理功能，与 StringIndexerPredictBatchOp 功能相同，是其升级版本，模型为分布式存储，提升了运行效率。支持多列同时转换。

由 StringIndexerTrainBatchOp 生成词典模型，将输入数据的字符串转化成词典模型中的ID

对于词典模型中不存在的字符串，提供了三种处理策略，"keep"表示用最大id加1代替, "skip"表示补null， "error"表示抛异常

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP"
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["football", "apple"],
    ["football", "apple"],
    ["football", "apple"],
    ["basketball", "apple"],
    ["basketball", "apple"],
    ["tennis", "pair"],
    ["tennis", "pair"],
    ["pingpang", "banana"],
    ["pingpang", "banana"],
    ["baseball", "banana"]
])

data = BatchOperator.fromDataframe(df, schemaStr='f0 string, f1 string')

stringindexer = StringIndexerTrainBatchOp()\
    .setSelectedCol("f0")\
    .setSelectedCols(["f1"])\
    .setStringOrderType("alphabet_asc")

model = stringindexer.linkFrom(data)

predictor = HugeStringIndexerPredictBatchOp()\
    .setSelectedCols(["f0", "f1"])\
    .setOutputCols(["f0_indexed", "f1_indexed"])

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HugeStringIndexerPredictBatchOpTest {
    @Test
```

```
    public void testStringIndexerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("football", "apple"),
            Row.of("football", "apple"),
            Row.of("football", "apple"),
            Row.of("basketball", "apple"),
            Row.of("basketball", "apple"),
            Row.of("tennis", "pair"),
            Row.of("tennis", "pair"),
            Row.of("pingpang", "banana"),
            Row.of("pingpang", "banana"),
            Row.of("baseball", "banana")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 string,f1
string");
        BatchOperator <?> stringindexer = new StringIndexerTrainBatchOp()
            .setSelectedCol("f0")
            .setSelectedCols("f1")
            .setStringOrderType("frequency_asc");
        BatchOperator <?> predictor = new
HugeStringIndexerPredictBatchOp().setSelectedCols("f0", "f1")
            .setOutputCols("f0_indexed", "f1_indexed");
        BatchOperator model = stringindexer.linkFrom(data);
        model.lazyPrint(10);
        BatchOperator result = predictor.linkFrom(model, data);
        result.print();
    }
}
```

运行结果

token	token_index
banana	5
football	6
basketball	1
pingpang	2
tennis	3
pair	4
baseball	0
apple	7

f0	f1	f0_indexed	f1_indexed

basketball	apple	1	7
pingpang	banana	2	5
football	apple	6	7
tennis	pair	3	4
tennis	pair	3	4
basketball	apple	1	7
football	apple	6	7
football	apple	6	7
pingpang	banana	2	5
baseball	banana	0	5

缺失值填充批预测 (ImputerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.ImputerPredictBatchOp

Python 类名：ImputerPredictBatchOp

功能介绍

数据缺失值填充处理，批式预测组件

运行时需要指定缺失值模型，由ImputerTrainBatchOp产生。缺失值填充的4种策略，即最大值、最小值、均值、指定数值，在生成缺失值模型时指定。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
```

```
            ["c", 100.9, 1],
            [None, None, None]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]

inOp = BatchOperator.fromDataframe(df_data, schemaStr='col1 string, col2
double, col3 double')

# train
trainOp = ImputerTrainBatchOp()\
           .setSelectedCols(selectedColNames)

model = trainOp.linkFrom(inOp)

# batch predict
predictOp = ImputerPredictBatchOp()
predictOp.linkFrom(model, inOp).print()

# stream predict
sinOp = StreamOperator.fromDataframe(df_data, schemaStr='col1 string, col2
double, col3 double')

predictStreamOp = ImputerPredictStreamOp(model)
predictStreamOp.linkFrom(sinOp).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ImputerPredictBatchOp;
import com.alibaba.alink.operator.batch.dataproc.ImputerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.dataproc.ImputerPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ImputerPredictBatchOpTest {
    @Test
```

```
    public void testImputerPredictBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1),
            Row.of(null, null, null)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df_data, "col1 string,
col2 double, col3 int");
        BatchOperator <?> trainOp = new ImputerTrainBatchOp()
            .setSelectedCols(selectedColNames);
        BatchOperator model = trainOp.linkFrom(inOp);
        BatchOperator <?> predictOp = new ImputerPredictBatchOp();
        predictOp.linkFrom(model, inOp).print();
        StreamOperator <?> sinOp = new MemSourceStreamOp(df_data, "col1 string,
col2 double, col3 int");
        StreamOperator <?> predictStreamOp = new ImputerPredictStreamOp(model);
        predictStreamOp.linkFrom(sinOp).print();
        StreamOperator.execute();
    }
}
```

运行结果

col1	col2	col3
a	10.000000	100
b	-2.500000	9
c	100.200000	1
d	-99.900000	100
a	1.400000	1
b	-2.200000	9
c	100.900000	1
null	15.414286	31

缺失值填充训练 (ImputerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.ImputerTrainBatchOp

Python 类名：ImputerTrainBatchOp

功能介绍

数据缺失值模型训练

缺失值填充支持4种策略，最大值、最小值、均值、指定数值。当策略为指定数值时，需要设置参数fillValue。

模型生成后处理其他数据参考ImputerPredictBatchOp

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
fillValue	填充缺失值	自定义的填充值。当strategy为value时，读取fillValue的值	String			null
strategy	缺失值填充规则	缺失值填充的规则，支持mean，max，min或者value。选择value时，需要读取fillValue的值	String		"MEAN", "MIN", "MAX", "VALUE"	"MEAN"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1],
            [None, None, None]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]

inOp = BatchOperator.fromDataframe(df_data, schemaStr='col1 string, col2
double, col3 double')

# train
trainOp = ImputerTrainBatchOp()\
           .setSelectedCols(selectedColNames)

model = trainOp.linkFrom(inOp)

# batch predict
predictOp = ImputerPredictBatchOp()
predictOp.linkFrom(model, inOp).print()

# stream predict
sinOp = StreamOperator.fromDataframe(df_data, schemaStr='col1 string, col2
double, col3 double')

predictStreamOp = ImputerPredictStreamOp(model)
predictStreamOp.linkFrom(sinOp).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
```

```
import com.alibaba.alink.operator.batch.dataproc.ImputerPredictBatchOp;
import com.alibaba.alink.operator.batch.dataproc.ImputerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.dataproc.ImputerPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ImputerTrainBatchOpTest {
    @Test
    public void testImputerTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1),
            Row.of(null, null, null)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df_data, "col1 string,
col2 double, col3 int");
        BatchOperator <?> trainOp = new ImputerTrainBatchOp()
            .setSelectedCols(selectedColNames);
        BatchOperator model = trainOp.linkFrom(inOp);
        BatchOperator <?> predictOp = new ImputerPredictBatchOp();
        predictOp.linkFrom(model, inOp).print();
        StreamOperator <?> sinOp = new MemSourceStreamOp(df_data, "col1 string,
col2 double, col3 int");
        StreamOperator <?> predictStreamOp = new ImputerPredictStreamOp(model);
        predictStreamOp.linkFrom(sinOp).print();
        StreamOperator.execute();
    }
}
```

运行结果

col1	col2	col3
a	10.000000	100
b	-2.500000	9

c	100.200000	1
d	-99.900000	100
a	1.400000	1
b	-2.200000	9
c	100.900000	1
null	15.414286	31

IndexToString预测 (IndexToStringPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.IndexToStringPredictBatchOp

Python 类名：IndexToStringPredictBatchOp

功能介绍

基于 StringIndexer 模型，将一列整数映射为字符串。

在批式预测中，IndexToStringPredictBatchOp 接收两个BatchOp的输入， 第一个输入为模型（StringIndexer的getModelData()获取，或者直接输入StringIndexerTrainBatchOp）， 第二个输入为要预测的数据。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelName	模型名字	模型名字	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df_data = pd.DataFrame([
    ["football"],
    ["football"],
    ["football"],
    ["basketball"],
    ["basketball"],
    ["tennis"],
])

train_data = BatchOperator.fromDataframe(df_data, schemaStr='f0 string')

stringIndexer = StringIndexer()\
    .setModelName("string_indexer_model")\
    .setSelectedCol("f0")\
    .setOutputCol("f0_indexed")\
    .setStringOrderType("frequency_asc").fit(train_data)

indexed = stringIndexer.transform(train_data)

indexToStrings = IndexToStringPredictBatchOp()\
    .setSelectedCol("f0_indexed")\
    .setOutputCol("f0_indxed_unindexed")

indexToStrings.linkFrom(stringIndexer.getModelData(), indexed).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.IndexToStringPredictBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.StringIndexer;
import com.alibaba.alink.pipeline.dataproc.StringIndexerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class IndexToStringPredictStreamOpTest {

    @Test
    public void testIndexToStringPredictStreamOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("football"),
            Row.of("football"),
            Row.of("football"),
```

```
            Row.of("basketball"),
            Row.of("basketball"),
            Row.of("tennis")
        );
        BatchOperator <?> train_data = new MemSourceBatchOp(df_data, "f0
string");
        StringIndexerModel stringIndexer = new StringIndexer()
            .setModelName("string_indexer_model")
            .setSelectedCol("f0")
            .setOutputCol("f0_indexed")
            .setStringOrderType("frequency_asc").fit(train_data);
        BatchOperator indexed = stringIndexer.transform(train_data);
        BatchOperator <?> indexToStrings = new IndexToStringPredictBatchOp()
            .setSelectedCol("f0_indexed")
            .setOutputCol("f0_indxed_unindexed");
        indexToStrings.linkFrom(stringIndexer.getModelData(), indexed).print();
    }
}
```

运行结果

f0	f0_indexed	f0_indxed_unindexed
football	2	football
football	2	football
football	2	football
basketball	1	basketball
basketball	1	basketball
tennis	0	tennis

JSON值抽取 (JsonValueBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.JsonValueBatchOp

Python 类名：JsonValueBatchOp

功能介绍

该组件完成json字符串中的信息抽取，按照用户给定的Path 抓取出相应的信息。该组件支持

- 按照多JsonPath规则编写的多条抽取规则。
- 指定输出列的数据类型

JsonPath表达式

	元素	含义	
	$	表示文档的根元素	
	@	表示文档的当前元素	
	.node_name 或 ['node_name']	匹配下级节点	
	[index]	选择数组中的元素	
	[start:end:step]	表示数组切片语法	

- 注意事项：jsonPath参数是字符串数组，长度应当和指定的输出列名称数目保持一致。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonPath	Json 路径数组	用来指定 Json 抽取的内容。	String[]	✓		
outputCols	输出结果列列名数组	输出结果列列名数组，必选	String[]	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
outputColTypes	输出结果列列类型数组	输出结果列类型数组	String[]			null

reservedCols	算法保留列名	算法保留列	String[]			null
skipFailed	是否跳过错误	当遇到抽取值为null 时是否跳过	boolean			false
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
     ["{a:boy,b:{b1:[1,2],b2:2}}"],
     ["{a:girl,b:{b1:[1,3],b2:2}}"]
 ])

data = BatchOperator.fromDataframe(df, schemaStr='str string')

JsonValueBatchOp()\
     .setJsonPath(["$.a", "$.b.b1[0]","$.b.b2"])\
     .setSelectedCol("str")\
     .setOutputCols(["f0","f1","f2"])\
     .linkFrom(data)\
     .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.JsonValueBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonValueStreamOpTest {
```

```
    @Test
    public void testJsonValueBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("{a:boy,b:{b1:[1,2],b2:2}}"),
            Row.of("{a:girl,b:{b1:[1,3],b2:2}}")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "str string");
        new JsonValueBatchOp()
            .setJsonPath("$.a", "$.b.b1[0]","$.b.b2")
            .setSelectedCol("str")
            .setOutputCols("f0", "f1","f2")
            .linkFrom(data)
            .print();
    }
}
```

运行结果

str	f0	f1	f2
{a:girl,b:{b1:[1,3],b2:2}}	girl	1	2
{a:boy,b:{b1:[1,2],b2:2}}	boy	1	2

表查找 (LookupBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.LookupBatchOp

Python 类名：LookupBatchOp

功能介绍

支持数据查找功能，支持多个key的查找，并将查找后的结果中的value列添加到待查询数据后面。与SQl语法中的inner join功能类似，当不存在重复的key时LookupBatchOp().setMapKeyCols("key_col_A").setMapValueCols("value_col").setSelectedCols("key_col_B").linkFrom(A, B)与 "SELECT A.value_col FROM A INNER JOIN B ON A.key_col_A = B.key_col_B"，但是需要注意：当数据B中存在多行相同的key时，只保留一个value，不会找到所有的value。

Table A

	key_col_A	value_col	
	Bob	98	
	Tom	72	

Table B

	key_col_B	age	
	Bob	11	
	Denny	10	

查找结果

	key_col_B	age	value_col
	Bob	11	98
	Denny	10	null

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		

mapKeyCols	Key列名	模型中对应的查找等值的列名	String[]			null
mapValueCols	Values列名	模型中需要拼接到样本中的列名	String[]			null
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamUpdateMethod	模型更新方法	模型更新方法，可选COMPLETE（全量更新）或者INCREMENT（增量更新）	String		"COMPLETE", "INCREMENT"	"CO

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["10", 2.0], ["1", 2.0], ["-3", 2.0], ["5", 1.0]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='f0 string, f1 double')

df2 = pd.DataFrame([
```

```
    ["1", "value1"], ["2", "value2"], ["5", "value5"]
])
modelOp = BatchOperator.fromDataframe(df2, schemaStr="key_col string, value_col
string")

LookupBatchOp().setMapKeyCols(["key_col"]).setMapValueCols(["value_col"]) \
    .setSelectedCols(["f0"]).linkFrom(modelOp, inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.LookupBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LookupBatchOpTest {
    @Test
    public void testLookupBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("10", 2.0), Row.of("1", 2.0), Row.of("-3", 2.0), Row.of("5",
1.0)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "f0 string, f1
double");
        List <Row> df2 = Arrays.asList(
            Row.of("1", "value1"), Row.of("2", "value2"), Row.of("5", "value5")
        );
        BatchOperator <?> modelOp = new MemSourceBatchOp(df2, "key_col string,
value_col string");
        new
LookupBatchOp().setMapKeyCols("key_col").setMapValueCols("value_col")
            .setSelectedCols("f0").linkFrom(modelOp, inOp).print();
    }
}
```

运行结果

f0	f1	value_col
10	2.0	null
1	2.0	value1

-3	2.0	null
5	1.0	value5

添加HBase数据 (LookupHBaseBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.LookupHBaseBatchOp

Python 类名：LookupHBaseBatchOp

功能介绍

LookupHBaseBatchOp ， 将HBase中的数据取出。 读HBase Plugin版。plugin版本为1.2.12。 读数据时，指定HBase的zookeeper地址，表名称，列簇名称。指定rowkey列（可以是多列）和要读取的数据列和格式（可以写多列）。 在使用时，需要先下载插件，详情请看https://www.yuque.com/pinshu/alink_guide/czg4cx

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
familyName	簇值	簇值	String	✓		
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
pluginVersion	插件版本号	插件版本号	String	✓		
rowKeyCols	rowkey所在列	rowkey所在列	String[]	✓		
tableName	HBase表名称	HBase表名称	String	✓		
zookeeperQuorum	Zookeeper quorum	Zookeeper quorum 地址	String	✓		
reservedCols	算法保留列名	算法保留列	String[]			null
timeout	HBase RPC 超时时间	HBase RPC 超时时间，单位毫秒	Integer			1000

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    ["1"],
    ["2"]
])

data = BatchOperator.fromDataframe(df, schemaStr='userid string')

lookupHBaseBatchOp = LookupHBaseBatchOp()\
    .setZookeeperQuorum("localhost:2181")\
    .setTableName("user")\
    .setRowKeyCols("userid")\
    .setFamilyName("color")\
    .setPluginVersion("1.2.12")\
    .setOutputSchemaStr("red long,black double,green int")
lookupHBaseBatchOp.linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.LookupHBaseBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HBaseTest {

    @Test
    public void testReadBatch() throws Exception {
        List <Row> datas = Arrays.asList(
            Row.of("1"),
            Row.of("2")
        );
        BatchOperator op = new MemSourceBatchOp(datas, "userid string");
        LookupHBaseBatchOp lookupHBaseBatchOp = new LookupHBaseBatchOp()
            .setZookeeperQuorum("localhost:2181")
            .setTableName("user")
            .setRowKeyCols("userid")
            .setFamilyName("color")
```

```
            .setPluginVersion("1.2.12")
            .setOutputSchemaStr("red long,black double,green int");
        lookupHBaseBatchOp.linkFrom(op).print();
    }
}
```

绝对值最大化批预测 (MaxAbsScalerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.MaxAbsScalerPredictBatchOp

Python 类名：MaxAbsScalerPredictBatchOp

功能介绍

- 绝对值最大标准化是对数据按照最大值和最小值进行标准化的组件, 将数据归一到-1和1之间。
- 需要读入MaxAbsScalerTrainBatchOp生成的模型

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])
```

```
colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]


inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')


# train
trainOp = MaxAbsScalerTrainBatchOp()\
           .setSelectedCols(selectedColNames)

trainOp.linkFrom(inOp)

# batch predict
predictOp = MaxAbsScalerPredictBatchOp()
predictOp.linkFrom(trainOp, inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.MaxAbsScalerPredictBatchOp;
import com.alibaba.alink.operator.batch.dataproc.MaxAbsScalerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MaxAbsScalerPredictBatchOpTest {
    @Test
    public void testMaxAbsScalerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
```

```
double, col3 int");
        BatchOperator <?> trainOp = new MaxAbsScalerTrainBatchOp()
            .setSelectedCols(selectedColNames);
        trainOp.linkFrom(inOp);
        BatchOperator <?> predictOp = new MaxAbsScalerPredictBatchOp();
        predictOp.linkFrom(trainOp, inOp).print();
    }
}
```

运行结果

col1	col2	col3
a	0.0991	1.0000
b	-0.0248	0.0900
c	0.9931	0.0100
d	-0.9901	1.0000
a	0.0139	0.0100
b	-0.0218	0.0900
c	1.0000	0.0100

绝对值最大化训练 (MaxAbsScalerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.MaxAbsScalerTrainBatchOp

Python 类名：MaxAbsScalerTrainBatchOp

功能介绍

- 绝对值最大标准化是对数据按照最大值和最小值进行标准化的组件, 将数据归一到-1和1之间。
- 使用绝对值最大标准化预测组件使用生成的模型，转换输入的数据

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])
```

```
colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]


inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')


# train
trainOp = MaxAbsScalerTrainBatchOp()\
           .setSelectedCols(selectedColNames)

trainOp.linkFrom(inOp)

# batch predict
predictOp = MaxAbsScalerPredictBatchOp()
predictOp.linkFrom(trainOp, inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.MaxAbsScalerPredictBatchOp;
import com.alibaba.alink.operator.batch.dataproc.MaxAbsScalerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MaxAbsScalerTrainBatchOpTest {
    @Test
    public void testMaxAbsScalerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
```

```
        BatchOperator <?> trainOp = new MaxAbsScalerTrainBatchOp()
            .setSelectedCols(selectedColNames);
        trainOp.linkFrom(inOp);
        BatchOperator <?> predictOp = new MaxAbsScalerPredictBatchOp();
        predictOp.linkFrom(trainOp, inOp).print();
    }
}
```

运行结果

col1	col2	col3
a	0.0991	1.0000
b	-0.0248	0.0900
c	0.9931	0.0100
d	-0.9901	1.0000
a	0.0139	0.0100
b	-0.0218	0.0900
c	1.0000	0.0100

归一化批预测 (MinMaxScalerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.MinMaxScalerPredictBatchOp

Python 类名：MinMaxScalerPredictBatchOp

功能介绍

数据归一化预测组件

将数据归一到minValue和maxValue之间，value最终结果为 (value - min) / (max - min) * (maxValue - minValue) + minValue，最终结果的范围为[minValue, maxValue]。

minValue和maxValue由用户指定，默认为0和1。

需要加载由MinMaxScalerTrainBatchOp训练的模型

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
```

```
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]


inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')


# train
trainOp = MinMaxScalerTrainBatchOp()\
          .setSelectedCols(selectedColNames)

trainOp.linkFrom(inOp)

# batch predict
predictOp = MinMaxScalerPredictBatchOp()
predictOp.linkFrom(trainOp, inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.MinMaxScalerPredictBatchOp;
import com.alibaba.alink.operator.batch.dataproc.MinMaxScalerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MinMaxScalerPredictBatchOpTest {
    @Test
    public void testMinMaxScalerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
```

```
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
        BatchOperator <?> trainOp = new MinMaxScalerTrainBatchOp()
            .setSelectedCols(selectedColNames);
        trainOp.linkFrom(inOp);
        BatchOperator <?> predictOp = new MinMaxScalerPredictBatchOp();
        predictOp.linkFrom(trainOp, inOp).print();
    }
}
```

运行结果

col1	col2	col3
a	0.5473	1.0000
b	0.4851	0.0808
c	0.9965	0.0000
d	0.0000	1.0000
a	0.5045	0.0000
b	0.4866	0.0808
c	1.0000	0.0000

归一化训练 (MinMaxScalerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.MinMaxScalerTrainBatchOp

Python 类名：MinMaxScalerTrainBatchOp

功能介绍

数据归一化组件

将数据归一到minValue和maxValue之间，value最终结果为 (value - min) / (max - min) * (maxValue - minValue) + minValue，最终结果的范围为[minValue, maxValue]。

minValue和maxValue由用户指定，默认为0和1。

生成的最大值最小值归一化模型在归一化预处理组件中使用。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
max	归一化的上界	归一化的上界	Double			1.0
min	归一化的下界	归一化的下界	Double			0.0

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]


inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')


# train
trainOp = MinMaxScalerTrainBatchOp()\
           .setSelectedCols(selectedColNames)

trainOp.linkFrom(inOp)

# batch predict
predictOp = MinMaxScalerPredictBatchOp()
predictOp.linkFrom(trainOp, inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.MinMaxScalerPredictBatchOp;
import com.alibaba.alink.operator.batch.dataproc.MinMaxScalerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
```

```
import java.util.List;

public class MinMaxScalerTrainBatchOpTest {
    @Test
    public void testMinMaxScalerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
        BatchOperator <?> trainOp = new MinMaxScalerTrainBatchOp()
            .setSelectedCols(selectedColNames);
        trainOp.linkFrom(inOp);
        BatchOperator <?> predictOp = new MinMaxScalerPredictBatchOp();
        predictOp.linkFrom(trainOp, inOp).print();
    }
}
```

运行结果

col1	col2	col3
a	0.5473	1.0000
b	0.4851	0.0808
c	0.9965	0.0000
d	0.0000	1.0000
a	0.5045	0.0000
b	0.4866	0.0808
c	1.0000	0.0000

MultiStringIndexer预测 (MultiStringIndexerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.MultiStringIndexerPredictBatchOp

Python 类名：MultiStringIndexerPredictBatchOp

功能介绍

多列字符串转换组件，输入的模型数据来自 MultiStringIndexerTrainBatchOp 组件的输出，训练的时候指定多个列，每个列单独编码。 这个组件为批式预测组件，预测时需要指定列名，列名必须与训练时列名相同。如果转换时指定了训练时不存在的列名，会报异常。 支持按照一定的次序编码。如随机、出现频次生序，出现频次降序、字符串生序、字符串降序5种方式。 设置 setStringOrderType 参数时分别对应 random frequency_asc frequency_desc alphabet_asc alphabet_desc。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP"
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["football", "apple"],
    ["football", "banana"],
    ["football", "banana"],
    ["basketball", "orange"],
    ["basketball", "grape"],
    ["tennis", "grape"],
])

data = BatchOperator.fromDataframe(df, schemaStr='f0 string,f1 string')

stringindexer = MultiStringIndexerTrainBatchOp() \
    .setSelectedCols(["f0", "f1"]) \
    .setStringOrderType("frequency_asc")

predictor = MultiStringIndexerPredictBatchOp().setSelectedCols(["f0",
"f1"]).setOutputCols(["f0_indexed", "f1_indexed"])

model = stringindexer.linkFrom(data)
predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.MultiStringIndexerPredictBatchOp;
import
com.alibaba.alink.operator.batch.dataproc.MultiStringIndexerTrainBatchOp;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MultiStringIndexerPredictBatchOpTest {
    @Test
    public void testMultiStringIndexerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("football", "apple"),
            Row.of("football", "banana"),
            Row.of("football", "banana"),
            Row.of("basketball", "orange"),
            Row.of("basketball", "grape"),
            Row.of("tennis", "grape")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 string,f1
string");
        BatchOperator <?> stringindexer = new MultiStringIndexerTrainBatchOp()
            .setSelectedCols("f0", "f1")
            .setStringOrderType("frequency_asc");
        BatchOperator <?> predictor = new
MultiStringIndexerPredictBatchOp().setSelectedCols("f0", "f1").setOutputCols(
            "f0_indexed", "f1_indexed");
        BatchOperator model = stringindexer.linkFrom(data);
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

f0	f1	f0_indexed	f1_indexed
basketball	orange	1	0
football	apple	2	1
tennis	grape	0	3
football	banana	2	2
basketball	grape	1	3
football	banana	2	2

MultiStringIndexer训练 (MultiStringIndexerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.MultiStringIndexerTrainBatchOp

Python 类名：MultiStringIndexerTrainBatchOp

功能介绍

MultiStringIndexer 训练组件的作用是训练一个模型用于将多列字符串映射为整数，训练的时候指定多个列，每个列单独编码。 支持按照一定的次序编码。如随机、出现频次生序，出现频次降序、字符串生序、字符串降序5种方式。 设置 setStringOrderType 参数时分别对应 random frequency_asc frequency_desc alphabet_asc alphabet_desc。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
stringOrderType	Token排序方法	Token排序方法	String		"RANDOM", "FREQUENCY_ASC", "FREQUENCY_DESC", "ALPHABET_ASC", "ALPHABET_DESC"	"RANDOM"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["football"],
```

```
    ["football"],
    ["football"],
    ["basketball"],
    ["basketball"],
    ["tennis"],
])


data = BatchOperator.fromDataframe(df, schemaStr='f0 string')

stringindexer = MultiStringIndexerTrainBatchOp() \
    .setSelectedCols(["f0"]) \
    .setStringOrderType("frequency_asc")

model = stringindexer.linkFrom(data)
model.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.MultiStringIndexerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MultiStringIndexerTrainBatchOpTest {
    @Test
    public void testMultiStringIndexerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("football"),
            Row.of("football"),
            Row.of("football"),
            Row.of("basketball"),
            Row.of("basketball"),
            Row.of("tennis")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 string");
        BatchOperator <?> stringindexer = new MultiStringIndexerTrainBatchOp()
            .setSelectedCols("f0")
            .setStringOrderType("frequency_asc");
        BatchOperator model = stringindexer.linkFrom(data);
        model.print();
```

```
    }
}
```

运行结果

column_index	token	token_index
-1	{"selectedCols":"[\"f0\"]","selectedColTypes":"[\"VARCHAR\"]"}	null
0	tennis	0
0	basketball	1
0	football	2

数据Rebalance (RebalanceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.RebalanceBatchOp

Python 类名：RebalanceBatchOp

功能介绍

该组件对数据进行 rebalance。

实现原理

将数据按轮转（round-robin）的方式划分分区，后续每个 worker 处理一个分区，各个分区间负载相等。 可以用于优化数据倾斜带来的性能问题。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
       ["0,0,0"],
       ["0.1,0.1,0.1"],
       ["0.2,0.2,0.2"],
       ["9,9,9"],
       ["9.1,9.1,9.1"],
       ["9.2,9.2,9.2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='Y string')

inOp.link(RebalanceBatchOp()).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.RebalanceBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RebalanceBatchOpTest {
    @Test
    public void testRebalanceBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("0,0,0"),
            Row.of("0.1,0.1,0.1"),
            Row.of("0.2,0.2,0.2"),
            Row.of("9,9,9"),
            Row.of("9.1,9.1,9.1"),
            Row.of("9.2,9.2,9.2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "Y string");
        inOp.link(new RebalanceBatchOp()).print();
    }
}
```

运行结果

Y
0,0,0
0.1,0.1,0.1
0.2,0.2,0.2
9,9,9
9.1,9.1,9.1
9.2,9.2,9.2

随机采样 (SampleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.SampleBatchOp

Python 类名：SampleBatchOp

功能介绍

本算子对数据进行随机抽样，每个样本都以相同的概率被抽到。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
ratio	采样比例	采样率，范围为[0, 1]	Double	✓	[0.0, 1.0]	
withReplacement	是否放回	是否有放回的采样，默认不放回	Boolean			false

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
       ["0,0,0"],
       ["0.1,0.1,0.1"],
       ["0.2,0.2,0.2"],
       ["9,9,9"],
       ["9.1,9.1,9.1"],
       ["9.2,9.2,9.2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='Y string')

sampleOp = SampleBatchOp()\
        .setRatio(0.3)\
```

```
        .setWithReplacement(False)

inOp.link(sampleOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.SampleBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SampleBatchOpTest {
    @Test
    public void testSampleBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("0,0,0"),
            Row.of("0.1,0.1,0.1"),
            Row.of("0.2,0.2,0.2"),
            Row.of("9,9,9"),
            Row.of("9.1,9.1,9.1"),
            Row.of("9.2,9.2,9.2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "Y string");
        BatchOperator <?> sampleOp = new SampleBatchOp()
            .setRatio(0.3)
            .setWithReplacement(false);
        inOp.link(sampleOp).print();
    }
}
```

运行结果

Y
0,0,0
0.2,0.2,0.2

固定条数随机采样 (SampleWithSizeBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.SampleWithSizeBatchOp

Python 类名：SampleWithSizeBatchOp

功能介绍

对数据按个数进行随机抽样，每个样本都以相同的概率被抽到。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
size	采样个数	采样个数	Integer	✓		
withReplacement	是否放回	是否有放回的采样，默认不放回	Boolean			false

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
  ["0,0,0"],
  ["0.1,0.1,0.1"],
  ["0.2,0.2,0.2"],
  ["9,9,9"],
  ["9.1,9.1,9.1"],
  ["9.2,9.2,9.2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='Y string')

sampleOp = SampleWithSizeBatchOp() \
  .setSize(2) \
```

```
  .setWithReplacement(False)

inOp.link(sampleOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.SampleWithSizeBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SampleWithSizeBatchOpTest {
    @Test
    public void testSampleWithSizeBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("0,0,0"),
            Row.of("0.1,0.1,0.1"),
            Row.of("0.2,0.2,0.2"),
            Row.of("9,9,9"),
            Row.of("9.1,9.1,9.1"),
            Row.of("9.2,9.2,9.2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "Y string");
        BatchOperator <?> sampleOp = new SampleWithSizeBatchOp()
            .setSize(2)
            .setWithReplacement(false);
        inOp.link(sampleOp).print();
    }
}
```

运行结果

Y
0,0,0
0.2,0.2,0.2

打乱数据顺序 (ShuffleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.ShuffleBatchOp

Python 类名：ShuffleBatchOp

功能介绍

将输入数据的顺序打乱。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
       ["0,0,0"],
       ["0.1,0.1,0.1"],
       ["0.2,0.2,0.2"],
       ["9,9,9"],
       ["9.1,9.1,9.1"],
       ["9.2,9.2,9.2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='Y string')

inOp.link(ShuffleBatchOp()).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
```

```
import com.alibaba.alink.operator.batch.dataproc.ShuffleBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ShuffleBatchOpTest {
    @Test
    public void testShuffleBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("0,0,0"),
            Row.of("0.1,0.1,0.1"),
            Row.of("0.2,0.2,0.2"),
            Row.of("9,9,9"),
            Row.of("9.1,9.1,9.1"),
            Row.of("9.2,9.2,9.2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "Y string");
        inOp.link(new ShuffleBatchOp()).print();
    }
}
```

运行结果

Y

0.2,0.2,0.2 9.2,9.2,9.2 9,9,9 9.1,9.1,9.1 0,0,0 0.1,0.1,0.1

单行拆分多行 (SplitArrayBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.SplitArrayBatchOp

Python 类名：SplitArrayBatchOp

功能介绍

列分割组件，将某一列的数据分割，输出多行数据。

例如，若输入数据为：

id	hosts
user_1	192.168.1.100,192.168.1.101
user_2	192.168.2.100

输入参数为：

- selectedColName = "hosts"
- keepColNames = "id"
- outputColName = "host"
- delimiter = ","

那么输出结果是：

id	host
user_1	192.168.1.100
user_1	192.168.1.101
user_2	192.168.2.100

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
delimiter	分隔符	分隔符，默认逗号	String			","
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null

reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ['Ohio', "2000,2001,2002"],
    ['Nevada', "2001,2002,2003"],
])


batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 string, f2
string')
op = SplitArrayBatchOp().setSelectedCol('f2')
batch_data = batch_data.link(op)

batch_data.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;

public class SplitPartBatchOpTest extends AlinkTestBase {

    @Test
    public void testSplitPart() throws Exception {
        Row[] rows = new Row[] {
            Row.of("Ohio", "2000,2001,2002"),
            Row.of("Nevada", "2001,2002,2003")
        };
        BatchOperator data = new MemSourceBatchOp(Arrays.asList(rows), "f1
```

```
string,f2 string");

        BatchOperator op = new SplitPartBatchOp()
            .setPart(2)
            .setSelectedCol("f2")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

f1	f2
Ohio	2000
Ohio	2001
Ohio	2002
Nevada	2001
Nevada	2002
Nevada	2003

数据拆分 (SplitBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.SplitBatchOp

Python 类名：SplitBatchOp

功能介绍

将输入数据按比例拆分为两部分。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
fraction	拆分到左端的数据比例	拆分到左端的数据比例	Double	✓	[0.0, 1.0]	
randomSeed	随机数种子	随机数种子	Integer			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([['Ohio', 2001, 1.7],
                        ['Ohio', 2002, 3.6],
                        ['Nevada', 2001, 2.4],
                        ['Nevada', 2002, 2.9]])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 string, f2
bigint, f3 double')

spliter = SplitBatchOp().setFraction(0.5)
spliter.linkFrom(batch_data)
spliter.lazyPrint(-1)
spliter.getSideOutput(0).lazyPrint(-1)

BatchOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.SplitBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SplitBatchOpTest {
    @Test
    public void testSplitBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9)
           );
        BatchOperator <?> batch_data = new MemSourceBatchOp(df_data, "f1
string, f2 int, f3 double");
        BatchOperator <?> spliter = new SplitBatchOp().setFraction(0.5);
        spliter.linkFrom(batch_data);
        spliter.lazyPrint(-1);
        spliter.getSideOutput(0).lazyPrint(-1);
        BatchOperator.execute();
    }
}
```

运行结果

f1	f2	f3
Ohio	2001	1.7000
Nevada	2002	2.9000

f1	f2	f3
Ohio	2002	3.6000
Nevada	2001	2.4000

单行拆分部分 (SplitPartBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.SplitPartBatchOp

Python 类名：SplitPartBatchOp

功能介绍

列分割组件，将某一列的数据分割，选取其中的部分进行输出。

例如，若输入数据为：

id	hosts
user_1	192.168.1.100,192.168.1.101
user_2	192.168.2.100
user_3	192.168.3.100, 192.168.3.101, 192.168.3.102

输入参数为：

- selectedColName = "hosts"
- keepColNames = "id"
- outputColName = "host"
- part = 1

那么输出结果是：

id	hosts
user_1	192.168.1.101
user_2	
user_3	192.168.3.101

默认分隔符为逗号，可自定义分隔符。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
part	输出部分序号	输出部分的序号，从0开始	Integer	✓		

selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
delimiter	分隔符	分隔符，默认逗号	String			","
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df_data = pd.DataFrame(    [ ['Ohio', "2000,2001,2002"],
                             ['Nevada', "2001,2002,2003"]])


batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 string, f2
string')
op = SplitPartBatchOp().setSelectedCol('f2').setPart(2)
batch_data = batch_data.link(op)

batch_data.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;

public class SplitPartBatchOpTest extends AlinkTestBase {

    @Test
    public void testSplitPart() throws Exception {
        Row[] rows = new Row[] {
```

```
            Row.of("Ohio", "2000,2001,2002"),
            Row.of("Nevada", "2001,2002,2003")
        };
        BatchOperator data = new MemSourceBatchOp(Arrays.asList(rows), "f1
string,f2 string");

        BatchOperator op = new SplitPartBatchOp()
            .setPart(2)
            .setSelectedCol("f2")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

f1	f2
Ohio	2002
Nevada	2003

标准化批预测 (StandardScalerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.StandardScalerPredictBatchOp

Python 类名：StandardScalerPredictBatchOp

功能介绍

标准化是对数据进行按正态化处理的组件

使用标准化训练组件训练的模型，对数据做标准化处理

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
```

```
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]


inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')


# train
trainOp = StandardScalerTrainBatchOp()\
          .setSelectedCols(selectedColNames)

trainOp.linkFrom(inOp)

# batch predict
predictOp = StandardScalerPredictBatchOp()
predictOp.linkFrom(trainOp, inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.StandardScalerPredictBatchOp;
import com.alibaba.alink.operator.batch.dataproc.StandardScalerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StandardScalerPredictBatchOpTest {
    @Test
    public void testStandardScalerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
```

```
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
        BatchOperator <?> trainOp = new StandardScalerTrainBatchOp()
            .setSelectedCols(selectedColNames);
        trainOp.linkFrom(inOp);
        BatchOperator <?> predictOp = new StandardScalerPredictBatchOp();
        predictOp.linkFrom(trainOp, inOp).print();
    }
}
```

运行结果

col1	col2	col3
a	-0.0784	1.4596
b	-0.2592	-0.4814
c	1.2270	-0.6521
d	-1.6687	1.4596
a	-0.2028	-0.6521
b	-0.2549	-0.4814
c	1.2371	-0.6521

标准化训练 (StandardScalerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.StandardScalerTrainBatchOp

Python 类名：StandardScalerTrainBatchOp

功能介绍

标准化是对数据进行按正态化处理的组件

训练过程计算数据的均值和标准差，在预测组件中使用模型结果

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
withMean	是否使用均值	是否使用均值，默认使用	Boolean			true
withStd	是否使用标准差	是否使用标准差，默认使用	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])
```

```
colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]


inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')


# train
trainOp = StandardScalerTrainBatchOp()\
           .setSelectedCols(selectedColNames)

trainOp.linkFrom(inOp)

# batch predict
predictOp = StandardScalerPredictBatchOp()
predictOp.linkFrom(trainOp, inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.StandardScalerPredictBatchOp;
import com.alibaba.alink.operator.batch.dataproc.StandardScalerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StandardScalerTrainBatchOpTest {
    @Test
    public void testStandardScalerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
```

```
        BatchOperator <?> trainOp = new StandardScalerTrainBatchOp()
            .setSelectedCols(selectedColNames);
        trainOp.linkFrom(inOp);
        BatchOperator <?> predictOp = new StandardScalerPredictBatchOp();
        predictOp.linkFrom(trainOp, inOp).print();
    }
}
```

运行结果

col1	col2	col3
a	-0.0784	1.4596
b	-0.2592	-0.4814
c	1.2270	-0.6521
d	-1.6687	1.4596
a	-0.2028	-0.6521
b	-0.2549	-0.4814
c	1.2371	-0.6521

分层随机采样 (StratifiedSampleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.StratifiedSampleBatchOp

Python 类名：StratifiedSampleBatchOp

功能介绍

分层采样组件。给定输入数据，本算法根据用户指定的不同类别的采样比例进行随机采样。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
strataCol	分层列	分层列	String	✓		
strataRatios	采用比率	采用比率, eg, a:0.1,b:0.3	String	✓		
strataRatio	采用比率	采用比率	Double			-1.0
withReplacement	是否放回	是否有放回的采样，默认不放回	Boolean			false

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ['a',0.0,0.0],
        ['a',0.2,0.1],
        ['b',0.2,0.8],
        ['b',9.5,9.7],
        ['b',9.1,9.6],
        ['b',9.3,9.9]
    ])
```

```
batchData = BatchOperator.fromDataframe(df, schemaStr='x1 string, x2 double, x3
double')
sampleOp = StratifiedSampleBatchOp()\
       .setStrataCol("x1")\
       .setStrataRatios("a:0.5,b:0.5")

batchData.link(sampleOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.StratifiedSampleBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StratifiedSampleBatchOpTest {
    @Test
    public void testStratifiedSampleBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 0.0, 0.0),
            Row.of("a", 0.2, 0.1),
            Row.of("b", 0.2, 0.8),
            Row.of("b", 9.5, 9.7),
            Row.of("b", 9.1, 9.6),
            Row.of("b", 9.3, 9.9)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "x1 string, x2
double, x3 double");
        BatchOperator <?> sampleOp = new StratifiedSampleBatchOp()
            .setStrataCol("x1")
            .setStrataRatios("a:0.5,b:0.5");
        batchData.link(sampleOp).print();
    }
}
```

运行结果

x1	x2	x3
a	0.0000	0.0000

b	9.5000	9.7000
b	9.1000	9.6000
b	9.3000	9.9000

固定条数分层随机采样 (StratifiedSampleWithSizeBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.StratifiedSampleWithSizeBatchOp

Python 类名：StratifiedSampleWithSizeBatchOp

功能介绍

固定条数分层随机采样组件。给定输入数据，本算法根据用户指定的不同类别的采样个数进行随机采样。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
strataCol	分层列	分层列	String	✓		
strataSizes	采样个数	采样个数, eg, a:10,b:30	String	✓		
strataSize	采样个数	采样个数	Integer			-1
withReplacement	是否放回	是否有放回的采样，默认不放回	Boolean			false

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ['a',0.0,0.0],
        ['a',0.2,0.1],
        ['b',0.2,0.8],
        ['b',9.5,9.7],
        ['b',9.1,9.6],
```

```
        ['b',9.3,9.9]
    ])


batchData = BatchOperator.fromDataframe(df, schemaStr='x1 string, x2 double, x3
double')
sampleOp = StratifiedSampleWithSizeBatchOp() \
       .setStrataCol("x1") \
       .setStrataSizes("a:1,b:2")

batchData.link(sampleOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.StratifiedSampleWithSizeBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StratifiedSampleWithSizeBatchOpTest {
    @Test
    public void testStratifiedSampleWithSizeBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 0.0, 0.0),
            Row.of("a", 0.2, 0.1),
            Row.of("b", 0.2, 0.8),
            Row.of("b", 9.5, 9.7),
            Row.of("b", 9.1, 9.6),
            Row.of("b", 9.3, 9.9)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "x1 string, x2
double, x3 double");
        BatchOperator <?> sampleOp = new StratifiedSampleWithSizeBatchOp()
            .setStrataCol("x1")
            .setStrataSizes("a:1,b:2");
        batchData.link(sampleOp).print();
    }
}
```

运行结果

x1	x2	x3
a	0.0000	0.0000
b	9.1000	9.6000
b	0.2000	0.8000

StringIndexer预测 (StringIndexerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.StringIndexerPredictBatchOp

Python 类名：StringIndexerPredictBatchOp

功能介绍

基于StringIndexer模型，将一列字符串映射为整数。该组件为批式组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[INTEGER, LONG, STRING]	
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null， "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP"
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["football"],
    ["football"],
    ["football"],
    ["basketball"],
    ["basketball"],
    ["tennis"],
])


data = BatchOperator.fromDataframe(df, schemaStr='f0 string')

stringindexer = StringIndexerTrainBatchOp() \
    .setSelectedCol("f0") \
    .setStringOrderType("frequency_asc")

predictor =
StringIndexerPredictBatchOp().setSelectedCol("f0").setOutputCol("f0_indexed")

model = stringindexer.linkFrom(data)
predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.StringIndexerPredictBatchOp;
import com.alibaba.alink.operator.batch.dataproc.StringIndexerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StringIndexerPredictBatchOpTest {
    @Test
```

```
    public void testStringIndexerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("football"),
            Row.of("football"),
            Row.of("football"),
            Row.of("basketball"),
            Row.of("basketball"),
            Row.of("tennis")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 string");
        BatchOperator <?> stringindexer = new StringIndexerTrainBatchOp()
            .setSelectedCol("f0")
            .setStringOrderType("frequency_asc");
        BatchOperator <?> predictor = new
StringIndexerPredictBatchOp().setSelectedCol("f0").setOutputCol(
            "f0_indexed");
        BatchOperator model = stringindexer.linkFrom(data);
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

f0	f0_indexed
football	2
football	2
football	2
basketball	1
basketball	1
tennis	0

StringIndexer训练 (StringIndexerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.StringIndexerTrainBatchOp

Python 类名：StringIndexerTrainBatchOp

功能介绍

StringIndexer训练组件的作用是训练一个模型用于将单列字符串映射为整数。

如将一列映射为整数，需指定 setSelectedCol 设定。

同时，该组件支持输入多列，生成一个映射词典，通过 setSelectedCols 设定其他需要补充的列名。

特征的排列顺序支持 random，frequency_asc，frequency_desc，alphabet_asc，alphabet_desc 五种排序方法。

注意：输入多列时，所有列必须为相同格式。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[INTEGER, LONG, STRING]	
modelName	模型名字	模型名字	String			
selectedCols	选中的列名数组	计算列对应的列名列表	String[]		所选列类型为[INTEGER, LONG, STRING]	null
stringOrderType	Token排序方法	Token排序方法	String		"RANDOM", "FREQUENCY_ASC", "FREQUENCY_DESC", "ALPHABET_ASC", "ALPHABET_DESC"	"RANDOM"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["football", "apple"],
    ["football", "apple"],
    ["football", "apple"],
    ["basketball", "apple"],
    ["basketball", "apple"],
    ["tennis", "pair"],
    ["tennis", "pair"],
    ["pingpang", "banana"],
    ["pingpang", "banana"],
    ["baseball", "banana"]
])


data = BatchOperator.fromDataframe(df, schemaStr='f0 string,f1 string')

stringindexer = StringIndexerTrainBatchOp() \
    .setSelectedCol("f0") \
    .setSelectedCols(["f1"]) \
    .setStringOrderType("alphabet_asc")

model = stringindexer.linkFrom(data)
model.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.StringIndexerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StringIndexerTrainBatchOpTest {
    @Test
    public void testAlphabetAsc() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("football", "apple"),
```

```
            Row.of("football", "apple"),
            Row.of("football", "apple"),
            Row.of("basketball", "apple"),
            Row.of("basketball", "apple"),
            Row.of("tennis", "pair"),
            Row.of("tennis", "pair"),
            Row.of("pingpang", "banana"),
            Row.of("pingpang", "banana"),
            Row.of("baseball", "banana")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 string,f1
string");
        BatchOperator <?> stringindexer = new StringIndexerTrainBatchOp()
            .setSelectedCol("f0")
            .setSelectedCols("f1")
            .setStringOrderType("alphabet_asc");
        BatchOperator model = stringindexer.linkFrom(data);
        model.print();
    }
}
```

运行结果

模型表:

token	token_index
pingpang	6
banana	1
baseball	2
basketball	3
pair	5
apple	0
football	4
tennis	7

张量转向量 (TensorToVectorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.TensorToVectorBatchOp

Python 类名：TensorToVectorBatchOp

功能介绍

转换张量类型为向量类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DOUBLE_TENSOR, FLOAT_TENSOR, INT_TENSOR, LONG_TENSOR]	
convertMethod	转换方法	张量转换为向量的方法，可取flatten, sum, mean, max, min.	String		"FLATTEN", "SUM", "MEAN", "MAX", "MIN"	"FLATTEN"
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ['DOUBLE#6#0.0 0.1 1.0 1.1 2.0 2.1']
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr = 'tensor string')

batch_data.link(TensorToVectorBatchOp().setSelectedCol("tensor")).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Collections;
import java.util.List;

public class TensorToVectorBatchOpTest {

    @Test
    public void testTensorToVectorBatchOp() throws Exception {
        List <Row> data = Collections.singletonList(Row.of("DOUBLE#6#0.0 0.1
1.0 1.1 2.0 2.1"));

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "tensor
```

```
string");

        memSourceBatchOp
            .link(
                new TensorToVectorBatchOp()
                    .setSelectedCol("tensor")
            )
            .print();
    }
}
```

运行结果

tensor
0.0 0.1 1.0 1.1 2.0 2.1

转MTable (ToMTableBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.ToMTableBatchOp

Python 类名：ToMTableBatchOp

功能介绍

将输入列转换为MTable类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
outputCol	输出结果列	输出结果列列名，可选，默认 null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
```

```
    ['{"data":{"col0":[1],"col1":["2"],"label":[0]},"schema":"col0 INT, col1
VARCHAR,label INT"}']
])

data = BatchOperator.fromDataframe(df_data, schemaStr = 'vec string')

ToMTableBatchOp().setSelectedCol("vec").linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

public class ToVectorDemoTest extends AlinkTestBase {
    @Test
    public void test() throws Exception {
        final String mTableStr = "{\"data\":{\"col0\":[1],\"col1\":
[\"2\"],\"label\":[0]},\"schema\":\"col0 INT, col1 VARCHAR,label INT\"}";

        Row[] rows = new Row[] {
            Row.of(mTableStr)
        };

        MemSourceBatchOp data = new MemSourceBatchOp(
            rows, new String[] {"m_table"}
        );
        new ToMTableBatchOp().setSelectedCol("vec").linkFrom(data).print();
    }
}
```

运行结果

vec
{"data":{"col0":[1],"col1":["2"],"label":[0]},"schema":"col0 INT,col1 VARCHAR,label INT"}

转Tensor (ToTensorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.ToTensorBatchOp

Python 类名：ToTensorBatchOp

功能介绍

将指定列转为 Alink 的张量类型。

如果指定列为 String 类型，并且值为 Alink 张量或者向量 toString 的结果，那么张量类型和形状将自动获取。 否则的话，需要指定张量类型和张量形状。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
tensorDataType	要转换的张量数据类型	要转换的张量数据类型。	String		"FLOAT", "DOUBLE", "INT", "LONG", "BOOLEAN", "BYTE", "UBYTE", "STRING"	

tensorShape	张量形状	张量的形状，数组类型。	Long[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame(["FLOAT#6#0.0 0.1 1.0 1.1 2.0 2.1 "])
source = BatchOperator.fromDataframe(df, schemaStr='vec string')

source.link(
    ToTensorBatchOp()
        .setSelectedCol("vec")
        .setTensorShape([2, 3])
        .setTensorDataType("float")
).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.dataproc.ToTensorBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

public class ToTensorTest {

    @Test
    public void testToTensorBatchOp() throws Exception {
        Row[] rows = new Row[] {
            Row.of("FLOAT#6#0.0 0.1 1.0 1.1 2.0 2.1 ")
        };
        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(rows, new
```

```
String[] {"vec"});

        memSourceBatchOp.link(
            new ToTensorBatchOp()
                .setSelectedCol("vec")
                .setTensorShape(2, 3)
                .setTensorDataType("float")
        ).print();
    }
}
```

运行结果

vec
FLOAT#2,3#0.0 0.1 1.0 1.1 2.0 2.1

转向量 (ToVectorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.ToVectorBatchOp

Python 类名：ToVectorBatchOp

功能介绍

将输入列转换为向量类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
vectorType	要转换的Vector类型。	要转换的Vector类型。	String		"DENSE", "SPARSE"	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df_data = pd.DataFrame([
    ['1 0 3 4']
])

data = BatchOperator.fromDataframe(df_data, schemaStr = 'vec string')

ToVectorBatchOp().setSelectedCol("vec").setVectorType("SPARSE").linkFrom(data).
print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.common.linalg.VectorType;
import com.alibaba.alink.operator.batch.dataproc.ToVectorBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

public class ToVectorTest extends AlinkTestBase {
    @Test
    public void test() throws Exception {
        final String vecStr = "1 0 3 4";
        Row[] rows = new Row[] {
            Row.of(vecStr)
        };
        MemSourceBatchOp data = new MemSourceBatchOp(
            rows, new String[] {"vec"}
        );
        new ToVectorBatchOp()
            .setSelectedCol("vec")
            .setVectorType(VectorType.SPARSE)
            .linkFrom(data)
            .print();
    }
}
```

运行结果

vec
$4$0:1.0 2:3.0 3:4.0

类型转换 (TypeConvertBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.TypeConvertBatchOp

Python 类名：TypeConvertBatchOp

功能介绍

类型转换是用来列类型进行转换的组件

组件可一次性转化多个列到指定的数据类型，但是这些列的数据类型只能为同一种，并且为JDBC Type。

支持的目标类型为 STRING, VARCHAR, FLOAT, DOUBLE, INT, BIGINT, LONG, BOOLEAN。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
targetType	目标类型	转换为的类型，类型应该为JDBC Type。	String	✓	"STRING", "VARCHAR", "FLOAT", "DOUBLE", "INT", "BIGINT", "LONG", "BOOLEAN"	
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([['Ohio', 2000, 1.5],
                        ['Ohio', 2001, 1.7],
                        ['Ohio', 2002, 3.6],
                        ['Nevada', 2001, 2.4],
                        ['Nevada', 2002, 2.9],
```

```
                          ['Nevada', 2003, 3.2],])


batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 string, f2
bigint, f3 double')
op =
TypeConvertBatchOp().setSelectedCols(['f2']).setTargetType('double').linkFrom(b
atch_data)
op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;

public class TypeConvertBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
        Row[] testArray = new Row[] {
            Row.of("Ohio", 2000L, 1.5),
            Row.of("Ohio", 2001L, 1.7),
            Row.of("Ohio", 2002L, 3.6),
            Row.of("Nevada", 2001L, 2.4),
            Row.of("Nevada", 2002L, 2.9),
            Row.of("Nevada", 2003L, 3.2)
        };

        MemSourceBatchOp inOp = new MemSourceBatchOp(Arrays.asList(testArray),
"f1 string, f2 bigint, f3 double");
        TypeConvertBatchOp op = new TypeConvertBatchOp()
            .setSelectedCols("f2")
            .setTargetType("double")
            .linkFrom(inOp);
        op.print();
    }
}
```

运行结果

f1	f2	f3

Ohio	2000.0000	1.5000
Ohio	2001.0000	1.7000
Ohio	2002.0000	3.6000
Nevada	2001.0000	2.4000
Nevada	2002.0000	2.9000
Nevada	2003.0000	3.2000

(UrlDecodeBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.UrlDecodeBatchOp

Python 类名：UrlDecodeBatchOp

功能介绍

对字符串列进行 url decode，将一些特殊字符和汉字进行解码。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[STRING]	
charset	字符集	字符集	String			"UTF-8"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['http%3A%2F%2Fwww.sina.com.cn'],
    ['http%3A%2F%2Fwww.abc.com']]
)
batch_data = BatchOperator.fromDataframe(df, schemaStr='f1 string',
op_type='batch')
op = UrlDecodeBatchOp().setSelectedCols(["f1"])
batch_data = batch_data.link(op)

batch_data.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;

public class UrlDecodeBatchOpTest extends AlinkTestBase {

    @Test
    public void testUrlDecodeDecode() throws Exception {
        Row[] rows = new Row[] {
            Row.of("http%3A%2F%2Fwww.sina.com.cn"),
            Row.of("http%3A%2F%2Fwww.abc.com")
        };

        BatchOperator data = new MemSourceBatchOp(Arrays.asList(rows), "f1
string");
        BatchOperator op = new
UrlDecodeBatchOp().setSelectedCols("f1").linkFrom(data);
        op.print();
    }
}
```

运行结果

f1
http://www.sina.com.cn
http://www.abc.com

(UrlEncodeBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.UrlEncodeBatchOp

Python 类名：UrlEncodeBatchOp

功能介绍

对字符串列进行 url encode，将一些特殊字符和汉字变为Encode编码。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[STRING]	
charset	字符集	字符集	String			"UTF-8"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['http://www.sina.com.cn'],
    ['http://www.abc.com']]
)
batch_data = BatchOperator.fromDataframe(df, schemaStr='f1 string',
op_type='batch')
op = UrlEncodeBatchOp().setSelectedCols(["f1"])
batch_data = batch_data.link(op)

batch_data.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class UrlEncodeBatchOpTest extends AlinkTestBase {

    @Test
    public void testUrlEncodeDecode() throws Exception {
        List <Row> rows = Arrays.asList(
            Row.of("http://www.sina.com.cn"),
            Row.of("http://www.abc.com")
        );

        BatchOperator data = new MemSourceBatchOp(rows, "f1 string");
        BatchOperator encoder = new
UrlEncodeBatchOp().setSelectedCols("f1").linkFrom(data);
        encoder.print();
    }
}
```

运行结果

f1
http%3A%2F%2Fwww.sina.com.cn
http%3A%2F%2Fwww.abc.com

向量转张量 (VectorToTensorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.VectorToTensorBatchOp

Python 类名：VectorToTensorBatchOp

功能介绍

转换向量类型为张量类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取error, skip	String		"ERROR", "SKIP"	"ERROR"
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null

tensorDataType	要转换的张量数据类型	要转换的张量数据类型。	String		"FLOAT", "DOUBLE", "INT", "LONG", "BOOLEAN", "BYTE", "UBYTE", "STRING"	
tensorShape	张量形状	张量的形状，数组类型。	Long[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ['0.0 0.1 1.0 1.1 2.0 2.1']
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr = 'vec string')

batch_data.link(VectorToTensorBatchOp().setSelectedCol("vec")).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Collections;
import java.util.List;

public class VectorToTensorBatchOpTest {

    @Test
    public void testVectorToTensorStreamOp() throws Exception {
        List <Row> data = Collections.singletonList(Row.of("0.0 0.1 1.0 1.1 2.0
2.1"));

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "vec
string");

        memSourceBatchOp
            .link(
                new VectorToTensorBatchOp()
                    .setSelectedCol("vec")
            )
            .print();
    }
}
```

运行结果

vec
DOUBLE#6#0.0 0.1 1.0 1.1 2.0 2.1

Velocity变量生成 (VelocityVariableBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.VelocityVariableBatchOp

Python 类名：VelocityVariableBatchOp

功能介绍

Velocity变量生成，又称作统计变量生成，又称作滑动窗口提特征，用于生成指定窗口长度的统计量作为feature。

参数说明

名称	中文名称	描述
objectCol	查询主体列	查询主体列
statCols	统计值列名列表	客体列列表(用于统计的列)
statNames	统计值列名列表	统计值列名列表，和客体列列表一一对应
statTypes	统计类型列表	统计类型列表，和客体列一一对应，包含count,sum,mean,max,min,stddev,stderror,var,sum2,sum3,sum4,missingCou
timeCol	时间戳列(TimeStamp)	时间戳列(TimeStamp)
timeInterval	时间间隔	时间间隔，单位秒
groupCol	分组单列名	分组单列名，可选

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

colNames = ["group", "object", "time", "f0", "f1"]
```

```
df = pd.DataFrame([
                ["a", "o1", 1527156582000, 1.2, -1.0],
                ["b", "o1", 1527156583000, 1.5, 1.3],
                ["a", "o1", 1527156584000, 1.8, 3.4],
                ["b", "o1", 1527156585000, 1.7, 4.6],
                ["a", "o1", 1527156586000, 1.9, 3.5],
                ["b", "o1", 1527156587000, 1.8, 3.7],
                ["a", "o1", 1527156588000, 1.8, 8.6],
                ["b", "o1", 1527156589000, 2.1, 9.4],
                ["a", "o1", 1527156590000, 2.4, 7.8],
                ["b", "o1", 1527156591000, 2.5, 1.4]
        ])



batchData = BatchOperator.fromDataframe(df, schemaStr='group string, object
string, time long, f0 double, f1 double')



groupColName = "group" #累计主体列(分组列)
objectColName = "object" #查询主体列(查询列)
timeColName = "time" #时间列
timeInterval = 3  #时间窗口3s

#计算的是sum(f0) as stat0, count(f0) as stat1, mean(f0) as stat2
statTypes = ["sum", "count", "mean"] #统计指标类型
statColNames = ["f0", "f0", "f0"] #统计列类型
statNames = ["stat0", "stat1", "stat2"] #统计指标名称

velocity = VelocityVariableBatchOp()\
                .setGroupCol(groupColName)\
                .setObjectCol(objectColName)\
                .setStatTypes(statTypes)\
                .setStatCols(statColNames)\
                .setStatNames(statNames)\
                .setTimeCol(timeColName)\
                .setTimeInterval(timeInterval)

batchData.link(velocity).print();
```

运行结果

group	object	time	f0	f1	stat0	stat1	stat2
a	o1	1527156582000	1.2	-1.0	0	0	0
a	o1	1527156584000	1.8	3.4	1.2	1.0	1.2

a	o1	1527156586000	1.9	3.5	1.8	1.0	1.8
a	o1	1527156588000	1.8	8.6	1.9	1.0	1.9
a	o1	1527156590000	2.4	7.8	1.8	1.0	1.8
b	o1	1527156583000	1.5	1.3	0	0	0
b	o1	1527156585000	1.7	4.6	1.5	1.0	1.5
b	o1	1527156587000	1.8	3.7	1.7	1.0	1.7
b	o1	1527156589000	2.1	9.4	1.8	1.0	1.8
b	o1	1527156591000	2.5	1.4	2.1	1.0	2.1

加权采样 (WeightSampleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.WeightSampleBatchOp

Python 类名：WeightSampleBatchOp

功能介绍

本算子是按照数据点的权重对数据按照比例进行加权采样，权重越大的数据点被采样的可能性越大。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
ratio	采样比例	采样率，范围为[0, 1]	Double	✓	[0.0, 1.0]	
weightCol	权重列名	权重列对应的列名	String	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
withReplacement	是否放回	是否有放回的采样，默认不放回	Boolean			false

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1.3, 1.1],
    ["b", 2.5, 0.9],
```

```
    ["c", 100.2, -0.01],
    ["d", 99.9, 100.9],
    ["e", 1.4, 1.1],
    ["f", 2.2, 0.9],
    ["g", 100.9, -0.01],
    ["j", 99.5, 100.9],
])



# batch source
inOp = BatchOperator.fromDataframe(df, schemaStr='id string, weight double,
value double')
sampleOp = WeightSampleBatchOp() \
  .setWeightCol("weight") \
  .setRatio(0.5) \
  .setWithReplacement(False)

inOp.link(sampleOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.WeightSampleBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class WeightSampleBatchOpTest {
    @Test
    public void testWeightSampleBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1.3, 1.1),
            Row.of("b", 2.5, 0.9),
            Row.of("c", 100.2, -0.01),
            Row.of("d", 99.9, 100.9),
            Row.of("e", 1.4, 1.1),
            Row.of("f", 2.2, 0.9),
            Row.of("g", 100.9, -0.01),
            Row.of("j", 99.5, 100.9)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id string, weight
double, value double");
```

```
        BatchOperator <?> sampleOp = new WeightSampleBatchOp()
            .setWeightCol("weight")
            .setRatio(0.5)
            .setWithReplacement(false);
        inOp.link(sampleOp).print();
    }
}
```

结果

id	weight	value
g	100.9000	-0.0100
d	99.9000	100.9000
c	100.2000	-0.0100
j	99.5000	100.9000

列数据转CSV (ColumnsToCsvBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.ColumnsToCsvBatchOp

Python 类名：ColumnsToCsvBatchOp

功能介绍

将数据格式从 Columns 转成 Csv

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = ColumnsToCsvBatchOp()\
    .setSelectedCols(["f0", "f1"])\
    .setReservedCols(["row"])\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.ColumnsToCsvBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ColumnsToCsvBatchOpTest {
    @Test
    public void testColumnsToCsvBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
```

```
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new ColumnsToCsvBatchOp()
            .setSelectedCols("f0", "f1")
            .setReservedCols("row")
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	csv
1	1.0,2.0
2	4.0,8.0

列数据转JSON (ColumnsToJsonBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.ColumnsToJsonBatchOp

Python 类名：ColumnsToJsonBatchOp

功能介绍

将数据格式从 Columns 转成 Json

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])
```

```
data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = ColumnsToJsonBatchOp()\
    .setSelectedCols(["f0", "f1"])\
    .setReservedCols(["row"])\
    .setJsonCol("json")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.ColumnsToJsonBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ColumnsToJsonBatchOpTest {
    @Test
    public void testColumnsToJsonBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new ColumnsToJsonBatchOp()
            .setSelectedCols("f0", "f1")
            .setReservedCols("row")
            .setJsonCol("json")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	json
1	{"f0":"1.0","f1":"2.0"}
2	{"f0":"4.0","f1":"8.0"}

列数据转KV (ColumnsToKvBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.ColumnsToKvBatchOp

Python 类名：ColumnsToKvBatchOp

功能介绍

将数据格式从 Columns 转成 Key-value格式

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
kvCol	KV列名	KV列的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = ColumnsToKvBatchOp()\
    .setSelectedCols(["f0", "f1"])\
    .setReservedCols(["row"])\
    .setKvCol("kv")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.ColumnsToKvBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ColumnsToKvBatchOpTest {
    @Test
    public void testColumnsToKvBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
```

```
        BatchOperator <?> op = new ColumnsToKvBatchOp()
            .setSelectedCols("f0", "f1")
            .setReservedCols("row")
            .setKvCol("kv")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	kv
1	f0:1.0,f1:2.0
2	f0:4.0,f1:8.0

列数据转三元组 (ColumnsToTripleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.ColumnsToTripleBatchOp

Python 类名：ColumnsToTripleBatchOp

功能介绍

将数据格式从 Columns 转成 Triple三元组数据

注意一条输入数据可能会产生多条输出数据。按照属性名称和属性值展开。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
tripleColumnValueSchemaStr	三元组结构中列信息和数据信息的Schema	三元组结构中列信息和数据信息的Schema	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			[]
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = ColumnsToTripleBatchOp()\
    .setSelectedCols(["f0", "f1"])\
    .setReservedCols(["row"])\
    .setTripleColumnValueSchemaStr("col string, val double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.ColumnsToTripleBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ColumnsToTripleBatchOpTest {
    @Test
    public void testColumnsToTripleBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
```

```
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new ColumnsToTripleBatchOp()
            .setSelectedCols("f0", "f1")
            .setReservedCols("row")
            .setTripleColumnValueSchemaStr("col string, val double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	col	val
1	f0	1.0
1	f1	2.0
2	f0	4.0
2	f1	8.0

列数据转向量 (ColumnsToVectorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.ColumnsToVectorBatchOp

Python 类名：ColumnsToVectorBatchOp

功能介绍

将数据格式从 Columns 转成 Vector 数据格式可以为数值类型，如int，float，long，double，也可以为能够转换为数值类型的字符串。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
vectorCol	向量列名	向量列对应的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
vectorSize	向量长度	向量长度	Long			-1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = ColumnsToVectorBatchOp()\
    .setSelectedCols(["f0", "f1"])\
    .setReservedCols(["row"])\
    .setVectorCol("vec")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.ColumnsToVectorBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ColumnsToVectorBatchOpTest {
    @Test
    public void testColumnsToVectorBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new ColumnsToVectorBatchOp()
            .setSelectedCols("f0", "f1")
            .setReservedCols("row")
            .setVectorCol("vec")
```

```
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	vec
1	1.0 2.0
1	4.0 8.0

CSV转列数据 (CsvToColumnsBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.CsvToColumnsBatchOp

Python 类名：CsvToColumnsBatchOp

功能介绍

将数据格式从 Csv 转成 Columns

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓	所选列类型为[STRING]	
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = CsvToColumnsBatchOp()\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .setReservedCols(["row"])\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.CsvToColumnsBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CsvToColumnsBatchOpTest {
    @Test
    public void testCsvToColumnsBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
```

```
        BatchOperator <?> op = new CsvToColumnsBatchOp()
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .setReservedCols("row")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	f0	f1
1	1.0	2.0
2	4.0	8.0

CSV转JSON (CsvToJsonBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.CsvToJsonBatchOp

Python 类名：CsvToJsonBatchOp

功能介绍

将数据格式从 Csv 转成 Json

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓	所选列类型为[STRING]	
jsonCol	JSON列名	JSON列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = CsvToJsonBatchOp()\
    .setCsvCol("csv").setSchemaStr("f0 double, f1 double")\
    .setReservedCols(["row"]).setJsonCol("json")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.CsvToJsonBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CsvToJsonBatchOpTest {
    @Test
    public void testCsvToJsonBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new CsvToJsonBatchOp()
            .setCsvCol("csv").setSchemaStr("f0 double, f1 double")
            .setReservedCols("row").setJsonCol("json")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	json
1	{"f1":"1.0","f2":"2.0"}
2	{"f2":"4.0","f4":"8.0"}

CSV转KV (CsvToKvBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.CsvToKvBatchOp

Python 类名：CsvToKvBatchOp

功能介绍

将数据格式从 Csv 转成 Key-value格式

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓	所选列类型为[STRING]	
kvCol	KV列名	KV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"

quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = CsvToKvBatchOp()\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .setReservedCols(["row"])\
    .setKvCol("kv")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.CsvToKvBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class CsvToKvBatchOpTest {
    @Test
    public void testCsvToKvBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new CsvToKvBatchOp()
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .setReservedCols("row")
            .setKvCol("kv")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	kv
1	f0:1.0,f1:2.0
2	f0:4.0,f1:8.0

CSV转三元组 (CsvToTripleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.CsvToTripleBatchOp

Python 类名：CsvToTripleBatchOp

功能介绍

将数据格式从 Csv 转成 Triple三元组

一条数据数据可能会产生多条三元组数据

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认
csvCol	CSV列名	CSV列的列名	String	✓	所选列类型为[STRING]	
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
tripleColumnValueSchemaStr	三元组结构中列信息和数据信息的Schema	三元组结构中列信息和数据信息的Schema	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","

handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERR
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			[]

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = CsvToTripleBatchOp()\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .setReservedCols(["row"])\
    .setTripleColumnValueSchemaStr("col string, val double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.CsvToTripleBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CsvToTripleBatchOpTest {
    @Test
    public void testCsvToTripleBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new CsvToTripleBatchOp()
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .setReservedCols("row")
            .setTripleColumnValueSchemaStr("col string, val double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	col	val
1	f0	1.0
1	f1	2.0
2	f0	4.0
2	f1	8.0

CSV转向量 (CsvToVectorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.CsvToVectorBatchOp

Python 类名：CsvToVectorBatchOp

功能介绍

将数据格式从 Csv 转成 Vector

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓	所选列类型为[STRING]	
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

vectorSize	向量长度	向量长度	Long			-1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = CsvToVectorBatchOp()\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .setReservedCols(["row"])\
    .setVectorCol("vec")\
    .setVectorSize(5)\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.CsvToVectorBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CsvToVectorBatchOpTest {
    @Test
```

```
    public void testCsvToVectorBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new CsvToVectorBatchOp()
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .setReservedCols("row")
            .setVectorCol("vec")
            .setVectorSize(5)
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	vec
1	$5$1.0 2.0
2	$5$4.0 8.0

JSON转列数据 (JsonToColumnsBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.JsonToColumnsBatchOp

Python 类名：JsonToColumnsBatchOp

功能介绍

将数据格式从 Json 转成 Columns

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓	所选列类型为[STRING]	
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
```

```
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = JsonToColumnsBatchOp()\
    .setJsonCol("json")\
    .setReservedCols(["row"])\
    .setSchemaStr("f0 double, f1 double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.JsonToColumnsBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonToColumnsBatchOpTest {
    @Test
    public void testJsonToColumnsBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new JsonToColumnsBatchOp()
            .setJsonCol("json")
            .setReservedCols("row")
            .setSchemaStr("f0 double, f1 double")
            .linkFrom(data);
        op.print();
```

```
    }
}
```

运行结果

row	f0	f1
1	1.0	2.0
2	4.0	8.0

JSON转CSV (JsonToCsvBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.JsonToCsvBatchOp

Python 类名：JsonToCsvBatchOp

功能介绍

将数据格式从 Json 转成 Csv，通过 setSchemaStr 设置csv每列对应的key。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
jsonCol	JSON列名	JSON列的列名	String	✓	所选列类型为[STRING]	
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = JsonToCsvBatchOp()\
    .setJsonCol("json")\
    .setReservedCols(["row"])\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.JsonToCsvBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonToCsvBatchOpTest {
    @Test
    public void testJsonToCsvBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
```

```
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new JsonToCsvBatchOp()
            .setJsonCol("json")
            .setReservedCols("row")
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	csv
1	1.0,2.0
2	4.0,8.0

JSON转KV (JsonToKvBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.JsonToKvBatchOp

Python 类名：JsonToKvBatchOp

功能介绍

将数据格式从 Json 转成 Key-value

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓	所选列类型为[STRING]	
kvCol	KV列名	KV列的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = JsonToKvBatchOp()\
    .setJsonCol("json")\
    .setReservedCols(["row"])\
    .setKvCol("kv")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.JsonToKvBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonToKvBatchOpTest {
    @Test
    public void testJsonToKvBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
```

```
        BatchOperator <?> op = new JsonToKvBatchOp()
            .setJsonCol("json")
            .setReservedCols("row")
            .setKvCol("kv")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	kv
1	f0:1.0,f1:2.0
2	f0:4.0,f1:8.0

JSON转三元组 (JsonToTripleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.JsonToTripleBatchOp

Python 类名：JsonToTripleBatchOp

功能介绍

将数据格式从 Json 转成 Triple，Json中的key赋值为一列，value赋值为一列。

setTripleColumnValueSchemaStr 分别设置为key值和value值的列名和列类型。一条输入数据可能产生多条输出。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓	所选列类型为[STRING]	
tripleColumnValueSchemaStr	三元组结构中列信息和数据信息的Schema	三元组结构中列信息和数据信息的Schema	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			[]

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = JsonToTripleBatchOp()\
    .setJsonCol("json")\
    .setReservedCols(["row"])\
    .setTripleColumnValueSchemaStr("col string, val double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.JsonToTripleBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonToTripleBatchOpTest {
    @Test
    public void testJsonToTripleBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
```

```
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new JsonToTripleBatchOp()
            .setJsonCol("json")
            .setReservedCols("row")
            .setTripleColumnValueSchemaStr("col string, val double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	col	val
1	f0	1.0
1	f1	2.0
2	f0	4.0
2	f1	8.0

JSON转向量 (JsonToVectorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.JsonToVectorBatchOp

Python 类名：JsonToVectorBatchOp

功能介绍

将数据格式从 Json 转成 Vector

要转换为向量，要求Json的key必须可以转换为int，value必须可以转换为数字。否则会报异常。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓	所选列类型为[STRING]	
vectorCol	向量列名	向量列对应的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null
vectorSize	向量长度	向量长度	Long			-1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
```

```
    ['1', '{"0":1.0,"1":2.0}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0', '1.0,2.0',
1.0, 2.0],
    ['2', '{"0":4.0,"1":8.0}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0', '4.0,8.0',
4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = JsonToVectorBatchOp()\
    .setJsonCol("json")\
    .setReservedCols(["row"])\
    .setVectorCol("vec")\
    .setVectorSize(5)\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.JsonToVectorBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonToVectorBatchOpTest {
    @Test
    public void testJsonToVectorBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"0\":\"1.0\",\"1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"0\":\"4.0\",\"1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new JsonToVectorBatchOp()
            .setJsonCol("json")
            .setReservedCols("row")
            .setVectorCol("vec")
            .setVectorSize(5)
            .linkFrom(data);
```

```
        op.print();
    }
}
```

运行结果

row	vec
1	$5$1.0 2.0
2	$5$4.0 8.0

KV转列数据 (KvToColumnsBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.KvToColumnsBatchOp

Python 类名：KvToColumnsBatchOp

功能介绍

将数据格式从 Kv 转成 Columns，将KV转换成不同的列。setSchemaStr 设置列名和数据类型，列名需要与KV中的Key保持一致。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
kvCol	KV列名	KV列的列名	String	✓	所选列类型为[STRING]	
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = KvToColumnsBatchOp()\
    .setKvCol("kv")\
    .setReservedCols(["row"])\
    .setSchemaStr("f0 double, f1 double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.KvToColumnsBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KvToColumnsBatchOpTest {
    @Test
    public void testKvToColumnsBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
```

```
        BatchOperator <?> op = new KvToColumnsBatchOp()
            .setKvCol("kv")
            .setReservedCols("row")
            .setSchemaStr("f0 double, f1 double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	f0	f1
1	1.0	2.0
2	4.0	8.0

KV转CSV (KvToCsvBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.KvToCsvBatchOp

Python 类名：KvToCsvBatchOp

功能介绍

将数据格式从 Key-value 转成 Csv setSchemaStr 设置csv每一列的key值。如果指定的key不在输入的KV中，留空。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
kvCol	KV列名	KV列的列名	String	✓	所选列类型为[STRING]	
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","

kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = KvToCsvBatchOp()\
    .setKvCol("kv")\
    .setReservedCols(["row"])\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.KvToCsvBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;
```

```
import java.util.Arrays;
import java.util.List;

public class KvToCsvBatchOpTest {
    @Test
    public void testKvToCsvBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new KvToCsvBatchOp()
            .setKvCol("kv")
            .setReservedCols("row")
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	csv
1	1.0,2.0
2	4.0,8.0

KV转JSON (KvToJsonBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.KvToJsonBatchOp

Python 类名：KvToJsonBatchOp

功能介绍

将数据格式从 Key-value 转成 Json

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓		
kvCol	KV列名	KV列的列名	String	✓	所选列类型为[STRING]	
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = KvToJsonBatchOp()\
    .setKvCol("kv")\
    .setReservedCols(["row"])\
    .setJsonCol("json")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.KvToJsonBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KvToJsonBatchOpTest {
    @Test
    public void testKvToJsonBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
```

```
        BatchOperator <?> op = new KvToJsonBatchOp()
            .setKvCol("kv")
            .setReservedCols("row")
            .setJsonCol("json")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	json
1	{"f0":"1.0","f1":"2.0"}
2	{"f0":"4.0","f1":"8.0"}

KV转三元组 (KvToTripleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.KvToTripleBatchOp

Python 类名：KvToTripleBatchOp

功能介绍

将数据格式从 Kv 转成 Triple，三元组。一条数据可能输出多条结果。 setTripleColumnValueSchemaStr 设置三元组格式，包含两列，一列为key的值，一般类型为string，如果key为数字字符串，也可以指定为int、long等类型。一列为value的值，如果指定为数字或者字符串格式。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
kvCol	KV列名	KV列的列名	String	✓	所选列类型为[STRING]	
tripleColumnValueSchemaStr	三元组结构中列信息和数据信息的Schema	三元组结构中列信息和数据信息的Schema	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"

kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			[]

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = KvToTripleBatchOp()\
    .setKvCol("kv")\
    .setReservedCols(["row"])\
    .setTripleColumnValueSchemaStr("col string, val double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.KvToTripleBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KvToTripleBatchOpTest {
    @Test
    public void testKvToTripleBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"4.0\",\"f1\":\"8.0\"}", "$3$0:4.0 1:8.0",
"f0:4.0,f1:8.0", "4.0,8.0", 4.0, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new KvToTripleBatchOp()
            .setKvCol("kv")
            .setReservedCols("row")
            .setTripleColumnValueSchemaStr("col string, val double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	col	val
1	f0	1.0
1	f1	2.0
2	f0	4.0
2	f1	8.0

KV转向量 (KvToVectorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.KvToVectorBatchOp

Python 类名：KvToVectorBatchOp

功能介绍

将数据格式从 Key-value 转成 Vector Kv中的key和value必须为数字，否则会出错报异常。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
kvCol	KV列名	KV列的列名	String	✓	所选列类型为[STRING]	
vectorCol	向量列名	向量列对应的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null
vectorSize	向量长度	向量长度	Long			-1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', '0:1.0,1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', '0:4.0,1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = KvToVectorBatchOp()\
    .setKvCol("kv")\
    .setReservedCols(["row"])\
    .setVectorCol("vec")\
    .setVectorSize(5)\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.KvToVectorBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KvToVectorBatchOpTest {
    @Test
    public void testKvToVectorBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"0:1.0,1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new KvToVectorBatchOp()
```

```
            .setKvCol("kv")
            .setReservedCols("row")
            .setVectorCol("vec")
            .setVectorSize(5)
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	vec
1	$5$1.0 2.0
2	$5$4.0 8.0

三元组转列数据 (TripleToColumnsBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.TripleToColumnsBatchOp

Python 类名：TripleToColumnsBatchOp

功能介绍

将数据格式从 Triple 转成 Columns

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
tripleColumnCol	三元组结构中列信息的列名	三元组结构中列信息的列名	String	✓	所选列类型为[STRING]	
tripleValueCol	三元组结构中数据信息的列名	三元组结构中数据信息的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
tripleRowCol	三元组结构中行信息的列名	三元组结构中行信息的列名	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1,'f1',1.0],
    [1,'f2',2.0],
    [2,'f1',4.0],
    [2,'f2',8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row double, col string, val
double")


op = TripleToColumnsBatchOp()\
    .setTripleRowCol("row")\
    .setTripleColumnCol("col")\
    .setTripleValueCol("val")\
    .setSchemaStr("f1 double, f2 double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.TripleToColumnsBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TripleToColumnsBatchOpTest {
    @Test
    public void testTripleToColumnsBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, "f1", 1.0),
            Row.of(1, "f2", 2.0),
            Row.of(2, "f1", 4.0)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df, "row int, col string,
val double");
        BatchOperator <?> op = new TripleToColumnsBatchOp()
            .setTripleRowCol("row")
            .setTripleColumnCol("col")
            .setTripleValueCol("val")
            .setSchemaStr("f1 double, f2 double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	f1	f2
1	1.0	2.0
2	4.0	8.0

三元组转CSV (TripleToCsvBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.TripleToCsvBatchOp

Python 类名：TripleToCsvBatchOp

功能介绍

将数据格式从 Triple 转成 Csv

三元组转换为key value对，setTripleRowCol 设置数据行信息的列名，这一列值相同的数据，会被合并成一个数据 setTripleColumnCol 设置为Column名称所在列名 setTripleValueCol 设置为Column值所在列名 setSchemaStr 设置输出的csv的Schema格式

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
tripleColumnCol	三元组结构中列信息的列名	三元组结构中列信息的列名	String	✓	所选列类型为[STRING]	
tripleValueCol	三元组结构中数据信息的列名	三元组结构中数据信息的列名	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","

handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
tripleRowCol	三元组结构中行信息的列名	三元组结构中行信息的列名	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1,'f1',1.0],
    [1,'f2',2.0],
    [2,'f1',4.0],
    [2,'f2',8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row double, col string, val
double")

op = TripleToCsvBatchOp()\
    .setTripleRowCol("row")\
    .setTripleColumnCol("col")\
    .setTripleValueCol("val")\
    .setCsvCol("csv")\
    .setSchemaStr("f1 string, f2 string")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.TripleToCsvBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TripleToCsvBatchOpTest {
    @Test
    public void testTripleToCsvBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, "f1", 1.0),
            Row.of(1, "f2", 2.0),
            Row.of(2, "f1", 4.0),
            Row.of(2, "f2", 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "row int, col string,
val double");
        BatchOperator <?> op = new TripleToCsvBatchOp()
            .setTripleRowCol("row")
            .setTripleColumnCol("col")
            .setTripleValueCol("val")
            .setCsvCol("csv")
            .setSchemaStr("f1 string, f2 string")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	csv
1	1.0,2.0
2	4.0,8.0

三元组转JSON (TripleToJsonBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.TripleToJsonBatchOp

Python 类名：TripleToJsonBatchOp

功能介绍

将数据格式从 Triple 转成 Json

三元组转换为json，setTripleRowCol 设置数据行信息的列名，这一列值相同的数据，会被合并成一个数据 setTripleColumnCol 设置为json中key所在的列名 setTripleValueCol 设置为json中value所在的列名

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓		
tripleColumnCol	三元组结构中列信息的列名	三元组结构中列信息的列名	String	✓		
tripleValueCol	三元组结构中数据信息的列名	三元组结构中数据信息的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
tripleRowCol	三元组结构中行信息的列名	三元组结构中行信息的列名	String			null

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1,'f1',1.0],
    [1,'f2',2.0],
    [2,'f1',4.0],
    [2,'f2',8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row double, col string, val
double")

op = TripleToJsonBatchOp()\
    .setTripleRowCol("row")\
    .setTripleColumnCol("col")\
    .setTripleValueCol("val")\
    .setJsonCol("json")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.TripleToJsonBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TripleToJsonBatchOpTest {
    @Test
    public void testTripleToJsonBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, "f1", 1.0),
            Row.of(1, "f2", 2.0),
            Row.of(2, "f1", 4.0),
            Row.of(2, "f2", 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "row int, col string,
val double");
        BatchOperator <?> op = new TripleToJsonBatchOp()
            .setTripleRowCol("row")
```

```
            .setTripleColumnCol("col")
            .setTripleValueCol("val")
            .setJsonCol("json")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	json
1	{"f1":"1.0","f2":"2.0"}
2	{"f1":"4.0","f2":"8.0"}

三元组转KV (TripleToKvBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.TripleToKvBatchOp

Python 类名：TripleToKvBatchOp

功能介绍

将数据格式从 Triple 转成 Key-value

三元组转换为key value对，setTripleRowCol 设置数据行信息的列名，这一列值相同的数据，会被合并成一个数据 setTripleColumnCol 设置为key名称的列名 setTripleValueCol 设置为value所在的列名

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
kvCol	KV列名	KV列的列名	String	✓		
tripleColumnCol	三元组结构中列信息的列名	三元组结构中列信息的列名	String	✓		
tripleValueCol	三元组结构中数据信息的列名	三元组结构中数据信息的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"

tripleRowCol	三元组结构中行信息的列名	三元组结构中行信息的列名	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1,'f1',1.0],
    [1,'f2',2.0],
    [2,'f1',4.0],
    [2,'f2',8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row double, col string, val
double")

op = TripleToKvBatchOp()\
    .setTripleRowCol("row")\
    .setTripleColumnCol("col")\
    .setTripleValueCol("val")\
    .setKvCol("kv")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.TripleToKvBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TripleToKvBatchOpTest {
```

```
    @Test
    public void testTripleToKvBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, "f1", 1.0),
            Row.of(1, "f2", 2.0),
            Row.of(2, "f1", 4.0),
            Row.of(2, "f2", 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "row int, col string,
val double");
        BatchOperator <?> op = new TripleToKvBatchOp()
            .setTripleRowCol("row")
            .setTripleColumnCol("col")
            .setTripleValueCol("val")
            .setKvCol("kv")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	kv
1	f1:1.0,f2:2.0
2	f1:4.0,f2:8.0

三元组转向量 (TripleToVectorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.TripleToVectorBatchOp

Python 类名：TripleToVectorBatchOp

功能介绍

将数据格式从 Triple 转成 Vector

三元组转换为向量，setTripleRowCol 设置数据行信息的列名，这一列值相同的数据，会被合并成一个向量。
setTripleColumnCol 设置向量的索引所在列，数值为向量的索引，因此需要为整型，int或者long。
setTripleValueCol 设置向量的值所在列，因此该列必须为数值类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
tripleColumnCol	三元组结构中列信息的列名	三元组结构中列信息的列名	String	✓	所选列类型为[INTEGER, LONG]	
tripleValueCol	三元组结构中数据信息的列名	三元组结构中数据信息的列名	String	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
vectorCol	向量列名	向量列对应的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"

tripleRowCol	三元组结构中行信息的列名	三元组结构中行信息的列名	String			null
vectorSize	向量长度	向量长度	Long			-1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1,1,1.0],
    [1,2,2.0],
    [2,1,4.0],
    [2,2,8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row double, col string, val
double")

op = TripleToVectorBatchOp()\
    .setTripleRowCol("row")\
    .setTripleColumnCol("col")\
    .setTripleValueCol("val")\
    .setVectorCol("vec")\
    .setVectorSize(5)\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.TripleToVectorBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
```

```
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TripleToVectorBatchOpTest {
    @Test
    public void testTripleToVectorBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, 1, 1.0),
            Row.of(1, 2, 2.0),
            Row.of(2, 1, 4.0),
            Row.of(2, 2, 8.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "row int, col int,
val double");
        BatchOperator <?> op = new TripleToVectorBatchOp()
            .setTripleRowCol("row")
            .setTripleColumnCol("col")
            .setTripleValueCol("val")
            .setVectorCol("vec")
            .setVectorSize(5)
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	vec
1	$5$1:1.0 2:2.0
2	$5$1:4.0 2:8.0

向量转列数据 (VectorToColumnsBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.VectorToColumnsBatchOp

Python 类名：VectorToColumnsBatchOp

功能介绍

将数据格式从 Vector 转成 Columns，vector中的数据拆分成多列

一条输入对应一条输出结果，输入的vector可以为稀疏格式，也可以为稠密格式。 vector的数据维度不需要保持一致。

setReservedCols设置保留的输入列。 设置SchemaStr时需要注意，字段个数必须小于等于vector列的最小维度。字段名称和类型用空格分隔，类型必须为double。 当vector维度大于SchemaStr中的字段个数时，输入vector中后面的维度会被忽略。如果vector在前面维度没有值，默认输出为0.0。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"

reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$4$0:1.0 3:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 2:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = VectorToColumnsBatchOp()\
    .setVectorCol("vec")\
    .setReservedCols(["row"])\
    .setSchemaStr("f0 double, f1 double, f2 double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.VectorToColumnsBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorToColumnsBatchOpTest {
    @Test
    public void testVectorToColumnsBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
```

```
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$4$0:1.0 3:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:4.0 2:8.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new VectorToColumnsBatchOp()
            .setVectorCol("vec")
            .setReservedCols("row")
            .setSchemaStr("f0 double, f1 double, f2 double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	f0	f1	f2
1	1.0000	0.0000	0.0000
2	4.0000	0.0000	8.0000

向量转CSV (VectorToCsvBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.VectorToCsvBatchOp

Python 类名：VectorToCsvBatchOp

功能介绍

将数据格式从 Vector 转成 Csv

一条输入对应一条输出结果，输入的vector可以为稀疏格式，也可以为稠密格式。 vector的数据维度不需要保持一致。

setCsvCol设置csv输出列名，setReservedCols设置保留的输入列。 设置SchemaStr时需要注意，字段个数必须小于等于vector列的最小维度。类型为string或者double。 当vector维度大于SchemaStr中的字段个数时，输入vector中后面的维度会被忽略。如果vector在前面维度没有值，输出中也会为空，对应位置会保留。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
csvFieldDelimiter	字段分隔符	字段分隔符	String			","

handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$4$0:1.0 3:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 2:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = VectorToCsvBatchOp()\
    .setVectorCol("vec")\
    .setReservedCols(["row"])\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.VectorToCsvBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorToCsvBatchOpTest {
    @Test
    public void testVectorToCsvBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$4$0:1.0 3:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:4.0 2:8.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new VectorToCsvBatchOp()
            .setVectorCol("vec")
            .setReservedCols("row")
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	csv
2	4.0,,8.0
1	1.0,,

向量转JSON (VectorToJsonBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.VectorToJsonBatchOp

Python 类名：VectorToJsonBatchOp

功能介绍

将数据格式从 Vector 转成 Json

一条输入数据对应一条输出数据，setJsonCol设置json输出列名，setReservedCols设置保留的输入列。 在输出json列中，vector的下标生成json中的key，vector的值生成json中的value。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = VectorToJsonBatchOp()\
    .setVectorCol("vec")\
    .setReservedCols(["row"])\
    .setJsonCol("json")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.VectorToJsonBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorToJsonBatchOpTest {
    @Test
    public void testVectorToJsonBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:4.0 1:8.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new VectorToJsonBatchOp()
            .setVectorCol("vec")
            .setReservedCols("row")
```

```
            .setJsonCol("json")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	json
1	{"0":"1.0","1":"2.0"}
2	{"0":"4.0","1":"8.0"}

向量转KV (VectorToKvBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.VectorToKvBatchOp

Python 类名：VectorToKvBatchOp

功能介绍

将数据格式从 Vector 转成 Key-value，批式组件

一条输入数据对应一条输出数据，setKvCol设置kv输出列名，setReservedCols设置保留的输入列。 在输出kv列中，vector的下标对应key，vector的值对应的value。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
kvCol	KV列名	KV列的列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"

reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = VectorToKvBatchOp()\
    .setVectorCol("vec")\
    .setReservedCols(["row"])\
    .setKvCol("kv")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.VectorToKvBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class VectorToKvBatchOpTest {
    @Test
    public void testVectorToKvBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:4.0 1:8.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new VectorToKvBatchOp()
            .setVectorCol("vec")
            .setReservedCols("row")
            .setKvCol("kv")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	kv
1	0:1.0,1:2.0
2	0:4.0,1:8.0

向量转三元组 (VectorToTripleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.format.VectorToTripleBatchOp

Python 类名：VectorToTripleBatchOp

功能介绍

将数据格式从 Vector 转成 Triple，批式组件

一条输入数据可能对应多条输出结果，输入中vector的维度为多少，就会生成多少条结果。 因此输出数据时，最好保留记录的ID字段，也就是代码示例中的"row"字段，通过setReservedCols参数指定。

输出结果中除包含通过setReservedCols中指定列外，会输出vector的下标和对应位置的值两列。列名和格式通过setTripleColumnValueSchemaStr指定。注意该schemaStr中并没有对格式做限定，但如果设置不对会导致运行失败。 下标列可以设置为int long和string类型，第二列只能设置为string或者double。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
tripleColumnValueSchemaStr	三元组结构中列信息和数据信息的Schema	三元组结构中列信息和数据信息的Schema	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"E

reservedCols	算法保留列名	算法保留列	String[]			[]

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = VectorToTripleBatchOp()\
    .setVectorCol("vec")\
    .setReservedCols(["row"])\
    .setTripleColumnValueSchemaStr("col string, val double")\
    .linkFrom(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.format.VectorToTripleBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorToTripleBatchOpTest {
    @Test
    public void testVectorToTripleBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
```

```
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0),
            Row.of("2", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:4.0 1:8.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new VectorToTripleBatchOp()
            .setVectorCol("vec")
            .setReservedCols("row")
            .setTripleColumnValueSchemaStr("col string, val double")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

row	col	val
2	0	4.0000
1	0	1.0000
1	1	2.0000
2	1	8.0000

向量聚合 (VectorAssemblerBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorAssemblerBatchOp

Python 类名：VectorAssemblerBatchOp

功能介绍

- 数据结构转换组件，将Table格式的数据转成tensor格式数据。
- 支持table中的多个 vector 列和数值列合并成一个vector 列。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]
])

data = BatchOperator.fromDataframe(df, schemaStr="f0 int, f1 int, f2 int")
colnames = ["f0","f1","f2"]
VectorAssemblerBatchOp().setSelectedCols(colnames)\
.setOutputCol("out").linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.vector.VectorAssemblerBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorAssemblerBatchOpTest {
    @Test
    public void testVectorAssemblerBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 3)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 int, f1 int, f2
int");
        new VectorAssemblerBatchOp().setSelectedCols("f0", "f1", "f2")
            .setOutputCol("out").linkFrom(data).print();
    }
}
```

运行结果

f0	f1	f2	out
2	1	1	2.0 1.0 1.0
3	2	1	3.0 2.0 1.0
4	3	2	4.0 3.0 2.0
2	4	1	2.0 4.0 1.0
2	2	1	2.0 2.0 1.0
4	3	2	4.0 3.0 2.0
1	2	1	1.0 2.0 1.0
5	3	3	5.0 3.0 3.0

二元向量函数 (VectorBiFunctionBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorBiFunctionBatchOp

Python 类名：VectorBiFunctionBatchOp

功能介绍

- 对两个向量进行操作的函数，支持minus(减),plus(加),dot(内积),merge(拼接),EuclidDistance(欧式距离),Cosine(cos值), ElementWiseMultiply(点乘)。
- 支持稀疏和稠密两种 Vector。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
biFuncName	函数名字	函数操作名称, 可取minus(减),plus(加),dot(内积),merge(拼接),EuclidDistance(欧式距离),Cosine(cos值), ElementWiseMultiply(点乘).	String	✓	"Minus", "Dot", "Plus", "Merge", "EuclidDistance", "Cosine", "ElementWiseMultiply"	
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		

reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1 2 3", "2 3 4"]
])
data = BatchOperator.fromDataframe(df, schemaStr="vec1 string, vec2 string")
VectorBiFunctionBatchOp() \
        .setSelectedCols(["vec1", "vec2"]) \

.setBiFuncName("minus").setOutputCol("vec_minus").linkFrom(data).print();
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorBiFunction;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;
```

```
import java.util.ArrayList;
import java.util.List;

public class ToVectorBiFunctionTest extends AlinkTestBase {
    @Test
    public void testBiVectorFunction() throws Exception {
        List <Row> df = new ArrayList <>();
        df.add(Row.of("1 2 3", "2 3 4"));
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec1 string, vec2
string");
         new VectorBiFunctionBatchOp()
            .setSelectedCols("vec1", "vec2")
            .setBiFuncName("minus")
            .setOutputCol("vec_minus").linkFrom(data).print();
    }
}
```

运行结果

vec1	vec2	vec_minus
1 2 3	2 3 4	-1.0 -1.0 -1.0

向量元素依次相乘 (VectorElementwiseProductBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorElementwiseProductBatchOp

Python 类名：VectorElementwiseProductBatchOp

功能介绍

Vector 中的每一个非零元素与scalingVector的每一个对应元素乘，返回乘积后的新vector。

ScalingVector 通过参数单独指定。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
scalingVector	尺度变化向量。	尺度的变化向量。	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

# load data
df = pd.DataFrame([
    ["1:3,2:4,4:7", 1],
    ["0:3,5:5", 3],
    ["2:4,4:5", 4]
])

data = BatchOperator.fromDataframe(df, schemaStr="vec string, id bigint")
vecEP = VectorElementwiseProductBatchOp().setSelectedCol("vec") \
    .setOutputCol("vec1") \
    .setScalingVector("$8$1:3.0 3:3.0 5:4.6")
data.link(vecEP).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorElementwiseProductBatchO
p;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorElementwiseProductBatchOpTest {
    @Test
    public void testVectorElementwiseProductBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:3,2:4,4:7", 1),
            Row.of("0:3,5:5", 3),
            Row.of("2:4,4:5", 4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string, id
int");
        BatchOperator <?> vecEP = new
VectorElementwiseProductBatchOp().setSelectedCol("vec")
            .setOutputCol("vec1")
            .setScalingVector("$8$1:3.0 3:3.0 5:4.6");
        data.link(vecEP).print();
```

```
    }
}
```

运行结果

vec	id	vec1
1:3,2:4,4:7	1	1:9.0 2:0.0 4:0.0
0:3,5:5	3	0:0.0 5:23.0
2:4,4:5	4	2:0.0 4:0.0

向量函数 (VectorFunctionBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorFunctionBatchOp

Python 类名：VectorFunctionBatchOp

功能介绍

- 获取一个向量的最大值、最小值，或者最大值、最小值的索引，或者对向量做尺度变换, 求NormL2, 求NormL1, 求NormL2Square, Normalize。
- 支持稀疏和稠密两种 Vector。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
funcName	函数名字	函数操作名称, 可取max（最大值）, min（最小值）, argMax（最大值索引）, argMin（最小值索引）, scale（尺度变换）, NormL2, NormL1, NormL2Square, Normalize	String	✓	"Max", "Min", "ArgMax", "ArgMin", "Scale", "NormL2", "NormL1", "NormL2Square", "Normalize"	
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
WithVariable	变量	函数中变量	String			
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null

reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1,"16.3, 1.1, 1.1"],
    [2,"16.8, 1.4, 1.5"],
    [3,"19.2, 1.7, 1.8"],
    [4,"10.0, 1.7, 1.7"],
    [5,"19.5, 1.8, 1.9"],
    [6,"20.9, 1.8, 1.8"],
    [7,"21.1, 1.9, 1.8"],
    [8,"20.9, 2.0, 2.1"],
    [9,"20.3, 2.3, 2.4"],
    [10,"22.0, 2.4, 2.5"]
])

opData = BatchOperator.fromDataframe(df, schemaStr="id bigint, vec string")
result = VectorFunctionBatchOp().setSelectedCol("vec")\
.setOutputCol("out").setFuncName("max").linkFrom(opData)
result.collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.stream.BatchOperator;
import com.alibaba.alink.operator.stream.dataproc.vector.VectorFunctionBatchOp;
import com.alibaba.alink.operator.stream.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.ArrayList;
import java.util.List;

public class VectorFunctionTest extends AlinkTestBase {

    @Test
    public void testVectorFunction() throws Exception {
        List <Row> df = new ArrayList <>();
        df.add(Row.of(1, "16.3, 1.1, 1.1"));
        df.add(Row.of(2, "16.8, 1.4, 1.5"));
        df.add(Row.of(3, "19.2, 1.7, 1.8"));
        df.add(Row.of(4, "10.0, 1.7, 1.7"));
        df.add(Row.of(5, "19.5, 1.8, 1.9"));
        df.add(Row.of(6, "20.9, 1.8, 1.8"));
        df.add(Row.of(7, "21.1, 1.9, 1.8"));
        df.add(Row.of(8, "20.9, 2.0, 2.1"));
        df.add(Row.of(9, "20.3, 2.3, 2.4"));
        df.add(Row.of(10, "22.0, 2.4, 2.5"));

        BatchOperator<?> streamData =  new MemSourceBatchOp(df, "id int, vec
string");

        new VectorFunctionBatchOp().setSelectedCol("vec")

.setOutputCol("out").setFuncName("max").linkFrom(streamData).print();
    }
}
```

运行结果

id	vec	out
1	16.3, 1.1, 1.1	16.3
2	16.8, 1.4, 1.5	16.8
3	19.2, 1.7, 1.8	19.2
4	10.0, 1.7, 1.7	10.0

5	19.5, 1.8, 1.9	19.5
6	20.9, 1.8, 1.8	20.9
7	21.1, 1.9, 1.8	21.1
8	20.9, 2.0, 2.1	20.9
9	20.3, 2.3, 2.4	20.3
10	22.0, 2.4, 2.5	22.0

向量缺失值填充预测 (VectorImputerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorImputerPredictBatchOp

Python 类名：VectorImputerPredictBatchOp

功能介绍

使用 Vector 缺失值填充模型对Vector数据进行数据填充。

输入数据包含 VectorImputerTrainBatchOp 输出的模型和要处理的数据。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1:3,2:4,4:7", 1],
    ["1:3,2:NaN", 3],
    ["2:4,4:5", 4]
])

data = BatchOperator.fromDataframe(df, schemaStr="vec string, id bigint")
vecFill = VectorImputerTrainBatchOp().setSelectedCol("vec")
```

```
model = data.link(vecFill)
VectorImputerPredictBatchOp().setOutputCol("vec1").linkFrom(model,
data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorImputerPredictBatchOp;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorImputerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorImputerPredictBatchOpTest {
    @Test
    public void testVectorImputerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:3,2:4,4:7", 1),
            Row.of("1:3,2:NaN", 3),
            Row.of("2:4,4:5", 4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string, id
int");
        BatchOperator <?> vecFill = new
VectorImputerTrainBatchOp().setSelectedCol("vec");
        BatchOperator <?> model = data.link(vecFill);
        new VectorImputerPredictBatchOp().setOutputCol("vec1").linkFrom(model,
data).print();
    }
}
```

运行结果

vec	id	vec1
1:3,2:4,4:7	1	1:3.0 2:4.0 4:7.0
1:3,2:NaN	3	1:3.0 2:4.0
2:4,4:5	4	2:4.0 4:5.0

向量缺失值填充训练 (VectorImputerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorImputerTrainBatchOp

Python 类名：VectorImputerTrainBatchOp

功能介绍

训练Vecotor 缺失值填充模型的组件，输出模型。

填充策略包含最大值，最小值，均值和指定数值4种。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
fillValue	填充缺失值	自定义的填充值。当strategy为value时，读取fillValue的值	Double			null
strategy	缺失值填充规则	缺失值填充的规则，支持mean，max，min或者value。选择value时，需要读取fillValue的值	String		"MEAN", "MIN", "MAX", "VALUE"	"MEAN"

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1:3,2:4,4:7", 1],
    ["1:3,2:NaN", 3],
    ["2:4,4:5", 4]
])

data = BatchOperator.fromDataframe(df, schemaStr="vec string, id bigint")
vecFill = VectorImputerTrainBatchOp().setSelectedCol("vec")
model = data.link(vecFill)
VectorImputerPredictBatchOp().setOutputCol("vec1").linkFrom(model,
data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorImputerPredictBatchOp;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorImputerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorImputerTrainBatchOpTest {
    @Test
    public void testVectorImputerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:3,2:4,4:7", 1),
            Row.of("1:3,2:NaN", 3),
            Row.of("2:4,4:5", 4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string, id
int");
        BatchOperator <?> vecFill = new
VectorImputerTrainBatchOp().setSelectedCol("vec");
        BatchOperator <?> model = data.link(vecFill);
        new VectorImputerPredictBatchOp().setOutputCol("vec1").linkFrom(model,
data).print();
```

```
    }
}
```

运行结果

vec	id	vec1
1:3,2:4,4:7	1	1:3.0 2:4.0 4:7.0
1:3,2:NaN	3	1:3.0 2:4.0
2:4,4:5	4	2:4.0 4:5.0

向量元素两两相乘 (VectorInteractionBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorInteractionBatchOp

Python 类名：VectorInteractionBatchOp

功能介绍

对两个vector 中的元素两两相乘，并组成一个新的向量。

输入的两个向量长度分别为m和n，生成的向量长度为m*n

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

备注：选择列的数目必须为两列

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$8$1:3,2:4,4:7", "$8$1:3,2:4,4:7"],
    ["$8$0:3,5:5", "$8$1:2,2:4,4:7"],
    ["$8$2:4,4:5", "$5$1:3,2:3,4:7"]
])
```

```
data = BatchOperator.fromDataframe(df, schemaStr="vec1 string, vec2 string")
vecInter =
VectorInteractionBatchOp().setSelectedCols(["vec1","vec2"]).setOutputCol("vec_p
roduct")
vecInter.linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorInteractionBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorInteractionBatchOpTest {
    @Test
    public void testVectorInteractionBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("$8$1:3,2:4,4:7", "$8$1:3,2:4,4:7"),
            Row.of("$8$0:3,5:5", "$8$1:2,2:4,4:7"),
            Row.of("$8$2:4,4:5", "$5$1:3,2:3,4:7")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec1 string, vec2
string");
        BatchOperator <?> vecInter = new
VectorInteractionBatchOp().setSelectedCols("vec1", "vec2").setOutputCol(
            "vec_product");
        vecInter.linkFrom(data).print();
    }
}
```

运行结果

vec1	vec2	vec_product
$8$1:3,2:4,4:7	$8$1:3,2:4,4:7	$64$9:9.0 10:12.0 12:21.0 17:12.0 18:16.0 20:28.0 33:21.0 34:28.0 36:49.0
$8$0:3,5:5	$8$1:2,2:4,4:7	$64$8:6.0 13:10.0 16:12.0 21:20.0 32:21.0 37:35.0
$8$2:4,4:5	$5$1:3,2:3,4:7	$40$10:12.0 12:15.0 18:12.0 20:15.0 34:28.0 36:35.0

向量绝对值最大化预测 (VectorMaxAbsScalerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorMaxAbsScalerPredictBatchOp

Python 类名：VectorMaxAbsScalerPredictBatchOp

功能介绍

vector绝对值最大标准化是对vector数据按照最大值和最小值进行标准化的组件, 将数据归一到-1和1之间。

预测组件使用 VectorMaxAbsScalerTrainBatchOp 训练生成的模型，处理数据之后生成结果数据。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
```

```
    ["c", "100.9, 1"]
])

data = BatchOperator.fromDataframe(df, schemaStr="col string, vec string")
trainOp = VectorMaxAbsScalerTrainBatchOp()\
          .setSelectedCol("vec")
model = trainOp.linkFrom(data)

batchPredictOp = VectorMaxAbsScalerPredictBatchOp()
batchPredictOp.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorMaxAbsScalerPredictBatch
Op;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorMaxAbsScalerTrainBatchOp
;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorMaxAbsScalerPredictBatchOpTest {
    @Test
    public void testVectorMaxAbsScalerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vec
string");
        BatchOperator <?> trainOp = new VectorMaxAbsScalerTrainBatchOp()
            .setSelectedCol("vec");
        BatchOperator <?> model = trainOp.linkFrom(data);
        BatchOperator <?> batchPredictOp = new
VectorMaxAbsScalerPredictBatchOp();
```

```
        batchPredictOp.linkFrom(model, data).print();
    }
}
```

运行结果

col	vec
a	0.09910802775024777 1.0
b	-0.024777006937561942 0.09
c	0.9930624380574826 0.01
d	-0.9900891972249752 1.0
a	0.013875123885034686 0.01
b	-0.02180376610505451 0.09
c	1.0 0.01

向量绝对值最大化训练 (VectorMaxAbsScalerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorMaxAbsScalerTrainBatchOp

Python 类名：VectorMaxAbsScalerTrainBatchOp

功能介绍

vector绝对值最大标准化是对vector数据按照数值最大绝对值进行标准化的组件, 将数据归一到-1和1之间。输入的向量维度可以不相同。

计算公式为 value / max(| value |)

该组件生成Vector绝对值最大标准化的模型

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为 [DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
```

```
])

data = BatchOperator.fromDataframe(df, schemaStr="col string, vec string")
trainOp = VectorMaxAbsScalerTrainBatchOp()\
           .setSelectedCol("vec")
model = trainOp.linkFrom(data)

batchPredictOp = VectorMaxAbsScalerPredictBatchOp()
batchPredictOp.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorMaxAbsScalerPredictBatch
Op;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorMaxAbsScalerTrainBatchOp
;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorMaxAbsScalerTrainBatchOpTest {
    @Test
    public void testVectorMaxAbsScalerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vec
string");
        BatchOperator <?> trainOp = new VectorMaxAbsScalerTrainBatchOp()
            .setSelectedCol("vec");
        BatchOperator <?> model = trainOp.linkFrom(data);
        BatchOperator <?> batchPredictOp = new
VectorMaxAbsScalerPredictBatchOp();
        batchPredictOp.linkFrom(model, data).print();
```

```
    }
}
```

运行结果

col	vec
a	0.09910802775024777 1.0
b	-0.024777006937561942 0.09
c	0.9930624380574826 0.01
d	-0.9900891972249752 1.0
a	0.013875123885034686 0.01
b	-0.02180376610505451 0.09
c	1.0 0.01

向量归一化预测 (VectorMinMaxScalerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorMinMaxScalerPredictBatchOp

Python 类名：VectorMinMaxScalerPredictBatchOp

功能介绍

vector归一化是对vector数据进行归一化处理的组件, 将数据归一到minValue和maxValue之间，value最终结果为 (value - min) / (max - min) * (maxValue - minValue) + minValue，最终结果的范围为[minValue, maxValue]。

minValue和maxValue由用户指定，默认为0和1。输入的向量维度可以不相同。

该组件为预测组件，加载模型后就可以处理数据

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
    ["c", "100.2, 1"],
```

```
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
])

data = BatchOperator.fromDataframe(df, schemaStr="col string, vec string")

trainOp = VectorMinMaxScalerTrainBatchOp()\
           .setSelectedCol("vec")
model = trainOp.linkFrom(data)

batchPredictOp = VectorMinMaxScalerPredictBatchOp()
batchPredictOp.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorMinMaxScalerPredictBatch
Op;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorMinMaxScalerTrainBatchOp
;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorMinMaxScalerPredictBatchOpTest {
    @Test
    public void testVectorMinMaxScalerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vec
string");
        BatchOperator <?> trainOp = new VectorMinMaxScalerTrainBatchOp()
```

```
            .setSelectedCol("vec");
        BatchOperator <?> model = trainOp.linkFrom(data);
        BatchOperator <?> batchPredictOp = new
VectorMinMaxScalerPredictBatchOp();
        batchPredictOp.linkFrom(model, data).print();
    }
}
```

运行结果

col	vec
a	0.5473107569721115 1.0
b	0.4850597609561753 0.08080808080808081
c	0.9965139442231076 0.0
d	0.0 1.0
a	0.5044820717131474 0.0
b	0.4865537848605578 0.08080808080808081
c	1.0 0.0

向量归一化训练 (VectorMinMaxScalerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorMinMaxScalerTrainBatchOp

Python 类名：VectorMinMaxScalerTrainBatchOp

功能介绍

vector归一化是对vector数据进行归一化处理的组件, 将数据归一到minValue和maxValue之间，value最终结果为 (value - min) / (max - min) * (maxValue - minValue) + minValue，最终结果的范围为[minValue, maxValue]。

minValue和maxValue由用户指定，默认为0和1。输入的向量维度可以不相同。

该组件为训练组件，生成的结果为模型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为 [DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
max	归一化的上界	归一化的上界	Double			1.0
min	归一化的下界	归一化的下界	Double			0.0

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
```

```
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
])

data = BatchOperator.fromDataframe(df, schemaStr="col string, vec string")

trainOp = VectorMinMaxScalerTrainBatchOp()\
           .setSelectedCol("vec")
model = trainOp.linkFrom(data)

batchPredictOp = VectorMinMaxScalerPredictBatchOp()
batchPredictOp.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorMinMaxScalerPredictBatch
Op;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorMinMaxScalerTrainBatchOp
;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorMinMaxScalerTrainBatchOpTest {
    @Test
    public void testVectorMinMaxScalerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vec
string");
```

```
        BatchOperator <?> trainOp = new VectorMinMaxScalerTrainBatchOp()
            .setSelectedCol("vec");
        BatchOperator <?> model = trainOp.linkFrom(data);
        BatchOperator <?> batchPredictOp = new
VectorMinMaxScalerPredictBatchOp();
        batchPredictOp.linkFrom(model, data).print();
    }
}
```

运行结果

col	vec
a	0.5473107569721115 1.0
b	0.4850597609561753 0.08080808080808081
c	0.9965139442231076 0.0
d	0.0 1.0
a	0.5044820717131474 0.0
b	0.4865537848605578 0.08080808080808081
c	1.0 0.0

向量标准化 (VectorNormalizeBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorNormalizeBatchOp

Python 类名：VectorNormalizeBatchOp

功能介绍

对 Vector 进行正则化操作。

指定参数范数的阶，例如p = 2, 对于向量，计算向量的平方和再开二次方记为norm，最终计算结果为

通过 setOutputCol，指定新生成的列名。如果不指定，默认替代输入列。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
p	范数的阶	范数的阶，默认2	Double			2.0
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ["1:3,2:4,4:7", 1],
    ["0:3,5:5", 3],
    ["2:4,4:5", 4]
])

data = BatchOperator.fromDataframe(df, schemaStr="vec string, id bigint")
VectorNormalizeBatchOp().setSelectedCol("vec").setOutputCol("vec_norm").linkFro
m(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.vector.VectorNormalizeBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorNormalizeBatchOpTest {
    @Test
    public void testVectorNormalizeBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:3,2:4,4:7", 1),
            Row.of("0:3,5:5", 3),
            Row.of("2:4,4:5", 4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string, id
int");
        new
VectorNormalizeBatchOp().setSelectedCol("vec").setOutputCol("vec_norm").linkFro
m(data).print();
    }
}
```

运行结果

vec	id	vec_norm
1:3,2:4,4:7	1	1:0.34874291623145787 2:0.46499055497527714 4:0.813733471206735

0:3,5:5	3	0:0.5144957554275265 5:0.8574929257125441
2:4,4:5	4	2:0.6246950475544243 4:0.7808688094430304

向量多项式展开 (VectorPolynomialExpandBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorPolynomialExpandBatchOp

Python 类名：VectorPolynomialExpandBatchOp

功能介绍

对 Vector 进行多项式展开，生成一个新的Vector。

调用 setDegree ，设置幂，默认2。调用 setOutputCol，设置生成的列名，如果不指定，默认替代输入列。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
degree	多项式阶数	多项式的阶数，默认2	Integer		[1, +inf)	2
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    ["$8$1:3,2:4,4:7"],
    ["$8$2:4,4:5"]
])

data = BatchOperator.fromDataframe(df, schemaStr="vec string")
VectorPolynomialExpandBatchOp().setSelectedCol("vec").setOutputCol("vec_out").l
inkFrom(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorPolynomialExpandBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorPolynomialExpandBatchOpTest {
    @Test
    public void testVectorPolynomialExpandBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("$8$1:3,2:4,4:7"),
            Row.of("$8$2:4,4:5")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string");
        new
VectorPolynomialExpandBatchOp().setSelectedCol("vec").setOutputCol("vec_out").l
inkFrom(data).print();
    }
}
```

运行结果

vec	vec_out
$8$1:3,2:4,4:7	$44$2:3.0 4:9.0 5:4.0 7:12.0 8:16.0 14:7.0 16:21.0 17:28.0 19:49.0
$8$2:4,4:5	$44$5:4.0 8:16.0 14:5.0 17:20.0 19:25.0

向量长度检验 (VectorSizeHintBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorSizeHintBatchOp

Python 类名：VectorSizeHintBatchOp

功能介绍

取出Vector 的size进行检测，并进行处理。

指定size大小和处理策略（抛异常error和跳过skip），当大小不匹配时，触发处理策略。 支持稀疏向量和稠密向量。如果不设置 OutputCol ，默认替换输入列。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
size	向量大小	用于判断向量的大小是否和设置的一致	Integer	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
outputCol	输出结果列	输出结果列列名，可选，默认 null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$8$1:3,2:4,4:7"],
    ["$8$2:4,4:5"]
])

data = BatchOperator.fromDataframe(df, schemaStr="vec string")
VectorSizeHintBatchOp().setSelectedCol("vec").setOutputCol("vec_hint").setHandl
eInvalidMethod("SKIP").setSize(8).linkFrom(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.vector.VectorSizeHintBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorSizeHintBatchOpTest {
    @Test
    public void testVectorSizeHintBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("$8$1:3,2:4,4:7"),
            Row.of("$8$2:4,4:5")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string");
        new
VectorSizeHintBatchOp().setSelectedCol("vec").setOutputCol("vec_hint").setHandl
eInvalidMethod("SKIP")
            .setSize(8).linkFrom(data).print();
    }
}
```

运行结果

vec	vec_hint

$8$1:3,2:4,4:7	$8$1:3.0 2:4.0 4:7.0
$8$2:4,4:5	$8$2:4.0 4:5.0

向量切片 (VectorSliceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorSliceBatchOp

Python 类名：VectorSliceBatchOp

功能介绍

取出Vector 中的若干列，组成一个新的Vector，输出到新的一列中。

可同时处理稀疏向量和稠密向量，如果为稠密向量，必须保证向量维度大于要抽取的最大下标。如下面代码中的例子，稠密向量长度必须大于3。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
indices	需要被提取的索引数组	需要被提取的索引数组	int[]			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1:3,2:4,4:7", 1],
    ["0:3,5:5", 3],
    ["2:4,4:5", 4]
```

```
])

data = BatchOperator.fromDataframe(df, schemaStr="vec string, id bigint")
vecSlice =
VectorSliceBatchOp().setSelectedCol("vec").setOutputCol("vec_slice").setIndices
([1,2,3])
vecSlice.linkFrom(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.vector.VectorSliceBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorSliceBatchOpTest {
    @Test
    public void testVectorSliceBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:3,2:4,4:7", 1),
            Row.of("0:3,5:5", 3),
            Row.of("2:4,4:5", 4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string, id
int");
        BatchOperator <?> vecSlice = new
VectorSliceBatchOp().setSelectedCol("vec").setOutputCol("vec_slice")
            .setIndices(new int[] {1, 2, 3});
        vecSlice.linkFrom(data).print();
    }
}
```

运行结果

vec	id	vec_slice
1:3,2:4,4:7	1	$3$0:3.0 1:4.0
0:3,5:5	3	3
2:4,4:5	4	$3$1:4.0

向量标准化预测 (VectorStandardScalerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorStandardScalerPredictBatchOp

Python 类名：VectorStandardScalerPredictBatchOp

功能介绍

标准化是对向量数据进行按正态化处理的组件

VectorStandardScalerTrainBatchOp 计算向量的每一列的均值和方差，组件可以指定默认均值为0，标准差为1。生成向量标准化的模型，在 VectorStandardScalerPredictBatchOp 中加载，对数据做标准化处理。

输入的向量可以同时包含稀疏向量和稠密向量，向量维度也可以不相同。输入稠密向量维度不够时，没有的维度默认为0。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
```

```
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
])

data = BatchOperator.fromDataframe(df, schemaStr="col string, vector string")
trainOp = VectorStandardScalerTrainBatchOp().setSelectedCol("vector")
model = trainOp.linkFrom(data)
VectorStandardScalerPredictBatchOp().linkFrom(model, data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorStandardScalerPredictBat
chOp;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorStandardScalerTrainBatch
Op;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorStandardScalerPredictBatchOpTest {
    @Test
    public void testVectorStandardScalerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vector
string");
        BatchOperator <?> trainOp = new
VectorStandardScalerTrainBatchOp().setSelectedCol("vector");
        BatchOperator <?> model = trainOp.linkFrom(data);
        new VectorStandardScalerPredictBatchOp().linkFrom(model, data).print();
```

```
    }
}
```

运行结果

col1	vec
a	-0.07835182408093559,1.4595814453461897
c	1.2269606224811418,-0.6520885789229323
b	-0.2549018445693762,-0.4814485769617911
a	-0.20280511721213143,-0.6520885789229323
c	1.237090541689495,-0.6520885789229323
b	-0.25924323851581327,-0.4814485769617911
d	-1.6687491397923802,1.4595814453461897

向量标准化训练 (VectorStandardScalerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorStandardScalerTrainBatchOp

Python 类名：VectorStandardScalerTrainBatchOp

功能介绍

标准化是对向量数据进行按正态化处理的组件

VectorStandardScalerTrainBatchOp 计算向量的每一列的均值和方差，组件可以指定默认均值为0，标准差为1。生成向量标准化的模型，在 VectorStandardScalerPredictBatchOp 中加载，对数据做标准化处理。

输入的向量可以同时包含稀疏向量和稠密向量，向量维度也可以不相同。输入稠密向量维度不够时，没有的维度默认为0。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
withMean	是否使用均值	是否使用均值，默认使用	Boolean			true
withStd	是否使用标准差	是否使用标准差，默认使用	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
])

data = BatchOperator.fromDataframe(df, schemaStr="col string, vector string")
trainOp = VectorStandardScalerTrainBatchOp().setSelectedCol("vector")
model = trainOp.linkFrom(data)
VectorStandardScalerPredictBatchOp().linkFrom(model, data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorStandardScalerPredictBat
chOp;
import
com.alibaba.alink.operator.batch.dataproc.vector.VectorStandardScalerTrainBatch
Op;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorStandardScalerTrainBatchOpTest {
    @Test
    public void testVectorStandardScalerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vector
string");
        BatchOperator <?> trainOp = new
```

```
VectorStandardScalerTrainBatchOp().setSelectedCol("vector");
        BatchOperator <?> model = trainOp.linkFrom(data);
        new VectorStandardScalerPredictBatchOp().linkFrom(model, data).print();
    }
}
```

运行结果

col1	vec
a	-0.07835182408093559,1.4595814453461897
c	1.2269606224811418,-0.6520885789229323
b	-0.2549018445693762,-0.4814485769617911
a	-0.20280511721213143,-0.6520885789229323
c	1.237090541689495,-0.6520885789229323
b	-0.25924323851581327,-0.4814485769617911
d	-1.6687491397923802,1.4595814453461897

VectorToColumnsLegacyBatchOp (VectorToColumnsLegacyBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorToColumnsLegacyBatchOp

Python 类名：VectorToColumnsLegacyBatchOp

功能介绍

- 数据结构转换组件，将vector格式的数据转成Table格式数据。这里，vector格式的数据是形如"1.1,1.3,1.2"的字符串; 转换后，结果变为三列，列名由用户指定。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCols	输出结果列列名数组	输出结果列列名数组，必选	String[]	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

data = np.array([\
["1:3,2:4,4:7", 1],\
["0:3,5:5", 3],\
["2:4,4:5", 4]])
df = pd.DataFrame({"vec":data[:,0], "id":data[:,1]})
opData = dataframeToOperator(df, schemaStr="vec string, id
bigint",op_type="batch")
vec2col = VectorToColumnsLegacyBatchOp().setSelectedCol("vec")\
.setOutputCols(["f0", "f1", "f2", "f3", "f4", "f5"]).linkFrom(opData)
vec2col.collectToDataframe()
```

运行结果

vec	id	f0	f1	f2	f3	f4	f5
1:3,2:4,4:7	1	0.0	3.0	4.0	0.0	7.0	0.0
0:3,5:5	3	3.0	0.0	0.0	0.0	0.0	5.0
2:4,4:5	4	0.0	0.0	4.0	0.0	5.0	0.0

VectorToTripleLegacyBatchOp (VectorToTripleLegacyBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dataproc.vector.VectorToTripleLegacyBatchOp

Python 类名：VectorToTripleLegacyBatchOp

功能介绍

数据结构转换组件，将稀疏向量转成三元组格式数据。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCols	输出结果列列名数组	输出结果列列名数组，必选	String[]	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		
reservedCols	算法保留列名	算法保留列	String[]			[]

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
data = np.array([['1', '2:2.0 3:2.2']])
df_data = pd.DataFrame(data)
schema = 'id string, vec string'
batch_data = dataframeToOperator(df_data, schemaStr=schema, op_type='batch')

op = VectorToTripleLegacyBatchOp().setSelectedCol('vec').setOutputCols("id, f2,
f3")
op.linkFrom(batch_data).collectToDataframe()
```

运行结果

id	f2	f3
1	2	2.0

1	3	2.2

SQL操作：As (AsBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.AsBatchOp

Python 类名：AsBatchOp

功能介绍

对批式数据进行sql的AS操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
clause	运算语句	运算语句	String	✓		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])

batch_data = BatchOperator.fromDataframe(df, schemaStr='f1 string, f2 bigint,
f3 double')
op = AsBatchOp().setClause("ff1,ff2,ff3")
batch_data = batch_data.link(op)
batch_data.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.AsBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AsBatchOpTest {
    @Test
    public void testAsBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> batch_data = new MemSourceBatchOp(df, "f1 string, f2
int, f3 double");
        BatchOperator <?> op = new AsBatchOp().setClause("ff1,ff2,ff3");
        batch_data = batch_data.link(op);
        batch_data.print();
    }
}
```

运行结果

ff1	ff2	ff3
Ohio	2000	1.5000
Ohio	2001	1.7000
Ohio	2002	3.6000
Nevada	2001	2.4000
Nevada	2002	2.9000
Nevada	2003	3.2000

SQL操作：Distinct (DistinctBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.DistinctBatchOp

Python 类名：DistinctBatchOp

功能介绍

对批式数据进行sql的DISTINCT操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])

batch_data = BatchOperator.fromDataframe(df, schemaStr='f1 string, f2 bigint,
f3 double')

batch_data.select('f1').link(DistinctBatchOp()).print()
```

Java 代码

```
import org.apache.flink.types.Row;
```

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.DistinctBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DistinctBatchOpTest {
    @Test
    public void testDistinctBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> batch_data = new MemSourceBatchOp(df, "f1 string, f2
int, f3 double");
        batch_data.select("f1").link(new DistinctBatchOp()).print();
    }
}
```

运行结果

f1	
Nevada	
Ohio	

SQL操作：Filter (FilterBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.FilterBatchOp

Python 类名：FilterBatchOp

功能介绍

对批式数据进行sql的FILTER操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
clause	运算语句	运算语句	String	✓		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])

batch_data = BatchOperator.fromDataframe(df, schemaStr='f1 string, f2 bigint,
f3 double')

op = FilterBatchOp().setClause("f1='Ohio'")
batch_data = batch_data.link(op)
batch_data.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.FilterBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FilterBatchOpTest {
    @Test
    public void testFilterBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> batch_data = new MemSourceBatchOp(df, "f1 string, f2
int, f3 double");
        BatchOperator <?> op = new FilterBatchOp().setClause("f1='Ohio'");
        batch_data = batch_data.link(op);
        batch_data.print();
    }
}
```

运行结果

f1	f2	f3
Ohio	2000	1.5000
Ohio	2001	1.7000
Ohio	2002	3.6000

SQL操作：FullOuterJoin (FullOuterJoinBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.FullOuterJoinBatchOp

Python 类名：FullOuterJoinBatchOp

功能介绍

对批式数据进行sql的FULL OUTER JOIN操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
joinPredicate	where语句	where语句	String	✓		
selectClause	select语句	select语句	String	✓		
type	join类型	join类型: "join", "leftOuterJoin", "rightOuterJoin" 或 "fullOuterJoin"	String		"JOIN", "LEFTOUTERJOIN", "RIGHTOUTERJOIN", "FULLOUTERJOIN"	"JOIN"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])
```

```
batch_data1 = BatchOperator.fromDataframe(df, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data2 = BatchOperator.fromDataframe(df, schemaStr='f1 string, f2 bigint,
f3 double')

op = FullOuterJoinBatchOp().setJoinPredicate("a.f1=b.f1 and
a.f2=b.f2").setSelectClause("a.f1, a.f2, a.f3")
result = op.linkFrom(batch_data1, batch_data2)
result.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.FullOuterJoinBatchOp;
import org.junit.Test;

public class FullOuterJoinBatchOpTest {
    @Test
    public void testFullOuterJoinBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data1 = new MemSourceBatchOp(df, "f1 string, f2 int,
f3 double");
        BatchOperator <?> data2 = new MemSourceBatchOp(df, "f1 string, f2 int,
f3 double");
        BatchOperator <?> joinOp =
            new FullOuterJoinBatchOp().setJoinPredicate("a.f1=b.f1 and
a.f2=b.f2").setSelectClause(
                "a.f1, a.f2, a.f3");
        joinOp.linkFrom(data1, data2).print();
    }
}
```

运行结果

f1	f2	f3
Nevada	2001	2.4000

Nevada	2002	2.9000
Nevada	2003	3.2000
Ohio	2000	1.5000
Ohio	2001	1.7000
Ohio	2002	3.6000

SQL操作：GroupBy (GroupByBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.GroupByBatchOp

Python 类名：GroupByBatchOp

功能介绍

对批式数据进行sql的GROUPBY操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
groupByPredicate	groupby语句	groupby语句	String	✓		
selectClause	select语句	select语句	String	✓		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])

batch_data = BatchOperator.fromDataframe(df, schemaStr='f1 string, f2 bigint,
f3 double')

op = GroupByBatchOp().setGroupByPredicate("f1").setSelectClause("f1,avg(f2) as
f2")
batch_data = batch_data.link(op)
```

```
batch_data.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GroupByBatchOpTest {
    @Test
    public void testGroupByBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> batch_data = new MemSourceBatchOp(df, "f1 string, f2
int, f3 double");
        BatchOperator <?> op = new
GroupByBatchOp().setGroupByPredicate("f1").setSelectClause("f1,avg(f2) as f2");
        batch_data = batch_data.link(op);
        batch_data.print();
    }
}
```

运行结果

f1	f2
Nevada	2002
Ohio	2001

SQL操作：IntersectAll (IntersectAllBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.IntersectAllBatchOp

Python 类名：IntersectAllBatchOp

功能介绍

对批式数据进行sql的INTERSECTALL操作。(一个数据集交另一个数据集，不去重)

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df1 = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])
df2 = pd.DataFrame([
    ['Nevada', 2001, 2.4],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])

batch_data1 = BatchOperator.fromDataframe(df1, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data2 = BatchOperator.fromDataframe(df2, schemaStr='f1 string, f2 bigint,
f3 double')
```

```
op = IntersectAllBatchOp()
op.linkFrom(batch_data1, batch_data2).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.IntersectAllBatchOp;
import org.junit.Test;

public class IntersectAllBatchOpTest {
    @Test
    public void testIntersectAllBatchOp() throws Exception {
        List <Row> df1 = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        List <Row> df2 = Arrays.asList(
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data1 = new MemSourceBatchOp(df1, "f1 string, f2 int,
f3 double");
        BatchOperator <?> data2 = new MemSourceBatchOp(df2, "f1 string, f2 int,
f3 double");
        BatchOperator <?> intersectAllOp = new IntersectAllBatchOp();
        intersectAllOp.linkFrom(data1, data2).print();
    }
}
```

运行结果

f1	f2	f3
Nevada	2001	2.4000
Nevada	2001	2.4000
Nevada	2003	3.2000

SQL操作：Intersect (IntersectBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.IntersectBatchOp

Python 类名：IntersectBatchOp

功能介绍

对批式数据进行sql的INTERSECT操作。(一个数据集交另一个数据集，去重)

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df1 = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])
df2 = pd.DataFrame([
    ['Nevada', 2001, 2.4],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])

batch_data1 = BatchOperator.fromDataframe(df1, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data2 = BatchOperator.fromDataframe(df2, schemaStr='f1 string, f2 bigint,
f3 double')
```

```
op = IntersectBatchOp()
op.linkFrom(batch_data1, batch_data2).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.IntersectBatchOp;
import org.junit.Test;

public class IntersectBatchOpTest {
    @Test
    public void testIntersectBatchOp() throws Exception {
        List <Row> df1 = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        List <Row> df2 = Arrays.asList(
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data1 = new MemSourceBatchOp(df1, "f1 string, f2 int,
f3 double");
        BatchOperator <?> data2 = new MemSourceBatchOp(df2, "f1 string, f2 int,
f3 double");
        BatchOperator <?> intersectAllOp = new IntersectBatchOp();
        intersectAllOp.linkFrom(data1, data2).print();
    }
}
```

运行结果

f1	f2	f3
Nevada	2001	2.4000
Nevada	2003	3.2000

SQL操作：Join (JoinBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.JoinBatchOp

Python 类名：JoinBatchOp

功能介绍

对批式数据进行sql的JOIN操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
joinPredicate	where语句	where语句	String	✓		
selectClause	select语句	select语句	String	✓		
type	join类型	join类型: "join", "leftOuterJoin", "rightOuterJoin" 或 "fullOuterJoin"	String		"JOIN", "LEFTOUTERJOIN", "RIGHTOUTERJOIN", "FULLOUTERJOIN"	"JOIN"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df1 = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])
```

```
df2 = pd.DataFrame([
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])

batch_data1 = BatchOperator.fromDataframe(df1, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data2 = BatchOperator.fromDataframe(df2, schemaStr='f1 string, f2 bigint,
f3 double')

op = JoinBatchOp().setJoinPredicate("a.f1=b.f1").setSelectClause("a.f1, a.f2,
a.f3")
op.linkFrom(batch_data1, batch_data2).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.JoinBatchOp;
import org.junit.Test;

public class JoinBatchOpTest {
    @Test
    public void testJoinBatchOp() throws Exception {
        List <Row> df1 = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        List <Row> df2 = Arrays.asList(
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data1 = new MemSourceBatchOp(df1, "f1 string, f2 int,
f3 double");
        BatchOperator <?> data2 = new MemSourceBatchOp(df2, "f1 string, f2 int,
f3 double");
        BatchOperator <?> joinOp = new
JoinBatchOp().setJoinPredicate("a.f1=b.f1").setSelectClause(
            "a.f1, a.f2, a.f3");
        joinOp.linkFrom(data1, data2).print();
    }
}
```

运行结果

f1	f2	f3
Nevada	2001	2.4000
Nevada	2002	2.9000
Nevada	2003	3.2000
Nevada	2001	2.4000
Nevada	2002	2.9000
Nevada	2003	3.2000

SQL操作：LeftOuterJoin (LeftOuterJoinBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.LeftOuterJoinBatchOp

Python 类名：LeftOuterJoinBatchOp

功能介绍

对批式数据进行sql的LEFT OUTER JOIN操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
joinPredicate	where语句	where语句	String	✓		
selectClause	select语句	select语句	String	✓		
type	join类型	join类型: "join", "leftOuterJoin", "rightOuterJoin" 或 "fullOuterJoin"	String		"JOIN", "LEFTOUTERJOIN", "RIGHTOUTERJOIN", "FULLOUTERJOIN"	"JOIN"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df1 = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])
```

```
df2 = pd.DataFrame([
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])

batch_data1 = BatchOperator.fromDataframe(df1, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data2 = BatchOperator.fromDataframe(df2, schemaStr='f1 string, f2 bigint,
f3 double')

op =
LeftOuterJoinBatchOp().setJoinPredicate("a.f1=b.f1").setSelectClause("a.f1,
a.f2, a.f3")
op.linkFrom(batch_data1, batch_data2).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.LeftOuterJoinBatchOp;
import org.junit.Test;

public class LeftOuterJoinBatchOpTest {
    @Test
    public void testLeftOuterJoinBatchOp() throws Exception {
        List <Row> df1 = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        List <Row> df2 = Arrays.asList(
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data1 = new MemSourceBatchOp(df1, "f1 string, f2 int,
f3 double");
        BatchOperator <?> data2 = new MemSourceBatchOp(df2, "f1 string, f2 int,
f3 double");
        BatchOperator <?> joinOp = new LeftOuterJoinBatchOp()
            .setJoinPredicate("a.f1=b.f1")
            .setSelectClause("a.f1, a.f2, a.f3");
        joinOp.linkFrom(data1, data2).print();
    }
}
```

运行结果

f1	f2	f3
Nevada	2001	2.4000
Nevada	2001	2.4000
Nevada	2002	2.9000
Nevada	2002	2.9000
Nevada	2003	3.2000
Nevada	2003	3.2000
Ohio	2000	1.5000
Ohio	2001	1.7000
Ohio	2002	3.6000

SQL操作：MinusAll (MinusAllBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.MinusAllBatchOp

Python 类名：MinusAllBatchOp

功能介绍

对批式数据进行sql的MINUS ALL操作。(一个数据集减去另一个数据集, 不去重)

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df1 = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2000, 1.5],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])
df2 = pd.DataFrame([
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])

batch_data1 = BatchOperator.fromDataframe(df1, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data2 = BatchOperator.fromDataframe(df2, schemaStr='f1 string, f2 bigint,
f3 double')

MinusAllBatchOp().linkFrom(batch_data1, batch_data2).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.MinusAllBatchOp;
import org.junit.Test;

public class MinusAllBatchOpTest {
    @Test
    public void testMinusAllBatchOp() throws Exception {
        List <Row> df1 = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        List <Row> df2 = Arrays.asList(
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data1 = new MemSourceBatchOp(df1, "f1 string, f2 int,
f3 double");
        BatchOperator <?> data2 = new MemSourceBatchOp(df2, "f1 string, f2 int,
f3 double");
        BatchOperator <?> minusAllOp = new MinusAllBatchOp();
        minusAllOp.linkFrom(data1, data2).print();
    }
}
```

运行结果

f1	f2	f3
Nevada	2002	2.9000
Ohio	2000	1.5000
Ohio	2000	1.5000
Ohio	2002	3.6000

SQL操作：Minus (MinusBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.MinusBatchOp

Python 类名：MinusBatchOp

功能介绍

对批式数据进行sql的MINUS操作。(一个数据集减去另一个数据集, 去重)

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df1 = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2000, 1.5],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])
df2 = pd.DataFrame([
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])

batch_data1 = BatchOperator.fromDataframe(df1, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data2 = BatchOperator.fromDataframe(df2, schemaStr='f1 string, f2 bigint,
f3 double')

MinusBatchOp().linkFrom(batch_data1, batch_data2).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.MinusBatchOp;
import org.junit.Test;

public class MinusBatchOpTest {
    @Test
    public void testMinusBatchOp() throws Exception {
        List <Row> df1 = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        List <Row> df2 = Arrays.asList(
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data1 = new MemSourceBatchOp(df1, "f1 string, f2 int,
f3 double");
        BatchOperator <?> data2 = new MemSourceBatchOp(df2, "f1 string, f2 int,
f3 double");
        BatchOperator <?> minusAllOp = new MinusBatchOp();
        minusAllOp.linkFrom(data1, data2).print();
    }
}
```

运行结果

f1	f2	f3
Nevada	2002	2.9000
Ohio	2000	1.5000
Ohio	2002	3.6000

SQL操作：OrderBy (OrderByBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.OrderByBatchOp

Python 类名：OrderByBatchOp

功能介绍

对批式数据进行sql的ORDER BY操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
clause	运算语句	运算语句	String	✓		
fetch	fetch的record数目	fetch的record数目	Integer			
limit	record的limit数	record的limit数	Integer			
offset	fetch的偏移值	fetch的偏移值	Integer			
order	排序方法	排序方法	String			"asc"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2000, 1.5],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])

batch_data = BatchOperator.fromDataframe(df, schemaStr='f1 string, f2 bigint,
f3 double')
```

```
batch_data.link(OrderByBatchOp().setClause("f2")).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.OrderByBatchOp;
import org.junit.Test;

public class OrderByBatchOpTest {
    @Test
    public void testOrderByBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f1 string, f2 int,
f3 double");
        data.link(new OrderByBatchOp().setClause("f2")).print();
    }
}
```

运行结果

f1	f2	f3
Ohio	2000	1.5000
Ohio	2000	1.5000
Nevada	2001	2.4000
Ohio	2002	3.6000
Nevada	2002	2.9000
Nevada	2003	3.2000

SQL操作：RightOuterJoin (RightOuterJoinBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.RightOuterJoinBatchOp

Python 类名：RightOuterJoinBatchOp

功能介绍

对批式数据进行sql的RIGHT OUTER JOIN操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
joinPredicate	where语句	where语句	String	✓		
selectClause	select语句	select语句	String	✓		
type	join类型	join类型: "join", "leftOuterJoin", "rightOuterJoin" 或 "fullOuterJoin"	String		"JOIN", "LEFTOUTERJOIN", "RIGHTOUTERJOIN", "FULLOUTERJOIN"	"JOIN"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df1 = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])
```

```
df2 = pd.DataFrame([
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])

batch_data1 = BatchOperator.fromDataframe(df1, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data2 = BatchOperator.fromDataframe(df2, schemaStr='f1 string, f2 bigint,
f3 double')

op =
RightOuterJoinBatchOp().setJoinPredicate("a.f1=b.f1").setSelectClause("a.f1,
a.f2, a.f3")
op.linkFrom(batch_data2, batch_data1).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.RightOuterJoinBatchOp;
import org.junit.Test;

public class RightOuterJoinBatchOpTest {
    @Test
    public void testRightOuterJoinBatchOp() throws Exception {
        List <Row> df1 = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        List <Row> df2 = Arrays.asList(
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data1 = new MemSourceBatchOp(df1, "f1 string, f2 int,
f3 double");
        BatchOperator <?> data2 = new MemSourceBatchOp(df2, "f1 string, f2 int,
f3 double");
        BatchOperator <?> joinOp = new RightOuterJoinBatchOp()
            .setJoinPredicate("a.f1=b.f1")
            .setSelectClause("a.f1, a.f2, a.f3");
        joinOp.linkFrom(data2, data1).print();
    }
}
```

运行结果

f1	f2	f3
Nevada	2001	2.4000
Nevada	2001	2.4000
Nevada	2001	2.4000
Nevada	2003	3.2000
Nevada	2003	3.2000
Nevada	2003	3.2000
null	null	null
null	null	null
null	null	null

SQL操作：Select (SelectBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.SelectBatchOp

Python 类名：SelectBatchOp

功能介绍

对批式数据进行sql的SELECT操作。

使用方式

SELECT 语句中支持的内置函数可以参考 Flink 对应版本的官方文档： System (Built-in) Functions 。 但需要注意，有些内置函数在旧 planner 中不支持，在这里列出：Unsupported Built-In Functions 。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
clause	运算语句	运算语句	String	✓		

代码示例

Python 代码

```
df = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])

batch_data = BatchOperator.fromDataframe(df, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data.link(SelectBatchOp().setClause("f1 as name")).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.SelectBatchOp;
```

```
import org.junit.Test;

public class SelectBatchOpTest {
    @Test
    public void testSelectBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f1 string, f2 int,
f3 double");
        data.link(new SelectBatchOp().setClause("f1 as name")).print();
    }
}
```

运行结果

name

Ohio Ohio Ohio Nevada Nevada Nevada

SqlCmd (SqlCmdBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.SqlCmdBatchOp

Python 类名：SqlCmdBatchOp

功能介绍

编写SQL语句处理批式数据。

内置函数

Sql组件支持一组用于数据转换的内置函数。Sql组件使用标准的ANSI SQL语法。

比较函数

SQL语法	描述
value1 = value2	等于。
value1 <> value2	不相等。
value1 > value2	比...更大。
value1 >= value2	大于或等于。
value1 < value2	少于。
value1 <= value2	小于等于。
value IS NULL	如果value为null，则返回TRUE。
value IS NOT NULL	如果value不为null，则返回TRUE。
value1 IS DISTINCT FROM value2	如果两个值不相等则返回TRUE，将null值视为相同。

value1 IS NOT DISTINCT FROM value2	如果两个值相等则返回TRUE，将null值视为相同。
value1 NOT BETWEEN value2 AND value3	如果value1小于value2或大于value3，则返回TRUE。
string1 LIKE string2	如果string1匹配模式string2，则返回TRUE。
string1 NOT LIKE string2	如果string1与模式string2不匹配，则返回TRUE。
string1 SIMILAR TO string2	如果string1匹配正则表达式string2，则返回TRUE。
string1 NOT SIMILAR TO string2	如果string1与正则表达式string2不匹配，则返回TRUE。
value IN (value [, value]*)	如果表达式存在于给定的表达式列表中，则返回TRUE。这是多个OR条件的简写。如果测试集包含NULL，则如果找不到该元素，则结果为NULL;如果可以找到，则结果为TRUE。如果元素为NULL，则结果始终为NULL。例如“42 IN（1,2,3）”导致FALSE。
value NOT IN (value [, value]*)	如果value不等于列表中的每个值，则返回TRUE。
EXISTS (sub-query)	如果子查询返回至少一行，则返回TRUE。仅在可以在连接和组操作中重写操作时才支持。
value IN (sub-query)	如果value等于子查询返回的行，则返回TRUE。尚未在流式传输环境中支持此操作。
value NOT IN (sub-query)	如果value不等于子查询返回的每一行，则返回TRUE。尚未在流式传输环境中支持此操作。

示例

1. select * from ${t0} where a='beijing'
2. select * from ${t0} where a IS NOT NULL
3. select * from ${t0} where product LIKE '%Rubber%'

逻辑函数

SQL语法	描述
boolean1 OR boolean2	如果boolean1为TRUE或boolean2为TRUE，则返回TRUE。支持三值逻辑。
boolean1 AND boolean2	如果boolean1和boolean2都为TRUE，则返回TRUE。支持三值逻辑。
NOT boolean	如果boolean不为TRUE，则返回TRUE;?如果布尔值为UNKNOWN，则返回UNKNOWN。
boolean IS FALSE	如果boolean为FALSE，则返回TRUE;?否则返回TRUE。如果布尔值为UNKNOWN，则返回FALSE。
boolean IS NOT FALSE	如果boolean不为FALSE，则返回TRUE;否则返回TRUE。如果布尔值为UNKNOWN，则返回TRUE。
boolean IS TRUE	如果boolean为TRUE，则返回TRUE;如果布尔值为UNKNOWN，则返回FALSE。

示例

1. select * from ${t0} where a IS NOT NULL AND b IS NOT NULL

算术函数

SQL语法	描述
+ numeric	返回数字。
- numeric	返回负数字。
numeric1 + numeric2	返回numeric1加上numeric2。
numeric1 - numeric2	返回numeric1减去numeric2。
numeric1 * numeric2	返回numeric1乘以numeric2。
numeric1 / numeric2	返回numeric1除以numeric2。
POWER(numeric1, numeric2)	返回numeric1上升到numeric2的幂。
ABS(numeric)	返回numeric的绝对值。
MOD(numeric1, numeric2)	返回numeric1的余数（模数）除以numeric2。仅当numeric1为负数时，结果才为负数。
SQRT(numeric)	返回数字的平方根。
LN(numeric)	返回的自然对数（以e为底）的数字。
LOG10(numeric)	返回数字的基数10对数。
EXP(numeric)	返回e提升到数字的幂。

CEIL(numeric)	将数字向上舍入，并返回大于或等于numeric的最小数字。
FLOOR(numeric)	将数字向下舍入，并返回小于或等于numeric的最大数字。
SIN(numeric)	计算给定数字的正弦值。
COS(numeric)	计算给定数字的余弦值。
TAN(numeric)	计算给定数字的正切值。
COT(numeric)	计算给定数字的余切值。
ASIN(numeric)	计算给定数字的反正弦值。
ACOS(numeric)	计算给定数字的反余弦值。
ATAN(numeric)	计算给定数字的反正切值。
DEGREES(numeric)	将数字从弧度转换为度数。
RADIANS(numeric)	将数字从度数转换为弧度。
SIGN(numeric)	计算给定数字的符号。
ROUND(numeric, int)	将给定数到整数位权小数点。
PI()	返回比pi更接近任何其他值的值。
E()	返回比e更接近任何其他值的值。
RAND()	返回介于0.0（包括）和1.0（不包括）之间的伪随机双精度值。
RAND(seed integer)	返回0.0（包括）和1.0（不包括）初始种子之间的伪随机双精度值。如果两个RAND函数具有相同的初始种子，则它们将返回相同的数字序列。
RAND_INTEGER(bound integer)	返回介于0.0（包括）和指定值（不包括）之间的伪随机整数值。
RAND_INTEGER(seed integer, bound integer)	返回0.0（包括）之间的伪随机整数值和具有初始种子的指定值（不包括）。如果两个RAND_INTEGER函数具有相同的初始种子和相同的绑定，则它们将返回相同的数字序列。
LOG(x numeric), LOG(base numeric, x numeric)	返回指定数量的指定基数的自然对数。如果使用一个参数调用，则此函数返回自然对数x。如果使用两个参数调用，则此函数返回x基数的对数b。x必须大于0.?b必须大于1。

示例

1. select ceil(a) as ca from ${t0}

字符串函数

SQL语法	描述

string1||string1	连接两个字符串。
CHAR_LENGTH(string)	返回字符串中的字符数。
UPPER(string)	返回转换为大写的字符串。
LOWER(string)	返回转换为小写的字符串。
POSITION(string1 IN string2)	返回第一次出现的位置字符串1的字符串2。
TRIM({ BOTH,LEADING,TRAILING } string1 FROM string2)	从string2中删除前导和/或尾随字符。默认情况下，两侧的空格都被删除。
OVERLAY(string1 PLACING string2 FROM integer [FOR integer2])	用string2替换string1的子字符串。
SUBSTRING(string FROM integer)	返回从给定点开始的字符串的子字符串。
SUBSTRING(string FROM integer FOR integer)	返回从具有给定长度的给定点开始的字符串的子字符串。
INITCAP(string)	返回字符串，每个字转换器的第一个字母为大写，其余为小写。单词是由非字母数字字符分隔的字母数字字符序列。
CONCAT(string1, string2,...)	返回连接参数产生的字符串。如果任何参数为NULL，则返回NULL。例如CONCAT("AA", "BB", "CC")返回AABBCC。
CONCAT_WS(separator, string1, string2,...)	返回使用分隔符连接参数产生的字符串。在要连接的字符串之间添加分隔符。返回NULL如果分隔符为NULL。CONCAT_WS（）不会跳过空字符串。但是，它会跳过任何NULL参数。例如CONCAT_WS("", "AA", "BB", "", "CC")返回AABB~~CC
BOOLEAN REGEXP(String str, String pattern)	指定str的字符串是否匹配指定的pattern进行正则匹配,str或者pattern为空或NULL返回false。
String REGEXP_REPLACE(String str, String pattern, String replacement)	用字符串replacement替换字符串str中正则模式为pattern的子串，返回新的字符串。正在匹配替换, 参数为null或者正则不合法返回null。
String REGEXP_EXTRACT(String str, String pattern, INT index)	使用正则模式pattern匹配抽取字符串str中的第index个子串，index 从1开始正在匹配提取, 参数为null或者正则不合法返回null。

示例

1. select substring(a from 2) as sub_a from ${t0}

条件函数

SQL语法	描述
CASE value WHEN value1 [, value11] *THEN result1 [WHEN valueN [, valueN1]* THEN resultN]* [ELSE resultZ] END	简单的案例。
CASE WHEN condition1 THEN result1 [WHEN conditionN THEN resultN]* [ELSE resultZ] END	搜索案例。
NULLIF(value, value)	如果值相同，则返回NULL。例如，NULLIF(5, 5)返回NULL; NULLIF(5, 0)返回5。
COALESCE(value, value [, value]*)	如果第一个值为null，则提供值。例如，COALESCE(NULL, 5)返回5。

示例

1. SELECT case f1 when 0 then 'zero' when 1 then 'one' when 2 then 'two' else 'other' end as f2 FROM ${t0}

类型转换功能

SQL语法	描述
CAST(value AS type)	将值转换为给定类型。type的取值包括：VARCHAR, BOOLEAN, TINYINT, SMALLINT, INTEGER/INT,, BIGINT, REAL/FLOAT, DOUBLE, DECIMAL, DATE, TIME, TIMESTAMP(3)

时间函数

SQL语法	描述
DATE string	将“yy-mm-dd”形式的日期字符串解析为SQL日期。
TIME string	将“hh：mm：ss”形式的时间字符串解析为SQL时间。
TIMESTAMP string	将“yy-mm-dd hh：mm：ss.fff”形式的时间戳字符串解析为SQL时间戳。
INTERVAL string range	对于SQL间隔为毫秒，以“dd hh：mm：ss.fff”形式解析间隔字符串，对于SQL间隔月，解析“yyyy-mm”。的间隔范围可以是例如DAY，MINUTE，DAY TO HOUR，或DAY TO SECOND的毫秒时间间隔; YEAR或YEAR TO MONTH间隔数月。例如INTERVAL '10 00:00:00.004' DAY TO SECOND，INTERVAL '10' DAY或INTERVAL '2-10' YEAR TO MONTH返回间隔。
CURRENT_DATE	以UTC时区返回当前SQL日期。
CURRENT_TIME	以UTC时区返回当前SQL时间。
CURRENT_TIMESTAMP	以UTC时区返回当前SQL时间戳。

LOCALTIME	返回本地时区的当前SQL时间。
LOCALTIMESTAMP	返回本地时区的当前SQL时间戳。
EXTRACT(timeintervalunit FROM temporal)	提取时间点或时间间隔的部分内容。将部件作为long值返回。例如，EXTRACT(DAY FROM DATE '2006-06-05')导致5。
FLOOR(timepoint TO timeintervalunit)	将时间点向下舍入到给定单位。例如，FLOOR(TIME '12:44:31' TO MINUTE)导致12:44:00。
CEIL(timepoint TO timeintervalunit)	将时间点舍入到给定单位。例如，CEIL(TIME '12:44:31' TO MINUTE)导致12:45:00。
QUARTER(date)	返回SQL日期的一年中的四分之一。例如，QUARTER(DATE '1994-09-27')导致3。
(timepoint, temporal) OVERLAPS (timepoint, temporal)	确定两个锚定时间间隔是否重叠。时间点和时间被转换为由两个时间点（开始，结束）定义的范围。该函数评估leftEnd >= rightStart && rightEnd >= leftStart。例如，(TIME '2:55:00', INTERVAL '1' HOUR) OVERLAPS (TIME '3:30:00', INTERVAL '2' HOUR)导致真实; (TIME '9:00:00', TIME '10:00:00') OVERLAPS (TIME '10:15:00', INTERVAL '3' HOUR)导致错误。
DATE_FORMAT(timestamp, format) (备注：该函数用的是GMT-0时区而不是本地时区)	timestamp使用指定format字符串格式化为字符串。格式必须与date_parse函数使用的MySQL日期格式语法兼容。格式规范在下面的日期格式说明表中给出。例如DATE_FORMAT(ts, '%Y, %d %M')，字符串格式为的结果"2017, 05 May"。
DATE_FORMAT_LTZ(timestamp, format) (备注：该函数用的是本地时区)	timestamp使用指定format字符串格式化为字符串。格式必须与date_parse函数使用的MySQL日期格式语法兼容。格式规范在下面的日期格式说明表中给出。例如DATE_FORMAT(ts, '%Y, %d %M')，字符串格式为的结果"2017, 05 May"。
TIMESTAMPADD(unit, interval, timestamp)	将（带符号）整数间隔添加到时间戳。某一间隔的单元由单元参数，它应为以下值中的一个给定的：SECOND，MINUTE，HOUR，DAY，WEEK，MONTH，QUARTER，或YEAR。例如，TIMESTAMPADD(WEEK, 1, '2003-01-02')导致2003-01-09。
String DATE_SUB(String startdate, INT days);String DATE_SUB(TIMESTAMP time, INT days)	为日期减去天数，日期格式可以是yyyy-MM-dd hh:mm:ss或yyyy-MM-dd或TIMESTAMP，返回String格式的日期yyyy-MM-dd，若有参数为null或解析错误，返回null。
String DATE_ADD(String startdate, INT days);String DATE_ADD(TIMESTAMP time, INT days)	返回指定startdate日期间隔后days天数的一个全新的String类型日期，日期格式可以是yyyy-MM-dd hh:mm:ss或yyyy-MM-dd或timestamp，返回string格式的日期yyyy-MM-dd，若有参数为null或解析错误，返回null。

INT DATEDIFF(String enddate, String startdate);INT DATEDIFF(TIMESTAMP enddate, String startdate);INT DATEDIFF(String enddate, TIMESTAMP startdate);INT DATEDIFF(TIMESTAMP enddate, TIMESTAMP startdate)	计算从enddate到startdate两个时间的天数差值，日期格式可以是yy-MM-dd HH:mm:ss或yy-MM-dd或timestamp，返回整数，若有参数为null或解析错误，返回null
TIMESTAMP TO_TIMESTAMP(BIGINT time);TIMESTAMP TO_TIMESTAMP(String date);TIMESTAMP TO_TIMESTAMP(String date, String format)	将bigint类型的日期或者String类型的日期转换成TimeStamp类型。
BIGINT UNIX_TIMESTAMP();BIGINT UNIX_TIMESTAMP(String date);BIGINT UNIX_TIMESTAMP(String date, String format)	两个参数，均为可选，无参数时返回当前时间的时间戳，单位为秒，第一个参数是字符串类型的时间，第二个参数是时间的格式，默认为yyyy-MM-dd hh:mm:ss，返回值是第一个参数转换成的长整型的时间戳，单位为秒，若有参数为null或解析错误，返回null。
String FROM_UNIXTIME(BIGINT unixtime [String format])	第一个参数unixtime为长整型，是以秒为单位的时间戳，第二个参数format可选，为日期格式，默认为yyyy-MM-dd HH:mm:ss，返回String类型的符合指定格式的日期，若有参数为null或解析错误，返回null。返回值为String类型的日期值, 默认日期格式：yyyy-MM-dd HH:mm:ss，若指定日期格式按指定格式输出 任一输入参数是NULL，返回NULL。
BIGINT NOW()	返回当前时区时间的时间戳，单位为秒，可以接受一个整型参数作为偏移值（单位：秒）

示例

1. select CURRENT_TIMESTAMP as ts from ${t0}
2. 获取当前时间，按指定格式输出为字符串： select FROM_UNIXTIME(NOW(), 'yyyy-MM-dd HH:mm:ss') as timestamp_str from ${t0}
3. 将timestamp格式化为字符串： select DATE_FORMAT_LTZ(gmt_create, 'yyyy-MM-dd HH:mm:ss') as timestamp_str from ${t0}

聚合函数

SQL语法	描述
COUNT(value [, value]*)	返回值不为null 的输入行数。
COUNT(*)	返回输入行数。
AVG(value)	返回所有输入值的数值的平均值（算术平均值）。

SUM(value)	返回所有输入值的数字总和。
MAX(value)	返回的最大值值在所有的输入值。
MIN(value)	返回的最小值的值在所有的输入值。
STDDEV_POP(value)	返回所有输入值的数字字段的总体标准偏差。
STDDEV_SAMP(value)	返回所有输入值的数字字段的样本标准偏差。
VAR_POP(value)	返回所有输入值中数字字段的总体方差（总体标准差的平方）。
VAR_SAMP(value)	返回所有输入值的数值字段的样本方差（样本标准差的平方）。
COLLECT(value)	返回值 s 的多集。null输入值将被忽略。如果仅添加空值，则返回空的multiset。
CONCAT_AGG(value, sep)	sep是分隔符，用指定的spearator做分隔符，连接value中的值。

示例

1. select class, max(sepalwidth) as max_sepalwidth from ${t0} group by class

分组函数

SQL语法	描述
GROUP_ID()	返回唯一标识分组键组合的整数。
GROUPING(expression)	如果表达式在当前行的分组集中汇总，则返回1，否则返回0。
GROUPING_ID(expression [, expression]*)	返回给定分组表达式的位向量。

访问函数

SQL语法	描述
tableName.compositeType.field	按名称访问Flink复合类型（如Tuple，POJO等）的字段并返回其值。
tableName.compositeType.*	将Flink复合类型（例如Tuple，POJO等）及其所有直接子类型转换为平面表示形式，其中每个子类型都是单独的字段。

数组函数

SQL语法	描述
ARRAY ‘[’ value [, value]* ‘]’	从值列表创建数组。

CARDINALITY(ARRAY)	返回数组的元素数。
array '[' index ']'	返回数组中特定位置的元素。指数从1开始。
ELEMENT(ARRAY)	返回具有单个元素的数组的唯一元素。null如果数组为空，则返回。如果数组有多个元素，则抛出异常。

保留关键字

虽然并非每个SQL功能都已实现，但某些字符串组合已被保留为关键字以供将来使用。如果要将以下字符串之一用作字段名称，请确保使用反引号将其包围（例如value，count）。A, ABS, ABSOLUTE, ACTION, ADA, ADD, ADMIN, AFTER, ALL, ALLOCATE, ALLOW, ALTER, ALWAYS, AND, ANY, ARE, ARRAY, AS, ASC, ASENSITIVE, ASSERTION, ASSIGNMENT, ASYMMETRIC, AT, ATOMIC, ATTRIBUTE, ATTRIBUTES, AUTHORIZATION, AVG, BEFORE, BEGIN, BERNOULLI, BETWEEN, BIGINT, BINARY, BIT, BLOB, BOOLEAN, BOTH, BREADTH, BY, C, CALL, CALLED, CARDINALITY, CASCADE, CASCADED, CASE, CAST, CATALOG, CATALOG_NAME, CEIL, CEILING, CENTURY, CHAIN, CHAR, CHARACTER, CHARACTERISTICTS, CHARACTERS, CHARACTER_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_NAME, CHARACTER_SET_SCHEMA, CHAR_LENGTH, CHECK, CLASS_ORIGIN, CLOB, CLOSE, COALESCE, COBOL, COLLATE, COLLATION, COLLATION_CATALOG, COLLATION_NAME, COLLATION_SCHEMA, COLLECT, COLUMN, COLUMN_NAME, COMMAND_FUNCTION, COMMAND_FUNCTION_CODE, COMMIT, COMMITTED, CONDITION, CONDITION_NUMBER, CONNECT, CONNECTION, CONNECTION_NAME, CONSTRAINT, CONSTRAINTS, CONSTRAINT_CATALOG, CONSTRAINT_NAME, CONSTRAINT_SCHEMA, CONSTRUCTOR, CONTAINS, CONTINUE, CONVERT, CORR, CORRESPONDING, COUNT, COVAR_POP, COVAR_SAMP, CREATE, CROSS, CUBE, CUME_DIST, CURRENT, CURRENT_CATALOG, CURRENT_DATE, CURRENT_DEFAULT_TRANSFORM_GROUP, CURRENT_PATH, CURRENT_ROLE, CURRENT_SCHEMA, CURRENT_TIME, CURRENT_TIMESTAMP, CURRENT_TRANSFORM_GROUP_FOR_TYPE, CURRENT_USER, CURSOR, CURSOR_NAME, CYCLE, DATA, DATABASE, DATE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, DAY, DEALLOCATE, DEC, DECADE, DECIMAL, DECLARE, DEFAULT, DEFAULTS, DEFERRABLE, DEFERRED, DEFINED, DEFINER, DEGREE, DELETE, DENSE_RANK, DEPTH, DEREF, DERIVED, DESC, DESCRIBE, DESCRIPTION, DESCRIPTOR, DETERMINISTIC, DIAGNOSTICS, DISALLOW, DISCONNECT, DISPATCH, DISTINCT, DOMAIN, DOUBLE, DOW, DOY, DROP, DYNAMIC, DYNAMIC_FUNCTION, DYNAMIC_FUNCTION_CODE, EACH, ELEMENT, ELSE, END, END-EXEC, EPOCH, EQUALS, ESCAPE, EVERY, EXCEPT, EXCEPTION, EXCLUDE, EXCLUDING, EXEC, EXECUTE, EXISTS, EXP, EXPLAIN, EXTEND, EXTERNAL, EXTRACT, FALSE, FETCH, FILTER, FINAL, FIRST, FIRST_VALUE, FLOAT, FLOOR, FOLLOWING, FOR, FOREIGN, FORTRAN, FOUND, FRAC_SECOND, FREE, FROM, FULL, FUNCTION, FUSION, G, GENERAL, GENERATED, GET, GLOBAL, GO, GOTO, GRANT, GRANTED, GROUP, GROUPING, HAVING, HIERARCHY, HOLD, HOUR, IDENTITY, IMMEDIATE, IMPLEMENTATION, IMPORT, IN, INCLUDING, INCREMENT, INDICATOR, INITIALLY, INNER, INOUT, INPUT, INSENSITIVE, INSERT, INSTANCE, INSTANTIABLE, INT, INTEGER, INTERSECT, INTERSECTION, INTERVAL, INTO, INVOKER, IS, ISOLATION, JAVA, JOIN, K, KEY, KEY_MEMBER, KEY_TYPE, LABEL, LANGUAGE, LARGE, LAST, LAST_VALUE, LATERAL, LEADING, LEFT, LENGTH, LEVEL, LIBRARY, LIKE, LIMIT, LN, LOCAL, LOCALTIME, LOCALTIMESTAMP, LOCATOR, LOWER, M, MAP, MATCH, MATCHED, MAX, MAXVALUE, MEMBER, MERGE, MESSAGE_LENGTH, MESSAGE_OCTET_LENGTH, MESSAGE_TEXT, METHOD, MICROSECOND, MILLENNIUM, MIN, MINUTE, MINVALUE, MOD, MODIFIES, MODULE, MONTH, MORE, MULTISET, MUMPS, NAME, NAMES, NATIONAL, NATURAL, NCHAR, NCLOB, NESTING, NEW, NEXT, NO, NONE, NORMALIZE, NORMALIZED, NOT, NULL, NULLABLE, NULLIF, NULLS, NUMBER, NUMERIC, OBJECT, OCTETS, OCTET_LENGTH, OF, OFFSET, OLD, ON, ONLY, OPEN, OPTION, OPTIONS, OR, ORDER, ORDERING, ORDINALITY, OTHERS, OUT, OUTER, OUTPUT, OVER, OVERLAPS,

OVERLAY, OVERRIDING, PAD, PARAMETER, PARAMETER_MODE, PARAMETER_NAME, PARAMETER_ORDINAL_POSITION, PARAMETER_SPECIFIC_CATALOG, PARAMETER_SPECIFIC_NAME, PARAMETER_SPECIFIC_SCHEMA, PARTIAL, PARTITION, PASCAL, PASSTHROUGH, PATH, PERCENTILE_CONT, PERCENTILE_DISC, PERCENT_RANK, PLACING, PLAN, PLI, POSITION, POWER, PRECEDING, PRECISION, PREPARE, PRESERVE, PRIMARY, PRIOR, PRIVILEGES, PROCEDURE, PUBLIC, QUARTER, RANGE, RANK, READ, READS, REAL, RECURSIVE, REF, REFERENCES, REFERENCING, REGR_AVGX, REGR_AVGY, REGR_COUNT, REGR_INTERCEPT, REGR_R2, REGR_SLOPE, REGR_SXX, REGR_SXY, REGR_SYY, RELATIVE, RELEASE, REPEATABLE, RESET, RESTART, RESTRICT, RESULT, RETURN, RETURNED_CARDINALITY, RETURNED_LENGTH, RETURNED_OCTET_LENGTH, RETURNED_SQLSTATE, RETURNS, REVOKE, RIGHT, ROLE, ROLLBACK, ROLLUP, ROUTINE, ROUTINE_CATALOG, ROUTINE_NAME, ROUTINE_SCHEMA, ROW, ROWS, ROW_COUNT, ROW_NUMBER, SAVEPOINT, SCALE, SCHEMA, SCHEMA_NAME, SCOPE, SCOPE_CATALOGS, SCOPE_NAME, SCOPE_SCHEMA, SCROLL, SEARCH, SECOND, SECTION, SECURITY, SELECT, SELF, SENSITIVE, SEQUENCE, SERIALIZABLE, SERVER, SERVER_NAME, SESSION, SESSION_USER, SET, SETS, SIMILAR, SIMPLE, SIZE, SMALLINT, SOME, SOURCE, SPACE, SPECIFIC, SPECIFICTYPE, SPECIFIC_NAME, SQL, SQLEXCEPTION, SQLSTATE, SQLWARNING, SQL_TSI_DAY, SQL_TSI_FRAC_SECOND, SQL_TSI_HOUR, SQL_TSI_MICROSECOND, SQL_TSI_MINUTE, SQL_TSI_MONTH, SQL_TSI_QUARTER, SQL_TSI_SECOND, SQL_TSI_WEEK, SQL_TSI_YEAR, SQRT, START, STATE, STATEMENT, STATIC, STDDEV_POP, STDDEV_SAMP, STREAM, STRUCTURE, STYLE, SUBCLASS_ORIGIN, SUBMULTISET, SUBSTITUTE, SUBSTRING, SUM, SYMMETRIC, SYSTEM, SYSTEM_USER, TABLE, TABLESAMPLE, TABLE_NAME, TEMPORARY, THEN, TIES, TIME, TIMESTAMP, TIMESTAMPADD, TIMESTAMPDIFF, TIMEZONE_HOUR, TIMEZONE_MINUTE, TINYINT, TO, TOP_LEVEL_COUNT, TRAILING, TRANSACTION, TRANSACTIONS_ACTIVE, TRANSACTIONS_COMMITTED, TRANSACTIONS_ROLLED_BACK, TRANSFORM, TRANSFORMS, TRANSLATE, TRANSLATION, TREAT, TRIGGER, TRIGGER_CATALOG, TRIGGER_NAME, TRIGGER_SCHEMA, TRIM, TRUE, TYPE, UESCAPE, UNBOUNDED, UNCOMMITTED, UNDER, UNION, UNIQUE, UNKNOWN, UNNAMED, UNNEST, UPDATE, UPPER, UPSERT, USAGE, USER, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_CODE, USER_DEFINED_TYPE_NAME, USER_DEFINED_TYPE_SCHEMA, USING, VALUE, VALUES, VARBINARY, VARCHAR, VARYING, VAR_POP, VAR_SAMP, VERSION, VIEW, WEEK, WHEN, WHENEVER, WHERE, WIDTH_BUCKET, WINDOW, WITH, WITHIN, WITHOUT, WORK, WRAPPER, WRITE, XML, YEAR, ZONE

日期格式说明符

日期格式说明符	描述
%a	缩写的工作日名称（Sun.. Sat）
%b	缩写的月份名称（Jan.. Dec）
%c	月，数字（1.. 12）
%D	这个月的一天，英语后缀（0th，1st，2nd，3rd，...）
%d	每月的某天，数字（01.. 31）
%e	每月的某天，数字（1.. 31）
%f	第二个分数（打印6位数：000000... 999000;解析时为1 - 9位数：0..999999999）（时间戳被截断为毫秒。）

%H	小时（00.. 23）
%h	小时（01.. 12）
%I	小时（01.. 12）
%i	分钟，数字（00.. 59）
%j	一年中的某一天（001.. 366）
%k	小时（0.. 23）
%l	小时（1.. 12）
%M	月份名称（January.. December）
%m	月，数字（01.. 12）
%p	AM 要么 PM
%r	时间，12小时（hh:mm:ss其次是AM或PM）
%S	秒（00... 59）
%s	秒（00... 59）
%T	时间，24小时（hh:mm:ss）
%U	周（00.. 53），周日是一周的第一天
%u	周（00.. 53），周一是一周的第一天
%V	周（01.. 53），周日是一周的第一天; 用于%X
%v	周（01.. 53），周一是一周的第一天; 用于%x
%W	平日名称（Sunday.. Saturday）
%w	星期几（0.. 6），星期日是一周的第一天
%X	星期日是星期的第一天的星期，数字，四位数; 用于%V
%x	一周的年份，星期一是一周的第一天，数字，四位数; 用于%v
%Y	年份，数字，四位数
%y	年份，数字（两位数）
%%	文字%字符
%x	x，对于x上面未列出的任何内容

常见错误

语句	错误信息	错误原因	解决办法

select average_pay, product_time as time, province from ${t0}	"实验运行失败Invalid Node[SqlCmd]: SQL parse failed. Encountered "as time" at line 1, column 29. Was expecting one of: "ORDER" ... "LIMIT" ... "OFFSET" ... "FETCH" ... "FROM" ... "," ... "AS" ... "AS" ... "AS" ... "AS" ... "AS" ... "." ... "NOT" ... "IN" ... "<" ... "<=" ... ">" ... ">=" ... "=" ... "<>" ... "!=" ... "BETWEEN" ... "LIKE" ... "SIMILAR" ... "+" ... "-" ... "*" ... "/" ... "%" ... "||" ... "AND" ... "OR" ... "IS" ... "MEMBER" ... "SUBMULTISET" ... "CONTAINS" ... "OVERLAPS" ... "EQUALS" ... "PRECEDES" ... "SUCCEEDS" ... "IMMEDIATELY" ... "MULTISET" ... "[" ... "UNION" ... "INTERSECT" ... "EXCEPT" ... "MINUS" ."	列名'time'与Flink保留字冲突	用反引号引用，如: select average_pay, product_time as time, province from ${t0}
if a < 0.5		不支持if	需要换成case when

参数说明

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])

data_op = BatchOperator.fromDataframe(df, schemaStr='f1 string, f2 bigint, f3
double')
result_op = SqlCmdBatchOp().setCommand("select * from ${t0} where
f1='Ohio'").linkFrom(data_op)
result_op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.SqlCmdBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SqlCmdBatchOppTest {
    @Test
    public void testSqlCmdBatchOp() throws Exception{
        List <Row> df = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nev", 2003, 3.2)
        );
        MemSourceBatchOp dataOp = new MemSourceBatchOp(df, "f1 string, f2 int,
f3 double");
        new SqlCmdBatchOp()
            .setCommand("select * from ${t0} where f1='Ohio'")
            .linkFrom(dataOp)
            .print();
    }
}
```

运行结果

f1	f2	f3
Ohio	2000	1.5000
Ohio	2001	1.7000
Ohio	2002	3.6000

SQL操作：UnionAll (UnionAllBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.UnionAllBatchOp

Python 类名：UnionAllBatchOp

功能介绍

对批式数据进行sql的UNION ALL操作。(不去重)

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df1 = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2000, 1.5],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])
df2 = pd.DataFrame([
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])

batch_data1 = BatchOperator.fromDataframe(df1, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data2 = BatchOperator.fromDataframe(df2, schemaStr='f1 string, f2 bigint,
f3 double')

UnionAllBatchOp().linkFrom(batch_data1, batch_data2).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.UnionBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class UnionAllBatchOpTest {
    @Test
    public void testUnionAllBatchOp() throws Exception {
        List <Row> df1 = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        List <Row> df2 = Arrays.asList(
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data1 = new MemSourceBatchOp(df1, "f1 string, f2 int,
f3 double");
        BatchOperator <?> data2 = new MemSourceBatchOp(df2, "f1 string, f2 int,
f3 double");
        BatchOperator <?> unionAll = new UnionAllBatchOp();
        unionAll.linkFrom(data1, data2).print();
    }
}
```

运行结果

f1	f2	f3
Ohio	2000	1.5000
Ohio	2000	1.5000
Ohio	2002	3.6000
Nevada	2001	2.4000

Nevada	2002	2.9000
Nevada	2003	3.2000
Nevada	2001	2.4000
Nevada	2003	3.2000

SQL操作：Union (UnionBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.UnionBatchOp

Python 类名：UnionBatchOp

功能介绍

对批式数据进行sql的UNION操作（去重）。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df1 = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2000, 1.5],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])
df2 = pd.DataFrame([
    ['Nevada', 2001, 2.4],
    ['Nevada', 2003, 3.2]
])

batch_data1 = BatchOperator.fromDataframe(df1, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data2 = BatchOperator.fromDataframe(df2, schemaStr='f1 string, f2 bigint,
f3 double')

UnionBatchOp().linkFrom(batch_data1, batch_data2).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.UnionBatchOp;
import org.junit.Test;

public class UnionBatchOpTest {
    @Test
    public void testUnionBatchOp() throws Exception {
        List <Row> df1 = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        List <Row> df2 = Arrays.asList(
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data1 = new MemSourceBatchOp(df1, "f1 string, f2 int,
f3 double");
        BatchOperator <?> data2 = new MemSourceBatchOp(df2, "f1 string, f2 int,
f3 double");
        BatchOperator <?> union = new UnionBatchOp();
        union.linkFrom(data1, data2).print();
    }
}
```

运行结果

f1	f2	f3
Nevada	2001	2.4000
Nevada	2002	2.9000
Nevada	2003	3.2000
Ohio	2000	1.5000
Ohio	2002	3.6000

SQL操作：Where (WhereBatchOp)

Java 类名：com.alibaba.alink.operator.batch.sql.WhereBatchOp

Python 类名：WhereBatchOp

功能介绍

对批式数据进行sql的WHERE操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
clause	运算语句	运算语句	String	✓		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['Ohio', 2000, 1.5],
    ['Ohio', 2001, 1.7],
    ['Ohio', 2002, 3.6],
    ['Nevada', 2001, 2.4],
    ['Nevada', 2002, 2.9],
    ['Nevada', 2003, 3.2]
])
batch_data = BatchOperator.fromDataframe(df, schemaStr='f1 string, f2 bigint,
f3 double')
batch_data.link(WhereBatchOp().setClause("f1='Ohio'"))
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.WhereBatchOp;
```

```
import org.junit.Test;

public class WhereBatchOpTest {
    @Test
    public void testWhereBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("Ohio", 2000, 1.5),
            Row.of("Ohio", 2001, 1.7),
            Row.of("Ohio", 2002, 3.6),
            Row.of("Nevada", 2001, 2.4),
            Row.of("Nevada", 2002, 2.9),
            Row.of("Nevada", 2003, 3.2)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f1 string, f2 int,
f3 double");
        data.link(new WhereBatchOp().setClause("f1='Ohio'")).print();
    }
}
```

运行结果

f1	f2	f3
Ohio	2000	1.5000
Ohio	2001	1.7000
Ohio	2002	3.6000

二值化 (BinarizerBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.BinarizerBatchOp

Python 类名：BinarizerBatchOp

功能介绍

给定一个阈值，将连续变量二值化（大于等于阈值转为1，小于阈值转为0）。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
threshold	二值化阈值	二值化阈值	Double			0.0
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.1, True, "2", "A"],
    [1.1, False, "2", "B"],
    [1.1, True, "1", "B"],
    [2.2, True, "1", "A"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='double double, bool boolean,
number int, str string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='double double, bool
boolean, number int, str string')

binarizer = BinarizerBatchOp().setSelectedCol("double").setThreshold(2.0)
binarizer.linkFrom(inOp1).print()

binarizer = BinarizerStreamOp().setSelectedCol("double").setThreshold(2.0)
binarizer.linkFrom(inOp2).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.BinarizerBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.feature.BinarizerStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class BinarizerBatchOpTest {
    @Test
    public void testBinarizerBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.1, true, 2, "A"),
            Row.of(1.1, false, 2, "B"),
```

```
            Row.of(1.1, true, 1, "B"),
            Row.of(2.2, true, 1, "A")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "double double, bool
boolean, number int, str string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "double double,
bool boolean, number int, str string");
        BatchOperator <?> binarizer = new
BinarizerBatchOp().setSelectedCol("double").setThreshold(2.0);
        binarizer.linkFrom(inOp1).print();
        StreamOperator <?> binarizer2 = new
BinarizerStreamOp().setSelectedCol("double").setThreshold(2.0);
        binarizer2.linkFrom(inOp2).print();
        StreamOperator.execute();
    }
}
```

运行结果

输出数据

double	bool	number	str
0.0000	true	2	A
0.0000	false	2	B
0.0000	true	1	B
1.0000	true	1	A

分箱预测 (BinningPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.BinningPredictBatchOp

Python 类名：BinningPredictBatchOp

功能介绍

使用等频，等宽或自动分箱的方法对数据进行分箱操作，输入为连续或离散的特征数据，输出为每个特征的分箱规则，在分箱组件中点键右键选择**我要分箱**，可以对分箱结果进行查看和编辑。

该组件为分箱预测批处理组件，在预测时需要指定编码方法，有WOE（直接输出bin的WOE值）、INDEX（输出bin的索引值）、 VECTOR（只有当前bin的索引值非0的向量）、ASSEMBLED_VECTOR（如果是多列，首先转换为VECTOR，输出多列的组合VECTOR）。

算法简介

信用评分卡模型在国外是一种成熟的预测方法，尤其在信用风险评估以及金融风险控制领域更是得到了比较广泛的使用，其原理是将模型变量WOE编码方式离散化之后运用logistic回归模型进行建模，其中本文介绍的分箱算法就是这里说的WOE编码过程。

分箱通常是指对连续变量进行区间划分，将连续变量划分成几个区间变量，主要目的是为了避免“过拟合”，使评分结果更具有稳健性和预测性。这里有个典型的例子就是：用决策树进行评分看起来效果不错，但往往都会因为”过拟合”，模型无法实际应用。

分箱支持等频、等宽两种方法，输入为连续或离散的特征数据，输出为每个特征的分箱规则，分箱结果支持合并、拆分等。

算法原理

分箱方法

等频分箱

等频分箱是对特征数据进行排序，按分位点的方式选取用户指定的N个分位点作为分箱边界，若相邻分位点相同则将两个分箱合并，因此分箱结果中有可能少于用户指定的分箱个数。

等宽分箱

等宽分箱是对特征数据按最大值和最小值等平均分成N份，在每一等份的边界作为分箱的边界。

离散变量分箱

字符串类型变量即为离散变量，离散变量的分箱不使用上述的三种分箱方法，对于离散变量的每一种取值会单独分成一个分箱，仅当某个取值小于用户指定的最小阈值时，该取值会被统一归入ELSE分箱中。

相关公式

WOE的计算公式如下:

$$ WOE_i=ln(\dfrac{py_i}{pn_i}) $$

其中，py_i是这个组中正样本个数占所有正样本个数的比例，pn_i是这个组中负样本个数占所有负样本个数的比例。

IV的计算公式如下：

$$ IV_i=(py_i-pn_i)*WOE_i $$

实际使用时，一般用IV评估变量的重要性，IV越大，变量对模型的影响越大，在单个变量内部，再利用WOE判断每个bin的区分度如何，WOE越大，区分度越大。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
defaultWoe	默认Woe，在woe为Nan或NULL时替换	默认Woe，在woe为Nan或NULL时替换	Double			NaN
dropLast	是否删除最后一个元素	删除最后一个元素是为了保证线性无关性。默认true	Boolean			true
encode	编码方法	编码方法	String		"WOE", "VECTOR", "ASSEMBLED_VECTOR", "INDEX"	"ASS
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替，"skip"表示补null，"error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEE

modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, 1.0, True, 0, "A", 1],
    [1, 2.1, False, 2, "B", 1],
    [2, 1.1, True, 3, "C", 1],
    [3, 2.2, True, 1, "E", 0],
    [4, 0.1, True, 2, "A", 0],
    [5, 1.5, False, -4, "D", 1],
    [6, 1.3, True, 1, "B", 0],
    [7, 0.2, True, -1, "A", 1],
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int,f0 double, f1
boolean, f2 int, f3 string, label int')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id int,f0 double, f1
boolean, f2 int, f3 string, label int')

train = BinningTrainBatchOp()\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setLabelCol("label")\
    .setPositiveLabelValueString("1")\
    .linkFrom(inOp1)
```

```
predict = BinningPredictBatchOp()\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setEncode("INDEX")\
    .setReservedCols(["id", "label"])\
    .linkFrom(train, inOp1)

predict.lazyPrint(10)
train.print()

predict = BinningPredictStreamOp(train)\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setEncode("INDEX")\
    .setReservedCols(["id", "label"])\
    .linkFrom(inOp2)

predict.print()
StreamOperator.execute()
```

Java代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.feature.BinningPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

/**
 * Unit test for Binning.
 */
public class BinningTrainBatchOpTest {

    @Test
    public void test() throws Exception {
        List<Row> list = Arrays.asList(
            Row.of(0, 1.0, true, 0, "A", 1),
            Row.of(1, 2.1, false, 2, "B", 1),
            Row.of(2, 1.1, true, 3, "C", 1),
            Row.of(3, 2.2, true, 1, "E", 0),
            Row.of(4, 0.1, true, 2, "A", 0),
```

```
            Row.of(5, 1.5, false, -4, "D", 1),
            Row.of(6, 1.3, true, 1, "B", 0),
            Row.of(7, 0.2, true, -1, "A", 1)
        );

        BatchOperator batchOperator = new MemSourceBatchOp(list, "id int, f0
double, f1 boolean, f2 int, f3 string, label int");
        StreamOperator streamOperator = new MemSourceStreamOp(list, "id int, f0
double, f1 boolean, f2 int, f3 string, label int");

        BatchOperator train = new BinningTrainBatchOp()
            .setSelectedCols("f0", "f1", "f2", "f3")
            .setLabelCol("label")
            .setPositiveLabelValueString("1")
            .linkFrom(batchOperator);

        BatchOperator predict = new BinningPredictBatchOp()
            .setSelectedCols("f0", "f1", "f2", "f3")
            .setReservedCols("id", "label")
            .setEncode("INDEX")
            .linkFrom(train, batchOperator);

        predict.lazyPrint(10);
        train.print();

        StreamOperator predictStream = new BinningPredictStreamOp(train)
            .setSelectedCols("f0", "f1", "f2", "f3")
            .setEncode("INDEX")
            .setReservedCols("id", "label")
            .linkFrom(streamOperator);

        predictStream.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型结果

```
                                FeatureBordersJson
0  [{"binDivideType":"QUANTILE","featureName":"f0...
1  [{"binDivideType":"QUANTILE","featureName":"f2...
2  [{"binDivideType":"DISCRETE","featureName":"f1...
3  [{"binDivideType":"DISCRETE","featureName":"f3...
```

预测结果

id	f0	f1	f2	f3	label
0	0	0	0	0	1
1	1	1	1	1	1
2	0	0	1	2	1
3	1	0	0	4	0
4	0	0	1	0	0
5	1	1	0	3	1
6	0	0	0	1	0
7	0	0	0	0	1

分箱训练 (BinningTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.BinningTrainBatchOp

Python 类名：BinningTrainBatchOp

功能介绍

使用等频，等宽或自动分箱的方法对数据进行分箱操作，输入为连续或离散的特征数据，输出为每个特征的分箱规则。

算法简介

分箱通常是指对连续变量进行区间划分，将连续变量划分成几个区间变量，主要目的是为了避免“过拟合”，使评分结果更具有稳健性和预测性。这里有个典型的例子就是：用决策树进行评分看起来效果不错，但往往都会因为”过拟合”，模型无法实际应用。

分箱支持等频、等宽两种方法，输入为连续或离散的特征数据，输出为每个特征的分箱规则，分箱结果支持合并、拆分等。

算法原理

分箱方法

等频分箱

等频分箱是对特征数据进行排序，按分位点的方式选取用户指定的N个分位点作为分箱边界，若相邻分位点相同则将两个分箱合并，因此分箱结果中有可能少于用户指定的分箱个数。

等宽分箱

等宽分箱是对特征数据按最大值和最小值等平均分成N份，在每一等份的边界作为分箱的边界。

离散变量分箱

字符串类型变量即为离散变量，离散变量的分箱不使用上述的三种分箱方法，对于离散变量的每一种取值会单独分成一个分箱，仅当某个取值小于用户指定的最小阈值时，该取值会被统一归入ELSE分箱中。

相关公式

WOE的计算公式如下:

$$ WOE_i=ln(\dfrac{py_i}{pn_i}) $$

其中，py_i是这个组中正样本个数占所有正样本个数的比例，pn_i是这个组中负样本个数占所有负样本个数的比例。

IV的计算公式如下：

$$ IV_i=(py_i-pn_i)*WOE_i $$

实际使用时，一般用IV评估变量的重要性，IV越大，变量对模型的影响越大，在单个变量内部，再利用WOE判断每个bin的区分度如何，WOE越大，区分度越大。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
binningMethod	连续特征分箱方法	连续特征分箱方法	String		"QUANTILE", "BUCKET"	"QUANTILE
discreteThresholds	离散个数阈值	离散个数阈值，低于该阈值的离散样本将不会单独成一个组别。	Integer			-214748364
discreteThresholdsArray	离散个数阈值	离散个数阈值，每一列对应数组中一个元素。	Integer[]			null
discreteThresholdsMap	离散分箱离散为ELSE的最小阈值, 形式如col0:3,col1:4	离散分箱离散为ELSE的最小阈值，形式如col0:3,col1:4。	String			null

fromUserDefined	是否读取用户自定义JSON	是否读取用户自定义JSON，true则为用户自定义分箱，false则按参数配置分箱。	Boolean			false
labelCol	标签列名	输入表中的标签列名	String			null
leftOpen	是否左开右闭	左开右闭为true，左闭右开为false	Boolean			true
numBuckets	quantile个数	quantile个数，对所有列有效。	Integer			2
numBucketsArray	quantile个数	quantile个数，每一列对应数组中一个元素。	Integer[]			null
numBucketsMap	用户定义的bucket个数，形式如col0:3,col1:4	用户定义的bucket个数，形式如col0:3,col1:4	String			null
positiveLabelValueString	正样本	正样本对应的字符串格式。	String			"1"

userDefinedBin	用户定义的bin的json	用户定义的bin的json	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, 1.0, True, 0, "A", 1],
    [1, 2.1, False, 2, "B", 1],
    [2, 1.1, True, 3, "C", 1],
    [3, 2.2, True, 1, "E", 0],
    [4, 0.1, True, 2, "A", 0],
    [5, 1.5, False, -4, "D", 1],
    [6, 1.3, True, 1, "B", 0],
    [7, 0.2, True, -1, "A", 1],
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int,f0 double, f1
boolean, f2 int, f3 string, label int')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id int,f0 double, f1
boolean, f2 int, f3 string, label int')

train = BinningTrainBatchOp()\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setLabelCol("label")\
    .setPositiveLabelValueString("1")\
    .linkFrom(inOp1)

predict = BinningPredictBatchOp()\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setEncode("INDEX")\
    .setReservedCols(["id", "label"])\
    .linkFrom(train, inOp1)

predict.lazyPrint(10)
train.print()

predict = BinningPredictStreamOp(train)\
```

```
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setEncode("INDEX")\
    .setReservedCols(["id", "label"])\
    .linkFrom(inOp2)

predict.print()
StreamOperator.execute()
```

Java代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.feature.BinningPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

/**
 * Unit test for Binning.
 */
public class BinningTrainBatchOpTest {

    @Test
    public void test() throws Exception {
        List<Row> list = Arrays.asList(
            Row.of(0, 1.0, true, 0, "A", 1),
            Row.of(1, 2.1, false, 2, "B", 1),
            Row.of(2, 1.1, true, 3, "C", 1),
            Row.of(3, 2.2, true, 1, "E", 0),
            Row.of(4, 0.1, true, 2, "A", 0),
            Row.of(5, 1.5, false, -4, "D", 1),
            Row.of(6, 1.3, true, 1, "B", 0),
            Row.of(7, 0.2, true, -1, "A", 1)
        );

        BatchOperator batchOperator = new MemSourceBatchOp(list, "id int, f0
double, f1 boolean, f2 int, f3 string, label int");
        StreamOperator streamOperator = new MemSourceStreamOp(list, "id int, f0
double, f1 boolean, f2 int, f3 string, label int");

        BatchOperator train = new BinningTrainBatchOp()
```

```
            .setSelectedCols("f0", "f1", "f2", "f3")
            .setLabelCol("label")
            .setPositiveLabelValueString("1")
            .linkFrom(batchOperator);

        BatchOperator predict = new BinningPredictBatchOp()
            .setSelectedCols("f0", "f1", "f2", "f3")
            .setReservedCols("id", "label")
            .setEncode("INDEX")
            .linkFrom(train, batchOperator);

        predict.lazyPrint(10);
        train.print();

        StreamOperator predictStream = new BinningPredictStreamOp(train)
            .setSelectedCols("f0", "f1", "f2", "f3")
            .setEncode("INDEX")
            .setReservedCols("id", "label")
            .linkFrom(streamOperator);

        predictStream.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型结果

```
                                FeatureBordersJson
0  [{"binDivideType":"QUANTILE","featureName":"f0...
1  [{"binDivideType":"QUANTILE","featureName":"f2...
2  [{"binDivideType":"DISCRETE","featureName":"f1...
3  [{"binDivideType":"DISCRETE","featureName":"f3...
```

预测结果

id	f0	f1	f2	f3	label
0	0	0	0	0	1
1	1	1	1	1	1
2	0	0	1	2	1
3	1	0	0	4	0
4	0	0	1	0	0

5	1	1	0	3	1
6	0	0	0	1	0
7	0	0	0	0	1

分桶 (BucketizerBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.BucketizerBatchOp

Python 类名：BucketizerBatchOp

功能介绍

给定切分点，将连续变量分桶，需要选择需要进行切分的单列或多列，同时给出选中每列的切分点，每列切分点都是一个double数组，需要严格递增。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
cutsArray	多列的切分点	多列的切分点	double[][]			
dropLast	是否删除最后一个元素	删除最后一个元素是为了保证线性无关性。默认true	Boolean			true
encode	编码方法	编码方法	String		"VECTOR", "ASSEMBLED_VECTOR", "INDEX"	"INDEX"
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP"

leftOpen	是否左开右闭	左开右闭为true，左闭右开为false	Boolean			true
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.1, True, "2", "A"],
    [1.1, False, "2", "B"],
    [1.1, True, "1", "B"],
    [2.2, True, "1", "A"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='double double, bool boolean,
number int, str string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='double double, bool
boolean, number int, str string')

bucketizer =
BucketizerBatchOp().setSelectedCols(["double","number"]).setCutsArray([[1.0,2.0
,2.2,4.0],[0.0,1.1]])
bucketizer.linkFrom(inOp1).print()
```

```
bucketizer =
BucketizerStreamOp().setSelectedCols(["double"]).setCutsArray([[2.0]])
bucketizer.linkFrom(inOp2).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.BucketizerBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.feature.BucketizerStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class BucketizerBatchOpTest {
    @Test
    public void testBucketizerBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.1, true, 2, "A"),
            Row.of(1.1, false, 2, "B"),
            Row.of(1.1, true, 1, "B"),
            Row.of(2.2, true, 1, "A")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "double double, bool
boolean, number int, str string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "double double,
bool boolean, number int, str string");
        BatchOperator <?> bucketizer = new
BucketizerBatchOp().setSelectedCols("double","number").setCutsArray(
            new double[] {1.0,2.0,2.2,4.0},new double[]{0.0,1.1});
        bucketizer.linkFrom(inOp1).print();
        StreamOperator <?> bucketizer2 = new
BucketizerStreamOp().setSelectedCols("double").setCutsArray(
            new double[] {2.0});
        bucketizer2.linkFrom(inOp2).print();
        StreamOperator.execute();
    }
}
```

运行结果

输出数据

批预测结果

double	bool	number	str
0	true	2	A
1	false	2	B
2	true	1	B
3	true	1	A

流预测结果

double	bool	number	str
0	false	2	B
0	true	1	B
0	true	2	A
1	true	1	A

卡方选择器 (ChiSqSelectorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.ChiSqSelectorBatchOp

Python 类名：ChiSqSelectorBatchOp

功能介绍

针对table数据，进行特征筛选。计算features和label列两两的卡方值，取最大的n个feature作为结果。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	
fdr	发现阈值	发现阈值, 默认值0.05	Double		
fpr	p value的阈值	p value的阈值，默认值0.05	Double		
fwe	错误率阈值	错误率阈值, 默认值0.05	Double		
numTopFeatures	最大的p-value列个数	最大的p-value列个数, 默认值50	Integer		
percentile	筛选的百分比	筛选的百分比，默认值0.1	Double		

selectorType	筛选类型	筛选类型，包含"NumTopFeatures","percentile", "fpr", "fdr", "fwe"五种。	String		"NumTopFeature "PERCENTILE", "FPR", "FDR", "FWE"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1, 1,2.0, True],
    ["c", 1, 2, -3.0, True],
    ["a", 2, 2,2.0, False],
    ["c", 0, 0, 0.0, False]
])

source = BatchOperator.fromDataframe(df, schemaStr='f_string string, f_long
long, f_int int, f_double double, f_boolean boolean')

selector = ChiSqSelectorBatchOp()\
            .setSelectedCols(["f_string", "f_long", "f_int", "f_double"])\
            .setLabelCol("f_boolean")\
            .setNumTopFeatures(2)

selector.linkFrom(source)

modelInfo: ChisqSelectorModelInfo = selector.collectModelInfo()

print(modelInfo.getColNames())
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.ChiSqSelectorBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.feature.ChisqSelectorModelInfo;
import org.junit.Test;
```

```
import java.util.Arrays;
import java.util.List;

public class ChiSqSelectorBatchOpTest {
    @Test
    public void testChiSqSelectorBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1L, 1, 2.0, true),
            Row.of("c", 1L, 2, -3.0, true),
            Row.of("a", 2L, 2, 2.0, false),
            Row.of("c", 0L, 0, 0.0, false)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df,
            "f_string string, f_long long, f_int int, f_double double,
f_boolean boolean");
        ChiSqSelectorBatchOp selector = new ChiSqSelectorBatchOp()
            .setSelectedCols("f_string", "f_long", "f_int", "f_double")
            .setLabelCol("f_boolean")
            .setNumTopFeatures(2);
        selector.linkFrom(source);
        ChisqSelectorModelInfo modelInfo = selector.collectModelInfo();
        System.out.println(modelInfo.toString());
    }
}
```

运行结果

```
------------------------- ChisqSelectorModelInfo -------------------------
Number of Selector Features: 2
Number of Features: 4
Type of Selector: NumTopFeatures
Number of Top Features: 2
Selector Indices:
    | ColName|ChiSquare|PValue| DF|Selected|
    |--------|---------|------|---|--------|
    |  f_long|        4|0.1353|  2|    true|
    |   f_int|        2|0.3679|  2|    true|
    |f_double|        2|0.3679|  2|   false|
    |f_string|        0|     1|  1|   false|
```

Cross特征预测 (CrossFeaturePredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.CrossFeaturePredictBatchOp

Python 类名：CrossFeaturePredictBatchOp

功能介绍

将选定的特征列组合成单个向量类型的特征。

使用方式

该组件是预测组件，需要配合预测组件 CrossFeatureTrainBatchOp 使用。

使用中指定输出列名（outputCol）即可。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
["1.0", "1.0", 1.0, 1],
["1.0", "1.0", 0.0, 1],
["1.0", "0.0", 1.0, 1],
["1.0", "0.0", 1.0, 1],
["2.0", "3.0", None, 0],
```

```
["2.0", "3.0", 1.0, 0],
["0.0", "1.0", 2.0, 0],
["0.0", "1.0", 1.0, 0]])
data = BatchOperator.fromDataframe(df, schemaStr="f0 string, f1 string, f2
double, label bigint")
train =
CrossFeatureTrainBatchOp().setSelectedCols(['f0','f1','f2']).linkFrom(data)
CrossFeaturePredictBatchOp().setOutputCol("cross").linkFrom(train,
data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.CrossFeaturePredictBatchOp;
import com.alibaba.alink.operator.batch.feature.CrossFeatureTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CrossFeaturePredictBatchOpTest {
    @Test
    public void testCrossFeaturePredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1.0", "1.0", 1.0, 1),
            Row.of("1.0", "1.0", 0.0, 1),
            Row.of("1.0", "0.0", 1.0, 1),
            Row.of("1.0", "0.0", 1.0, 1),
            Row.of("2.0", "3.0", null, 0),
            Row.of("2.0", "3.0", 1.0, 0),
            Row.of("0.0", "1.0", 2.0, 0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 string, f1
string, f2 double, label int");
        BatchOperator <?> train = new
CrossFeatureTrainBatchOp().setSelectedCols("f0", "f1", "f2").linkFrom(data);
        new CrossFeaturePredictBatchOp().setOutputCol("cross").linkFrom(train,
data).print();
    }
}
```

运行结果

f0	f1	f2	label	cross
1.0	1.0	1.0000	1	$36$0:1.0
1.0	1.0	0.0000	1	$36$9:1.0
1.0	0.0	1.0000	1	$36$6:1.0
1.0	0.0	1.0000	1	$36$6:1.0
2.0	3.0	null	0	$36$22:1.0
2.0	3.0	1.0000	0	$36$4:1.0
0.0	1.0	2.0000	0	$36$29:1.0
0.0	1.0	1.0000	0	$36$2:1.0

Cross特征训练 (CrossFeatureTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.CrossFeatureTrainBatchOp

Python 类名：CrossFeatureTrainBatchOp

功能介绍

将选定的特征列组合成单个向量类型的特征。

使用方式

该组件是训练组件，需要配合预测组件 CrossFeaturePredictBatch/StreamOp 使用。

为了训练模型，需要指定参与组合的特征列名（selectedCols）。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
["1.0", "1.0", 1.0, 1],
["1.0", "1.0", 0.0, 1],
["1.0", "0.0", 1.0, 1],
["1.0", "0.0", 1.0, 1],
["2.0", "3.0", None, 0],
["2.0", "3.0", 1.0, 0],
["0.0", "1.0", 2.0, 0],
["0.0", "1.0", 1.0, 0]])
data = BatchOperator.fromDataframe(df, schemaStr="f0 string, f1 string, f2
double, label bigint")
```

```
train =
CrossFeatureTrainBatchOp().setSelectedCols(['f0','f1','f2']).linkFrom(data)
CrossFeaturePredictBatchOp().setOutputCol("cross").linkFrom(train,
data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.CrossFeaturePredictBatchOp;
import com.alibaba.alink.operator.batch.feature.CrossFeatureTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CrossFeatureTrainBatchOpTest {
    @Test
    public void testCrossFeatureTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1.0", "1.0", 1.0, 1),
            Row.of("1.0", "1.0", 0.0, 1),
            Row.of("1.0", "0.0", 1.0, 1),
            Row.of("1.0", "0.0", 1.0, 1),
            Row.of("2.0", "3.0", null, 0),
            Row.of("2.0", "3.0", 1.0, 0),
            Row.of("0.0", "1.0", 2.0, 0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 string, f1
string, f2 double, label int");
        BatchOperator <?> train = new
CrossFeatureTrainBatchOp().setSelectedCols("f0", "f1", "f2").linkFrom(data);
        new CrossFeaturePredictBatchOp().setOutputCol("cross").linkFrom(train,
data).print();
    }
}
```

运行结果

f0	f1	f2	label	cross
1.0	1.0	1.0000	1	$36$0:1.0
1.0	1.0	0.0000	1	$36$9:1.0

1.0	0.0	1.0000	1	$36$6:1.0
1.0	0.0	1.0000	1	$36$6:1.0
2.0	3.0	null	0	$36$22:1.0
2.0	3.0	1.0000	0	$36$4:1.0
0.0	1.0	2.0000	0	$36$29:1.0
0.0	1.0	1.0000	0	$36$2:1.0

离散余弦变换 (DCTBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.DCTBatchOp

Python 类名：DCTBatchOp

功能介绍

DCT(Discrete Cosine Transform), 又叫做离散余弦变换，是对数据进行离散余弦变换，可以用来做视频编码，图像压缩等，经过变换后数据会有更好的聚集性。 输入是vector列，vector的size为n, 那经过变换后的输出变成size为n的vector。

算法原理

DCT变换就是输入信号为实偶函数的DFT变换

$$F(u,v) = \frac{1}{2N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)cos[\frac{\pi}{N}u(x+\frac{1}{2})]cos[\frac{\pi}{N}v(y+\frac{1}{2})]$$

$$f(x,y) = \frac{1}{2N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)cos[\frac{\pi}{N}u(x+\frac{1}{2})]cos[\frac{\pi}{N}v(y+\frac{1}{2})]$$

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
inverse	是否为逆变换	是否为逆变换，false表示正变换，true表示逆变换。默认正变换。	Boolean			false
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["-0.6264538 0.1836433"],
    ["-0.8356286 1.5952808"],
    ["0.3295078 -0.8204684"],
    ["0.4874291 0.7383247"],
    ["0.5757814 -0.3053884"],
    ["1.5117812 0.3898432"],
    ["-0.6212406 -2.2146999"],
    ["11.1249309 9.9550664"],
    ["9.9838097 10.9438362"],
    ["10.8212212 10.5939013"],
    ["10.9189774 10.7821363"],
    ["10.0745650 8.0106483"],
    ["10.6198257 9.9438713"],
    ["9.8442045 8.5292476"],
    ["9.5218499 10.4179416"],
])

data = BatchOperator.fromDataframe(df, schemaStr='features string')

dct = DCTBatchOp() \
    .setSelectedCol("features") \
    .setOutputCol("result")

dct.linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.DCTBatchOp;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DCTBatchOpTest {
    @Test
    public void testDCTBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("-0.6264538 0.1836433"),
            Row.of("-0.8356286 1.5952808"),
            Row.of("0.3295078 -0.8204684"),
            Row.of("0.4874291 0.7383247"),
            Row.of("0.5757814 -0.3053884"),
            Row.of("1.5117812 0.3898432"),
            Row.of("-0.6212406 -2.2146999"),
            Row.of("11.1249309 9.9550664"),
            Row.of("9.9838097 10.9438362"),
            Row.of("10.8212212 10.5939013"),
            Row.of("10.9189774 10.7821363"),
            Row.of("10.0745650 8.0106483"),
            Row.of("10.6198257 9.9438713"),
            Row.of("9.8442045 8.5292476"),
            Row.of("9.5218499 10.4179416")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "features string");
        BatchOperator <?> dct = new DCTBatchOp()
            .setSelectedCol("features")
            .setOutputCol("result");
        dct.linkFrom(data).print();
    }
}
```

运行结果

features	result
-0.6264538 0.1836433	-0.31311430733060563 -0.5728251528295567
-0.8356286 1.5952808	0.5371552219632794 -1.7189125211901217
0.3295078 -0.8204684	-0.34716156955541605 0.8131559692231375
0.4874291 0.7383247	0.866738824045179 -0.17740998012986753
0.5757814 -0.3053884	0.19119672388537412 0.6230811409567939
1.5117812 0.3898432	1.3446515085097996 0.7933299678708727

-0.6212406 -2.2146999	-2.005312758591568 1.126745876574769
11.1249309 9.9550664	14.905809038224113 0.8272191210194105
9.9838097 10.9438362	14.798080330160849 -0.6788412482687869
10.8212212 10.5939013	15.142778339690611 0.1607394427886475
10.9189774 10.7821363	15.345004656570287 0.09676126975502636
10.0745650 8.0106483	12.788176963635138 1.4594094943741613
10.6198257 9.9438713	14.54072959496546 0.4779719400128842
9.8442045 8.5292476	12.991992573716212 0.9298149409580412
9.5218499 10.4179416	14.099561785095878 -0.6336325176349823

等宽离散化预测 (EqualWidthDiscretizerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.EqualWidthDiscretizerPredictBatchOp

Python 类名：EqualWidthDiscretizerPredictBatchOp

功能介绍

等宽离散是常见的离散化方法，可以用于计算选定数值列的分位点，然后对数据进行离散化

算法原理

该算法将取值范围划为一些区间，每个区间都有相同的组距，也就是数据范围/组数，通过训练可以得到一系列分为点， 然后使用这些分位点进行预测。 该组件可以对所有列使用同一个分组数量，也可以每一列对应一个分组数量。预测结果可以是特征值或一系列0/1离散特征。

编码结果

Encode ——> INDEX

预测结果为单个token的index

Encode ——> VECTOR

预测结果为稀疏向量:

```
1. dropLast为true,向量中非零元个数为0或者1
2. dropLast为false,向量中非零元个数必定为1
```

Encode ——> ASSEMBLED_VECTOR

编码方式为"ASSEMBLED_VECTOR"时，必须设置一个输出列，输出结果为稀疏向量,是各列VECTOR格式的预测,按照选择顺序拼接的结果。

向量维度

Encode ——> Vector

$$ vectorSize = numBuckets - dropLast(true: 1, false: 0) + (handleInvalid: keep(1), skip(0), error(0)) $$

```
numBuckets: 训练参数

dropLast: 预测参数

handleInvalid: 预测参数
```

Token index

Encode ——> Vector

```
1. 正常数据: 唯一的非零元为数据所在的bucket,若 dropLast为true, 最大的bucket的值会被丢掉, 预测结果为全零元

2. null:
    2.1 handleInvalid为keep: 唯一的非零元为:numBuckets - dropLast(true: 1, false: 0)
    2.2 handleInvalid为skip: null
    2.3 handleInvalid为error: 报错
```

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
dropLast	是否删除最后一个元素	删除最后一个元素是为了保证线性无关性。默认true	Boolean			true
encode	编码方法	编码方法	String		"VECTOR", "ASSEMBLED_VECTOR", "INDEX"	"INDEX
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP
modelFilePath	模型的文件路径	模型的文件路径	String			null

outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1, 1.1],
    ["b", -2, 0.9],
    ["c", 100, -0.01],
    ["d", -99, 100.9],
    ["a", 1, 1.1],
    ["b", -2, 0.9],
    ["c", 100, -0.01],
    ["d", -99, 100.9]
])

batchSource =  BatchOperator.fromDataframe(df,schemaStr="f_string string,
f_long long, f_double double")

trainOp = EqualWidthDiscretizerTrainBatchOp(). \
setSelectedCols(['f_long', 'f_double']). \
setNumBuckets(5). \
linkFrom(batchSource)

EqualWidthDiscretizerPredictBatchOp(). \
setSelectedCols(['f_long', 'f_double']). \
```

```
linkFrom(trainOp,batchSource). \
print()

trainOp = EqualWidthDiscretizerTrainBatchOp().setSelectedCols(['f_long',
'f_double']). \
setNumBucketsArray([5,3]). \
linkFrom(batchSource)

EqualWidthDiscretizerPredictBatchOp(). \
setSelectedCols(['f_long', 'f_double']). \
linkFrom(trainOp,batchSource). \
print()

EqualWidthDiscretizerPredictBatchOp(). \
setEncode("ASSEMBLED_VECTOR"). \
setSelectedCols(['f_long', 'f_double']). \
setOutputCols(["assVec"]). \
linkFrom(trainOp,batchSource).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.feature.EqualWidthDiscretizerPredictBatchOp;
import
com.alibaba.alink.operator.batch.feature.EqualWidthDiscretizerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.feature.EqualWidthDiscretizerPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.params.feature.HasEncodeWithoutWoe.Encode;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class EqualWidthDiscretizerPredictBatchOpTest {
    @Test
    public void testEqualWidthDiscretizerPredictBatchOp2() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1, 1.1),
            Row.of("b", -2, 0.9),
            Row.of("c", 100, -0.01),
            Row.of("d", -99, 100.9),
```

```
            Row.of("a", 1, 1.1),
            Row.of("b", -2, 0.9),
            Row.of("c", 100, -0.01),
            Row.of("d", -99, 100.9)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, "f_string
string, f_long int, f_double double");

        BatchOperator <?> trainOp = new
EqualWidthDiscretizerTrainBatchOp().setSelectedCols("f_long", "f_double")
            .setNumBuckets(5).linkFrom(batchSource);

        new
EqualWidthDiscretizerPredictBatchOp().setSelectedCols("f_long","f_double")
            .linkFrom(trainOp, batchSource).print();

        BatchOperator trainOp2 = new
EqualWidthDiscretizerTrainBatchOp().setSelectedCols("f_long", "f_double")
            .setNumBucketsArray(5,3).linkFrom(batchSource);

        new
EqualWidthDiscretizerPredictBatchOp().setSelectedCols("f_long","f_double")
            .linkFrom(trainOp2,batchSource).print();

        new
EqualWidthDiscretizerPredictBatchOp().setSelectedCols("f_long","f_double")
            .setEncode(Encode.ASSEMBLED_VECTOR)
            .setOutputCols("assVec")
            .linkFrom(trainOp2,batchSource).print();
    }
}
```

运行结果

f_string	f_long	f_double
a	2	0
b	2	0
c	4	0
d	0	4
a	2	0
b	2	0
c	4	0

d	0	4

f_string	f_long	f_double
a	2	0
b	2	0
c	4	0
d	0	2
a	2	0
b	2	0
c	4	0
d	0	2

f_string	f_long	f_double	assVec
a	1	1.1000	$8$2:1.0 5:1.0
b	-2	0.9000	$8$2:1.0 5:1.0
c	100	-0.0100	$8$5:1.0
d	-99	100.9000	$8$0:1.0
a	1	1.1000	$8$2:1.0 5:1.0
b	-2	0.9000	$8$2:1.0 5:1.0
c	100	-0.0100	$8$5:1.0
d	-99	100.9000	$8$0:1.0

等宽离散化训练 (EqualWidthDiscretizerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.EqualWidthDiscretizerTrainBatchOp

Python 类名：EqualWidthDiscretizerTrainBatchOp

功能介绍

等宽离散可以计算选定数值列的分位点，每个区间都有相同的组距，也就是数据范围/组数，通过训练可以得到一系列分为点， 然后使用这些分位点进行预测。 其中可以所有列使用同一个分组数量，也可以每一列对应一个分组数量。预测结果可以是特征值或一系列0/1离散特征。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
leftOpen	是否左开右闭	左开右闭为true，左闭右开为false	Boolean			true
numBuckets	quantile个数	quantile个数，对所有列有效。	Integer			2
numBucketsArray	quantile个数	quantile个数，每一列对应数组中一个元素。	Integer[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1, 1.1],
    ["b", -2, 0.9],
    ["c", 100, -0.01],
    ["d", -99, 100.9],
    ["a", 1, 1.1],
    ["b", -2, 0.9],
    ["c", 100, -0.01],
    ["d", -99, 100.9]
])

batchSource =  BatchOperator.fromDataframe(df,schemaStr="f_string string,
f_long long, f_double double")

trainOp = EqualWidthDiscretizerTrainBatchOp(). \
setSelectedCols(['f_long', 'f_double']). \
setNumBuckets(5). \
linkFrom(batchSource)

EqualWidthDiscretizerPredictBatchOp(). \
setSelectedCols(['f_long', 'f_double']). \
linkFrom(trainOp,batchSource). \
print()

trainOp = EqualWidthDiscretizerTrainBatchOp().setSelectedCols(['f_long',
'f_double']). \
setNumBucketsArray([5,3]). \
linkFrom(batchSource)

EqualWidthDiscretizerPredictBatchOp(). \
setSelectedCols(['f_long', 'f_double']). \
linkFrom(trainOp,batchSource). \
print()

EqualWidthDiscretizerPredictBatchOp(). \
setEncode("ASSEMBLED_VECTOR"). \
setSelectedCols(['f_long', 'f_double']). \
setOutputCols(["assVec"]). \
linkFrom(trainOp,batchSource).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.feature.EqualWidthDiscretizerPredictBatchOp;
import
com.alibaba.alink.operator.batch.feature.EqualWidthDiscretizerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.feature.EqualWidthDiscretizerPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.params.feature.HasEncodeWithoutWoe.Encode;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class EqualWidthDiscretizerTrainBatchOpTest {
    @Test
    public void testEqualWidthDiscretizerTrainBatchOp2() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1, 1.1),
            Row.of("b", -2, 0.9),
            Row.of("c", 100, -0.01),
            Row.of("d", -99, 100.9),
            Row.of("a", 1, 1.1),
            Row.of("b", -2, 0.9),
            Row.of("c", 100, -0.01),
            Row.of("d", -99, 100.9)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, "f_string
string, f_long int, f_double double");

        BatchOperator <?> trainOp = new
EqualWidthDiscretizerTrainBatchOp().setSelectedCols("f_long", "f_double")
            .setNumBuckets(5).linkFrom(batchSource);

        new
EqualWidthDiscretizerPredictBatchOp().setSelectedCols("f_long","f_double")
            .linkFrom(trainOp, batchSource).print();

        BatchOperator trainOp2 = new
EqualWidthDiscretizerTrainBatchOp().setSelectedCols("f_long", "f_double")
            .setNumBucketsArray(5,3).linkFrom(batchSource);

        new
EqualWidthDiscretizerPredictBatchOp().setSelectedCols("f_long","f_double")
```

```
            .linkFrom(trainOp2,batchSource).print();

        new
EqualWidthDiscretizerPredictBatchOp().setSelectedCols("f_long","f_double")
            .setEncode(Encode.ASSEMBLED_VECTOR)
            .setOutputCols("assVec")
            .linkFrom(trainOp2,batchSource).print();
    }
}
```

运行结果

f_string	f_long	f_double
a	2	0
b	2	0
c	4	0
d	0	4
a	2	0
b	2	0
c	4	0
d	0	4

f_string	f_long	f_double
a	2	0
b	2	0
c	4	0
d	0	2
a	2	0
b	2	0
c	4	0
d	0	2

f_string	f_long	f_double	assVec
a	1	1.1000	$8$2:1.0 5:1.0

b	-2	0.9000	$8$2:1.0 5:1.0
c	100	-0.0100	$8$5:1.0
d	-99	100.9000	$8$0:1.0
a	1	1.1000	$8$2:1.0 5:1.0
b	-2	0.9000	$8$2:1.0 5:1.0
c	100	-0.0100	$8$5:1.0
d	-99	100.9000	$8$0:1.0

特征哈希 (FeatureHasherBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.FeatureHasherBatchOp

Python 类名：FeatureHasherBatchOp

功能介绍

将多个特征组合成一个特征向量。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
categoricalCols	离散特征列名	离散特征列名	String[]			
numFeatures	向量维度	生成向量长度	Integer			262144
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.1, True, "2", "A"],
    [1.1, False, "2", "B"],
    [1.1, True, "1", "B"],
    [2.2, True, "1", "A"]
```

```
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='double double, bool boolean,
number int, str string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='double double, bool
boolean, number int, str string')

hasher = FeatureHasherBatchOp().setSelectedCols(["double", "bool", "number",
"str"]).setOutputCol("output").setNumFeatures(200)
hasher.linkFrom(inOp1).print()

hasher = FeatureHasherStreamOp().setSelectedCols(["double", "bool", "number",
"str"]).setOutputCol("output").setNumFeatures(200)
hasher.linkFrom(inOp2).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.FeatureHasherBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.feature.FeatureHasherStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FeatureHasherBatchOpTest {
    @Test
    public void testFeatureHasherBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.1, true, 2, "A"),
            Row.of(1.1, false, 2, "B"),
            Row.of(1.1, true, 1, "B"),
            Row.of(2.2, true, 1, "A")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "double double, bool
boolean, number int, str string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "double double,
bool boolean, number int, str string");
        BatchOperator <?> hasher = new
FeatureHasherBatchOp().setSelectedCols("double", "bool", "number", "str")
```

```
            .setOutputCol("output").setNumFeatures(200);
        hasher.linkFrom(inOp1).print();
        StreamOperator <?> hasher2 = new
FeatureHasherStreamOp().setSelectedCols("double", "bool", "number", "str")
            .setOutputCol("output").setNumFeatures(200);
        hasher2.linkFrom(inOp2).print();
        StreamOperator.execute();
    }
}
```

运行结果

输出数据

double	bool	number	str	output
1.1000	true	2	A	$200$13:2.0 38:1.1 45:1.0 195:1.0
1.1000	false	2	B	$200$13:2.0 30:1.0 38:1.1 76:1.0
1.1000	true	1	B	$200$13:1.0 38:1.1 76:1.0 195:1.0
2.2000	true	1	A	$200$13:1.0 38:2.2 45:1.0 195:1.0

Hash Cross特征 (HashCrossFeatureBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.HashCrossFeatureBatchOp

Python 类名：HashCrossFeatureBatchOp

功能介绍

将选定的离散列组合成单列的向量类型的数据。

算法原理

将选定列的数据的字符串形式以逗号为分隔符拼接起来，然后使用 `murmur3_32` 函数得到哈希值，并将哈希值通过平移的方式转换至 `[0, 特征数)` 之间。

使用方式

使用需要设置选取列的列名（ `selectCols` ）和输出列名（ `outputCol` ），特征数通过参数 `numFeatures` 设置，特征数也是 输出列中向量的长度。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
numFeatures	向量维度	生成向量长度	Integer			262144
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    ["1.0", "1.0", 1.0, 1],
    ["1.0", "1.0", 0.0, 1],
    ["1.0", "0.0", 1.0, 1],
    ["1.0", "0.0", 1.0, 1],
    ["2.0", "3.0", None, 0],
    ["2.0", "3.0", 1.0, 0],
    ["0.0", "1.0", 2.0, 0],
    ["0.0", "1.0", 1.0, 0]])
data = BatchOperator.fromDataframe(df, schemaStr="f0 string, f1 string, f2
double, label bigint")
cross = HashCrossFeatureBatchOp().setSelectedCols(['f0', 'f1',
'f2']).setOutputCol('cross').setNumFeatures(4)
print(cross.linkFrom(data))
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.HashCrossFeatureBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HashCrossFeatureBatchOpTest {
    @Test
    public void testHashCrossFeatureBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1.0", "1.0", 1.0, 1),
            Row.of("1.0", "1.0", 0.0, 1),
            Row.of("1.0", "0.0", 1.0, 1),
            Row.of("1.0", "0.0", 1.0, 1),
            Row.of("2.0", "3.0", null, 0),
            Row.of("2.0", "3.0", 1.0, 0),
            Row.of("0.0", "1.0", 2.0, 0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 string, f1
string, f2 double, label bigint");
        BatchOperator <?> cross = new
HashCrossFeatureBatchOp().setSelectedCols("f0", "f1",
"f2").setOutputCol("cross")
            .setNumFeatures(4);
        System.out.print(cross.linkFrom(data));
```

```
    }
}
```

运行结果

f0	f1	f2	label	cross
1.0	1.0	0.0000	1	$36$33:1.0
1.0	1.0	1.0000	1	$36$15:1.0
1.0	0.0	1.0000	1	$36$33:1.0
1.0	0.0	1.0000	1	$36$33:1.0
2.0	3.0	1.0000	0	$36$20:1.0
2.0	3.0	None	0	36
0.0	1.0	1.0000	0	$36$28:1.0
0.0	1.0	2.0000	0	$36$33:1.0

线性特征重要性 (LinearModelFeatureImportanceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.LinearModelFeatureImportanceBatchOp

Python 类名：LinearModelFeatureImportanceBatchOp

功能介绍

- 计算线性模型的特征重要性, 支持稀疏和稠密。
- 输入为训练好的线性模型和待预测的数据。
- 输出为各个特征的重要性值、系数和均值等。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为 [DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
```

```
    [1, 2, 1],
    [5, 3, 2]])

input = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label int')

# load data
dataTest = input
colnames = ["f0","f1"]
lr =
LogisticRegressionTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = input.link(lr)

LinearModelFeatureImportanceBatchOp().linkFrom(model, dataTest).print()
```

运行结果

col_name	feat_importance	coefficient	mean
f0	28.130170	20.741863	2.875
f1	7.938565	8.574630	2.500

多热编码预测 (MultiHotPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.MultiHotPredictBatchOp

Python 类名：MultiHotPredictBatchOp

功能介绍

multi-hot编码，也称多热编码，是与独热编码相对应的一种编码方式。该编码对每一个字符串特征列按照指定分隔符进行分割，分割得到的值存在m个可能值，那么经过多热编码后就变成了m个二元特征。对每一字段编码将会把该字段分割后的每一个值映射到唯一的编码。 因此，编码后的数据会变成稀疏数据，输出结果也是kv的稀疏结构。

组件为多热编码的批式预测组件。

编码结果

输入

col_0	col_1
"a b"	"1 2"
"b c"	"1 3"
"c d"	"1 4"
"a d"	"3 2"
"d e"	null
NULL	"2 3"

Encode ——> VECTOR

预测结果为稀疏向量:

向量中非零元个数必定为1, 只能是一个稀疏向量$5$0:1.0 4:1.0或者NULL。

Encode ——> ASSEMBLED_VECTOR

预测结果为稀疏向量,是预测选择列中,各列预测为VECTOR时,按照选择顺序ASSEMBLE的结果。

向量维度

Encode ——> Vector

$$ vectorSize = distinct token Number + enableElse(true: 1, false:0) + (handleInvalid: keep(1), skip(0), error(0))
$$

```
distinct token Number: 训练集中指定列的去重后的token数目

enableElse: 训练时若填写discreteThresholds或discreteThresholdsArray则为true，默认为
false

handleInvalid: 预测参数
```

举例

输入列为col_0

```
1. 如果没有填写discreteThresholds，那么enableElse为false，distinct token Number为
(a,b,c,d,e)一共5个token
        1.1.1 handleInvalid为keep: vectorSize=(5 + 0 + 1 = 6)
        1.2.2 handleInvalid为skip: vectorSize=(5 + 0 + 0 = 5)
        1.2.3 handleInvalid为error: vectorSize=(5 + 0 + 0 = 5)
2. 如果discreteThresholds为2, 那么enableElse为true, distinct token Number为
(a,b,c,d,e)一共5个token
        1.1.1 handleInvalid为keep: vectorSize=(5 + 1 + 1 = 7)
        1.2.2 handleInvalid为skip: vectorSize=(5 + 1 + 0 = 6)
        1.2.3 handleInvalid为error: vectorSize=(5 + 1 + 0 = 6)
```

Token index

Encode ——> Vector

```
1. 训练集中出现过的token: 预测值为模型中token对应的token_index

2. 训练集中未出现过的token:
    3.1 enableElse为true
        3.1.1 handleInvalid为keep: 预测值为:distinct token Number + 1
        3.1.2 handleInvalid为skip: 预测值为:distinct token Number
        3.1.3 handleInvalid为error: 预测值为:distinct token Number

    3.2 enableElse为false
        3.2.1 handleInvalid为keep: 预测值为:distinct token Number
        3.2.2 handleInvalid为skip: 无index
        3.2.3 handleInvalid为error: 报错
```

举例

输入列为col_0

1. 如果没有填写discreteThresholds, 假设模型中a,b,c,d,e对应的token index为0,1,2,3,4

1.1 handleInvalid为keep

col_0	Encode为VECTOR的输出
"a b"	$6$0:1.0 1:1.0
"b c"	$6$1:1.0 2:1.0
"c d"	$6$3:1.0 3:1.0
"a d"	$6$0:1.0 3:1.0
"d e"	$6$0:3.0 4:1.0
NULL	NULL

1.2 handleInvalid为skip

col_0	Encode为VECTOR的输出
"a b"	$5$0:1.0 1:1.0
"b c"	$5$1:1.0 2:1.0
"c d"	$5$3:1.0 3:1.0
"a d"	$5$0:1.0 3:1.0
"d e"	$5$0:3.0 4:1.0
NULL	NULL

1.3 handleInvalid为error: 直接报错

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
outputCols	输出结果列列名数组	输出结果列列名数组，必选	String[]	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为 [STRING]	
encode	编码方法	编码方法	String		"VECTOR", "ASSEMBLED_VECTOR"	"ASSEM

handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP"
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

# load data
df = pd.DataFrame([
    ["a b", 1],
    ["b c", 1],
    ["c d", 1],
    ["a d", 2],
    ["d e", 2],
    [None, 1]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='query string, weight long')
```

```
# multi hot train
multi_hot = MultiHotTrainBatchOp().setSelectedCols(["query"])
model = inOp.link(multi_hot)
model.print()

# batch predict
predictor =
MultiHotPredictBatchOp().setSelectedCols(["query"]).setOutputCols(["output"])
print(BatchOperator.collectToDataframe(predictor.linkFrom(model, inOp)))
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.MultiHotPredictBatchOp;
import com.alibaba.alink.operator.batch.feature.MultiHotTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MultiHotPredictBatchOpTest {
    @Test
    public void testMultiHotPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a b", 1),
            Row.of("b c", 1),
            Row.of("c d", 1),
            Row.of("a d", 2),
            Row.of("d e", 2),
            Row.of(null, 1)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "query string, weight
int");
        BatchOperator <?> multi_hot = new
MultiHotTrainBatchOp().setSelectedCols("query");
        BatchOperator <?> model = inOp.link(multi_hot);
        model.print();
        BatchOperator <?> predictor = new
MultiHotPredictBatchOp().setSelectedCols("query").setOutputCols("output");
        predictor.linkFrom(model, inOp).print();
    }
}
```

运行结果

模型

model_id	model_info
0	{"delimiter":"\" \""}
1048576	["query","a",0,2]
2097152	["query","b",1,2]
3145728	["query","c",2,2]
4194304	["query","d",3,3]
5242880	["query","e",4,1]

结果

query	weight	output
a b	1	$6$0:1.0 1:1.0
b c	1	$6$1:1.0 2:1.0
c d	1	$6$2:1.0 3:1.0
a d	2	$6$0:1.0 3:1.0
d e	2	$6$3:1.0 4:1.0
null	1	null

多热编码训练 (MultiHotTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.MultiHotTrainBatchOp

Python 类名：MultiHotTrainBatchOp

功能介绍

multi-hot编码，也称多热编码，是与独热编码相对应的一种编码方式。该编码对每一个字符串特征列按照指定分隔符进行分割，分割得到的值存在m个可能值，那么经过多热编码后就变成了m个二元特征。对每一字段编码将会把该字段分割后的每一个值映射到唯一的编码。 因此，编码后的数据会变成稀疏数据，输出结果也是kv的稀疏结构。

编码结果

输入

col_0	col_1
"a b"	"1 2"
"b c"	"1 3"
"c d"	"1 4"
"a d"	"3 2"
"d e"	null
NULL	"2 3"

Encode ——> VECTOR

预测结果为稀疏向量:

向量中非零元个数必定为1, 只能是一个稀疏向量$5$0:1.0 4:1.0或者NULL。

Encode ——> ASSEMBLED_VECTOR

预测结果为稀疏向量,是预测选择列中,各列预测为VECTOR时,按照选择顺序ASSEMBLE的结果。

向量维度

Encode ——> Vector

$$ vectorSize = distinct token Number + enableElse(true: 1, false:0) + (handleInvalid: keep(1), skip(0), error(0))
$$

```
distinct token Number: 训练集中指定列的去重后的token数目

enableElse: 训练时若填写discreteThresholds或discreteThresholdsArray则为true，默认为false

handleInvalid: 预测参数
```

举例

输入列为col_0

```
1. 如果没有填写discreteThresholds，那么enableElse为false，distinct token Number为(a,b,c,d,e)一共5个token
        1.1.1 handleInvalid为keep: vectorSize=(5 + 0 + 1 = 6)
        1.2.2 handleInvalid为skip: vectorSize=(5 + 0 + 0 = 5)
        1.2.3 handleInvalid为error: vectorSize=(5 + 0 + 0 = 5)
2. 如果discreteThresholds为2, 那么enableElse为true, distinct token Number为(a,b,c,d,e)一共5个token
        1.1.1 handleInvalid为keep: vectorSize=(5 + 1 + 1 = 7)
        1.2.2 handleInvalid为skip: vectorSize=(5 + 1 + 0 = 6)
        1.2.3 handleInvalid为error: vectorSize=(5 + 1 + 0 = 6)
```

Token index

Encode ——> Vector

```
1. 训练集中出现过的token: 预测值为模型中token对应的token_index

2. 训练集中未出现过的token:
    3.1 enableElse为true
        3.1.1 handleInvalid为keep: 预测值为:distinct token Number + 1
        3.1.2 handleInvalid为skip: 预测值为:distinct token Number
        3.1.3 handleInvalid为error: 预测值为:distinct token Number

    3.2 enableElse为false
        3.2.1 handleInvalid为keep: 预测值为:distinct token Number
        3.2.2 handleInvalid为skip: 无index
        3.2.3 handleInvalid为error: 报错
```

举例

输入列为col_0

1. 如果没有填写discreteThresholds, 假设模型中a,b,c,d,e对应的token index为0,1,2,3,4

1.1 handleInvalid为keep

col_0	Encode为VECTOR的输出

"a b"	$6$0:1.0 1:1.0
"b c"	$6$1:1.0 2:1.0
"c d"	$6$3:1.0 3:1.0
"a d"	$6$0:1.0 3:1.0
"d e"	$6$0:3.0 4:1.0
NULL	NULL

1.2 handleInvalid为skip

col_0	Encode为VECTOR的输出
"a b"	$5$0:1.0 1:1.0
"b c"	$5$1:1.0 2:1.0
"c d"	$5$3:1.0 3:1.0
"a d"	$5$0:1.0 3:1.0
"d e"	$5$0:3.0 4:1.0
NULL	NULL

1.3 handleInvalid为error: 直接报错

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[STRING]	
delimiter	分隔符	用来分割字符串	String			" "

discreteThresholds	离散个数阈值	离散个数阈值，低于该阈值的离散样本将不会单独成一个组别。	Integer			-2147483648
discreteThresholdsArray	离散个数阈值	离散个数阈值，每一列对应数组中一个元素。	Integer[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

# load data
df = pd.DataFrame([
    ["a b", 1],
    ["b c", 1],
    ["c d", 1],
    ["a d", 2],
    ["d e", 2],
    [None, 1]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='query string, weight long')

# multi hot train
multi_hot = MultiHotTrainBatchOp().setSelectedCols(["query"])
model = inOp.link(multi_hot)
model.print()

# batch predict
predictor =
MultiHotPredictBatchOp().setSelectedCols(["query"]).setOutputCols(["output"])
print(BatchOperator.collectToDataframe(predictor.linkFrom(model, inOp)))
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.MultiHotPredictBatchOp;
import com.alibaba.alink.operator.batch.feature.MultiHotTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MultiHotTrainBatchOpTest {
    @Test
    public void testMultiHotTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a b", 1),
            Row.of("b c", 1),
            Row.of("c d", 1),
            Row.of("a d", 2),
            Row.of("d e", 2),
            Row.of(null, 1)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "query string, weight
int");
        BatchOperator <?> multi_hot = new
MultiHotTrainBatchOp().setSelectedCols("query");
        BatchOperator <?> model = inOp.link(multi_hot);
        model.print();
        BatchOperator <?> predictor = new
MultiHotPredictBatchOp().setSelectedCols("query").setOutputCols("output");
        predictor.linkFrom(model, inOp).print();
    }
}
```

运行结果

模型

model_id	model_info
0	{"delimiter":"\" \""}
1048576	["query","a",0,2]
2097152	["query","b",1,2]
3145728	["query","c",2,2]

4194304	["query","d",3,3]
5242880	["query","e",4,1]

结果

query	weight	output
a b	1	$6$0:1.0 1:1.0
b c	1	$6$1:1.0 2:1.0
c d	1	$6$2:1.0 3:1.0
a d	2	$6$0:1.0 3:1.0
d e	2	$6$3:1.0 4:1.0
null	1	null

独热编码预测 (OneHotPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.OneHotPredictBatchOp

Python 类名：OneHotPredictBatchOp

功能介绍

one-hot编码，也称独热编码，对于每一个特征，如果它有m个可能值，那么经过 独热编码后，就变成了m个二元特征。并且，这些特征互斥，每次只有一个激活。 因此，数据会变成稀疏的，输出结果也是kv的稀疏结构。

组件为独热编码的批式预测组件。

编码结果

输入

selectedCol0	selectedCol1
a	1
b	1
c	1
d	2
a	2
b	2
c	2
e	null
NULL	2

Encode ——> INDEX

预测结果为单个token的index，如0, 1, 2 ...

Encode ——> VECTOR

预测结果为稀疏向量:

```
1. dropLast为true,向量中非零元个数为0或者1, 如$5, $5$0:1.0或者NULL。
2. dropLast为false,向量中非零元个数必定为1, 只能是$5$0:1.0或者NULL。
```

Encode ——> ASSEMBLED_VECTOR

预测结果为稀疏向量,是预测选择列中,各列预测为VECTOR时,按照选择顺序ASSEMBLE的结果。

向量维度

Encode ——> Vector

$$ vectorSize = distinct token Number - dropLast(true: 1, false: 0) + enableElse(true: 1, false:0) + (handleInvalid: keep(1), skip(0), error(0)) $$

```
distinct token Number: 训练集中指定列的去重后的token数目

dropLast: 预测参数

enableElse: 训练时若填写discreteThresholds或discreteThresholdsArray则为true，默认为false

handleInvalid: 预测参数
```

举例

输入列为selectedCol0

```
1. 如果没有填写discreteThresholds，那么enableElse为false，distinct token Number为(a,b,c,d,e)一共5个token
    1.1 dropLast为True
        1.1.1 handleInvalid为keep: vectorSize=(5 - 1 + 0 + 1 = 5)
        1.1.2 handleInvalid为skip: vectorSize=(5 - 1 + 0 + 0 = 4)
        1.1.3 handleInvalid为error: vectorSize=(5 - 1 + 0 + 0 = 4)
    1.2 dropLast为False
        1.1.1 handleInvalid为keep: vectorSize=(5 - 0 + 0 + 1 = 6)
        1.2.2 handleInvalid为skip: vectorSize=(5 - 0 + 0 + 0 = 5)
        1.2.3 handleInvalid为error: vectorSize=(5 - 0 + 0 + 0 = 5)
2. 如果discreteThresholds为2, 那么enableElse为true, distinct token Number为(a,b,c)一共3个token
    2.1 dropLast为True
        1.1.1 handleInvalid为keep: vectorSize=(3 - 1 + 1 + 1 = 4)
        1.1.2 handleInvalid为skip: vectorSize=(3 - 1 + 1 + 0 = 3)
        1.1.3 handleInvalid为error: vectorSize=(3 - 1 + 1 + 0 = 3)
    2.2 dropLast为False
        1.1.1 handleInvalid为keep: vectorSize=(3 - 0 + 1 + 1 = 5)
        1.2.2 handleInvalid为skip: vectorSize=(3 - 0 + 1 + 0 = 4)
        1.2.3 handleInvalid为error: vectorSize=(3 - 0 + 1 + 0 = 4)
```

Token index

Encode ——> Vector

```
1. 训练集中出现过的token: 预测值为模型中token对应的token_index,若 dropLast为true, token_index最大的值会被丢掉，预测结果为全零元
```

```
2. null:
    2.1 handleInvalid为keep: 预测值为distinct token Number - dropLast(true: 1,
false: 0)
    2.2 handleInvalid为skip: null
    2.3 handleInvalid为error: 报错

3. 训练集中未出现过的token:
    3.1 enableElse为true
        3.1.1 handleInvalid为keep: 预测值为:distinct token Number -
dropLast(true: 1, false: 0) + 1
        3.1.2 handleInvalid为skip: 预测值为:distinct token Number -
dropLast(true: 1, false: 0)
        3.1.3 handleInvalid为error: 预测值为:distinct token Number -
dropLast(true: 1, false: 0)

    3.2 enableElse为false
        3.2.1 handleInvalid为keep: 预测值为:distinct token Number -
dropLast(true: 1, false: 0)
        3.2.2 handleInvalid为skip: null
        3.2.3 handleInvalid为error: 报错
```

举例

输入列为selectedCol0

1. 如果没有填写discreteThresholds

   假设模型中a,b,c,d,e对应的token index为0,1,2,3,4

   1.1 dropLast为True

   1.1.1 handleInvalid为keep

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $5$0:1.0 | | b | 1 | $5$1:1.0 | | c | 2 | $5$2:1.0 | | d | 3 | $5$3:1.0 | | e | 4 (Encode为IDNEX时,dropLast不起作用) | 5 (最大的token index被drop了) | | NULL | 5 | $5$4:1.0 |

   1.1.2 handleInvalid为skip: vectorSize=(5 - 1 + 0 + 0 = 4)

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $4$0:1.0 | | b | 1 | $4$1:1.0 | | c | 2 | $4$2:1.0 | | d | 3 | $4$3:1.0 | | e | 4 (Encode为IDNEX时,dropLast不起作用) | 4 (最大的token index被drop了) | | NULL | NULL | NULL |

1.1.3 handleInvalid为error: 直接报错

```
1.2 dropLast为False
1.1.1 handleInvalid为keep
```

selectedCol0	Encode为INDEX的输出	Encode为VECTOR的输出
a	0	$6$0:1.0

b	1	$6$1:1.0
c	2	$6$2:1.0
d	3	$6$3:1.0
e	4 (Encode为IDNEX时,dropLast不起作用)	$6$4:1.0
NULL	5	$6$5:1.0

1.2.2 handleInvalid为skip

selectedCol0	Encode为INDEX的输出	Encode为VECTOR的输出
a	0	$5$0:1.0
b	1	$5$1:1.0
c	2	$5$2:1.0
d	3	$5$3:1.0
e	4 (Encode为IDNEX时,dropLast不起作用)	$5$4:1.0
NULL	NULL	NULL

```
1.2.3 handleInvalid为error: 直接报错
```

1. 如果discreteThresholds为2 假设模型中a,b,c对应的token index为0,1,2 2.1 dropLast为True 1.1.1 handleInvalid为keep:

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $4$0:1.0 | | b | 1 | $4$1:1.0 | | c | 2 | 4 (最大的token index被drop了) | | d | 4 | $4$3:1.0 (unknown token) | | e | 4 | $4$3:1.0 (unknown token) | | NULL | 3 | $4$2:1.0 |

   1.1.2 handleInvalid为skip:

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $3$0:1.0 | | b | 1 | $3$1:1.0 | | c | 2 | 3 (最大的token index被drop了) | | d | 3 | $3$2:1.0 (unknown token) | | e | 3 | $4$2:1.0 (unknown token) | | NULL | NULL | NULL |

   ```
   1.1.3 handleInvalid为error: 直接报错
   ```

   2.2 dropLast为False 1.1.1 handleInvalid为keep:

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $5$0:1.0 | | b | 1 | $5$1:1.0 | | c | 2 | $5$2:1.0 | | d | 4 | $5$4:1.0 (unknown token) | | e | 4 | $5$4:1.0 (unknown token) | | NULL | 3 | $5$3:1.0 |

   1.2.2 handleInvalid为skip:

| selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $4$0:1.0 | | b | 1 | $4$1:1.0 | | c | 2 | $4$2:1.0 | | d | 3 | $4$3:1.0 (unknown token) | | e | 3 | $4$3:1.0 (unknown token) | | NULL | NULL | NULL |

1.2.3 handleInvalid为error: 直接报错

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
dropLast	是否删除最后一个元素	删除最后一个元素是为了保证线性无关性。默认true	Boolean			true
encode	编码方法	编码方法	String		"VECTOR", "ASSEMBLED_VECTOR", "INDEX"	"ASSE
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1],
    ["b", 1],
    ["c", 1],
    ["e", 2],
    ["a", 2],
    ["b", 1],
    ["c", 2],
    ["d", 2],
    [None, 1]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='query string, weight long')

# one hot train
one_hot = OneHotTrainBatchOp().setSelectedCols(["query"])
model = inOp.link(one_hot)
model.lazyPrint(10)

# batch predict
predictor = OneHotPredictBatchOp().setOutputCols(["output"])
predictor.linkFrom(model, inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.OneHotPredictBatchOp;
import com.alibaba.alink.operator.batch.feature.OneHotTrainBatchOp;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class OneHotPredictBatchOpTest {
    @Test
    public void testOneHotPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1),
            Row.of("b", 1),
            Row.of("c", 1),
            Row.of("e", 2),
            Row.of("a", 2),
            Row.of("b", 1),
            Row.of("c", 2),
            Row.of("d", 2),
            Row.of(null, 1)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "query string, weight
int");
        BatchOperator <?> one_hot = new
OneHotTrainBatchOp().setSelectedCols("query");
        BatchOperator <?> model = inOp.link(one_hot);
        model.lazyPrint(10);
        BatchOperator <?> predictor = new
OneHotPredictBatchOp().setOutputCols("output");
        predictor.linkFrom(model, inOp).print();
    }
}
```

运行结果

模型

column_index	token	token_index
-1	{"selectedCols":"[\"query\"]","selectedColTypes":"[\"VARCHAR\"]","enableElse":"false"}	null
0	a	0
0	b	1
0	c	2
0	d	3

0	e	4

预测

query	weight	output
a	1	$5$0:1.0
b	1	$5$1:1.0
c	1	$5$2:1.0
e	2	5
a	2	$5$0:1.0
b	1	$5$1:1.0
c	2	$5$2:1.0
d	2	$5$3:1.0
null	1	$5$4:1.0

独热编码训练 (OneHotTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.OneHotTrainBatchOp

Python 类名：OneHotTrainBatchOp

功能介绍

one-hot编码，也称独热编码，对于每一个特征，如果它有m个可能值，那么经过 独热编码后，就变成了m个二元特征。并且，这些特征互斥，每次只有一个激活。 因此，数据会变成稀疏的，输出结果也是kv的稀疏结构。

编码结果

输入

selectedCol0	selectedCol1
a	1
b	1
c	1
d	2
a	2
b	2
c	2
e	null
NULL	2

Encode ——> INDEX

预测结果为单个token的index，如0, 1, 2 ...

Encode ——> VECTOR

预测结果为稀疏向量:

```
1. dropLast为true,向量中非零元个数为0或者1, 如$5, $5$0:1.0或者NULL。
2. dropLast为false,向量中非零元个数必定为1, 只能是$5$0:1.0或者NULL。
```

Encode ——> ASSEMBLED_VECTOR

预测结果为稀疏向量,是预测选择列中,各列预测为VECTOR时,按照选择顺序ASSEMBLE的结果。

向量维度

Encode ——> Vector

$$ vectorSize = distinct token Number - dropLast(true: 1, false: 0) + enableElse(true: 1, false:0) + (handleInvalid: keep(1), skip(0), error(0)) $$

```
distinct token Number: 训练集中指定列的去重后的token数目

dropLast: 预测参数

enableElse: 训练时若填写discreteThresholds或discreteThresholdsArray则为true，默认为false

handleInvalid: 预测参数
```

举例

输入列为selectedCol0

```
1. 如果没有填写discreteThresholds，那么enableElse为false，distinct token Number为(a,b,c,d,e)一共5个token
    1.1 dropLast为True
        1.1.1 handleInvalid为keep: vectorSize=(5 - 1 + 0 + 1 = 5)
        1.1.2 handleInvalid为skip: vectorSize=(5 - 1 + 0 + 0 = 4)
        1.1.3 handleInvalid为error: vectorSize=(5 - 1 + 0 + 0 = 4)
    1.2 dropLast为False
        1.1.1 handleInvalid为keep: vectorSize=(5 - 0 + 0 + 1 = 6)
        1.2.2 handleInvalid为skip: vectorSize=(5 - 0 + 0 + 0 = 5)
        1.2.3 handleInvalid为error: vectorSize=(5 - 0 + 0 + 0 = 5)
2. 如果discreteThresholds为2, 那么enableElse为true, distinct token Number为(a,b,c)一共3个token
    2.1 dropLast为True
        1.1.1 handleInvalid为keep: vectorSize=(3 - 1 + 1 + 1 = 4)
        1.1.2 handleInvalid为skip: vectorSize=(3 - 1 + 1 + 0 = 3)
        1.1.3 handleInvalid为error: vectorSize=(3 - 1 + 1 + 0 = 3)
    2.2 dropLast为False
        1.1.1 handleInvalid为keep: vectorSize=(3 - 0 + 1 + 1 = 5)
        1.2.2 handleInvalid为skip: vectorSize=(3 - 0 + 1 + 0 = 4)
        1.2.3 handleInvalid为error: vectorSize=(3 - 0 + 1 + 0 = 4)
```

Token index

Encode ——> Vector

```
1. 训练集中出现过的token: 预测值为模型中token对应的token_index,若 dropLast为true, token_index最大的值会被丢掉，预测结果为全零元

2. null:
```

```
    2.1 handleInvalid为keep: 预测值为distinct token Number - dropLast(true: 1,
false: 0)
    2.2 handleInvalid为skip: null
    2.3 handleInvalid为error: 报错

3. 训练集中未出现过的token:
    3.1 enableElse为true
        3.1.1 handleInvalid为keep: 预测值为:distinct token Number -
dropLast(true: 1, false: 0) + 1
        3.1.2 handleInvalid为skip: 预测值为:distinct token Number -
dropLast(true: 1, false: 0)
        3.1.3 handleInvalid为error: 预测值为:distinct token Number -
dropLast(true: 1, false: 0)

    3.2 enableElse为false
        3.2.1 handleInvalid为keep: 预测值为:distinct token Number -
dropLast(true: 1, false: 0)
        3.2.2 handleInvalid为skip: null
        3.2.3 handleInvalid为error: 报错
```

举例

输入列为selectedCol0

1. 如果没有填写discreteThresholds

   假设模型中a,b,c,d,e对应的token index为0,1,2,3,4

   1.1 dropLast为True

   1.1.1 handleInvalid为keep

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $5$0:1.0 | | b | 1 | $5$1:1.0 | | c | 2 | $5$2:1.0 | | d | 3 | $5$3:1.0 | | e | 4 (Encode为IDNEX时,dropLast不起作用) | 5 (最大的token index被drop了) | | NULL | 5 | $5$4:1.0 |

   1.1.2 handleInvalid为skip: vectorSize=(5 - 1 + 0 + 0 = 4)

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $4$0:1.0 | | b | 1 | $4$1:1.0 | | c | 2 | $4$2:1.0 | | d | 3 | $4$3:1.0 | | e | 4 (Encode为IDNEX时,dropLast不起作用) | 4 (最大的token index被drop了) | | NULL | NULL | NULL |

1.1.3 handleInvalid为error: 直接报错

```
1.2 dropLast为False
1.1.1 handleInvalid为keep
```

selectedCol0	Encode为INDEX的输出	Encode为VECTOR的输出
a	0	$6$0:1.0
b	1	$6$1:1.0

c	2	$6$2:1.0
d	3	$6$3:1.0
e	4 (Encode为IDNEX时,dropLast不起作用)	$6$4:1.0
NULL	5	$6$5:1.0

1.2.2 handleInvalid为skip

selectedCol0	Encode为INDEX的输出	Encode为VECTOR的输出
a	0	$5$0:1.0
b	1	$5$1:1.0
c	2	$5$2:1.0
d	3	$5$3:1.0
e	4 (Encode为IDNEX时,dropLast不起作用)	$5$4:1.0
NULL	NULL	NULL

```
1.2.3 handleInvalid为error: 直接报错
```

1. 如果discreteThresholds为2 假设模型中a,b,c对应的token index为0,1,2 2.1 dropLast为True 1.1.1 handleInvalid为keep:

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $4$0:1.0 | | b | 1 | $4$1:1.0 | | c | 2 | 4 (最大的token index被drop了) | | d | 4 | $4$3:1.0 (unknown token) | | e | 4 | $4$3:1.0 (unknown token) | | NULL | 3 | $4$2:1.0 |

   1.1.2 handleInvalid为skip:

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $3$0:1.0 | | b | 1 | $3$1:1.0 | | c | 2 | 3 (最大的token index被drop了) | | d | 3 | $3$2:1.0 (unknown token) | | e | 3 | $4$2:1.0 (unknown token) | | NULL | NULL | NULL |

   ```
   1.1.3 handleInvalid为error: 直接报错
   ```

   2.2 dropLast为False 1.1.1 handleInvalid为keep:

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $5$0:1.0 | | b | 1 | $5$1:1.0 | | c | 2 | $5$2:1.0 | | d | 4 | $5$4:1.0 (unknown token) | | e | 4 | $5$4:1.0 (unknown token) | | NULL | 3 | $5$3:1.0 |

   1.2.2 handleInvalid为skip:

   | selectedCol0 | Encode为INDEX的输出 | Encode为VECTOR的输出 | | --- | --- | --- | | a | 0 | $4$0:1.0 | | b | 1 | $4$1:1.0 | | c | 2 | $4$2:1.0 | | d | 3 | $4$3:1.0 (unknown token) | | e | 3 | $4$3:1.0 (unknown token) | | NULL | NULL | NULL |

1.2.3 handleInvalid为error: 直接报错

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
discreteThresholds	离散个数阈值	离散个数阈值，低于该阈值的离散样本将不会单独成一个组别。	Integer			-2147483648
discreteThresholdsArray	离散个数阈值	离散个数阈值，每一列对应数组中一个元素。	Integer[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1],
    ["b", 1],
    ["c", 1],
    ["e", 2],
    ["a", 2],
    ["b", 1],
    ["c", 2],
```

```
    ["d", 2],
    [None, 1]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='query string, weight long')

# one hot train
one_hot = OneHotTrainBatchOp().setSelectedCols(["query"])
model = inOp.link(one_hot)
model.lazyPrint(10)

# batch predict
predictor = OneHotPredictBatchOp().setOutputCols(["output"])
predictor.linkFrom(model, inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.OneHotPredictBatchOp;
import com.alibaba.alink.operator.batch.feature.OneHotTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class OneHotTrainBatchOpTest {
    @Test
    public void testOneHotTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1),
            Row.of("b", 1),
            Row.of("c", 1),
            Row.of("e", 2),
            Row.of("a", 2),
            Row.of("b", 1),
            Row.of("c", 2),
            Row.of("d", 2),
            Row.of(null, 1)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "query string, weight
int");
        BatchOperator <?> one_hot = new
OneHotTrainBatchOp().setSelectedCols("query");
        BatchOperator <?> model = inOp.link(one_hot);
```

```
        model.lazyPrint(10);
        BatchOperator <?> predictor = new
OneHotPredictBatchOp().setOutputCols("output");
        predictor.linkFrom(model, inOp).print();
    }
}
```

运行结果

模型

column_index	token	token_index
-1	{"selectedCols":"[\"query\"]","selectedColTypes":"[\"VARCHAR\"]","enableElse":"false"}	null
0	a	0
0	b	1
0	c	2
0	d	3
0	e	4

预测

query	weight	output
a	1	$5$0:1.0
b	1	$5$1:1.0
c	1	$5$2:1.0
e	2	5
a	2	$5$0:1.0
b	1	$5$1:1.0
c	2	$5$2:1.0
d	2	$5$3:1.0
null	1	$5$4:1.0

特征构造：OverWindow (OverWindowBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.OverWindowBatchOp

Python 类名：OverWindowBatchOp

功能介绍

批式OverWindow特征构造组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
clause	运算语句	运算语句	String	✓		
orderBy	排序列	排序列	String	✓		
groupCols	分组列名数组	分组列名，多列，可选，默认不选	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 8, 2.2],
    [2, 7, 3.3],
    [5, 6, 4.4],
    [5, 6, 5.5],
    [7, 5, 6.6],
    [1, 8, 1.1],
    [1, 9, 1.0],
    [7, 5, 7.7],
    [9, 5, 8.8],
```

```
    [9, 4, 9.8],
    [19, 4, 8.8]])
data = BatchOperator.fromDataframe(df, schemaStr="f0 bigint, f1 bigint, f2
double")
OverWindowBatchOp().setOrderBy("f0, f1 desc").setClause("count_preceding(*) as
cc").linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.OverWindowBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class OverWindowBatchOpTest {
    @Test
    public void testOverWindowBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 8, 2.2),
            Row.of(2, 7, 3.3),
            Row.of(5, 6, 4.4),
            Row.of(5, 6, 5.5),
            Row.of(7, 5, 6.6),
            Row.of(1, 8, 1.1),
            Row.of(1, 9, 1.0),
            Row.of(7, 5, 7.7),
            Row.of(9, 5, 8.8),
            Row.of(9, 4, 9.8)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "f0 int, f1 int, f2
double");
        new OverWindowBatchOp().setOrderBy("f0, f1
desc").setClause("count_preceding(*) as cc").linkFrom(data).print();
    }
}
```

运行结果

f0	f1	f2	cc
1	9	1.0000	0

1	8	1.1000	1
2	8	2.2000	2
2	7	3.3000	3
5	6	4.4000	4
5	6	5.5000	5
7	5	6.6000	6
7	5	7.7000	7
9	5	8.8000	8
9	4	9.8000	9

样本稳定指数 (PSIBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.PSIBatchOp

Python 类名：PSIBatchOp

功能介绍

样本稳定指数是衡量样本变化所产生的偏移量的一种重要指标，通常用来衡量样本的稳定程度，比如样本在两个月份之间的变化是否稳定。通常变量的PSI值在0.1以下表示变化不太显著，在0.1到0.25之间表示有比较显著的变化，大于0.25表示变量变化比较剧烈，需要特殊关注。

PSI计算公式：

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, True, 0, "A", 1],
    [2.1, False, 2, "B", 1],
    [1.1, True, 3, "C", 1],
    [2.2, True, 1, "E", 0],
    [0.1, True, 2, "A", 0],
    [1.5, False, -4, "D", 1],
    [1.3, True, 1, "B", 0],
    [0.2, True, -1, "A", 1],
])
```

```
inOp1 = BatchOperator.fromDataframe(df, schemaStr='f0 double, f1 boolean, f2
int, f3 string, label int')

split = SplitBatchOp().setFraction(0.8).linkFrom(inOp1)
test1 = split
test2 = split.getSideOutput(0)

train = BinningTrainBatchOp().setSelectedCols(["f0", "f1", "f2",
"f3"]).setLabelCol("label").setPositiveLabelValueString("1").linkFrom(inOp1)
op = PSIBatchOp().setSelectedCols(["f0", "f1", "f2", "f3"]).linkFrom(train,
test1, test2)
op.print()
```

运行结果

featureName index values testPercentage basePercentage \ 0 f2 2.0 NaN 0.0 0.00
1 f2 0.0 (-inf,1] 50.0 66.67
2 f0 2.0 NaN 0.0 0.00
3 f1 2.0 NaN 0.0 0.00
4 f1 0.0 false 50.0 16.67
5 f3 2.0 E 0.0 16.67
6 f3 3.0 A 50.0 33.33
7 f0 NaN NaN NaN NaN
8 f2 NaN NaN NaN NaN
9 f1 3.0 ELSE 0.0 0.00
10 f1 1.0 true 50.0 83.33
11 f3 5.0 NaN 0.0 0.00
12 f3 6.0 ELSE 0.0 0.00
13 f3 0.0 B 0.0 33.33
14 f3 1.0 C 0.0 16.67
15 f3 4.0 D 50.0 0.00
16 f2 1.0 (1,+inf) 50.0 33.33
17 f0 0.0 (-inf,1.3] 50.0 66.67
18 f0 1.0 (1.3,+inf) 50.0 33.33
19 f1 NaN NaN NaN NaN
20 f3 NaN NaN NaN NaN

```
testSubBase  lnTestDivBase     psi
```

0 0.00 NaN NaN
1 -16.67 -0.29 0.0480
2 0.00 NaN NaN
3 0.00 NaN NaN
4 33.33 1.10 0.3661
5 -16.67 NaN NaN
6 16.67 0.41 0.0676
7 NaN NaN 0.1156
8 NaN NaN 0.1156

9 0.00 NaN NaN
10 -33.33 -0.51 0.1702
11 0.00 NaN NaN
12 0.00 NaN NaN
13 -33.33 NaN NaN
14 -16.67 NaN NaN
15 50.00 NaN NaN
16 16.67 0.41 0.0676
17 -16.67 -0.29 0.0480
18 16.67 0.41 0.0676
19 NaN NaN 0.5363
20 NaN NaN 0.0676

主成分分析预测 (PcaPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.PcaPredictBatchOp

Python 类名：PcaPredictBatchOp

功能介绍

主成分分析，是考察多个变量间相关性一种多元统计方法，研究如何通过少数几个主成分来揭示多个变量间的内部结构，即从原始变量中导出少数几个主成分，使它们尽可能多地保留原始变量的信息，且彼此间互不相关，作为新的综合指标。详细介绍请见维基百科链接wiki。

主成分分析功能包含主成分分析训练和主成分分析预测（批和流)。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
        [0.0,0.0,0.0],
        [0.1,0.2,0.1],
        [0.2,0.2,0.8],
        [9.0,9.5,9.7],
        [9.1,9.1,9.6],
        [9.2,9.3,9.9]
])

# batch source
inOp = BatchOperator.fromDataframe(df, schemaStr='x1 double, x2 double, x3
double')

trainOp = PcaTrainBatchOp()\
       .setK(2)\
       .setSelectedCols(["x1","x2","x3"])

predictOp = PcaPredictBatchOp()\
        .setPredictionCol("pred")

# batch train
inOp.link(trainOp)

# batch predict
predictOp.linkFrom(trainOp,inOp)

predictOp.print()

# stream predict
inStreamOp = StreamOperator.fromDataframe(df, schemaStr='x1 double, x2 double,
x3 double')

predictStreamOp = PcaPredictStreamOp(trainOp)\
        .setPredictionCol("pred")

predictStreamOp.linkFrom(inStreamOp)

predictStreamOp.print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.PcaPredictBatchOp;
import com.alibaba.alink.operator.batch.feature.PcaTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.feature.PcaPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class PcaPredictBatchOpTest {
    @Test
    public void testPcaPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0.0, 0.0, 0.0),
            Row.of(0.1, 0.2, 0.1),
            Row.of(0.2, 0.2, 0.8),
            Row.of(9.0, 9.5, 9.7),
            Row.of(9.1, 9.1, 9.6),
            Row.of(9.2, 9.3, 9.9)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "x1 double, x2
double, x3 double");
        BatchOperator <?> trainOp = new PcaTrainBatchOp()
            .setK(2)
            .setSelectedCols("x1", "x2", "x3");
        BatchOperator <?> predictOp = new PcaPredictBatchOp()
            .setPredictionCol("pred");
        inOp.link(trainOp);
        predictOp.linkFrom(trainOp, inOp);
        predictOp.print();
        StreamOperator <?> inStreamOp = new MemSourceStreamOp(df, "x1 double,
x2 double, x3 double");
        StreamOperator <?> predictStreamOp = new PcaPredictStreamOp(trainOp)
            .setPredictionCol("pred");
        predictStreamOp.linkFrom(inStreamOp);
        predictStreamOp.print();
        StreamOperator.execute();
    }
}
```

运行结果

x1	x2	x3	pred
9.0	9.5	9.7	3.2280384305400736,1.1516225426477789E-4
0.2	0.2	0.8	0.13565076707329407,0.09003329494282108
9.2	9.3	9.9	3.250783163664603,0.0456526246528135
9.1	9.1	9.6	3.182618319978973,0.027469531992220464
0.1	0.2	0.1	0.045855205015063565,-0.012182917696915518
0.0	0.0	0.0	0.0,0.0

主成分分析训练 (PcaTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.PcaTrainBatchOp

Python 类名：PcaTrainBatchOp

功能介绍

主成分分析，是考察多个变量间相关性一种多元统计方法，研究如何通过少数几个主成分来揭示多个变量间的内部结构，即从原始变量中导出少数几个主成分，使它们尽可能多地保留原始变量的信息，且彼此间互不相关，作为新的综合指标。详细介绍请见维基百科链接wiki。

主成分分析功能包含主成分分析训练和主成分分析预测（批和流)。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
k	降维后的维度	降维后的维度	Integer	✓	[1, +inf)	
calculationType	计算类型	计算类型，包含"CORR", "COV"两种。	String		"CORR", "COV"	"CORR"
selectedCols	选中的列名数组	计算列对应的列名列表	String[]		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        [0.0,0.0,0.0],
        [0.1,0.2,0.1],
        [0.2,0.2,0.8],
        [9.0,9.5,9.7],
        [9.1,9.1,9.6],
        [9.2,9.3,9.9]
])

# batch source
inOp = BatchOperator.fromDataframe(df, schemaStr='x1 double, x2 double, x3
double')

trainOp = PcaTrainBatchOp()\
       .setK(2)\
       .setSelectedCols(["x1","x2","x3"])

predictOp = PcaPredictBatchOp()\
        .setPredictionCol("pred")

# batch train
inOp.link(trainOp)

# batch predict
predictOp.linkFrom(trainOp,inOp)

predictOp.print()

# stream predict
inStreamOp = StreamOperator.fromDataframe(df, schemaStr='x1 double, x2 double,
x3 double')

predictStreamOp = PcaPredictStreamOp(trainOp)\
        .setPredictionCol("pred")

predictStreamOp.linkFrom(inStreamOp)

predictStreamOp.print()
```

```
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.PcaPredictBatchOp;
import com.alibaba.alink.operator.batch.feature.PcaTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.feature.PcaPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class PcaTrainBatchOpTest {
    @Test
    public void testPcaTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0.0, 0.0, 0.0),
            Row.of(0.1, 0.2, 0.1),
            Row.of(0.2, 0.2, 0.8),
            Row.of(9.0, 9.5, 9.7),
            Row.of(9.1, 9.1, 9.6),
            Row.of(9.2, 9.3, 9.9)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "x1 double, x2
double, x3 double");
        BatchOperator <?> trainOp = new PcaTrainBatchOp()
            .setK(2)
            .setSelectedCols("x1", "x2", "x3");
        BatchOperator <?> predictOp = new PcaPredictBatchOp()
            .setPredictionCol("pred");
        inOp.link(trainOp);
        predictOp.linkFrom(trainOp, inOp);
        predictOp.print();
        StreamOperator <?> inStreamOp = new MemSourceStreamOp(df, "x1 double,
x2 double, x3 double");
        StreamOperator <?> predictStreamOp = new PcaPredictStreamOp(trainOp)
            .setPredictionCol("pred");
        predictStreamOp.linkFrom(inStreamOp);
        predictStreamOp.print();
        StreamOperator.execute();
```

```
    }
}
```

运行结果

x1	x2	x3	pred
9.0	9.5	9.7	3.2280384305400736,1.1516225426477789E-4
0.2	0.2	0.8	0.13565076707329407,0.09003329494282108
9.2	9.3	9.9	3.250783163664603,0.0456526246528135
9.1	9.1	9.6	3.182618319978973,0.027469531992220464
0.1	0.2	0.1	0.045855205015063565,-0.012182917696915518
0.0	0.0	0.0	0.0,0.0

分位数离散化预测 (QuantileDiscretizerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.QuantileDiscretizerPredictBatchOp

Python 类名：QuantileDiscretizerPredictBatchOp

功能介绍

分位点离散可以计算选定列的分位点，然后使用这些分位点进行离散化。 生成选中列对应的q-quantile，其中可以所有列指定一个，也可以每一列对应一个

编码结果

Encode ——> INDEX

预测结果为单个token的index

Encode ——> VECTOR

预测结果为稀疏向量:

```
1. dropLast为true,向量中非零元个数为0或者1
2. dropLast为false,向量中非零元个数必定为1
```

Encode ——> ASSEMBLED_VECTOR

预测结果为稀疏向量,是预测选择列中,各列预测为VECTOR时,按照选择顺序ASSEMBLE的结果。

向量维度

Encode ——> Vector

$$ vectorSize = numBuckets - dropLast(true: 1, false: 0) + (handleInvalid: keep(1), skip(0), error(0)) $$

```
numBuckets: 训练参数

dropLast: 预测参数

handleInvalid: 预测参数
```

Token index

Encode ——> Vector

```
1. 正常数据: 唯一的非零元为数据所在的bucket,若 dropLast为true, 最大的bucket的值会被丢掉,
预测结果为全零元
```

```
2. null:
    2.1 handleInvalid为keep: 唯一的非零元为:numBuckets - dropLast(true: 1, false:
0)
    2.2 handleInvalid为skip: null
    2.3 handleInvalid为error: 报错
```

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
dropLast	是否删除最后一个元素	删除最后一个元素是为了保证线性无关性。默认true	Boolean			true
encode	编码方法	编码方法	String		"VECTOR", "ASSEMBLED_VECTOR", "INDEX"	"INDEX
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP
modelFilePath	模型的文件路径	模型的文件路径	String			null

outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1, 1, 2.0, True],
    ["c", 1, 2, -3.0, True],
    ["a", 2, 2, 2.0, False],
    ["c", 0, 0, 0.0, False]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f_string string, f_long long, f_int int, f_double double,
f_boolean boolean')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f_string string, f_long long, f_int int, f_double double,
f_boolean boolean')

trainOp = QuantileDiscretizerTrainBatchOp()\
    .setSelectedCols(['f_double'])\
    .setNumBuckets(8)\
    .linkFrom(batchSource)
predictBatchOp = QuantileDiscretizerPredictBatchOp()\
    .setSelectedCols(['f_double'])
predictStreamOp = QuantileDiscretizerPredictStreamOp(trainOp)\
```

```
    .setSelectedCols(['f_double'])

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource) .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.feature.QuantileDiscretizerPredictBatchOp;
import
com.alibaba.alink.operator.batch.feature.QuantileDiscretizerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.feature.QuantileDiscretizerPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class QuantileDiscretizerPredictBatchOpTest {
    @Test
    public void testQuantileDiscretizerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1, 1, 2.0, true),
            Row.of("c", 1, 2, -3.0, true),
            Row.of("a", 2, 2, 2.0, false),
            Row.of("c", 0, 0, 0.0, false)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df,
            "f_string string, f_long int, f_int int, f_double double, f_boolean
boolean");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df,
            "f_string string, f_long int, f_int int, f_double double, f_boolean
boolean");
        BatchOperator <?> trainOp = new QuantileDiscretizerTrainBatchOp()
            .setSelectedCols("f_double")
            .setNumBuckets(8)
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new
QuantileDiscretizerPredictBatchOp()
```

```
            .setSelectedCols("f_double");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        StreamOperator <?> predictStreamOp = new
QuantileDiscretizerPredictStreamOp(trainOp)
            .setSelectedCols("f_double");
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f_string	f_long	f_int	f_double	f_boolean
a	1	1	2	true
c	1	2	0	true
a	2	2	2	false
c	0	0	1	false

分位数离散化训练 (QuantileDiscretizerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.QuantileDiscretizerTrainBatchOp

Python 类名：QuantileDiscretizerTrainBatchOp

功能介绍

分位点离散可以计算选定列的分位点，然后使用这些分位点进行离散化。 生成选中列对应的q-quantile，其中可以所有列指定一个，也可以每一列对应一个

编码结果

Encode ——> INDEX

预测结果为单个token的index

Encode ——> VECTOR

预测结果为稀疏向量:

```
1. dropLast为true,向量中非零元个数为0或者1
2. dropLast为false,向量中非零元个数必定为1
```

Encode ——> ASSEMBLED_VECTOR

预测结果为稀疏向量,是预测选择列中,各列预测为VECTOR时,按照选择顺序ASSEMBLE的结果。

向量维度

Encode ——> Vector

$$ vectorSize = numBuckets - dropLast(true: 1, false: 0) + (handleInvalid: keep(1), skip(0), error(0)) $$

```
numBuckets: 训练参数

dropLast: 预测参数

handleInvalid: 预测参数
```

Token index

Encode ——> Vector

```
1. 正常数据: 唯一的非零元为数据所在的bucket,若 dropLast为true, 最大的bucket的值会被丢掉,
预测结果为全零元

2. null:
```

```
    2.1 handleInvalid为keep: 唯一的非零元为:numBuckets - dropLast(true: 1, false:
0)
    2.2 handleInvalid为skip: null
    2.3 handleInvalid为error: 报错
```

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
leftOpen	是否左开右闭	左开右闭为true，左闭右开为false	Boolean			true
numBuckets	quantile个数	quantile个数，对所有列有效。	Integer			2
numBucketsArray	quantile个数	quantile个数，每一列对应数组中一个元素。	Integer[]			null

代码示例

Python 代码

```python
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1, 1, 2.0, True],
    ["c", 1, 2, -3.0, True],
    ["a", 2, 2, 2.0, False],
    ["c", 0, 0, 0.0, False]
])
```

```
batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f_string string, f_long long, f_int int, f_double double,
f_boolean boolean')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f_string string, f_long long, f_int int, f_double double,
f_boolean boolean')

trainOp = QuantileDiscretizerTrainBatchOp()\
    .setSelectedCols(['f_double'])\
    .setNumBuckets(8)\
    .linkFrom(batchSource)
predictBatchOp = QuantileDiscretizerPredictBatchOp()\
    .setSelectedCols(['f_double'])
predictStreamOp = QuantileDiscretizerPredictStreamOp(trainOp)\
    .setSelectedCols(['f_double'])

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource) .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.feature.QuantileDiscretizerPredictBatchOp;
import
com.alibaba.alink.operator.batch.feature.QuantileDiscretizerTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.feature.QuantileDiscretizerPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class QuantileDiscretizerTrainBatchOpTest {
    @Test
    public void testQuantileDiscretizerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1, 1, 2.0, true),
            Row.of("c", 1, 2, -3.0, true),
```

```
            Row.of("a", 2, 2, 2.0, false),
            Row.of("c", 0, 0, 0.0, false)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df,
            "f_string string, f_long int, f_int int, f_double double, f_boolean
boolean");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df,
            "f_string string, f_long int, f_int int, f_double double, f_boolean
boolean");
        BatchOperator <?> trainOp = new QuantileDiscretizerTrainBatchOp()
            .setSelectedCols("f_double")
            .setNumBuckets(8)
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new
QuantileDiscretizerPredictBatchOp()
            .setSelectedCols("f_double");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        StreamOperator <?> predictStreamOp = new
QuantileDiscretizerPredictStreamOp(trainOp)
            .setSelectedCols("f_double");
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f_string	f_long	f_int	f_double	f_boolean
a	1	1	2	true
c	1	2	0	true
a	2	2	2	false
c	0	0	1	false

决策树模型编码 (TreeModelEncoderBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.TreeModelEncoderBatchOp

Python 类名：TreeModelEncoderBatchOp

功能介绍

使用树模型，将输入数据编码为特征。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
```

```
gbdtTrainOp = GbdtTrainBatchOp()\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setLabelCol('label')\
    .linkFrom(batchSource)

encoderBatchOp = TreeModelEncoderBatchOp()\
    .setPredictionCol("encoded_features")
encoderStreamOp = TreeModelEncoderStreamOp(gbdtTrainOp)\
    .setPredictionCol("encoded_features")

encoderBatchOp.linkFrom(gbdtTrainOp, batchSource).print()
encoderStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.TreeModelEncoderBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.feature.TreeModelEncoderStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;

import com.alibaba.alink.operator.batch.classification.GbdtTrainBatchOp;

import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TreeModelEncoderBatchOpTest {
    @Test
    public void testTreeModelEncoderBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator batchSource = new MemSourceBatchOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator streamSource = new MemSourceStreamOp(
```

```
            df, "f0 double, f1 string, f2 int, f3 int, label int");

        BatchOperator gbdtTrainOp = new GbdtTrainBatchOp()
            .setFeatureCols(new String[] {"f0", "f1", "f2", "f3"})
            .setLabelCol("label")
            .linkFrom(batchSource);

        BatchOperator encoderBatchOp = new TreeModelEncoderBatchOp()
            .setPredictionCol("encoded_features");
        StreamOperator encoderStreamOp =new
TreeModelEncoderStreamOp(gbdtTrainOp)
            .setPredictionCol("encoded_features");

        encoderBatchOp.linkFrom(gbdtTrainOp, batchSource).print();
        encoderStreamOp.linkFrom(streamSource).print();

        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$123$0:1.0 2:1.0 4:1.0 6:1.0 8:1.0 10:1.0 12:1.0 14:1.0 16:1.0 18:1.0 20:1.0 22:1.0 24:1.0 26:1.0 28:1.0 30:1.0 32:1.0 34:1.0 36:1.0 38:1.0 40:1.0 42:1.0 44:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0 120:1.0 121:1.0 122:1.0

2.0000	B	1	1	0	$123$0:1.0 2:1.0 4:1.0 6:1.0 8:1.0 10:1.0 12:1.0 14:1.0 16:1.0 18:1.0 20:1.0 22:1.0 24:1.0 26:1.0 28:1.0 30:1.0 32:1.0 34:1.0 36:1.0 38:1.0 40:1.0 42:1.0 44:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0 120:1.0 121:1.0 122:1.0
3.0000	C	2	2	1	$123$1:1.0 3:1.0 5:1.0 7:1.0 9:1.0 11:1.0 13:1.0 15:1.0 17:1.0 19:1.0 21:1.0 23:1.0 25:1.0 27:1.0 29:1.0 31:1.0 33:1.0 35:1.0 37:1.0 39:1.0 41:1.0 43:1.0 45:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0 120:1.0 121:1.0 122:1.0
4.0000	D	3	3	1	$123$1:1.0 3:1.0 5:1.0 7:1.0 9:1.0 11:1.0 13:1.0 15:1.0 17:1.0 19:1.0 21:1.0 23:1.0 25:1.0 27:1.0 29:1.0 31:1.0 33:1.0 35:1.0 37:1.0 39:1.0 41:1.0 43:1.0 45:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0 120:1.0 121:1.0 122:1.0

向量卡方选择器 (VectorChiSqSelectorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.feature.VectorChiSqSelectorBatchOp

Python 类名：VectorChiSqSelectorBatchOp

功能介绍

针对vector数据，进行特征筛选

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTO SPARSE_VECTO STRING, VECTO
fdr	发现阈值	发现阈值, 默认值0.05	Double		
fpr	p value的阈值	p value的阈值，默认值0.05	Double		
fwe	错误率阈值	错误率阈值, 默认值0.05	Double		
numTopFeatures	最大的p-value列个数	最大的p-value列个数, 默认值50	Integer		
percentile	筛选的百分比	筛选的百分比，默认值0.1	Double		

selectorType	筛选类型	筛选类型，包含"NumTopFeatures","percentile", "fpr", "fdr", "fwe"五种。	String		"NumTopFeature "PERCENTILE", "FPR", "FDR", "FWE"

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

无python接口

Java 代码

```
package javatest.com.alibaba.alink.batch.feature;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.VectorChiSqSelectorBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;

public class VectorChiSqSelectorBatchOpTest {

    @Test
    public void testVectorChiSqSelectorBatchOp() throws Exception {
        Row[] testArray = new Row[] {
            Row.of(7, "0.0  0.0  18.0  1.0", 1.0),
            Row.of(8, "0.0  1.0  12.0  0.0", 0.0),
            Row.of(9, "1.0  0.0  15.0  0.1", 0.0),
        };

        String[] colNames = new String[] {"id", "features", "clicked"};

        MemSourceBatchOp source = new
MemSourceBatchOp(Arrays.asList(testArray), colNames);

        VectorChiSqSelectorBatchOp test = new VectorChiSqSelectorBatchOp()
            .setSelectedCol("features")
            .setLabelCol("clicked");

        test.linkFrom(source);
```

```
        test.lazyPrintModelInfo();

        BatchOperator.execute();
    }

}
```

运行结果

```
------------------------- ChisqSelectorModelInfo -------------------------
Number of Selector Features: 4
Number of Features: 4
Type of Selector: NumTopFeatures
Number of Top Features: 50

Selector Indices:

    |VectorIndex|ChiSquare|PValue| DF|Selected|
    |-----------|---------|------|---|--------|
    |          3|        3|0.2231|  2|    true|
    |          2|        3|0.2231|  2|    true|
    |          0|     0.75|0.3865|  1|    true|
    |          1|     0.75|0.3865|  1|    true|
```

Bert文本嵌入 (BertTextEmbeddingBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.BertTextEmbeddingBatchOp

Python 类名：BertTextEmbeddingBatchOp

功能介绍

把文本输入到 BERT 模型，提取某一编码层的 pooled output 作为该句子的 embedding 结果。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
bertModelName	BERT模型名字	BERT模型名字：Base-Chinese,Base-Multilingual-Cased,Base-Uncased,Base-Cased	String			"Base-Chinese"
doLowerCase	是否将文本转换为小写	是否将文本转换为小写，默认根据模型自动决定	Boolean			null
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4
layer	输出第几层 encoder layer 的结果	输出第几层 encoder layer 的结果， -1 表示最后一层，-2 表示倒数第2层，以此类推	Integer			-1

maxSeqLength	句子截断长度	句子截断长度	Integer			128
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 'An english sentence.'],
    [2, '这是一个中文句子']
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 bigint, f2
string')

BertTextEmbeddingBatchOp() \
    .setSelectedCol("f2") \
    .setOutputCol("embedding") \
    .setLayer(-2) \
    .linkFrom(batch_data) \
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.common.MLEnvironmentFactory;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.BertTextEmbeddingBatchOp;
import org.junit.Test;

public class BertTextEmbeddingBatchOpTest {

    @Test
    public void testBertTextEmbeddingBatchOp() throws Exception {
        Row[] rows1 = new Row[] {
            Row.of(1L, "An english sentence."),
```

```
            Row.of(2L, "这是一个中文句子"),
        };

        BatchOperator <?> data = BatchOperator.fromTable(
            MLEnvironmentFactory.getDefault().createBatchTable(rows1, new
String[] {"f1", "f2"}));

        BertTextEmbeddingBatchOp bertEmb = new BertTextEmbeddingBatchOp()
            .setSelectedCol("f2").setOutputCol("embedding").setLayer(-2)
            .setDoLowerCase(true)
            .setIntraOpParallelism(4)
            .linkFrom(data);
        bertEmb.print();
    }
}
```

运行结果

f1	f2	embedding
1	An english sentence.	-0.4501993 0.06074004 0.121287264 -0.27875 0.3...
2	这是一个中文句子	-0.8317032 0.32284066 -0.12233654 -0.6955824 0...

文本特征生成预测 (DocCountVectorizerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.DocCountVectorizerPredictBatchOp

Python 类名：DocCountVectorizerPredictBatchOp

功能介绍

根据文本中词语的特征信息，将每条文本转化为稀疏向量。

使用方式

该组件是预测组件，需要配合训练组件 DocCountVectorizerTrainBatchOp 使用。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
```

```
    [4, u'二手郁达夫文集 （ 国内版 ） 全十二册馆藏书']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int, text string')

segment = SegmentBatchOp().setSelectedCol("text").linkFrom(inOp1)
train =
DocCountVectorizerTrainBatchOp().setSelectedCol("text").linkFrom(segment)
predictBatch =
DocCountVectorizerPredictBatchOp().setSelectedCol("text").linkFrom(train,
segment)
train.lazyPrint(-1)
predictBatch.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.DocCountVectorizerPredictBatchOp;
import com.alibaba.alink.operator.batch.nlp.DocCountVectorizerTrainBatchOp;
import com.alibaba.alink.operator.batch.nlp.SegmentBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DocCountVectorizerPredictBatchOpTest {
    @Test
    public void testDocCountVectorizerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集 （ 国内版 ） 全十二册馆藏书")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        BatchOperator <?> segment = new
SegmentBatchOp().setSelectedCol("text").linkFrom(inOp1);
        BatchOperator <?> train = new
DocCountVectorizerTrainBatchOp().setSelectedCol("text").linkFrom(segment);
        BatchOperator <?> predictBatch = new
DocCountVectorizerPredictBatchOp().setSelectedCol("text").linkFrom(train,
            segment);
```

```
        train.lazyPrint(-1);
        predictBatch.print();
    }
}
```

运行结果

模型数据

model_id	model_info
0	{"minTF":"1.0","featureType":"\"WORD_COUNT\""}
1048576	{"f0":"）","f1":0.6931471805599453,"f2":0}
2097152	{"f0":"（","f1":0.6931471805599453,"f2":1}
3145728	{"f0":"馆藏","f1":1.0986122886681098,"f2":2}
4194304	{"f0":"郁达夫","f1":1.0986122886681098,"f2":3}
5242880	{"f0":"选读","f1":1.0986122886681098,"f2":4}
6291456	{"f0":"象棋","f1":1.0986122886681098,"f2":5}
7340032	{"f0":"谢恩","f1":1.0986122886681098,"f2":6}
8388608	{"f0":"美国","f1":1.0986122886681098,"f2":7}
9437184	{"f0":"索引","f1":1.0986122886681098,"f2":8}
10485760	{"f0":"糖尿病","f1":1.0986122886681098,"f2":9}
11534336	{"f0":"电磁","f1":1.0986122886681098,"f2":10}
12582912	{"f0":"版","f1":1.0986122886681098,"f2":11}
13631488	{"f0":"正版","f1":1.0986122886681098,"f2":12}
14680064	{"f0":"李宜燮","f1":1.0986122886681098,"f2":13}
15728640	{"f0":"旧书","f1":1.0986122886681098,"f2":14}
16777216	{"f0":"文集","f1":1.0986122886681098,"f2":15}
17825792	{"f0":"文献","f1":1.0986122886681098,"f2":16}
18874368	{"f0":"文学","f1":1.0986122886681098,"f2":17}
19922944	{"f0":"成像","f1":1.0986122886681098,"f2":18}
20971520	{"f0":"思","f1":1.0986122886681098,"f2":19}
22020096	{"f0":"图解","f1":1.0986122886681098,"f2":20}

23068672	{"f0":"国内","f1":1.0986122886681098,"f2":21}
24117248	{"f0":"南开大学","f1":1.0986122886681098,"f2":22}
25165824	{"f0":"华龄","f1":1.0986122886681098,"f2":23}
26214400	{"f0":"十二册","f1":1.0986122886681098,"f2":24}
27262976	{"f0":"医学","f1":1.0986122886681098,"f2":25}
28311552	{"f0":"出版社","f1":0.6931471805599453,"f2":26}
29360128	{"f0":"全","f1":1.0986122886681098,"f2":27}
30408704	{"f0":"入门","f1":1.0986122886681098,"f2":28}
31457280	{"f0":"二手","f1":0.0,"f2":29}
32505856	{"f0":"书","f1":1.0986122886681098,"f2":30}
33554432	{"f0":"主编","f1":1.0986122886681098,"f2":31}
34603008	{"f0":"中国","f1":1.0986122886681098,"f2":32}
35651584	{"f0":"下册","f1":1.0986122886681098,"f2":33}
36700160	{"f0":":","f1":1.0986122886681098,"f2":34}
37748736	{"f0":"9787310003969","f1":1.0986122886681098,"f2":35}
38797312	{"f0":"/","f1":1.0986122886681098,"f2":36}

批预测结果

id	text
0	$37$10:1.0 14:1.0 18:1.0 25:1.0 29:1.0 34:1.0
1	$37$0:1.0 1:1.0 4:1.0 7:1.0 13:1.0 17:1.0 22:1.0 26:1.0 29:1.0 33:1.0 35:1.0
2	$37$5:1.0 6:1.0 12:1.0 19:1.0 20:1.0 23:1.0 26:1.0 28:1.0 29:1.0 31:1.0 36:2.0
3	$37$8:1.0 9:1.0 16:1.0 29:1.0 32:1.0
4	$37$0:1.0 1:1.0 2:1.0 3:1.0 11:1.0 15:1.0 21:1.0 24:1.0 27:1.0 29:1.0 30:1.0

文本特征生成训练 (DocCountVectorizerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.DocCountVectorizerTrainBatchOp

Python 类名：DocCountVectorizerTrainBatchOp

功能介绍

根据文本中词语的特征信息，将每条文本转化为稀疏向量。

使用方式

该组件是训练组件，需要配合预测组件 DocCountVectorizerPredictBatch/StreamOp 使用。

文本内容列（SelectedCol）中的内容用于统计词语的统计信息，需要是用空格分隔的词语。

将文本转换为稀疏向量时，每个唯一的词语将对应向量中的一个唯一的索引值。 而向量中对应索引的值表示这个词语在文本中的特征信息，可以通过参数 featureType 来选择不同的特征。

在转换时，所使用的词语集合还可以通过参数来进行筛选：

- maxDF/minDF：根据包含词语的文本次数（DF）进行筛选（当设置值在[0,1)区间时，表示占总文本数的比例）；
- minTF：仅在预测单条文本时起作用，根据词语在当前文本中的出现的次数进行筛选（当设置值在[0,1)区间时，表示占当前文本总次数的比例）；
- vocabSize：根据词语在所有文本中出现的总次数排序，只使用前 vocabSize 个词语。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]
featureType	特征类型	生成特征向量的类型，支持IDF/WORD_COUNT/TF_IDF/Binary/TF	String		"IDF", "WORD_COUNT", "TF_IDF", "BINARY", "TF"

maxDF	最大词频	如果一个词出现的文档次数大于maxDF,这个词不会被包含在字典中。maxDF可以是具体的词频也可以是整体词频的比例，如果minDF在[0,1)区间，会被认为是比例。	Double		
minDF	最小文档词频	如果一个词出现的文档次数小于minDF,这个词不会被包含在字典中。minTF可以是具体的词频也可以是整体词频的比例，如果minDF在[0,1)区间，会被认为是比例。	Double		
minTF	最低词频	最低词频，如果词频小于minTF,这个词会被忽略掉。minTF可以是具体的词频也可以是整体词频的比例，如果minTF在[0,1)区间，会被认为是比例。	Double		
vocabSize	字典库大小	字典库大小，如果总词数目大于这个值，那个文档频率低的词会被过滤掉。	Integer		

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集（ 国内版 ）全十二册馆藏书']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int, text string')

segment = SegmentBatchOp().setSelectedCol("text").linkFrom(inOp1)
train =
DocCountVectorizerTrainBatchOp().setSelectedCol("text").linkFrom(segment)
predictBatch =
DocCountVectorizerPredictBatchOp().setSelectedCol("text").linkFrom(train,
segment)
train.lazyPrint(-1)
predictBatch.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.DocCountVectorizerPredictBatchOp;
import com.alibaba.alink.operator.batch.nlp.DocCountVectorizerTrainBatchOp;
import com.alibaba.alink.operator.batch.nlp.SegmentBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DocCountVectorizerTrainBatchOpTest {
    @Test
    public void testDocCountVectorizerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集（ 国内版 ）全十二册馆藏书")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        BatchOperator <?> segment = new
SegmentBatchOp().setSelectedCol("text").linkFrom(inOp1);
        BatchOperator <?> train = new
DocCountVectorizerTrainBatchOp().setSelectedCol("text").linkFrom(segment);
        BatchOperator <?> predictBatch = new
DocCountVectorizerPredictBatchOp().setSelectedCol("text").linkFrom(train,
            segment);
        train.lazyPrint(-1);
        predictBatch.print();
    }
}
```

运行结果

模型数据

model_id	model_info
0	{"minTF":"1.0","featureType":"\"WORD_COUNT\""}
1048576	{"f0":"）","f1":0.6931471805599453,"f2":0}

2097152	{"f0":"（","f1":0.6931471805599453,"f2":1}
3145728	{"f0":"馆藏","f1":1.0986122886681098,"f2":2}
4194304	{"f0":"郁达夫","f1":1.0986122886681098,"f2":3}
5242880	{"f0":"选读","f1":1.0986122886681098,"f2":4}
6291456	{"f0":"象棋","f1":1.0986122886681098,"f2":5}
7340032	{"f0":"谢恩","f1":1.0986122886681098,"f2":6}
8388608	{"f0":"美国","f1":1.0986122886681098,"f2":7}
9437184	{"f0":"索引","f1":1.0986122886681098,"f2":8}
10485760	{"f0":"糖尿病","f1":1.0986122886681098,"f2":9}
11534336	{"f0":"电磁","f1":1.0986122886681098,"f2":10}
12582912	{"f0":"版","f1":1.0986122886681098,"f2":11}
13631488	{"f0":"正版","f1":1.0986122886681098,"f2":12}
14680064	{"f0":"李宜燮","f1":1.0986122886681098,"f2":13}
15728640	{"f0":"旧书","f1":1.0986122886681098,"f2":14}
16777216	{"f0":"文集","f1":1.0986122886681098,"f2":15}
17825792	{"f0":"文献","f1":1.0986122886681098,"f2":16}
18874368	{"f0":"文学","f1":1.0986122886681098,"f2":17}
19922944	{"f0":"成像","f1":1.0986122886681098,"f2":18}
20971520	{"f0":"思","f1":1.0986122886681098,"f2":19}
22020096	{"f0":"图解","f1":1.0986122886681098,"f2":20}
23068672	{"f0":"国内","f1":1.0986122886681098,"f2":21}
24117248	{"f0":"南开大学","f1":1.0986122886681098,"f2":22}
25165824	{"f0":"华龄","f1":1.0986122886681098,"f2":23}
26214400	{"f0":"十二册","f1":1.0986122886681098,"f2":24}
27262976	{"f0":"医学","f1":1.0986122886681098,"f2":25}
28311552	{"f0":"出版社","f1":0.6931471805599453,"f2":26}
29360128	{"f0":"全","f1":1.0986122886681098,"f2":27}
30408704	{"f0":"入门","f1":1.0986122886681098,"f2":28}

31457280	{"f0":"二手","f1":0.0,"f2":29}
32505856	{"f0":"书","f1":1.0986122886681098,"f2":30}
33554432	{"f0":"主编","f1":1.0986122886681098,"f2":31}
34603008	{"f0":"中国","f1":1.0986122886681098,"f2":32}
35651584	{"f0":"下册","f1":1.0986122886681098,"f2":33}
36700160	{"f0":":","f1":1.0986122886681098,"f2":34}
37748736	{"f0":"9787310003969","f1":1.0986122886681098,"f2":35}
38797312	{"f0":"/","f1":1.0986122886681098,"f2":36}

批预测结果

id	text
0	$37$10:1.0 14:1.0 18:1.0 25:1.0 29:1.0 34:1.0
1	$37$0:1.0 1:1.0 4:1.0 7:1.0 13:1.0 17:1.0 22:1.0 26:1.0 29:1.0 33:1.0 35:1.0
2	$37$5:1.0 6:1.0 12:1.0 19:1.0 20:1.0 23:1.0 26:1.0 28:1.0 29:1.0 31:1.0 36:2.0
3	$37$8:1.0 9:1.0 16:1.0 29:1.0 32:1.0
4	$37$0:1.0 1:1.0 2:1.0 3:1.0 11:1.0 15:1.0 21:1.0 24:1.0 27:1.0 29:1.0 30:1.0

文本哈希特征生成预测 (DocHashCountVectorizerPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.DocHashCountVectorizerPredictBatchOp

Python 类名：DocHashCountVectorizerPredictBatchOp

功能介绍

根据文本中词语的特征信息，将每条文本转化为固定长度的稀疏向量。

在转换时，每个词语会通过哈希函数映射到稀疏向量的一个索引值，映射到同一个索引值的多个词语将看作同一个词语来统计特征信息。

使用方式

该组件是预测组件，需要配合训练组件 DocHashCountVectorizerTrainBatchOp 使用。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集 （ 国内版 ） 全十二册馆藏书']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int, text string')

segment = SegmentBatchOp().setSelectedCol("text").linkFrom(inOp1)
train =
DocHashCountVectorizerTrainBatchOp().setSelectedCol("text").linkFrom(segment)
predictBatch =
DocHashCountVectorizerPredictBatchOp().setSelectedCol("text").linkFrom(train,
segment)
train.lazyPrint(-1)
predictBatch.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.nlp.DocHashCountVectorizerPredictBatchOp;
import com.alibaba.alink.operator.batch.nlp.DocHashCountVectorizerTrainBatchOp;
import com.alibaba.alink.operator.batch.nlp.SegmentBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DocHashCountVectorizerPredictBatchOpTest {
    @Test
    public void testDocHashCountVectorizerPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集 （ 国内版 ） 全十二册馆藏书")
```

```
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        BatchOperator <?> segment = new
SegmentBatchOp().setSelectedCol("text").linkFrom(inOp1);
        BatchOperator <?> train = new
DocHashCountVectorizerTrainBatchOp().setSelectedCol("text").linkFrom(segment);
        BatchOperator <?> predictBatch = new
DocHashCountVectorizerPredictBatchOp().setSelectedCol("text").linkFrom(
            train, segment);
        train.lazyPrint(-1);
        predictBatch.print();
    }
}
```

运行结果

模型数据

model_id	
0	{"numFeatures":"262144","minTF":"1.0","featureType":"\"WORD_COUNT\""}
1048576	{"0":-0.6061358035703156,"37505":1.0986122886681098,"180035":1.0986122886681098,"214

批预测结果

id	text
0	$262144$10121:1.0 64444:1.0 119456:1.0 206232:1.0 210357:1.0 256946:1.0
1	$262144$0:6.0 37505:1.0 46743:1.0 93228:1.0 119217:1.0 138080:1.0 141480:1.0 172901:1.0 206232:1.0 216139:1.0 226698:1.0 254628:1.0
2	$262144$40170:1.0 70777:1.0 96509:1.0 126159:1.0 158267:1.0 181703:1.0 206232:1.0 216139:1.0 232884:1.0 250534:2.0 259932:1.0
3	$262144$206232:1.0 214785:1.0 251090:1.0 255656:1.0 261064:1.0
4	$262144$0:4.0 87711:1.0 138080:1.0 162140:1.0 180035:1.0 195777:1.0 206232:1.0 219988:1.0 241122:1.0 254628:1.0 257763:1.0 259051:1.0

文本哈希特征生成训练 (DocHashCountVectorizerTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.DocHashCountVectorizerTrainBatchOp

Python 类名：DocHashCountVectorizerTrainBatchOp

功能介绍

根据文本中词语的特征信息，将每条文本转化为固定长度的稀疏向量。

在转换时，每个词语会通过哈希函数映射到稀疏向量的一个索引值，映射到同一个索引值的多个词语将看作同一个词语来统计特征信息。

使用方式

该组件是训练组件，需要配合预测组件 DocHashCountVectorizerPredictBatch/StreamOp 使用。

文本内容列（SelectedCol）中的内容用于统计词语的统计信息，需要是用空格分隔的词语。

将文本转换为稀疏向量时，需要指定向量维度（numFeatures）。 每个词语会通过哈希函数映射到一个 [0, numFeatures) 内的索引值，映射到同一个索引值的多个词语将看作同一个词语来统计特征信息。 而向量中对应索引的值表示对应的词语在文本中的特征信息，可以通过参数 featureType 来选择不同的特征。

在转换时，所使用的词语集合还可以通过参数来进行筛选：

- maxDF/minDF：根据包含词语的文本次数（DF）进行筛选（当设置值在[0,1)区间时，表示占总文本数的比例）；
- minTF：仅在预测单条文本时起作用，根据词语在当前文本中的出现的次数进行筛选（当设置值在[0,1)区间时，表示占当前文本总次数的比例）。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]
featureType	特征类型	生成特征向量的类型，支持IDF/WORD_COUNT/TF_IDF/Binary/TF	String		"IDF", "WORD_COUNT" "TF_IDF", "BINARY", "TF"

minDF	最小文档词频	如果一个词出现的文档次数小于minDF,这个词不会被包含在字典中。minTF可以是具体的词频也可以是整体词频的比例，如果minDF在[0,1)区间，会被认为是比例。	Double		
minTF	最低词频	最低词频，如果词频小于minTF,这个词会被忽略掉。minTF可以是具体的词频也可以是整体词频的比例，如果minTF在[0,1)区间，会被认为是比例。	Double		
numFeatures	向量维度	生成向量长度	Integer		

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集（ 国内版 ）全十二册馆藏书']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int, text string')

segment = SegmentBatchOp().setSelectedCol("text").linkFrom(inOp1)
train =
DocHashCountVectorizerTrainBatchOp().setSelectedCol("text").linkFrom(segment)
predictBatch =
DocHashCountVectorizerPredictBatchOp().setSelectedCol("text").linkFrom(train,
segment)
train.lazyPrint(-1)
predictBatch.print()

inOp2 = StreamOperator.fromDataframe(df, schemaStr='id int, text string')

segment2 = SegmentStreamOp().setSelectedCol("text").linkFrom(inOp2)
predictStream =
DocHashCountVectorizerPredictStreamOp(train).setSelectedCol("text").linkFrom(se
gment2)
```

```
predictStream.print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.nlp.DocHashCountVectorizerPredictBatchOp;
import com.alibaba.alink.operator.batch.nlp.DocHashCountVectorizerTrainBatchOp;
import com.alibaba.alink.operator.batch.nlp.SegmentBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.nlp.DocHashCountVectorizerPredictStreamOp;
import com.alibaba.alink.operator.stream.nlp.SegmentStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DocHashCountVectorizerTrainBatchOpTest {
    @Test
    public void testDocHashCountVectorizerTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集（ 国内版 ）全十二册馆藏书")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        BatchOperator <?> segment = new
SegmentBatchOp().setSelectedCol("text").linkFrom(inOp1);
        BatchOperator <?> train = new
DocHashCountVectorizerTrainBatchOp().setSelectedCol("text").linkFrom(segment);
        BatchOperator <?> predictBatch = new
DocHashCountVectorizerPredictBatchOp().setSelectedCol("text").linkFrom(
            train, segment);
        train.lazyPrint(-1);
        predictBatch.print();
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text
string");
        StreamOperator <?> segment2 = new
```

```
SegmentStreamOp().setSelectedCol("text").linkFrom(inOp2);
        StreamOperator <?> predictStream = new
DocHashCountVectorizerPredictStreamOp(train).setSelectedCol("text")
            .linkFrom(segment2);
        predictStream.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型数据

model_id	
0	{"numFeatures":"262144","minTF":"1.0","featureType":"\"WORD_COUNT\""}
1048576	{"0":-0.6061358035703156,"180035":1.0986122886681098,"37505":1.0986122886681098,"214

批预测结果

id	text
0	$262144$10121:1.0 64444:1.0 119456:1.0 206232:1.0 210357:1.0 256946:1.0
1	$262144$0:6.0 37505:1.0 46743:1.0 93228:1.0 119217:1.0 138080:1.0 141480:1.0 172901:1.0 206232:1.0 216139:1.0 226698:1.0 254628:1.0
2	$262144$40170:1.0 70777:1.0 96509:1.0 126159:1.0 158267:1.0 181703:1.0 206232:1.0 216139:1.0 232884:1.0 250534:2.0 259932:1.0
3	$262144$206232:1.0 214785:1.0 251090:1.0 255656:1.0 261064:1.0
4	$262144$0:4.0 87711:1.0 138080:1.0 162140:1.0 180035:1.0 195777:1.0 206232:1.0 219988:1.0 241122:1.0 254628:1.0 257763:1.0 259051:1.0

流预测结果

id	text
4	$262144$0:4.0 87711:1.0 138080:1.0 162140:1.0 180035:1.0 195777:1.0 206232:1.0 219988:1.0 241122:1.0 254628:1.0 257763:1.0 259051:1.0
3	$262144$206232:1.0 214785:1.0 251090:1.0 255656:1.0 261064:1.0
1	$262144$0:6.0 37505:1.0 46743:1.0 93228:1.0 119217:1.0 138080:1.0 141480:1.0 172901:1.0 206232:1.0 216139:1.0 226698:1.0 254628:1.0
0	$262144$10121:1.0 64444:1.0 119456:1.0 206232:1.0 210357:1.0 256946:1.0

2	$262144$40170:1.0 70777:1.0 96509:1.0 126159:1.0 158267:1.0 181703:1.0 206232:1.0 216139:1.0 232884:1.0 250534:2.0 259932:1.0

文档向量生成 (DocVecFromWordVecBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.DocVecFromWordVecBatchOp

Python 类名：DocVecFromWordVecBatchOp

功能介绍

DocVecFromWordVec是根据word2vec的结果和文档的分词结果，将文档转成向量，向量维数保持与词的维数一致，同时每个维度通过对文档中的词求平均或者最大或者最小取得。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
predMethod	向量组合方法	预测文档向量时，需要用到的方法。支持三种方法：平均（avg），最小（min）和最大（max），默认值为平均	String		"AVG", "SUM", "MIN", "MAX"	"AVG"

wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集（ 国内版 ）全十二册馆藏书']
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='id long, text string')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='id long, text string')

segment = SegmentBatchOp()\
    .setSelectedCol('text')\
    .setOutputCol('segment')

segmentStream = SegmentStreamOp()\
    .setSelectedCol('text')\
    .setOutputCol('segment')

w2v = Word2VecTrainBatchOp()\
    .setSelectedCol('segment')\
    .setVectorSize(2)\
    .setMinCount(1)

docvec = DocVecFromWordVecBatchOp()\
    .setSelectedCol('segment')\
    .setPredMethod('avg')\
    .setOutputCol('output')
```

```
segmented = batchSource.link(segment)
model = segmented.link(w2v)
docvec.linkFrom(model, segmented).print()

docvecStream = DocVecFromWordVecStreamOp(model)\
    .setSelectedCol('segment')\
    .setPredMethod('avg')\
    .setOutputCol('output')

docvecStream.linkFrom(streamSource.link(segmentStream)).print()

StreamOperator.execute()
```

运行结果

```
   id                                       text  \
0   0                                二手旧书:医学电磁成像
1   1  二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969
2   2                    二手正版图解象棋入门/谢恩思主编/华龄出版社
3   3                                二手中国糖尿病文献索引
4   4                     二手郁达夫文集（ 国内版 ）全十二册馆藏书

                                          segment  \
0                                   二手 旧书 : 医学 电磁 成像
1  二手 美国 文学 选读 （  下册  ） 李宜燮 南开大学 出版社  97873100...
2                  二手 正版 图解 象棋 入门 / 谢恩 思 主编 / 华龄 出版社
3                                   二手 中国 糖尿病 文献 索引
4                  二手 郁达夫 文集 （  国内 版  ） 全 十二册 馆藏 书

                                  output
0  0.4896821898767483 0.45953670334632435
1  0.38892360975895096 0.5566970765129314
2   0.5599209241585106 0.4733199256278301
3  0.4992423120206364 0.41816740880009445
4   0.5056449699037094 0.6150642543802489
```

词云 (DocWordCloudBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.DocWordCloudBatchOp

Python 类名：DocWordCloudBatchOp

功能介绍

输入包含文章合集的表，该组件能计算文章合集中的高频词，并产生词云可视化。

组件名称：com.alibaba.alink.batchoperator.nlp.DocWordCloudBatchOp

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
topN	取前n个	取前n个	Integer		[1, +inf)	100
wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1, "I wish I can grow up quickly"],
    [2, "present in two or more JARS."],
    [3, "I wish I can grow up more quickly"],
    [4, "present in two or more more more JARS."],
    [5, "I wish I can grow up quickly"],
    [6, "present present in two two two or more JARS."],
])

data = BatchOperator.fromDataframe(df, schemaStr="id long, sentence string")
```

```
wordCountBatchOp = DocWordCloudBatchOp().setSelectedCol("sentence")
wordCountBatchOp.linkFrom(data).print()
```

运行结果

```
       word  count
0         I      6
1      more      6
2       two      5
3   present      4
4     JARS.      3
5        in      3
6        or      3
7      wish      3
8       can      3
9      grow      3
10  quickly      3
11       up      3
```

文本词频统计 (DocWordCountBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.DocWordCountBatchOp

Python 类名：DocWordCountBatchOp

功能介绍

对文本列按行统计词语的出现的频数。

使用方式

文本内容列（ContentCol）中的内容用于统计词频，需要是用分隔符分隔的词语。 其中，分隔符可以通过参数 wordDelimiter 来设置，默认是空格（" "），可以为正则表达式。 文本内容列可以使用分词（SegmentBatchOp）组件的输出结果列，同时也可以在之前接入停用词过滤（StopWordsRemoverBatchOp）组件去掉常见的高频词。

文本 ID 列（DocIdCol）用于标识每一行数据，从而可以将输出结果中的词语、频数数据与输入行对应起来。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
contentCol	文本列	文本列名	String	✓	所选列类型为[STRING]	
docIdCol	文档ID列	文档ID列名	String	✓		
wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
```

```
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集（ 国内版 ）全十二册馆藏书']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int, text string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id int, text string')

segment =
SegmentBatchOp().setSelectedCol("text").setOutputCol("segment").linkFrom(inOp1)
remover =
StopWordsRemoverBatchOp().setSelectedCol("segment").setOutputCol("remover").lin
kFrom(segment)
wordCount =
DocWordCountBatchOp().setContentCol("remover").setDocIdCol("id").linkFrom(remov
er)
wordCount.print()

segment2 =
SegmentStreamOp().setSelectedCol("text").setOutputCol("segment").linkFrom(inOp2
)
remover2 =
StopWordsRemoverStreamOp().setSelectedCol("segment").setOutputCol("remover").li
nkFrom(segment2)
wordCount2 =
DocWordCountStreamOp().setContentCol("remover").setDocIdCol("id").linkFrom(remo
ver2)
wordCount2.print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.DocWordCountBatchOp;
import com.alibaba.alink.operator.batch.nlp.SegmentBatchOp;
import com.alibaba.alink.operator.batch.nlp.StopWordsRemoverBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.nlp.DocWordCountStreamOp;
import com.alibaba.alink.operator.stream.nlp.SegmentStreamOp;
import com.alibaba.alink.operator.stream.nlp.StopWordsRemoverStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class DocWordCountBatchOpTest {
    @Test
    public void testDocWordCountBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集 （ 国内版 ） 全十二册馆藏书")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text
string");
        BatchOperator <?> segment =
            new
SegmentBatchOp().setSelectedCol("text").setOutputCol("segment").linkFrom(inOp1)
;
        BatchOperator <?> remover = new
StopWordsRemoverBatchOp().setSelectedCol("segment").setOutputCol("remover")
            .linkFrom(segment);
        BatchOperator <?> wordCount = new
DocWordCountBatchOp().setContentCol("remover").setDocIdCol("id").linkFrom(
            remover);
        wordCount.print();
        StreamOperator <?> segment2 = new
SegmentStreamOp().setSelectedCol("text").setOutputCol("segment").linkFrom(
            inOp2);
        StreamOperator <?> remover2 = new
StopWordsRemoverStreamOp().setSelectedCol("segment").setOutputCol("remover")
            .linkFrom(segment2);
        StreamOperator <?> wordCount2 = new
DocWordCountStreamOp().setContentCol("remover").setDocIdCol("id").linkFrom(
            remover2);
        wordCount2.print();
        StreamOperator.execute();
    }
}
```

运行结果

批运行结果

id	word	cnt
0	医学	1

0	电磁	1
0	成像	1
0	旧书	1
0	二手	1
1	美国	1
1	出版社	1
1	选读	1
1	文学	1
1	二手	1
1	下册	1
1	南开大学	1
1	9787310003969	1
1	李宜燮	1
2	出版社	1
2	主编	1
2	谢恩	1
2	二手	1
2	正版	1
2	入门	1
2	象棋	1
2	华龄	1
2	思	1
2	图解	1
3	中国	1
3	文献	1
3	索引	1
3	糖尿病	1
3	二手	1

4	国内	1
4	十二册	1
4	文集	1
4	书	1
4	二手	1
4	全	1
4	版	1
4	馆藏	1
4	郁达夫	1

流运行结果

id	word	cnt
4	国内	1
4	十二册	1
4	文集	1
4	书	1
4	二手	1
4	全	1
4	版	1
4	馆藏	1
4	郁达夫	1
3	中国	1
3	文献	1
3	索引	1
3	糖尿病	1
3	二手	1
1	美国	1
1	出版社	1
1	选读	1

1	文学	1
1	二手	1
1	下册	1
1	南开大学	1
1	9787310003969	1
1	李宜燮	1
0	医学	1
0	电磁	1
0	成像	1
0	旧书	1
0	二手	1
2	出版社	1
2	主编	1
2	谢恩	1
2	二手	1
2	正版	1
2	入门	1
2	象棋	1
2	华龄	1
2	思	1
2	图解	1

关键词抽取 (KeywordsExtractionBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.KeywordsExtractionBatchOp

Python 类名：KeywordsExtractionBatchOp

功能介绍

从每行文本中抽取与文本意义最相关的若干词语。

算法原理

组件支持两种抽取关键词的方法：基于 TextRank 和基于 TF-IDF。

TextRank

TextRank 受到网页间关系的 PageRank 算法启发，利用局部词汇之间关系（共现窗口）构建图，计算词的重要性，选取权重大的作为关键词。

在构建的图中，每个词语对应一个节点 $V{.}$。 *两个不同的词语 i, j 只要在同一个窗口中共同出现过，对应节点间就存在两条有向边 $e*{ij}$ 和 $e{ji}$，权重分别为 1，即 $w*{ij} = w_{ji} = 1$。*

每个节点初始重要性值 $WS(V*{*}) = \frac{1}{|Out(V{*})|}$，并按照下面公式进行迭代更新直至收敛：

$$WS(V*i) = (1 - d) + d * \Sigma*{Vj\in In(V_i)} \frac{w*{ji}}{\Sigma*{V_k\in Out(V_j)}w*{jk} }WS(V_j).$$

其中，d 是阻尼系数。

TF-IDF

同时考虑所有文本，计算每个词语的 TF-IDF 值。然后在每行文本中，选取最大的若干个作为关键词。

使用方式

文本列通过参数 selectedCol 指定，需要是空格分隔的词语。 文本列可以使用分词（SegmentBatchOp）组件的输出结果列，同时也可以在之前接入停用词过滤（StopWordsRemoverBatchOp）组件去掉常见的高频词。

使用的提取方法通过参数 method 在指定，提取的关键词数目通过参数 topN 指定。

当使用基于 TextRank 的方法时，需要设置窗口大小 windowSize、最大迭代步数 maxIter、收敛阈值 epsilon 和阻尼稀疏 dampingFactor。

使用基于 TF-IDF 的方法不需要指定其他参数。

文献索引

TextRank：https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

TF-IDF：https://en.wikipedia.org/wiki/Tf%E2%80%93idf

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
dampingFactor	阻尼系数	阻尼系数	Double			0.85
epsilon	收敛阈值	收敛阈值	Double			1.0E-6
maxIter	最大迭代步数	最大迭代步数，默认为 100	Integer		[1, +inf)	100
method	抽取关键词的方法	抽取关键词的方法，支持TF_IDF和TEXT_RANK	String		"TEXT_RANK", "TF_IDF"	"TEXT_RANK"
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
topN	前N的数据	挑选最近的N个数据	Integer		[1, +inf)	10

windowSize	窗口大小	窗口大小	Integer		[1, +inf)	2

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集（ 国内版 ）全十二册馆藏书']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int, text string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id int, text string')

segment = SegmentBatchOp().setSelectedCol("text").linkFrom(inOp1)
remover = StopWordsRemoverBatchOp().setSelectedCol("text").linkFrom(segment)
keywords =
KeywordsExtractionBatchOp().setSelectedCol("text").setMethod("TF_IDF").setTopN(
3).linkFrom(remover)
keywords.print()

segment2 = SegmentStreamOp().setSelectedCol("text").linkFrom(inOp2)
remover2 = StopWordsRemoverStreamOp().setSelectedCol("text").linkFrom(segment2)
keywords2 =
KeywordsExtractionStreamOp().setSelectedCol("text").setTopN(3).linkFrom(remover
2)
keywords2.print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.KeywordsExtractionBatchOp;
import com.alibaba.alink.operator.batch.nlp.SegmentBatchOp;
import com.alibaba.alink.operator.batch.nlp.StopWordsRemoverBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.nlp.KeywordsExtractionStreamOp;
```

```
import com.alibaba.alink.operator.stream.nlp.SegmentStreamOp;
import com.alibaba.alink.operator.stream.nlp.StopWordsRemoverStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KeywordsExtractionBatchOpTest {
    @Test
    public void testKeywordsExtractionBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集 （ 国内版 ） 全十二册馆藏书")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text
string");
        BatchOperator <?> segment = new
SegmentBatchOp().setSelectedCol("text").linkFrom(inOp1);
        BatchOperator <?> remover = new
StopWordsRemoverBatchOp().setSelectedCol("text").linkFrom(segment);
        BatchOperator <?> keywords =
            new
KeywordsExtractionBatchOp().setSelectedCol("text").setMethod("TF_IDF").setTopN(
                3).linkFrom(remover);
        keywords.print();
        StreamOperator <?> segment2 = new
SegmentStreamOp().setSelectedCol("text").linkFrom(inOp2);
        StreamOperator <?> remover2 = new
StopWordsRemoverStreamOp().setSelectedCol("text").linkFrom(segment2);
        StreamOperator <?> keywords2 = new
KeywordsExtractionStreamOp().setSelectedCol("text").setTopN(3).linkFrom(
            remover2);
        keywords2.print();
        StreamOperator.execute();
    }
}
```

运行结果

批运行结果

text	id
索引 糖尿病 文献	3
旧书 成像 医学	0
李宜燮 9787310003969 美国	1
华龄 思 谢恩	2
国内 十二册 书	4

流运行结果

id	text
3	中国 文献 糖尿病
4	郁达夫 馆藏 文集
0	旧书 电磁 医学
1	美国 出版社 文学
2	正版 华龄 图解

标签Word2Vec (LabeledWord2VecBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.LabeledWord2VecBatchOp

Python 类名：LabeledWord2VecBatchOp

功能介绍

Word2Vec是Google在2013年开源的一个将词表转为向量的算法，其利用神经网络，可以通过训练，将词映射到K维度空间向量，甚至对于表示词的向量进行操作还能和语义相对应，由于其简单和高效引起了很多人的关注。

Google Word2Vec的工具包相关链接：https://code.google.com/p/word2vec/

支持metapath2vec++训练：metapath2vec: Scalable Representation Learning forHeterogeneous Networks

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	计算列对应的列名	计算列对应的列名	String	✓	所选列类型为 [STRING]	
typeCol	节点类型列名	用来指定节点类型列	String	✓		
vertexCol	节点列名	用来指定节点列	String	✓	所选列类型为 [STRING]	
alpha	学习率	学习率	Double			0.025
batchSize	batch大小	batch大小, 按行计算	Integer		[1, +inf)	
minCount	最小词频	最小词频	Integer			5
negative	负采样大小	负采样大小	Integer			5
numIter	迭代次数	迭代次数，默认为1。	Integer			1
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"true"
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	100
window	窗口大小	窗口大小	Integer			5

wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["Bob Lucy Bella"]
])

source = BatchOperator.fromDataframe(df, schemaStr="tokens string")

df = pd.DataFrame([
    ["Bob", "A"],
    ["Bella", "A"],
    ["Karry", "A"],
    ["Lucy", "B"],
    ["Alice", "B"],
    ["Lisa", "B"]
])

type = BatchOperator.fromDataframe(df, schemaStr="node string, type string")

labeledWord2vecBatchOp = LabeledWord2VecBatchOp() \
  .setSelectedCol("tokens")         \
  .setVertexCol("node")             \
  .setTypeCol("type")               \
  .setMinCount(1)                   \
  .setVectorSize(4)
labeledWord2vecBatchOp.linkFrom(source, type).print()
```

运行结果

word	vec
Lucy	0.03437602147459984,-0.04761518910527229,0.012536839582026005,-0.09563367068767548
Bob	0.057709891349077225,0.08290477842092514,-0.06487766653299332,0.026675613597035408

Bella	0.02439533919095993,0.07039660215377808,-0.04170553758740425,-0.061801809817552567

NGram (NGramBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.NGramBatchOp

Python 类名：NGramBatchOp

功能介绍

对每行文本生成对应的 NGram 结果。

算法原理

N-Gram 是一种基于统计语言模型的算法，它将文本里面的内容进行大小为 N 的滑动窗口操作，形成了长度是 N 的片段序列。

使用方式

该组件对于文本内容列（SelectedCol）中的每一行文本进行 N-Gram 处理，产生一行输出。 N 的值通过参数 n 设置。

输入每行文本中各个词语间需要以空格进行分割，可以使用分词（SegmentBatchOp）组件的输出结果列。 输出一行中包含多个 N-Gram 结果，各个结果间用空格分隔；单个结果中各个词语间用下划线连接。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
n	nGram长度	nGram长度	Integer			2
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, 'That is an English Book!'],
    [1, 'Do you like math?'],
    [2, 'Have a good day!']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id long, text string')

op = NGramBatchOp().setSelectedCol("text")
op.linkFrom(inOp1).print()

inOp2 = StreamOperator.fromDataframe(df, schemaStr='id long, text string')
op2 = NGramStreamOp().setSelectedCol("text")
op2.linkFrom(inOp2).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.NGramBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.nlp.NGramStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class NGramBatchOpTest {
    @Test
    public void testNGramBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "That is an English Book!"),
            Row.of(1, "Do you like math?"),
            Row.of(2, "Have a good day!")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        BatchOperator <?> op = new NGramBatchOp().setSelectedCol("text");
        op.linkFrom(inOp1).print();
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text
string");
        StreamOperator <?> op2 = new NGramStreamOp().setSelectedCol("text");
```

```
        op2.linkFrom(inOp2).print();
        StreamOperator.execute();
    }
}
```

运行结果

批运行结果

id	text
0	That_is is_an an_English English_Book!
1	Do_you you_like like_math?
2	Have_a a_good good_day!

流运行结果

id	text
1	Do_you you_like like_math?
2	Have_a a_good good_day!
0	That_is is_an an_English English_Book!

RegexTokenizer (RegexTokenizerBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.RegexTokenizerBatchOp

Python 类名：RegexTokenizerBatchOp

功能介绍

通过正则表达式对文本进行切分或者匹配操作。

使用方式

文本列通过参数 selectedCol 指定，切分或者匹配用的正则表达式通过参数 pattern 指定。

当参数 gaps 为 True 时，对文本进行切分操作（类似于分词）； 当参数 gaps 为 为 False 时，将提取匹配正则表达式的词语。

对于处理后的结果，还可以通过参数 minTokenLength 根据长度筛掉词语，或者通过参数 toLowerCase 将所有词语转为小写。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
gaps	切分/匹配	如果gaps为True，pattern用于切分文档；如果gaps为False，会提取出匹配pattern的词。	Boolean			true
minTokenLength	词语最短长度	词语的最短长度，小于这个值的词语会被过滤掉	Integer			1
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
pattern	分隔符/正则匹配符	如果gaps为True，pattern用于切分文档；如果gaps为False，会提取出匹配pattern的词。	String			"\s+"

reservedCols	算法保留列名	算法保留列	String[]			null
toLowerCase	是否转换为小写	转换为小写	Boolean			true
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, 'That is an English Book!'],
    [1, 'Do you like math?'],
    [2, 'Have a good day!']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id long, text string')
op =
RegexTokenizerBatchOp().setSelectedCol("text").setGaps(False).setToLowerCase(Tr
ue).setOutputCol(
    "token").setPattern("\\w+")

op.linkFrom(inOp1).print()

inOp2 = StreamOperator.fromDataframe(df, schemaStr='id long, text string')
op2 =
RegexTokenizerStreamOp().setSelectedCol("text").setGaps(False).setToLowerCase(T
rue).setOutputCol(
    "token").setPattern("\\w+")
op2.linkFrom(inOp2).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;
```

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.RegexTokenizerBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.nlp.RegexTokenizerStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RegexTokenizerBatchOpTest {
    @Test
    public void testRegexTokenizerBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "That is an English Book!"),
            Row.of(1, "Do you like math?"),
            Row.of(2, "Have a good day!")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        BatchOperator <?> op = new
RegexTokenizerBatchOp().setSelectedCol("text").setGaps(false).setToLowerCase(tr
ue)
            .setOutputCol("token").setPattern("\\w+");
        op.linkFrom(inOp1).print();
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text
string");
        StreamOperator <?> op2 =
            new
RegexTokenizerStreamOp().setSelectedCol("text").setGaps(false).setToLowerCase(t
rue)
                .setOutputCol("token").setPattern("\\w+");
        op2.linkFrom(inOp2).print();
        StreamOperator.execute();
    }
}
```

运行结果

批运行结果

id	text	token
0	That is an English Book!	that is an english book
1	Do you like math?	do you like math

2	Have a good day!	have a good day

流运行结果

id	text	token
0	That is an English Book!	that is an english book
1	Do you like math?	do you like math
2	Have a good day!	have a good day

分词 (SegmentBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.SegmentBatchOp

Python 类名：SegmentBatchOp

功能介绍

对文本进行分词，分词后各个词语间用空格分隔。

使用方式

文本列通过参数 selectedCol 指定。 词典文件可以从这里 查看。 通过参数 userDefinedDict 可以添加额外的词语。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为 [STRING]	
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
userDefinedDict	用户自定义字典	用户自定义字典	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
```

```
    [4, u'二手郁达夫文集 （ 国内版 ） 全十二册馆藏书']
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text string')

SegmentBatchOp() \
    .setSelectedCol("text") \
    .setOutputCol("segment") \
    .linkFrom(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.SegmentBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SegmentBatchOpTest {
    @Test
    public void testSegmentBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集 （ 国内版 ） 全十二册馆藏书")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text
string");
        new SegmentBatchOp()
            .setSelectedCol("text")
            .setOutputCol("segment")
            .linkFrom(inOp).print();
    }
}
```

运行结果

id	text	segment
0	二手旧书:医学电磁成像	二手 旧书 : 医学 电磁 成像

1	二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969	二手 美国 文学 选读 （ 下册 ） 李宜燮 南开大学 出版社 9787310003969
2	二手正版图解象棋入门/谢恩思主编/华龄出版社	二手 正版 图解 象棋 入门 / 谢恩 思 主编 / 华龄 出版社
3	二手中国糖尿病文献索引	二手 中国 糖尿病 文献 索引
4	二手郁达夫文集（ 国内版 ）全十二册馆藏书	二手 郁达夫 文集 （ 国内 版 ） 全 十二册 馆藏 书

停用词过滤 (StopWordsRemoverBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.StopWordsRemoverBatchOp

Python 类名：StopWordsRemoverBatchOp

功能介绍

过滤文本中的噪声词语（例如：的、是、啊等）。

使用方式

文本列通过参数 selectedCol 指定，需要是空格分隔的词语，可以使用分词（SegmentBatchOp）组件的输出结果列。 可以通过参数 caseSensitive 设置过滤时是否大小写敏感。

噪声词表可以从这里 查看。 通过参数 stopWords 可以添加额外的噪声词语。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为 [STRING]	
caseSensitive	是否大小写敏感	大小写敏感	Boolean			false
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
stopWords	用户自定义停用词表	用户自定义停用词表	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集（ 国内版 ）全十二册馆藏书']])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int, text string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id int, text string')

segment =
SegmentBatchOp().setSelectedCol("text").setOutputCol("segment").linkFrom(inOp1)
remover =
StopWordsRemoverBatchOp().setSelectedCol("segment").setOutputCol("remover").lin
kFrom(segment)
remover.print()

segment2 =
SegmentStreamOp().setSelectedCol("text").setOutputCol("segment").linkFrom(inOp2
)
remover2 =
StopWordsRemoverStreamOp().setSelectedCol("segment").setOutputCol("remover").li
nkFrom(segment2)
remover2.print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.SegmentBatchOp;
import com.alibaba.alink.operator.batch.nlp.StopWordsRemoverBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.nlp.SegmentStreamOp;
import com.alibaba.alink.operator.stream.nlp.StopWordsRemoverStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StopWordsRemoverBatchOpTest {
    @Test
    public void testStopWordsRemoverBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
```

```
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text
string");
        BatchOperator <?> segment =
            new
SegmentBatchOp().setSelectedCol("text").setOutputCol("segment").linkFrom(inOp1)
;
        BatchOperator <?> remover = new
StopWordsRemoverBatchOp().setSelectedCol("segment").setOutputCol("remover")
            .linkFrom(segment);
        remover.print();
        StreamOperator <?> segment2 = new
SegmentStreamOp().setSelectedCol("text").setOutputCol("segment").linkFrom(
            inOp2);
        StreamOperator <?> remover2 = new
StopWordsRemoverStreamOp().setSelectedCol("segment").setOutputCol("remover")
            .linkFrom(segment2);
        remover2.print();
        StreamOperator.execute();
    }
}
```

运行结果

批运行结果

id	text	segment	remover
0	二手旧书:医学电磁成像	二手 旧书 : 医学 电磁 成像	二手 旧书 医学 电磁 成像
1	二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969	二手 美国 文学 选读 （ 下册 ） 李宜燮 南开大学 出版社 9787310003969	二手 美国 文学 选读 下册 李宜燮 南开大学 出版社 9787310003969
2	二手正版图解象棋入门/谢恩思主编/华龄出版社	二手 正版 图解 象棋 入门 / 谢恩思 主编 / 华龄 出版社	二手 正版 图解 象棋 入门 谢恩 思 主编 华龄 出版社
3	二手中国糖尿病文献索引	二手 中国 糖尿病 文献 索引	二手 中国 糖尿病 文献 索引

流运行结果

id	text	segment	remover

0	二手旧书:医学电磁成像	二手 旧书 : 医学 电磁 成像	二手 旧书 医学 电磁 成像
3	二手中国糖尿病文献索引	二手 中国 糖尿病 文献 索引	二手 中国 糖尿病 文献 索引
1	二手美国文学选读（ 下册 ） 李宜燮南开大学出版社 9787310003969	二手 美国 文学 选读 （ 下册 ） 李宜燮 南开大学 出版社 9787310003969	二手 美国 文学 选读 下册 李宜燮 南开大学 出版社 9787310003969
2	二手正版图解象棋入门/谢恩思主编/华龄出版社	二手 正版 图解 象棋 入门 / 谢恩思 主编 / 华龄 出版社	二手 正版 图解 象棋 入门 谢恩 思 主编 华龄 出版社

TF-IDF (TfidfBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.TfidfBatchOp

Python 类名：TfidfBatchOp

功能介绍

计算文本中词语的 TF-IDF 值。

算法原理

TF-IDF（term frequency–inverse document frequency）是一种用于资讯检索与文本挖掘的统计方法，用以评估单个词语对于一个文件集的重要程度。

词频（term frequency）$tf(t, d)$ 表示词语 t 在文档 d 中出现的频率：$\frac{f_{t,d}}{\Sigma{t'\in d}f_{t', d}}$, *其中 $f*{t,d}$表示词语 t 在文本 d 中出现的次数。

逆文档评率（inverse document frequency）$idf(t, D)$ 衡量一个词在语料库 D（所有文本）中提供的信息量：$\log\frac{|D|}{1 + |{d\in D:t\in D}|}$， 其中分子是所有文本的数量，分母是有词语 t 出现的文本的数量。

最终，文本 d 中的词语 t 在该语料库 D 的 TF-IDF 值就是 $tf(t, d) * idf(t, D)$。

使用方式

TF-IDF 加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。

在 Alink 中使用时，输入数据不需要为原始的文本，而是文本进行词频（DocWordCountBatchOp）统计的结果，记录了在每个文本中各词出现的次数。 组件需要设置文档 ID 列（docIdCol），单词列（wordCol）和词频列（countCol）。

文献索引

TF-IDF: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
countCol	词频列	词频列名	String	✓	所选列类型为 [LONG]	
docIdCol	文档ID列	文档ID列名	String	✓		
wordCol	单词列	单词列名	String	✓	所选列类型为 [STRING]	

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集 （ 国内版 ） 全十二册馆藏书']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int, text string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id int, text string')

segment =
SegmentBatchOp().setSelectedCol("text").setOutputCol("segment").linkFrom(inOp1)
remover =
StopWordsRemoverBatchOp().setSelectedCol("segment").setOutputCol("remover").lin
kFrom(segment)
wordCount =
DocWordCountBatchOp().setContentCol("remover").setDocIdCol("id").linkFrom(remov
er)
tfidf =
TfidfBatchOp().setDocIdCol("id").setWordCol("word").setCountCol("cnt").linkFrom
(wordCount)
tfidf.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.DocWordCountBatchOp;
import com.alibaba.alink.operator.batch.nlp.SegmentBatchOp;
import com.alibaba.alink.operator.batch.nlp.StopWordsRemoverBatchOp;
import com.alibaba.alink.operator.batch.nlp.TfidfBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TfidfBatchOpTest {
    @Test
```

```
    public void testTfidfBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集 （ 国内版 ） 全十二册馆藏书")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text
string");
        BatchOperator <?> segment =
            new
SegmentBatchOp().setSelectedCol("text").setOutputCol("segment").linkFrom(inOp1)
;
        BatchOperator <?> remover = new
StopWordsRemoverBatchOp().setSelectedCol("segment").setOutputCol("remover")
            .linkFrom(segment);
        BatchOperator <?> wordCount = new
DocWordCountBatchOp().setContentCol("remover").setDocIdCol("id").linkFrom(
            remover);
        BatchOperator <?> tfidf = new
TfidfBatchOp().setDocIdCol("id").setWordCol("word").setCountCol("cnt").linkFrom
(
            wordCount);
        tfidf.print();
    }
}
```

运行结果

id	word	cnt	total_word_count	doc_cnt	total_doc_count	tf	
0	医学	1	5	1	5	0.2000	0
0	电磁	1	5	1	5	0.2000	0
2	入门	1	10	1	5	0.1000	0
4	文集	1	9	1	5	0.1111	0
4	版	1	9	1	5	0.1111	0
4	十二册	1	9	1	5	0.1111	0
4	书	1	9	1	5	0.1111	0
4	馆藏	1	9	1	5	0.1111	0

4	郁达夫	1	9	1	5	0.1111	0
0	成像	1	5	1	5	0.2000	0
0	旧书	1	5	1	5	0.2000	0
0	二手	1	5	5	5	0.2000	-
1	二手	1	9	5	5	0.1111	-
1	9787310003969	1	9	1	5	0.1111	0
2	二手	1	10	5	5	0.1000	-
2	华龄	1	10	1	5	0.1000	0
2	思	1	10	1	5	0.1000	0
4	二手	1	9	5	5	0.1111	-
3	索引	1	5	1	5	0.2000	0
3	二手	1	5	5	5	0.2000	-
1	出版社	1	9	2	5	0.1111	0
2	出版社	1	10	2	5	0.1000	0
2	谢恩	1	10	1	5	0.1000	0
2	象棋	1	10	1	5	0.1000	0
3	糖尿病	1	5	1	5	0.2000	0
1	选读	1	9	1	5	0.1111	0
2	正版	1	10	1	5	0.1000	0
1	文学	1	9	1	5	0.1111	0
1	南开大学	1	9	1	5	0.1111	0
3	中国	1	5	1	5	0.2000	0
1	下册	1	9	1	5	0.1111	0
2	图解	1	10	1	5	0.1000	0
4	全	1	9	1	5	0.1111	0
1	美国	1	9	1	5	0.1111	0
1	李宜燮	1	9	1	5	0.1111	0
2	主编	1	10	1	5	0.1000	0

4	国内	1	9	1	5	0.1111	0
3	文献	1	5	1	5	0.2000	0

文本分解 (TokenizerBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.TokenizerBatchOp

Python 类名：TokenizerBatchOp

功能介绍

对文本按空白符进行切分操作。

使用方式

文本列通过参数 selectedCol 指定，输出列通过 outputCol 指定。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, 'That is an English Book!'],
    [1, 'Do you like math?'],
    [2, 'Have a good day!']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id long, text string')

op = TokenizerBatchOp().setSelectedCol("text")
```

```
op.linkFrom(inOp1).print()

inOp2 = StreamOperator.fromDataframe(df, schemaStr='id long, text string')
op2 = TokenizerStreamOp().setSelectedCol("text")
op2.linkFrom(inOp2).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.TokenizerBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.nlp.TokenizerStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TokenizerBatchOpTest {
    @Test
    public void testTokenizerBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "That is an English Book!"),
            Row.of(1, "Do you like math?"),
            Row.of(2, "Have a good day!")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        BatchOperator <?> op = new TokenizerBatchOp().setSelectedCol("text");
        op.linkFrom(inOp1).print();
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text
string");
        StreamOperator <?> op2 = new
TokenizerStreamOp().setSelectedCol("text");
        op2.linkFrom(inOp2).print();
        StreamOperator.execute();
    }
}
```

运行结果

批运行结果

id	text
0	that is an english book!
1	do you like math?
2	have a good day!

流运行结果

id	text
1	do you like math?
0	that is an english book!
2	have a good day!

Word2Vec (Word2VecBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.Word2VecBatchOp

Python 类名：Word2VecBatchOp

功能介绍

Word2Vec是Google在2013年开源的一个将词表转为向量的算法，其利用神经网络，可以通过训练，将词映射到K维度空间向量，甚至对于表示词的向量进行操作还能和语义相对应，由于其简单和高效引起了很多人的关注。

Word2Vec的工具包相关链接：https://code.google.com/p/word2vec/

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	计算列对应的列名	计算列对应的列名	String	✓	所选列类型为 [STRING]	
alpha	学习率	学习率	Double			0.025
batchSize	batch大小	batch大小, 按行计算	Integer		[1, +inf)	
minCount	最小词频	最小词频	Integer			5
negative	负采样大小	负采样大小	Integer			5
numIter	迭代次数	迭代次数，默认为1。	Integer			1
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"true"
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	100
window	窗口大小	窗口大小	Integer			5
wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["A B C"]
])

source = BatchOperator.fromDataframe(df, schemaStr='tokens string')

word2vecBatchOp = Word2VecBatchOp() \
  .setSelectedCol("tokens")         \
  .setMinCount(1)                   \
  .setVectorSize(4)
word2vecBatchOp.linkFrom(source).print()
```

运行结果

word	vec
A	0.024366257712244987,0.07037621736526489,-0.04168345779180527,-0.06180821731686592
B	0.05771925672888756,0.08288027346134186,-0.06486544758081436,0.026565641164779663
C	0.034414440393447876,-0.047638311982154846,0.012538374401628971,-0.09579437971115112

备注

如果不输入vecTable的情况下是随机初始化，可能会造成两次结果相同的item的embedding结果绝对值差别比较大，请注意

Word2Vec预测 (Word2VecPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.Word2VecPredictBatchOp

Python 类名：Word2VecPredictBatchOp

功能介绍

Word2Vec是Google在2013年开源的一个将词表转为向量的算法，其利用神经网络，可以通过训练，将词映射到K维度空间向量，甚至对于表示词的向量进行操作还能和语义相对应，由于其简单和高效引起了很多人的关注。

Word2Vec的工具包相关链接：https://code.google.com/p/word2vec/

预测是根据word2vec的结果和文档的分词结果，将文档转成向量，向量维数保持与词的维数一致，同时每个维度通过对文档中的词求平均或者最大或者最小取得。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
predMethod	向量组合方法	预测文档向量时，需要用到的方法。支持三种方法：平均（avg），最小（min）和最大（max），默认值为平均	String		"AVG", "SUM", "MIN", "MAX"	"AVG"
reservedCols	算法保留列名	算法保留列	String[]			null
wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["A B C"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='tokens string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='tokens string')
train =
Word2VecTrainBatchOp().setSelectedCol("tokens").setMinCount(1).setVectorSize(4)
.linkFrom(inOp1)
predictBatch =
Word2VecPredictBatchOp().setSelectedCol("tokens").linkFrom(train, inOp1)

train.lazyPrint(-1)
predictBatch.print()

predictStream =
Word2VecPredictStreamOp(train).setSelectedCol("tokens").linkFrom(inOp2)
predictStream.print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.Word2VecPredictBatchOp;
import com.alibaba.alink.operator.batch.nlp.Word2VecTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.nlp.Word2VecPredictStreamOp;
```

```
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class Word2VecPredictBatchOpTest {
    @Test
    public void testWord2VecPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("A B C")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "tokens string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "tokens string");
        BatchOperator <?> train = new
Word2VecTrainBatchOp().setSelectedCol("tokens").setMinCount(1).setVectorSize(4)
            .linkFrom(inOp1);
        BatchOperator <?> predictBatch = new
Word2VecPredictBatchOp().setSelectedCol("tokens").linkFrom(train, inOp1);
        train.lazyPrint(-1);
        predictBatch.print();
        StreamOperator <?> predictStream = new
Word2VecPredictStreamOp(train).setSelectedCol("tokens").linkFrom(inOp2);
        predictStream.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型结果

word	vec
A	0.7309136238338743 0.8314290437797685 0.24048455175042288 0.6063329203030643
C	0.7309085567091897 0.10053583269390566 0.41008295020646984 0.4074737375159046
B	0.7310876997238699 0.29335938660122723 0.901396784395289 0.004137313321518908

批预测结果

tokens	
0.7309699600889779 0.40844142102496706 0.5173214287840605 0.3393146570468293	

流预测结果

tokens	
0.7309691109963297 0.4083920636901659 0.5173538721894075 0.3392825036669853	

Word2Vec训练 (Word2VecTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.Word2VecTrainBatchOp

Python 类名：Word2VecTrainBatchOp

功能介绍

Word2Vec是Google在2013年开源的一个将词表转为向量的算法，其利用神经网络，可以通过训练，将词映射到K维度空间向量，甚至对于表示词的向量进行操作还能和语义相对应，由于其简单和高效引起了很多人的关注。

Word2Vec的工具包相关链接：https://code.google.com/p/word2vec/

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为 [STRING]	
alpha	学习率	学习率	Double			0.025
minCount	最小词频	最小词频	Integer			5
numIter	迭代次数	迭代次数，默认为1。	Integer			1
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"true"
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	100
window	窗口大小	窗口大小	Integer			5
wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ["A B C"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='tokens string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='tokens string')
train =
Word2VecTrainBatchOp().setSelectedCol("tokens").setMinCount(1).setVectorSize(4)
.linkFrom(inOp1)
predictBatch =
Word2VecPredictBatchOp().setSelectedCol("tokens").linkFrom(train, inOp1)

train.lazyPrint(-1)
predictBatch.print()

predictStream =
Word2VecPredictStreamOp(train).setSelectedCol("tokens").linkFrom(inOp2)
predictStream.print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.Word2VecPredictBatchOp;
import com.alibaba.alink.operator.batch.nlp.Word2VecTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.nlp.Word2VecPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class Word2VecTrainBatchOpTest {
    @Test
    public void testWord2VecTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("A B C")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "tokens string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "tokens string");
```

```
        BatchOperator <?> train = new
Word2VecTrainBatchOp().setSelectedCol("tokens").setMinCount(1).setVectorSize(4)
            .linkFrom(inOp1);
        BatchOperator <?> predictBatch = new
Word2VecPredictBatchOp().setSelectedCol("tokens").linkFrom(train, inOp1);
        train.lazyPrint(-1);
        predictBatch.print();
        StreamOperator <?> predictStream = new
Word2VecPredictStreamOp(train).setSelectedCol("tokens").linkFrom(inOp2);
        predictStream.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型结果

word	vec
A	0.7308596353097189 0.8314177144978963 0.24043567184236792 0.6063183430688116
C	0.7309068666584959 0.10053389527357781 0.41008241284786995 0.40747231850240395
B	0.7311470683768729 0.29342648043578945 0.9014165072579701 0.0041863689268244915

批预测结果

tokens	
0.7309711901150291 0.40845936340242117 0.5173115306494026 0.33932567683268	

流预测结果

tokens	
0.7309691109963297 0.4083920636901659 0.5173538721894075 0.3392825036669853	

单词计数 (WordCountBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.WordCountBatchOp

Python 类名：WordCountBatchOp

功能介绍

输出文本列所有词语和对应频数。

使用方式

文本内容列（SelectedCol）中的内容用于统计词频，需要是用分隔符分隔的词语。 其中，分隔符可以通过参数 wordDelimiter 来设置，默认是空格（" "），可以为正则表达式。 文本内容列可以使用分词（SegmentBatchOp）组件的输出结果列，同时也可以在之前接入停用词过滤（StopWordsRemoverBatchOp）组件去掉常见的高频词。

需要注意的是，该组件统计的是文本内容列总体的词频，而文本词频统计组件（DocWordCountBatchOp）是按行统计词频。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["doc0", "中国 的 文化"],
    ["doc1", "只要 功夫 深"],
    ["doc2", "北京 的 拆迁"],
    ["doc3", "人名 的 名义"]
```

```
])

source = BatchOperator.fromDataframe(df, "id string, content string")
wordCountBatchOp = WordCountBatchOp()\
    .setSelectedCol("content")\
    .setWordDelimiter(" ")\
    .linkFrom(source)
wordCountBatchOp.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.nlp.WordCountBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class WordCountBatchOpTest {
    @Test
    public void testWordCountBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("doc0", "中国 的 文化"),
            Row.of("doc1", "只要 功夫 深"),
            Row.of("doc2", "北京 的 拆迁"),
            Row.of("doc3", "人名 的 名义")
        );
        BatchOperator <?> source = new MemSourceBatchOp(df, "id string, content
string");
        BatchOperator <?> wordCountBatchOp = new WordCountBatchOp()
            .setSelectedCol("content")
            .setWordDelimiter(" ")
            .linkFrom(source);
        wordCountBatchOp.print();
    }
}
```

运行结果

word	cnt
深	1
拆迁	1

北京	1
文化	1
中国	1
的	3
人名	1
只要	1
名义	1
功夫	1

单词识别 (WordRecognitionBatchOp)

Java 类名：com.alibaba.alink.operator.batch.nlp.WordRecognitionBatchOp

Python 类名：WordRecognitionBatchOp

功能介绍

从语料库中学习出候选词组，输入的词语无需分词。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
prefixThre	前缀阈值	前缀阈值	Double			0.0
solidationThre	凝固程度阈值	凝固程度阈值	Double			0.0
suffixThre	后缀阈值	后缀阈值	Double			0.0
windowSize	窗口大小	窗口大小	Integer		[1, +inf)	2

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集（ 国内版 ）全十二册馆藏书']
])
```

```
inOp = BatchOperator.fromDataframe(df, schemaStr='id int, text string')
op =
WordRecognitionBatchOp().setSelectedCol("text").setWindowSize(4).linkFrom(inOp)
op.print()
```

运行结果

```
  ngram lw  ngram_cnt  left_entropy  right_entropy  left_cnt rw  max_cnt  \
0    二手  二          5      1.609438       0.000000         6  手        5
1    出版  出          2      0.000000       0.693147         2  版        2

   solidation
0    0.166667
1    0.500000
```

全表统计 (AllStatBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.AllStatBatchOp

Python 类名：AllStatBatchOp

功能介绍

全表统计是用来计算到整个表的统计量，用于batch数据源。

对于每个选择的变量，根据其为数值类型（DOUBLE、INT或LONG）还是字符串类型（STRING），所计算的统计量略有不同。

对于数值型变量，所计算的统计量包括基本统计量（basic_stats）、前K大（topK）、前K小（bottomK）、频次统计（freq）和直方图（histogram），其中基本统计量包括count, sum, min, max, mean, variance, standardVariance, standardError, skewness, countMissValue。

对于字符串类型的变量，其只有基本统计量（basic_stats）和频次统计（freq），并且基本统计量中只有count和countMissValue两项。

对于统计结果，除了在输出桩中返回之外，也可以通过可视化大屏直接查看（见下文例子）。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选中的列名数组	计算列对应的列名列表	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
statLevel	统计级别	将计算的统计级别。L1 has basic statistics; L2 has basic statistics and cov/corr; L3 has basic statistics, cov/corr, histogram, freq, and top/bottom k.	String		"L1", "L2", "L3"	"L1"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0.0,0.0,0.0],
    [0.1,0.2,0.1],
    [0.2,0.2,0.8],
    [9.0,9.5,9.7],
    [9.1,9.1,9.6],
    [9.2,9.3,9.9]
])
data = BatchOperator.fromDataframe(df, schemaStr="x1 double, x2 double, x3
double")
allStatOp= AllStatBatchOp()\
    .setSelectedCols(["x1","x2"])
data.link(allStatOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.statistics.AllStatBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AllStatBatchOpTest {

    @Test
    public void testAllStatBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0.0, 0.0, 0.0),
            Row.of(0.1, 0.2, 0.1),
            Row.of(0.2, 0.2, 0.8),
            Row.of(9.0, 9.5, 9.7),
            Row.of(9.1, 9.1, 9.6),
            Row.of(9.2, 9.3, 9.9));

        BatchOperator <?> data = new MemSourceBatchOp(df, "x1 double, x2
```

```
double, x3 double");

        AllStatBatchOp allStatOp = new AllStatBatchOp()
            .setSelectedCols(new String[] {"x1", "x2"});

        data.link(allStatOp).print();
    }
}
```

结果运行结果

stat	x1	x2
count	6	6
sum	27.599999999999998	28.3
sum2	248.49999999999997	259.63
sum3	2261.2679999999996	2415.3190000000004
sum4	20582.427399999997	22483.0819
mean	4.6	4.716666666666667
max	9.2	9.5
min	0.0	0.0
range	9.2	9.5
countMissValue	0	0
standardDeviation	4.930314391598166	5.022914160790195
variance	24.308	25.229666666666663
standardError	2.0127924218193325	2.0505961192893265
moment2	41.416666666666664	43.27166666666667
moment3	376.87799999999993	402.55316666666675
moment4	3430.404566666666	3747.180316666667
centralMoment2	20.256666666666664	21.02472222222222
centralMoment3	0.0	0.12192592592600704
centralMoment4	410.87256666666735	443.53848032407313
skewness	0.0	0.0012647382766099678
kurtosis	-1.9986839400638958	-1.9966076839585

cv	1.0718074764343841	1.0649287973406774

AD检验 (AndersonDarlingTestBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.AndersonDarlingTestBatchOp

Python 类名：AndersonDarlingTestBatchOp

功能介绍

检验是否服从正态分布

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ["1", "a", 1.3, 1.1],
        ["1", "b", -2.5, 0.9],
        ["2", "c", 100.2, -0.01],
        ["2", "d", -99.9, 100.9],
        ["1", "a", 1.4, 1.1],
        ["1", "b", -2.2, 0.9],
        ["2", "c", 100.9, -0.01],
        ["2", "d", -99.5, 100.9]
])
```

```
batchData = BatchOperator.fromDataframe(df, schemaStr='id string, col1 string,
col2 double, col3 double')

ad = AndersonDarlingTestBatchOp()\
            .setSelectedCol("col3")

batchData.link(ad).print()
```

运行结果

```
               test_name  ad_value    pvalue
0  Anderson_Darling_Test  2.061553  0.000031
```

卡方检验 (ChiSquareTestBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.ChiSquareTestBatchOp

Python 类名：ChiSquareTestBatchOp

功能介绍

卡法独立性检验是检验两个因素（各有两项或以上的分类）之间是否相互影响的问题，其零假设是两因素之间相互独立。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ['a1','b1','c1'],
            ['a1','b2','c1'],
            ['a1','b1','c2'],
            ['a2','b1','c1'],
            ['a2','b2','c2'],
            ['a2', 'b1','c1']
])

batchData = BatchOperator.fromDataframe(df, schemaStr='x1 string, x2 string, x3
string')

chisqTest = ChiSquareTestBatchOp()\
```

```
            .setSelectedCols(["x1","x2"])\
            .setLabelCol("x3")

batchData.link(chisqTest).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.statistics.ChiSquareTestBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ChiSquareTestBatchOpTest {
    @Test
    public void testChiSquareTestBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a1", "b1", "c1"),
            Row.of("a1", "b2", "c1"),
            Row.of("a1", "b1", "c2"),
            Row.of("a2", "b1", "c1"),
            Row.of("a2", "b2", "c2"),
            Row.of("a2", "b1", "c1")
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "x1 string, x2
string, x3 string");
        BatchOperator <?> chisqTest = new ChiSquareTestBatchOp()
            .setSelectedCols("x1", "x2")
            .setLabelCol("x3");
        batchData.link(chisqTest).print();
    }
}
```

运行结果

col	chi2_result
x1	{"comment":"pearson test","df":1.0,"p":1.0,"value":0.0}
x2	{"comment":"pearson test","df":1.0,"p":0.5402913746074196,"value":0.37500000000000006}

相关系数 (CorrelationBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.CorrelationBatchOp

Python 类名：CorrelationBatchOp

功能介绍

- 相关系数算法用于计算一个矩阵中每一列之间的相关系数，范围在[-1,1]之间。计算的时候，count数按两列间同时非空的元素个数计算，两两列之间可能不同。
- 支持Pearson和Spearman两种相关系数
- 只支持数字类型列。如果想计算vector列，请参考VectorCorrelationBatchOp

使用方式

- 通过collectCorrelation()获取CorrelationMetric.

```
corr = corrOp.collectCorrelation()
print(corr)
print(corr.getCorrelation())
print(corr.getCorrelation()[1][1])
```

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默
method	方法	方法：包含"PEARSON"和"SPEARMAN"两种，PEARSON。	String		"PEARSON", "SPEARMAN"	"PEA
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
          [0.0,0.0,0.0],
          [0.1,0.2,0.1],
          [0.2,0.2,0.8],
          [9.0,9.5,9.7],
          [9.1,9.1,9.6],
          [9.2,9.3,9.9]])

source = BatchOperator.fromDataframe(df, schemaStr='x1 double, x2 double, x3
double')


corr = CorrelationBatchOp()\
            .setSelectedCols(["x1","x2","x3"])

corr = source.link(corr).collectCorrelation()
print(corr)
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.statistics.CorrelationBatchOp;
import
com.alibaba.alink.operator.common.statistics.basicstatistic.CorrelationResult;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CorrelationBatchOpTest {
    @Test
    public void testCorrelationBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0.0, 0.0, 0.0),
            Row.of(0.1, 0.2, 0.1),
            Row.of(0.2, 0.2, 0.8),
            Row.of(9.0, 9.5, 9.7),
            Row.of(9.1, 9.1, 9.6)
```

```
        );
        BatchOperator <?> source = new MemSourceBatchOp(df, "x1 double, x2
double, x3 double");
        CorrelationBatchOp corr = new CorrelationBatchOp()
            .setSelectedCols("x1", "x2", "x3");
        CorrelationResult correlationResult =
source.link(corr).collectCorrelation();
        System.out.println(correlationResult);
    }
}
```

运行结果

colName	x1	x2	x3
x1	1.0000	0.9994	0.9990
x2	0.9994	1.0000	0.9986
x3	0.9990	0.9986	1.0000

协方差 (CovBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.CovBatchOp

Python 类名：CovBatchOp

功能介绍

用于计算定长窗口内数据的协方差。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ["1", "a", 1.3, 1.1],
        ["1", "b", -2.5, 0.9],
        ["2", "c", 100.2, -0.01],
        ["2", "d", -99.9, 100.9],
        ["1", "a", 1.4, 1.1],
        ["1", "b", -2.2, 0.9],
        ["2", "c", 100.9, -0.01],
        ["2", "d", -99.5, 100.9]
        ])
```

```
batchData = BatchOperator.fromDataframe(df, schemaStr='id string, col1 string,
col2 double, col3 double')

ad = CovBatchOp()\
            .setSelectedCols(["col3", "col2"])

batchData.link(ad).print()
```

运行结果

```
          col3         col2
0  2153.212021 -2873.573464
1 -2873.573464  5730.834107
```

KS检验 (KolmogorovSmirnovTestBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.KolmogorovSmirnovTestBatchOp

Python 类名：KolmogorovSmirnovTestBatchOp

功能介绍

对连续一维数据的概率分布检验

- 具体细节请参考：https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ["1", "a", 1.3, 1.1],
        ["1", "b", -2.5, 0.9],
        ["2", "c", 100.2, -0.01],
        ["2", "d", -99.9, 100.9],
        ["1", "a", 1.4, 1.1],
        ["1", "b", -2.2, 0.9],
        ["2", "c", 100.9, -0.01],
        ["2", "d", -99.5, 100.9]
```

```
        ])

batchData = BatchOperator.fromDataframe(df, schemaStr='id string, col1 string,
col2 double, col3 double')

ks = KolmogorovSmirnovTestBatchOp()\
            .setSelectedCol("col3")

batchData.link(ks).print()
```

运行结果

```
                test_name  ks_value    pvalue
0  Kolmogorov_Smirnov_Test  0.447907  0.080714
```

洛伦兹曲线 (LorenzCurveBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.LorenzCurveBatchOp

Python 类名：LorenzCurveBatchOp

功能介绍

洛伦兹曲线研究的是国民收入在国民之间的分配问题。为了研究国民收入在国民之间的分配问题，美国统计学家（或说奥地利统计学家）M.O.洛伦兹（Max Otto Lorenz，1903- ）1907年（或说1905年）提出了著名的洛伦兹曲线。 意大利经济学家基尼在此基础上定义了基尼系数。
画一个矩形，矩形的高衡量社会财富的百分比， 将之分为N等份，每一等分为1/N的社会总财富。在矩形的长上，将所有家庭从最贫者到最富者自左向右排列， 也分为N等分，第一个等份代表收入最低的1/N的家庭。在这个矩形中，将每1/N的家庭所有拥有的财富的占比累 积起来，并将相应的点画在图中，便得到了一条曲线就是洛伦兹曲线。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
quantileNum	多分类数目	多分类数目	Integer			100

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, 1, 1.0],
    [1.0, 2, 1.0],
    [2.0, 3, 1.0],
    [3.0, 4, 1.0],
    [4.0, 2, 1.0],
    [3.0, 1, 1.0],
    [2.0, 4, 1.0],
    [4.0, 1, 1.0]
])
```

```
source = BatchOperator.fromDataframe(df, schemaStr='f0 double, f1 int, f2
double')

perc = LorenzCurveBatchOp()\
    .setSelectedCol("f0")\
    .setQuantileNum(100)

source.link(perc).print()
```

运行结果

quntile	lorenz_value
0	0.1
1	0.1
2	0.1
3	0.1
4	0.1
...	...
95	1.0
96	1.0
97	1.0
98	1.0
99	1.0

PP图 (PPPlotBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.PPPlotBatchOp

Python 类名：PPPlotBatchOp

功能介绍

P-P图是用于检验变量是否近似于服从特定分布的图形。其绘制了变量实际累计概率与在相应分布中的累计概率之间的关系。当数据服从指定分布时，P-P图中各点连线近似于一条直线。 目前仅支持与正态分布的累积概率进行比较，以选定变量数据的均值与方差作为正态分布参数。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ["1", "a", 1.3, 1.1],
        ["1", "b", -2.5, 0.9],
        ["2", "c", 100.2, -0.01],
        ["2", "d", -99.9, 100.9],
        ["1", "a", 1.4, 1.1],
        ["1", "b", -2.2, 0.9],
        ["2", "c", 100.9, -0.01],
        ["2", "d", -99.5, 100.9]
```

```
        ])

batchData = BatchOperator.fromDataframe(df, schemaStr='id string, col1 string,
col2 double, col3 double')

ppPlot = PPPlotBatchOp()\
            .setSelectedCols(["col2", "col3"])

batchData.link(ppPlot).print()
```

运行结果

```
   colname       x theoretical  empirical
0     col2  -99.90   0.0935598      0.125
1     col2  -99.50    0.094446      0.250
2     col2   -2.50    0.487025      0.375
3     col2   -2.20    0.488605      0.500
4     col2    1.30    0.507048      0.625
5     col2    1.40    0.507575      0.750
6     col2  100.20    0.907265      0.875
7     col2  100.90    0.908791      1.000
8     col3   -0.01    0.289602      0.250
9     col3    0.90    0.296347      0.500
10    col3    1.10    0.297839      0.750
11    col3  100.90    0.947396      1.000
```

分位数 (QuantileBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.QuantileBatchOp

Python 类名：QuantileBatchOp

功能介绍

quantile

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
quantileNum	分位个数	分位个数	Integer	✓	[0, +inf)	
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
roundMode	取整的模式	取整的模式。当q是组的个数，k 是第k个组，total是总的样本大小时，第k组的边界索引应为(1.0 / q) *(total - 1)* k。这个值应该为整数，所以需要取整，取整时用到这个参数。	String		"CEIL", "FLOOR", "ROUND"	"ROUND"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        [1.0, 0],
        [2.0, 1],
        [3.0, 2],
        [4.0, 3]
    ])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 double, f1 double')
StreamData = StreamOperator.fromDataframe(df, schemaStr='f0 double, f1 double')

QuantileBatchOp()\
    .setSelectedCol("f0")\
    .setQuantileNum(100)\
    .linkFrom(batchData)\
    .print()

QuantileStreamOp()\
    .setSelectedCols(["f0"])\
    .setQuantileNum(100)\
    .linkFrom(StreamData)\
    .print()

StreamOperator.execute()
```

运行结果

```
      f0  quantile
0    1.0         0
1    1.0         1
2    1.0         2
3    1.0         3
4    1.0         4
..   ...       ...
96   4.0        96
97   4.0        97
98   4.0        98
99   4.0        99
100  4.0       100

[101 rows x 2 columns]
```

排行榜 (RankingListBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.RankingListBatchOp

Python 类名：RankingListBatchOp

功能介绍

排行榜是用来计算分组榜单的，例如数据是

age	job	balance
58	management	2143
44	technician	29
33	entrepreneur	2
47	blue-collar	1506
33	unknown	1
35	management	231
28	management	447
42	entrepreneur	2
58	retired	121
43	technician	593
41	admin.	270
29	admin.	390
53	technician	6
58	technician	71
57	services	162
51	retired	229
45	admin.	13

选择marital（婚姻状况）作为分组列， age（年龄）作为主体列， balance(净资产)作为计算列，计算指标是sum，
 那么结果

marital	age	sum(balance)	rank
divorced	54	318166.0	1
divorced	56	283257.0	2

divorced	59	281327.0	3
divorced	58	263003.0	4
married	37	1389347.0	1
married	45	1372091.0	2
married	39	1301587.0	3
married	36	1274481.0	4
single	32	1365481.0	1
single	31	1348609.0	2
single	30	1287184.0	3
single	33	1044254.0	4

可以看出，离婚人群中50岁往上的资产比较多，结婚人群中35到45岁资产比较多，单身的30到35资产比较多。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
objectCol	主体列	主体列	String	✓		
addedCols	附加列	附加列	String[]			null
addedStatTypes	附加列统计类型	附加列统计类型	String[]			null
groupCol	分组单列名	分组单列名，可选	String			null

groupValues	计算分组	计算分组, 分组列选择时必选, 用逗号分隔	String[]			null
isDescending	是否降序	是否降序	Boolean			false
statCol	计算列	计算列	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
statType	统计类型	统计类型	String		"count", "countTotal", "min", "max", "sum", "mean", "variance"	"count"
topN	个数	个数	Integer			10

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ["1", "a", 1.3, 1.1],
        ["1", "b", -2.5, 0.9],
        ["2", "c", 100.2, -0.01],
        ["2", "d", -99.9, 100.9],
        ["1", "a", 1.4, 1.1],
        ["1", "b", -2.2, 0.9],
        ["2", "c", 100.9, -0.01],
        ["2", "d", -99.5, 100.9]
])

batchData = BatchOperator.fromDataframe(df, schemaStr='id string, col1 string,
col2 double, col3 double')
```

```
rankList = RankingListBatchOp()\
            .setGroupCol("id")\
            .setGroupValues(["1", "2"])\
            .setObjectCol("col1")\
            .setStatCol("col2")\
            .setStatType("sum")\
            .setTopN(20)


batchData.link(rankList).print()
```

运行结果

id	col1	col2	rank
1	b	-4.7	1
1	a	2.7	2
2	d	-199.4	1
2	c	201.1	2

散点图 (ScatterPlotBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.ScatterPlotBatchOp

Python 类名：ScatterPlotBatchOp

功能介绍

该组件读入一批点的x坐标和y坐标绘制散点图。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
groupCol	分组列名	分组列名	String			null
tagCol	标签列名	标签列名	String			null
tensorCol	向量列名	向量列名	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
xCol	xColName	xColName	String			null
yCol	yColName	yColName	String			null

散点图矩阵 (ScatterPlotMatrixBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.ScatterPlotMatrixBatchOp

Python 类名：ScatterPlotMatrixBatchOp

功能介绍

该组件提供对数据分布的一种可视化形式。对于具有多列的表类型数据，该组件对数据在任意两列上的分布绘制散点图，形成矩阵形式，从而能同时观察数据在多个列上的分布情况。

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [5,2,3.5,1,'Iris-versicolor'],
    [5.1,3.7,1.5,0.4,'Iris-setosa'],
    [6.4,2.8,5.6,2.2,'Iris-virginica'],
    [6,2.9,4.5,1.5,'Iris-versicolor'],
    [4.9,3,1.4,0.2,'Iris-setosa'],
    [5.7,2.6,3.5,1,'Iris-versicolor'],
    [4.6,3.6,1,0.2,'Iris-setosa'],
    [5.9,3,4.2,1.5,'Iris-versicolor'],
    [6.3,2.8,5.1,1.5,'Iris-virginica'],
    [4.7,3.2,1.3,0.2,'Iris-setosa'],
    [5.1,3.3,1.7,0.5,'Iris-setosa'],
    [5.5,2.4,3.8,1.1,'Iris-versicolor'],
])

source = BatchOperator.fromDataframe(df, schemaStr='sepal_length double,
sepal_width double, petal_length double, petal_width double, category string')

op = ScatterPlotMatrixBatchOp() \
        .setSelectedCols(["sepal_length", "sepal_width", "category",
"petal_length", "petal_width"]) \
        .setVizName("Scatter")

source.link(op).print()
```

运行结果

	sepal_length	sepal_width	category	petal_length	petal_width
0	5.9	3.2	2.0	4.8	1.8
1	7.4	2.8	3.0	6.1	1.9
2	5.8	2.7	3.0	5.1	1.9
3	7.7	2.6	3.0	6.9	2.3
4	6.5	3.2	3.0	5.1	2.0
5	6.9	3.2	3.0	5.7	2.3
6	6.7	3.1	2.0	4.4	1.4
7	5.8	4.0	1.0	1.2	0.2
8	6.7	3.0	3.0	5.2	2.3
9	7.9	3.8	3.0	6.4	2.0

全表统计 (SummarizerBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.SummarizerBatchOp

Python 类名：SummarizerBatchOp

功能介绍

全表统计用来计算整表的统计量, 包含count(个数),numValidValue(有效值个数), numMissingValue(缺失值个数), sum(求和), mean(均值), standardDeviation(标准差), variance(方差), min(最小值), max(最大值), normL1(L1范数), normL2(L2范数)。

结果可以使用collectSummary获取TableSummary, 通过TableSummary获取对应的结果, 也可以直接打印。

另外, 对所有的BatchOp, 可以直接获取Op输出表的统计量。具体使用方式如下,

使用方式

- 打印统计结果.

```
summary = summarizer.linkFrom(source).collectSummary()
print(summary)
```

- 获取相应的统计值

```
summary = summarizer.linkFrom(source).collectSummary()
print(summary.sum('f_double'))
print(summary.mean('f_double'))
print(summary.variance('f_double'))
print(summary.standardDeviation('f_double'))
print(summary.min('f_double'))
print(summary.max('f_double'))
print(summary.normL1('f_double'))
print(summary.normL2('f_double'))
print(summary.numValidValue('f_double'))
print(summary.numMissingValue('f_double'))
```

- 对Op的输出表做统计

```
source.lazyPrintStatistics()
BatchOperator.execute()
```

- 获取Op输出表的TableSummary

```
summary = source..collectStatistics()
```

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1, 1,2.0, True],
    ["c", 1, 2, -3.0, True],
    ["a", 2, 2,2.0, False],
    ["c", 0, 0, 0.0, False]
])
source = BatchOperator.fromDataframe(df, schemaStr='f_string string, f_long
long, f_int int, f_double double, f_boolean boolean')

summarizer = SummarizerBatchOp()\
    .setSelectedCols(["f_long", "f_int", "f_double"])

summary = summarizer.linkFrom(source).collectSummary()

print(summary)
```

Java 代码

```
package com.alibaba.alink.operator.batch.statistics;

import org.apache.flink.api.common.typeinfo.TypeInformation;
import org.apache.flink.table.api.TableSchema;
import org.apache.flink.table.api.Types;
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import
```

```
com.alibaba.alink.operator.common.statistics.basicstatistic.TableSummary;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Assert;
import org.junit.Test;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class SummarizerBatchOpTest extends AlinkTestBase {

  @Test
  public void test() {
    Row[] testArray =
            new Row[] {
                    Row.of("a", 1L, 1, 2.0, true),
                    Row.of(null, 2L, 2, -3.0, true),
                    Row.of("c", null, null, 2.0, false),
                    Row.of("a", 0L, 0, null, null),
            };

    String[] colNames = new String[] {"f_string", "f_long", "f_int",
"f_double", "f_boolean"};

    MemSourceBatchOp source = new MemSourceBatchOp(Arrays.asList(testArray),
colNames);

    SummarizerBatchOp summarizer = new SummarizerBatchOp()
            .setSelectedCols("f_double", "f_int");

    summarizer.linkFrom(source);

    TableSummary srt = summarizer.collectSummary();

    System.out.println(srt.toString());

    Assert.assertEquals(srt.getColNames().length, 2);
    Assert.assertEquals(srt.count(), 4);
    Assert.assertEquals(srt.numMissingValue("f_double"), 1, 10e-4);
    Assert.assertEquals(srt.numValidValue("f_double"), 3, 10e-4);
    Assert.assertEquals(srt.max("f_double"), 2.0, 10e-4);
    Assert.assertEquals(srt.min("f_int"), 0.0, 10e-4);
    Assert.assertEquals(srt.mean("f_double"), 0.3333333333333333, 10e-4);
    Assert.assertEquals(srt.variance("f_double"), 8.333333333333334, 10e-4);
    Assert.assertEquals(srt.standardDeviation("f_double"), 2.886751345948129,
10e-4);
    Assert.assertEquals(srt.normL1("f_double"), 7.0, 10e-4);
    Assert.assertEquals(srt.normL2("f_double"), 4.123105625617661, 10e-4);
```

```
  }
}
```

运行结果

Summary:

colName	count	missing	sum	mean	variance	min	max
f_long	4	0	4	1	0.6667	0	2
f_int	4	0	5	1.25	0.9167	0	2
f_double	4	0	1	0.25	5.5833	-3	2

向量卡方检验 (VectorChiSquareTestBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.VectorChiSquareTestBatchOp

Python 类名：VectorChiSquareTestBatchOp

功能介绍

针对vector数据，进行卡方检验

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为 [DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

无python接口

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.statistics.VectorChiSquareTestBatchOp;
import org.junit.Test;
```

```
import java.util.Arrays;

public class VectorChiSquareTestBatchOpTest {

    @Test
    public void testVectorChiSquareTestBatchOp() throws Exception {
        Row[] testArray = new Row[] {
            Row.of(7, "0.0  0.0  18.0  1.0", 1.0),
            Row.of(8, "0.0  1.0  12.0  0.0", 0.0),
            Row.of(9, "1.0  0.0  15.0  0.1", 0.0),
        };

        String[] colNames = new String[] {"id", "features", "clicked"};

        MemSourceBatchOp source = new
MemSourceBatchOp(Arrays.asList(testArray), colNames);

        VectorChiSquareTestBatchOp test = new VectorChiSquareTestBatchOp()
            .setSelectedCol("features")
            .setLabelCol("clicked");

        test.linkFrom(source);

        test.lazyPrintChiSquareTest();

        BatchOperator.execute();
    }

}
```

运行结果

ChiSquareTest:

col	p	value	df
0	0.3865	0.75	1
1	0.3865	0.75	1
2	0.2231	3	2
3	0.2231	3	2

向量相关系数 (VectorCorrelationBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.VectorCorrelationBatchOp

Python 类名：VectorCorrelationBatchOp

功能介绍

- 相关系数算法用于计算一个矩阵中每一列之间的相关系数，范围在[-1,1]之间。计算的时候，count数按两列间同时非空的元素个数计算，两两列之间可能不同。
- 支持Pearson和Spearman两种相关系数
- 只支持Vector类型列。如果想计算数字类型列，请参考CorrelationBatchOp

使用方式

- 通过collectCorrelation()获取CorrelationMetric.

```
corr = corrOp.collectCorrelation()
print(corr)
print(corr.getCorrelation())
print(corr.getCorrelation()[1][1])
```

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
method	方法	方法：包含"PEARSON"和"SPEARMAN"两种，PEARSON。	String		"PEARSON", "SPEARMAN"	"I

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

无python接口

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.statistics.VectorCorrelationBatchOp;
import org.junit.Test;

import java.util.Arrays;

public class VectorCorrelationBatchOpTest {

    @Test
    public void testVectorCorrelationBatchOp() throws Exception {
        Row[] testArray = new Row[] {
            Row.of(7, "0.0  0.0  18.0  1.0", 1.0),
            Row.of(8, "0.0  1.0  12.0  0.0", 0.0),
            Row.of(9, "1.0  0.0  15.0  0.1", 0.0),
        };

        String[] colNames = new String[] {"id", "features", "clicked"};

        MemSourceBatchOp source = new
MemSourceBatchOp(Arrays.asList(testArray), colNames);

        VectorCorrelationBatchOp test = new VectorCorrelationBatchOp()
            .setSelectedCol("features");

        test.linkFrom(source);

        test.lazyPrintCorrelation();

        BatchOperator.execute();
    }
}
```

运行结果

Correlation:

colName	0	1	2	3
0	1	-0.5	0	-0.4193
1	-0.5	1	-0.866	-0.5766

2	0	-0.866	1	0.9078
3	-0.4193	-0.5766	0.9078	1

向量全表统计 (VectorSummarizerBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.VectorSummarizerBatchOp

Python 类名：VectorSummarizerBatchOp

功能介绍

- 全表统计用来计算整表的统计量, 包含count(个数),numValidValue(有效值个数), numMissingValue(缺失值个数), sum(求和), mean(均值), standardDeviation(标准差), variance(方差), min(最小值), max(最大值), normL1(L1范数), normL2(L2范数)。
- 只支持vector列
- 只支持java接口

使用方式

- 打印统计结果.

```
test.lazyPrintVectorSummary();
```

- 获取相应的统计值

```
    summary = summarizer.linkFrom(source).collectVectorSummary()
    print(summary.sum('f_double'))
    print(summary.mean('f_double'))
    print(summary.variance('f_double'))
    print(summary.standardDeviation('f_double'))
    print(summary.min('f_double'))
    print(summary.max('f_double'))
    print(summary.normL1('f_double'))
    print(summary.normL2('f_double'))
    print(summary.numValidValue('f_double'))
    print(summary.numMissingValue('f_double'))

## 参数说明


| 名称 | 中文名称 | 描述 | 类型 | 是否必须? | 取值范围 | 默认值 |
| --- | --- | --- | --- | --- | --- | --- |
| selectedCol | 选中的列名 | 计算列对应的列名 | String | ✓ | 所选列类型为
[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR] |  |



## 代码示例
```

```
** 以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！**

### Python 代码
无python接口

### Java 代码
```java
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.statistics.VectorSummarizerBatchOp;
import org.junit.Test;

import java.util.Arrays;

public class VectorSummarizerBatchOpTest {

    @Test
    public void testVectorCorrelationBatchOp() throws Exception {
        Row[] testArray = new Row[] {
            Row.of(7, "0.0  0.0  18.0  1.0", 1.0),
            Row.of(8, "0.0  1.0  12.0  0.0", 0.0),
            Row.of(9, "1.0  0.0  15.0  0.1", 0.0),
        };

        String[] colNames = new String[] {"id", "features", "clicked"};

        MemSourceBatchOp source = new
MemSourceBatchOp(Arrays.asList(testArray), colNames);

        VectorSummarizerBatchOp test = new VectorSummarizerBatchOp()
            .setSelectedCol("features");

        test.linkFrom(source);

        test.lazyPrintVectorSummary();

        BatchOperator.execute();
    }
}
```

运行结果

DenseVectorSummary:

id	count	sum	mean	variance	stdDev	min	max	normL1	normL2
```

0	3	1	0.3333	0.3333	0.5774	0	1	1	1
1	3	1	0.3333	0.3333	0.5774	0	1	1	1
2	3	45	15	9	3	12	18	45	26.3249
3	3	1.1	0.3667	0.3033	0.5508	0	1	1.1	1.005

Som (SomBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.analysis.SomBatchOp

Python 类名：SomBatchOp

功能介绍

Self-Organized Map 算法，是一种高维数据可视化算法。

参考：https://clarkdatalabs.github.io/soms/SOM_NBA

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
vdim	向量长度	向量长度	Integer	✓		
vectorCol	vector列名	vector列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
xdim	x方向网格数	x方向网格数	Integer	✓		
ydim	y方向网格数	y方向网格数	Integer	✓		
debug	是否打开调试	是否打开调试	Boolean			false
evaluation	是否每轮评估迭代结果	是否每轮评估迭代结果	Boolean			false
learnRate	学习率	学习率	Double			0.5
numIters	迭代轮数	迭代轮数	Integer			100
sigma	neighborhood函数方差	neighborhood函数方差	Double			1.0

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [5,2,3.5,1,'Iris-versicolor'],
    [5.1,3.7,1.5,0.4,'Iris-setosa'],
    [6.4,2.8,5.6,2.2,'Iris-virginica'],
    [6,2.9,4.5,1.5,'Iris-versicolor'],
    [4.9,3,1.4,0.2,'Iris-setosa'],
    [5.7,2.6,3.5,1,'Iris-versicolor'],
    [4.6,3.6,1,0.2,'Iris-setosa'],
    [5.9,3,4.2,1.5,'Iris-versicolor'],
    [6.3,2.8,5.1,1.5,'Iris-virginica'],
    [4.7,3.2,1.3,0.2,'Iris-setosa'],
    [5.1,3.3,1.7,0.5,'Iris-setosa'],
    [5.5,2.4,3.8,1.1,'Iris-versicolor'],
])

source = BatchOperator.fromDataframe(df, schemaStr='sepal_length double,
sepal_width double, petal_length double, petal_width double, category string')

va = VectorAssemblerBatchOp().setSelectedCols(["sepal_length", "sepal_width"])
\
  .setOutputCol("features")

som = SomBatchOp()\
  .setXdim(2) \
  .setYdim(4) \
  .setVdim(2) \
  .setSigma(1.0) \
  .setNumIters(10) \
  .setVectorCol("features")

source.link(va).link(som).print()
```

运行结果

	meta	xidx	yidx	weights	cnt
0	4,4,2,r	0	0	6.098901,2.6216097	8
1	4,4,2,r	0	1	5.7773294,2.6960063	13
2	4,4,2,r	0	2	5.4155393,2.6422026	7
3	4,4,2,r	0	3	4.960137,2.6060686	6
4	4,4,2,r	1	0	6.3758574,2.8099446	9
5	4,4,2,r	1	1	6.1287007,2.9054923	11
6	4,4,2,r	1	2	5.422063,2.9557745	5
7	4,4,2,r	1	3	4.8237553,3.0189981	16

```
8   4,4,2,r     2     0   6.771961,2.9707642    8
9   4,4,2,r     2     1   6.541121,3.1317835   13
10  4,4,2,r     2     2     5.73227,3.323552    4
11  4,4,2,r     2     3  5.0270276,3.4258003   17
12  4,4,2,r     3     0   7.296618,3.1325939   11
13  4,4,2,r     3     1   6.956377,3.2311482    7
14  4,4,2,r     3     2  5.9284854,3.5446692    4
15  4,4,2,r     3     3   5.236388,3.7456865   11
```

Som画图 (SomPlotBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.analysis.SomPlotBatchOp

Python 类名：SomPlotBatchOp

功能介绍

该组件用于对SOM组件产出的模型进行绘图，在可视化大屏直接查看模型信息

参数说明

Tsne (TsneBatchOp)

Java 类名：com.alibaba.alink.operator.batch.statistics.analysis.TsneBatchOp

Python 类名：TsneBatchOp

功能介绍

t-SNE(http://lvdmaaten.github.io/tsne/)是一种将高维数据降维的算法。
 本组件可将高维数据降维，特别地，支持对降至2维的数据绘制散点图。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
outputDim	outputDim	outputDim	Integer			2
perplexity	perplexity	perplexity	Double			50.0
randomSeed	randomSeed	randomSeed	Integer			-1
reservedCols	算法保留列名	算法保留列	String[]			null
theta	theta	theta	Double			0.5

备注：

- 对于高维数据（100维以上），可支持的规模在20万样本左右
- 对于低维数据（数十维），可支持的规模在100万样本左右
- 关于运行资源：tSNE是单机算法，因此**worker数设为1即可**，内存8G以上
- 性能评估 | case | 样本数 | 维数 | 时间 | | --- | --- | --- | --- | | 1 | 70000 | 784 | 13m | | 2 | 630000 | 64 | 3h |

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [5,2,3.5,1,'Iris-versicolor'],
    [5.1,3.7,1.5,0.4,'Iris-setosa'],
    [6.4,2.8,5.6,2.2,'Iris-virginica'],
    [6,2.9,4.5,1.5,'Iris-versicolor'],
    [4.9,3,1.4,0.2,'Iris-setosa'],
    [5.7,2.6,3.5,1,'Iris-versicolor'],
    [4.6,3.6,1,0.2,'Iris-setosa'],
    [5.9,3,4.2,1.5,'Iris-versicolor'],
    [6.3,2.8,5.1,1.5,'Iris-virginica'],
    [4.7,3.2,1.3,0.2,'Iris-setosa'],
    [5.1,3.3,1.7,0.5,'Iris-setosa'],
    [5.5,2.4,3.8,1.1,'Iris-versicolor'],
])

source = BatchOperator.fromDataframe(df, schemaStr='sepal_length double,
sepal_width double, petal_length double, petal_width double, category string')

va = VectorAssemblerBatchOp()\
    .setSelectedCols(["sepal_length", "sepal_width", "petal_length",
"petal_width"]) \
    .setOutputCol("features")

tsne = TsneBatchOp()\
    .setVectorCol("features")\
    .setOutputCol("xy")

source.link(va).link(tsne).print()
```

运行结果

```
     sepal_length  sepal_width  ...         features
xy
0             5.1          3.5  ...  5.1 3.5 1.4 0.2      1.5238329866266036
10.98943682214784
1             5.0          2.0  ...  5.0 2.0 3.5 1.0   -0.6428129573234383
-1.6899062329318093
2             5.1          3.7  ...  5.1 3.7 1.5 0.4      1.606258057625356
11.220010392753794
3             6.4          2.8  ...  6.4 2.8 5.6 2.2    -1.0427522852929407
-7.307929734935246
```

```
4             6.0          2.9  ...  6.0 2.9 4.5 1.5    -0.6031932003546642
-4.191947334076536
5             4.9          3.0  ...  4.9 3.0 1.4 0.2     0.9515272728748377
10.249849225819032
6             5.7          2.6  ...  5.7 2.6 3.5 1.0  -0.36782368346229943
-2.1300900246298373
7             4.6          3.6  ...  4.6 3.6 1.0 0.2     2.0129659333153356
10.487812103300815
```

Bert文本分类预测 (BertTextClassifierPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.BertTextClassifierPredictBatchOp

Python 类名：BertTextClassifierPredictBatchOp

功能介绍

与 BERT 文本分类训练组件对应的预测组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
inferBatchSize	推理数据批大小	推理数据批大小	Integer			256
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv"
schema = "label bigint, review string";
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schema) \
    .setIgnoreFirstLine(True)
data = data.where("review is not null")
data = data.firstN(300)
model = CsvSourceBatchOp() \
    .setFilePath("http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/bert_text_classifier_model.csv") \
    .setSchemaStr("model_id bigint, model_info string, label_value bigint")
predict = BertTextClassifierPredictBatchOp() \
```

```
    .setPredictionCol("pred") \
    .setPredictionDetailCol("pred_detail") \
    .linkFrom(model, data)
predict.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.BertTextClassifierPredictBatchO
p;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class BertTextClassifierPredictBatchOpTest {

    @Test
    public void testBertTextClassifierPredictBatchOp() throws Exception {
        String url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv";
        String schema = "label bigint, review string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schema)
            .setIgnoreFirstLine(true);
        data = data.where("review is not null");
        data = data.firstN(300);
        BatchOperator <?> model = new CsvSourceBatchOp()
            .setFilePath("http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/bert_text_classifier_model.csv")
            .setSchemaStr("model_id bigint, model_info string, label_value
bigint");
        BertTextClassifierPredictBatchOp predict = new
BertTextClassifierPredictBatchOp()
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail")
            .linkFrom(model, data);
        predict.print();
    }
}
```

运行结果

label	review	pred	pred_detail

1	大堂不错，有四星的样子，房间的设施一般，感觉有点旧，卫生间细节不错，各种配套东西都不错，感觉...	1	{"0":0.2737436294555664,"1":0.7262563705444336}
1	装修较旧,特别是地毯的材质颜色显得较脏,与四星的评级很不相称,门童很热情,赞一个	1	{"0":0.2737436294555664,"1":0.7262563705444336}
1	住过好多次这家酒店了，上次来到前台，服务员能准确的报出我的名字，感觉很亲切。四星级就是不一样...	1	{"0":0.2737436294555664,"1":0.7262563705444336}
1	非常不错的酒店，依山傍水，里面大片森林，散散步很不错，坐在湖边也休息也是不错的选择；房间很幽...	1	{"0":0.2737436294555664,"1":0.7262563705444336}
1	不知怎么回事，一直想给泰安华侨大厦点评，却一直未成功。因为酒店的服务很好。前一段时间和同事一...	1	{"0":0.2737436294555664,"1":0.7262563705444336}
...	...	...	...
1	还行吧，不算失望，离地铁2号线要走500M左右，但在中环旁边，自己开车去比较方便。	1	{"0":0.2737436294555664,"1":0.7262563705444336}
1	在携程上订了一天，实际入住两天，酒店照携程价加收了一天房费，并为我免费升级了商务房。中间问酒...	1	{"0":0.2737436294555664,"1":0.7262563705444336}
1	很不错的酒店。虽然离地铁站有一定距离，可是还是挺方便的。房间的床很大很舒服。但是相对的，别的...	1	{"0":0.2737436294555664,"1":0.7262563705444336}
1	超赞！虽然窗正对中环，但一点也不觉得吵。房间很干净、整齐，给人很舒服的感觉，服务也很好，价格...	1	{"0":0.2737436294555664,"1":0.7262563705444336}
1	7月25日到家人到泰山玩，通过携程订的华侨大厦的房间（说是搞活动，花280升级到360的房间...	1	{"0":0.2737436294555664,"1":0.7262563705444336}

Bert文本分类训练 (BertTextClassifierTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.BertTextClassifierTrainBatchOp

Python 类名：BertTextClassifierTrainBatchOp

功能介绍

在预训练的 BERT 模型的基础上增加一个全连接层，用于进行文本分类。

参数说明

名称	中文名称	描述
labelCol	标签列名	输入表中的标签列名
textCol	文本列	文本列
batchSize	数据批大小	数据批大小
bertModelName	BERT模型名字	BERT模型名字： Base-Chinese,Base-Multilingual-Case
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estima` 为所有 worker 都能访问到的目录
customConfigJson	自定义参数	对应 https://github.com/alibaba/EasyTransfer/blob/master/eas 中的config_json
intraOpParallelism	Op 间并发度	Op 间并发度
learningRate	学习率	学习率
maxSeqLength	句子截断长度	句子截断长度
numEpochs	epoch 数	epoch 数
numFineTunedLayers	微调层数	微调层数

numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设（需要取整），否则为总并发度减去 Worker 角色数。
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设（需要取整），否则为总并发度减去 PS 角色数。
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文且目录名与压缩文件主文件名一致，可以使用 http://, http是目录，那么只能使用本地路径，即 file://。
removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv"
schema = "label bigint, review string"
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schema) \
    .setIgnoreFirstLine(True)
data = data.where("review is not null")

train = BertTextClassifierTrainBatchOp() \
    .setTextCol("review") \
    .setLabelCol("label") \
    .setNumEpochs(2.) \
    .setNumFineTunedLayers(1) \
    .setMaxSeqLength(128) \
    .setBertModelName("Base-Chinese") \
    .linkFrom(data)

AkSinkBatchOp() \
    .setFilePath("/tmp/bert_text_classifier_model.ak") \
    .setOverwriteSink(True) \
    .linkFrom(train)
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.BertTextClassifierTrainBatchOp;
import com.alibaba.alink.operator.batch.sink.AkSinkBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class BertTextClassifierTrainBatchOpTest {

    @Test
    public void testBertTextClassifierTrainBatchOp() throws Exception {
        String url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv";
        String schema = "label bigint, review string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schema)
            .setIgnoreFirstLine(true);
        data = data.where("review is not null");

        BertTextClassifierTrainBatchOp train = new
BertTextClassifierTrainBatchOp()
            .setTextCol("review")
            .setLabelCol("label")
            .setNumEpochs(2.)
            .setNumFineTunedLayers(1)
            .setMaxSeqLength(128)
            .setBertModelName("Base-Chinese")
            .linkFrom(data);

        new AkSinkBatchOp()
            .setFilePath("/tmp/bert_text_classifier_model.ak")
            .setOverwriteSink(true)
            .linkFrom(train);
        BatchOperator.execute();
    }
}
```

Bert文本对分类预测 (BertTextPairClassifierPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.BertTextPairClassifierPredictBatchOp

Python 类名：BertTextPairClassifierPredictBatchOp

功能介绍

与 BERT 文本对分类训练组件对应的预测组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
inferBatchSize	推理数据批大小	推理数据批大小	Integer			256
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv"
schemaStr = "f_quality bigint, f_id_1 string, f_id_2 string, f_string_1 string,
f_string_2 string"
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schemaStr) \
    .setFieldDelimiter("\t") \
    .setIgnoreFirstLine(True) \
    .setQuoteChar(None)
data = data.firstN(300)
model = CsvSourceBatchOp() \
```

```
    .setFilePath("http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/bert_text_pair_classifier_model.csv") \
    .setSchemaStr("model_id bigint, model_info string, label_value bigint")
predict = BertTextPairClassifierPredictBatchOp() \
    .setPredictionCol("pred") \
    .setPredictionDetailCol("pred_detail") \
    .setReservedCols(["f_quality"]) \
    .linkFrom(model, data)
predict.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.BertTextPairClassifierPredictBa
tchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class BertTextPairClassifierPredictBatchOpTest {

    @Test
    public void testBertTextPairClassifierPredictBatchOpTest() throws Exception
{
        String url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv";
        String schemaStr = "f_quality bigint, f_id_1 string, f_id_2 string,
f_string_1 string, f_string_2 string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schemaStr)
            .setFieldDelimiter("\t")
            .setIgnoreFirstLine(true)
            .setQuoteChar(null);
        data = data.firstN(300);
        BatchOperator <?> model = new CsvSourceBatchOp()
            .setFilePath("http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/bert_text_pair_classifier_model.csv")
            .setSchemaStr("model_id bigint, model_info string, label_value
bigint");
        BertTextPairClassifierPredictBatchOp predict = new
BertTextPairClassifierPredictBatchOp()
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail")
            .setReservedCols("f_quality")
            .linkFrom(model, data);
        predict.print();
```

```
    }
}
```

运行结果

f_quality	pred	pred_detail
1	1	{"0":0.07034707069396973,"1":0.9296529293060303}
0	1	{"0":0.07034707069396973,"1":0.9296529293060303}
1	1	{"0":0.07034707069396973,"1":0.9296529293060303}
0	1	{"0":0.07034707069396973,"1":0.9296529293060303}
1	1	{"0":0.07034707069396973,"1":0.9296529293060303}
...	...	...
1	1	{"0":0.0704156756401062,"1":0.9295843243598938}
1	1	{"0":0.0704156756401062,"1":0.9295843243598938}
1	1	{"0":0.0704156756401062,"1":0.9295843243598938}
1	1	{"0":0.0704156756401062,"1":0.9295843243598938}
0	1	{"0":0.0704156756401062,"1":0.9295843243598938}

Bert文本对分类训练 (BertTextPairClassifierTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.BertTextPairClassifierTrainBatchOp

Python 类名：BertTextPairClassifierTrainBatchOp

功能介绍

在预训练的 BERT 模型的基础上增加一个全连接层，用于进行文本对分类任务。

参数说明

名称	中文名称	描述
labelCol	标签列名	输入表中的标签列名
textCol	文本列	文本列
textPairCol	文本对列	文本对列
batchSize	数据批大小	数据批大小
bertModelName	BERT模型名字	BERT模型名字： Base-Chinese,Base-Multilingual-Case
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estima` 为所有 worker 都能访问到的目录
customConfigJson	自定义参数	对应 https://github.com/alibaba/EasyTransfer/blob/master/eas 中的config_json
intraOpParallelism	Op 间并发度	Op 间并发度
learningRate	学习率	学习率

maxSeqLength	句子截断长度	句子截断长度
numEpochs	epoch 数	epoch 数
numFineTunedLayers	微调层数	微调层数
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设（需要取整），否则为总并发度减去 Worker 角色数。
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设（需要取整），否则为总并发度减去 PS 角色数。
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文且目录名与压缩文件主文件名一致，可以使用 http://, http是目录，那么只能使用本地路径，即 file://。
removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv"
schemaStr = "f_quality bigint, f_id_1 string, f_id_2 string, f_string_1 string,
f_string_2 string"
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schemaStr) \
    .setFieldDelimiter("\t") \
    .setIgnoreFirstLine(True) \
    .setQuoteChar(None)
data = ShuffleBatchOp().linkFrom(data)

train = BertTextPairClassifierTrainBatchOp() \

.setTextCol("f_string_1").setTextPairCol("f_string_2").setLabelCol("f_quality")
\
    .setNumEpochs(0.1) \
    .setMaxSeqLength(32) \
    .setNumFineTunedLayers(1) \
    .setBertModelName("Base-Uncased") \
```

```
    .linkFrom(data)

AkSinkBatchOp() \
    .setFilePath("/tmp/bert_text_pair_classifier_model.ak") \
    .setOverwriteSink(True) \
    .linkFrom(train)
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.common.utils.JsonConverter;
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.BertTextPairClassifierTrainBatc
hOp;
import com.alibaba.alink.operator.batch.dataproc.ShuffleBatchOp;
import com.alibaba.alink.operator.batch.sink.AkSinkBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.google.common.collect.ImmutableMap;
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

public class BertTextPairClassifierTrainBatchOpTest {

    @Test
    public void testBertTextPairClassifierTrainBatchOp() throws Exception {
        String url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv";
        String schemaStr = "f_quality bigint, f_id_1 string, f_id_2 string,
f_string_1 string, f_string_2 string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schemaStr)
            .setFieldDelimiter("\t")
            .setIgnoreFirstLine(true)
            .setQuoteChar(null);
        data = new ShuffleBatchOp().linkFrom(data);

        Map <String, Map <String, Object>> customConfig = new HashMap <>();
        customConfig.put("train_config", ImmutableMap.of("optimizer_config",
ImmutableMap.of("learning_rate", 0.01)));

        BertTextPairClassifierTrainBatchOp train = new
BertTextPairClassifierTrainBatchOp()
```

```
.setTextCol("f_string_1").setTextPairCol("f_string_2").setLabelCol("f_quality")
            .setNumEpochs(0.1)
            .setMaxSeqLength(32)
            .setNumFineTunedLayers(1)
            .setCustomJsonJson(JsonConverter.toJson(customConfig))
            .setBertModelName("Base-Uncased")
            .linkFrom(data);

        new AkSinkBatchOp()
            .setFilePath("/tmp/bert_text_pair_classifier_model.ak")
            .setOverwriteSink(true)
            .linkFrom(train);
        BatchOperator.execute();
    }
}
```

C45决策树分类预测 (C45PredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.C45PredictBatchOp

Python 类名：C45PredictBatchOp

功能介绍

- c45是一种常用的树模型
- c45组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
```

```
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = C45TrainBatchOp()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = C45PredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = C45PredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.C45PredictBatchOp;
import com.alibaba.alink.operator.batch.classification.C45TrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.classification.C45PredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class C45PredictBatchOpTest {
    @Test
    public void testC45PredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
```

```
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new C45TrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new C45PredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new C45PredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

C45决策树分类训练 (C45TrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.C45TrainBatchOp

Python 类名：C45TrainBatchOp

功能介绍

- c45是一种常用的树模型
- c45组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	

createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String			"series"
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			128
maxDepth	树的深度限制	树的深度限制	Integer			2147483647
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2147483647
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer			64
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.0

minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.0
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			2
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = C45TrainBatchOp()\
    .setLabelCol('label')\
```

```
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = C45PredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = C45PredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.C45PredictBatchOp;
import com.alibaba.alink.operator.batch.classification.C45TrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.classification.C45PredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class C45TrainBatchOpTest {
    @Test
    public void testC45TrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new C45TrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
```

```
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new C45PredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new C45PredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

CART决策树分类预测 (CartPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.CartPredictBatchOp

Python 类名：CartPredictBatchOp

功能介绍

- cart是一种常用的树模型
- cart组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
```

```
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = CartTrainBatchOp()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = CartPredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = CartPredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.CartPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.CartTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.classification.CartPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CartPredictBatchOpTest {
    @Test
    public void testCartPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
```

```
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new CartTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new CartPredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new CartPredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

CART决策树分类训练 (CartTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.CartTrainBatchOp

Python 类名：CartTrainBatchOp

功能介绍

- cart是一种常用的树模型
- cart组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	

createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String			"series"
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			128
maxDepth	树的深度限制	树的深度限制	Integer			2147483647
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2147483647
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer			64
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.0

minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.0
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			2
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = CartTrainBatchOp()\
    .setLabelCol('label')\
```

```
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = CartPredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = CartPredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.CartPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.CartTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.classification.CartPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CartTrainBatchOpTest {
    @Test
    public void testCartTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new CartTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
```

```
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new CartPredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new CartPredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

预测结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

决策树预测 (DecisionTreePredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.DecisionTreePredictBatchOp

Python 类名：DecisionTreePredictBatchOp

功能介绍

- 决策树组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
```

```
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = DecisionTreeTrainBatchOp()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = DecisionTreePredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = DecisionTreePredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.DecisionTreePredictBatchOp;
import
com.alibaba.alink.operator.batch.classification.DecisionTreeTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.classification.DecisionTreePredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DecisionTreePredictBatchOpTest {
    @Test
    public void testDecisionTreePredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
```

```
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new DecisionTreeTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new DecisionTreePredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new
DecisionTreePredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

决策树训练 (DecisionTreeTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.DecisionTreeTrainBatchOp

Python 类名：DecisionTreeTrainBatchOp

功能介绍

- 决策树组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String			"serie

maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			128
maxDepth	树的深度限制	树的深度限制	Integer			2147
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2147
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer			64
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.0
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.0
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			2

treeType	模型中树的类型	模型中树的类型，三种选项可选：树为一种方式gini, infoGain, infoGainRatio	String		"GINI", "INFOGAIN", "INFOGAINRATIO"	"GINI
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = DecisionTreeTrainBatchOp()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = DecisionTreePredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = DecisionTreePredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
```

```
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.DecisionTreePredictBatchOp;
import
com.alibaba.alink.operator.batch.classification.DecisionTreeTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.classification.DecisionTreePredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DecisionTreeTrainBatchOpTest {
    @Test
    public void testDecisionTreeTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new DecisionTreeTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new DecisionTreePredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new
DecisionTreePredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
```

```
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

FM分类预测 (FmClassifierPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.FmClassifierPredictBatchOp

Python 类名：FmClassifierPredictBatchOp

功能介绍

FM即因子分解机（Factor Machine），它的特点是考虑了特征之间的相互作用，是一种非线性模型。该组件使用FM模型解决分类问题。

算法原理

FM模型是线性模型的升级，是在线性表达式后面加入了新的交叉项特征及对应的权值，FM模型的表达式如下所示：

$$y = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

这里我们使用 Adagrad 优化算法求解该模型。算法原理细节可以参考文献[1]。

算法使用

FM算法是推荐领域被验证的效果较好的推荐方案之一，在电商、广告、视频、信息流、游戏的推荐领域有广泛应用。

文献

[1] S. Rendle, "Factorization Machines," 2010 IEEE International Conference on Data Mining, 2010, pp. 995-1000, doi: 10.1109/ICDM.2010.127.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		

modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
       ["1:1.1 3:2.0", 1.0],
       ["2:2.1 10:3.1", 1.0],
       ["1:1.2 5:3.2", 0.0],
       ["3:1.2 7:4.2", 0.0]
])

input = BatchOperator.fromDataframe(df, schemaStr='kv string, label double')
dataTest = input
fm = FmClassifierTrainBatchOp().setVectorCol("kv").setLabelCol("label")
model = input.link(fm)
```

```
predictor = FmClassifierPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, dataTest).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.FmClassifierPredictBatchOp;
import
com.alibaba.alink.operator.batch.classification.FmClassifierTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmClassifierPredictBatchOpTest {
    @Test
    public void testFmClassifierPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:1.1 3:2.0", 1.0),
            Row.of("2:2.1 10:3.1", 1.0),
            Row.of("1:1.2 5:3.2", 0.0),
            Row.of("3:1.2 7:4.2", 0.0)
        );
        BatchOperator <?> input = new MemSourceBatchOp(df, "kv string, label
double");
        BatchOperator dataTest = input;
        BatchOperator <?> fm = new
FmClassifierTrainBatchOp().setVectorCol("kv").setLabelCol("label");
        BatchOperator model = input.link(fm);
        BatchOperator <?> predictor = new
FmClassifierPredictBatchOp().setPredictionCol("pred");
        predictor.linkFrom(model, dataTest).print();
    }

}
```

运行结果

kv	label	pred
1:1.1 3:2.0	1.0	1.0
2:2.1 10:3.1	1.0	1.0

1:1.2 5:3.2	0.0	0.0
3:1.2 7:4.2	0.0	0.0

FM分类训练 (FmClassifierTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.FmClassifierTrainBatchOp

Python 类名：FmClassifierTrainBatchOp

功能介绍

FM即因子分解机（Factor Machine），它的特点是考虑了特征之间的相互作用，是一种非线性模型。该组件使用FM模型解决分类问题。

算法原理

FM模型是线性模型的升级，是在线性表达式后面加入了新的交叉项特征及对应的权值，FM模型的表达式如下所示：

$$y = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

这里我们使用 Adagrad 优化算法求解该模型。算法原理细节可以参考文献[1]。

算法使用

FM算法是推荐领域被验证的效果较好的推荐方案之一，在电商、广告、视频、信息流、游戏的推荐领域有广泛应用。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献

[1] S. Rendle, "Factorization Machines," 2010 IEEE International Conference on Data Mining, 2010, pp. 995-1000, doi: 10.1109/ICDM.2010.127.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		

batchSize	迭代数据batch size	数据batch size	Integer			-1
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1.0E-6
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
initStdev	初始化参数的标准差	初始化参数的标准差	Double			0.05
lambda0	常数项正则化系数	常数项正则化系数	Double			0.0
lambda1	线性项正则化系数	线性项正则化系数	Double			0.0
lambda2	二次项正则化系数	二次项正则化系数	Double			0.0
learnRate	学习率	学习率	Double			0.01
numEpochs	epoch数	epoch数	Integer			10
numFactor	因子数	因子数	Integer			10
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true
withLinearItem	是否含有线性项	是否含有线性项	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
       ["1:1.1 3:2.0", 1.0],
       ["2:2.1 10:3.1", 1.0],
       ["1:1.2 5:3.2", 0.0],
       ["3:1.2 7:4.2", 0.0]
])

input = BatchOperator.fromDataframe(df, schemaStr='kv string, label double')
dataTest = input

# load data
dataTest = input
fm = FmClassifierTrainBatchOp().setVectorCol("kv").setLabelCol("label")
model = input.link(fm)

predictor = FmClassifierPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, dataTest).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.FmClassifierPredictBatchOp;
import
com.alibaba.alink.operator.batch.classification.FmClassifierTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmClassifierTrainBatchOpTest {
    @Test
    public void testFmClassifierTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:1.1 3:2.0", 1.0),
            Row.of("2:2.1 10:3.1", 1.0),
            Row.of("1:1.2 5:3.2", 0.0),
            Row.of("3:1.2 7:4.2", 0.0)
        );
        BatchOperator <?> input = new MemSourceBatchOp(df, "kv string, label
double");
        BatchOperator dataTest = input;
        BatchOperator <?> fm = new
FmClassifierTrainBatchOp().setVectorCol("kv").setLabelCol("label");
        BatchOperator model = input.link(fm);
        BatchOperator <?> predictor = new
FmClassifierPredictBatchOp().setPredictionCol("pred");
        predictor.linkFrom(model, dataTest).print();
    }
}
```

运行结果

kv	label	pred
1:1.1 3:2.0	1.0	1.0
2:2.1 10:3.1	1.0	1.0
1:1.2 5:3.2	0.0	0.0
3:1.2 7:4.2	0.0	0.0

备注

该组件的输入为训练数据，输出为Fm分类模型。

GBDT分类器预测 (GbdtPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.GbdtPredictBatchOp

Python 类名：GbdtPredictBatchOp

功能介绍

功能介绍

梯度提升决策树(Gradient Boosting Decision Trees)二分类，是经典的基于梯度提升的有监督学习模型，可以用来解决二分类问题

算法原理

梯度提升决策树模型构建了一个由多棵决策树组成的组合模型。每一棵决策树对应一个弱学习器，将这些弱学习器组合在一起，可以达到比较好的分类或回归效果。

梯度提升的基本递推结构为：

$$F*{m}(x) = F*{m-1}(x) + \beta_{m}h(x;a_m)$$

其中 $h(x;a*m)$ *通常为一棵 CART[2] 决策树，* ${a_m}$ *为在这棵决策树下的分割变量，* $*beta*{m}h(x;a_m)$ 为在 $h(x;a_m)$ 约束下的步长，通过这个递推结构即可得出最终模型。

算法使用

对于一些常见的二分类问题，都可以使用这个算法解决，模型拥有较好的性能，且拥有不错的可解释性，在实际场景中，应用较为广泛。

- 支持连续特征和离散特征
- 支持数据采样和特征采样
- 目标分类必须为两个

文献或出处

1. Friedman J H. Greedy function approximation: a gradient boosting machine[J]. Annals of statistics, 2001: 1189-1232.
2. Breiman, L., Friedman, J. H., Olshen, R. and Stone, C. (1983). Classification and Regression Trees. Wadsworth, Belmont, CA.
3. weka
4. xgboost
5. lightgbm

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
```

```
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = GbdtTrainBatchOp()\
    .setLearningRate(1.0)\
    .setNumTrees(3)\
    .setMinSamplesPerLeaf(1)\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = GbdtPredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = GbdtPredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.GbdtPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.GbdtTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.classification.GbdtPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GbdtPredictBatchOpTest {
    @Test
    public void testGbdtPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
```

```
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new GbdtTrainBatchOp()
            .setLearningRate(1.0)
            .setNumTrees(3)
            .setMinSamplesPerLeaf(1)
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new GbdtPredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new GbdtPredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":0.9849144946061075,"1":0.01508550539389248}
2.0000	B	1	1	0	0	{"0":0.9849144946061075,"1":0.01508550539389248}
3.0000	C	2	2	1	1	{"0":0.015085505393892529,"1":0.9849144946061075}
4.0000	D	3	3	1	1	{"0":0.015085505393892529,"1":0.9849144946061075}

GBDT分类器训练 (GbdtTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.GbdtTrainBatchOp

Python 类名：GbdtTrainBatchOp

功能介绍

梯度提升决策树(Gradient Boosting Decision Trees)二分类，是经典的基于梯度提升的有监督学习模型，可以用来解决二分类问题

算法原理

梯度提升决策树模型构建了一个由多棵决策树组成的组合模型。每一棵决策树对应一个弱学习器，将这些弱学习器组合在一起，可以达到比较好的分类或回归效果。

梯度提升的基本递推结构为：

$$F*{m}(x) = F*{m-1}(x) + \beta_{m}h(x;a_m)$$

其中 $h(x;a*m)$ *通常为一棵 CART[2] 决策树，* ${a_m}$ *为在这棵决策树下的分割变量，* $\beta*{m}h(x;a_m)$ 为在 $h(x;a_m)$ 约束下的步长，通过这个递推结构即可得出最终模型。

算法使用

对于一些常见的二分类问题，都可以使用这个算法解决，模型拥有较好的性能，且拥有不错的可解释性，在实际场景中，应用较为广泛。

- 支持连续特征和离散特征
- 支持数据采样和特征采样
- 目标分类必须为两个

文献或出处

1. Friedman J H. Greedy function approximation: a gradient boosting machine[J]. Annals of statistics, 2001: 1189-1232.
2. Breiman, L., Friedman, J. H., Olshen, R. and Stone, C. (1983). Classification and Regression Trees. Wadsworth, Belmont, CA.
3. weka
4. xgboost
5. lightgbm

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	

labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
criteria	树分裂的策略	树分裂的策略，可以为PAI, XGBOOST	String		"PAI", "XGBOOST"	"P
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	nu
featureImportanceType	特征重要性类型	特征重要性类型（默认为GAIN）	String		"WEIGHT", "GAIN", "COVER"	"G
featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double			1.
gamma	xgboost中的l2正则项	xgboost中的l2正则项	Double			0.
lambda	xgboost中的l1正则项	xgboost中的l1正则项	Double			0.
learningRate	学习率	学习率（默认为0.3）	Double			0.
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			12
maxDepth	树的深度限制	树的深度限制	Integer			6

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			1(
minSumHessianPerLeaf	叶子节点最小Hessian值	叶子节点最小Hessian值（默认为0）	Double			0.
newtonStep	是否使用二阶梯度	是否使用二阶梯度	Boolean			tr
numTrees	模型中树的棵数	模型中树的棵数	Integer			1(
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double			1.
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	nu
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	nu

参数建议

对于训练效果来说，比较重要的参数是 树的棵树+学习率、叶子节点最小样本数、单颗树最大深度、特征采样比例。

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = GbdtTrainBatchOp()\
    .setLearningRate(1.0)\
    .setNumTrees(3)\
    .setMinSamplesPerLeaf(1)\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = GbdtPredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = GbdtPredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.GbdtPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.GbdtTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.classification.GbdtPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GbdtTrainBatchOpTest {
    @Test
    public void testGbdtTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new GbdtTrainBatchOp()
            .setLearningRate(1.0)
            .setNumTrees(3)
            .setMinSamplesPerLeaf(1)
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new GbdtPredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new GbdtPredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":0.9849144946061075,"1":0.01508550539389248}
2.0000	B	1	1	0	0	{"0":0.9849144946061075,"1":0.01508550539389248}
3.0000	C	2	2	1	1	{"0":0.015085505393892529,"1":0.9849144946061075}
4.0000	D	3	3	1	1	{"0":0.015085505393892529,"1":0.9849144946061075}

最近邻（大规模版本） (HugeKnnBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.HugeKnnBatchOp

Python 类名：HugeKnnBatchOp

功能介绍

KNN (K Nearest Neighbor）是一种分类算法。 KNN算法的核心思想是如果一个样本在特征空间中的k个最相邻的样本中的大多数属于某一个类别， 则该样本也属于这个类别，并具有这个类别上样本的特性。

这里Huge版KNN是针对训练数据（即字典表）较大时进行优化的KNN版本。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE"	"EUCLIDEAN"
featureCols	特征列名数组	特征列名数组，默认全选	String[]			null
k	topK	topK	Integer			10
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1, 0, 0],
    [2, 8, 8],
    [1, 1, 2],
    [2, 9, 10],
    [1, 3, 1],
    [2, 10, 7]
])

dataSourceOp = BatchOperator.fromDataframe(df, schemaStr="label int, f0 int, f1
int")

op =
HugeKnnBatchOp().setDistanceType("EUCLIDEAN").setK(2).setLabelCol("label").setF
eatureCols(["f0", "f1"]).setPredictionCol("pred").linkFrom(dataSourceOp,
dataSourceOp)
op.print()
```

Java代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;

public class HugeKnnBatchOpTest extends AlinkTestBase {
    @Test
    public void test1() throws Exception {
        Row[] rows = new Row[] {
```

```
            Row.of(1, 0, 0),
            Row.of(2, 8, 8),
            Row.of(1, 1, 2),
            Row.of(2, 9, 10),
            Row.of(1, 3, 1),
            Row.of(2, 10, 7)
        };

        MemSourceBatchOp inputData = new MemSourceBatchOp(Arrays.asList(rows),
"label int, f0 int, f1 int");

        HugeKnnBatchOp op = new HugeKnnBatchOp()
            .setDistanceType("EUCLIDEAN")
            .setK(2)
            .setLabelCol("label")
            .setFeatureCols("f0", "f1")
            .setPredictionCol("pred")
            .linkFrom(inputData, inputData);
        op.print();
    }
}
```

运行结果

label	f0	f1	pred
1	0	0	1
1	1	2	1
1	3	1	1
2	8	8	2
2	9	10	2
2	10	7	2

ID3决策树分类预测 (Id3PredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.Id3PredictBatchOp

Python 类名：Id3PredictBatchOp

功能介绍

- id3是一种常用的树模型
- id3组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
```

```
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = Id3TrainBatchOp()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = Id3PredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = Id3PredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.Id3PredictBatchOp;
import com.alibaba.alink.operator.batch.classification.Id3TrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.classification.Id3PredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class Id3PredictBatchOpTest {
    @Test
    public void testId3PredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
```

```
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new Id3TrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new Id3PredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new Id3PredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

ID3决策树分类训练 (Id3TrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.Id3TrainBatchOp

Python 类名：Id3TrainBatchOp

功能介绍

- id3是一种常用的树模型
- id3组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	

createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String			"series"
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			128
maxDepth	树的深度限制	树的深度限制	Integer			2147483647
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2147483647
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer			64
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.0

minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.0
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			2
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = Id3TrainBatchOp()\
    .setLabelCol('label')\
```

```
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = Id3PredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = Id3PredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.Id3PredictBatchOp;
import com.alibaba.alink.operator.batch.classification.Id3TrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.classification.Id3PredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class Id3TrainBatchOpTest {
    @Test
    public void testId3TrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new Id3TrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
```

```
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new Id3PredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new Id3PredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

KerasSequential分类预测 (KerasSequentialClassifierPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.KerasSequentialClassifierPredictBatchOp

Python 类名：KerasSequentialClassifierPredictBatchOp

功能介绍

与 KerasSequential分类训练组件对应的预测组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
inferBatchSize	推理数据批大小	推理数据批大小	Integer			256
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
source = CsvSourceBatchOp() \
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/random_tensor.csv") \
    .setSchemaStr("tensor string, label int")

source = ToTensorBatchOp() \
    .setSelectedCol("tensor") \
    .setTensorDataType("DOUBLE") \
    .setTensorShape([200, 3]) \
    .linkFrom(source)

trainBatchOp = KerasSequentialClassifierTrainBatchOp() \
```

```
    .setTensorCol("tensor") \
    .setLabelCol("label") \
    .setLayers([
        "Conv1D(256, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Dropout(0.1)",
        "MaxPooling1D(pool_size=8)",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Flatten()"
    ]) \
    .setOptimizer("Adam()") \
    .setNumEpochs(1) \
    .linkFrom(source)

predictBatchOp = KerasSequentialClassifierPredictBatchOp() \
    .setPredictionCol("pred") \
    .setPredictionDetailCol("pred_detail") \
    .setReservedCols(["label"]) \
    .linkFrom(trainBatchOp, source)
predictBatchOp.lazyPrint(10)
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.KerasSequentialClassifierPredic
tBatchOp;
import
com.alibaba.alink.operator.batch.classification.KerasSequentialClassifierTrainB
atchOp;
import com.alibaba.alink.operator.batch.dataproc.ToTensorBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class KerasSequentialClassifierPredictBatchOpTest {

    @Test
    public void testKerasSequentialClassifierPredictBatchOp() throws Exception
{
        BatchOperator<?> source = new CsvSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/random_tensor.csv")
            .setSchemaStr("tensor string, label int");

        source = new ToTensorBatchOp()
```

```
            .setSelectedCol("tensor")
            .setTensorDataType("DOUBLE")
            .setTensorShape(200, 3)
            .linkFrom(source);

        KerasSequentialClassifierTrainBatchOp trainBatchOp = new
KerasSequentialClassifierTrainBatchOp()
            .setTensorCol("tensor")
            .setLabelCol("label")
            .setLayers(new String[] {
                "Conv1D(256, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Dropout(0.1)",
                "MaxPooling1D(pool_size=8)",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Flatten()"
            })
            .setOptimizer("Adam()")
            .setNumEpochs(1)
            .linkFrom(source);

        KerasSequentialClassifierPredictBatchOp predictBatchOp = new
KerasSequentialClassifierPredictBatchOp()
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail")
            .setReservedCols("label")
            .linkFrom(trainBatchOp, source);
        predictBatchOp.lazyPrint(10);
        BatchOperator.execute();
    }
}
```

运行结果

label	pred	pred_detail
0	0	{"0":0.636155836712713,"1":0.36384416328728697}
1	0	{"0":0.6334926095655181,"1":0.3665073904344819}
1	0	{"0":0.6381823204965642,"1":0.3618176795034358}
1	0	{"0":0.6376416296248051,"1":0.362358370375195}
1	0	{"0":0.6345856985385896,"1":0.36541430146141035}
1	0	{"0":0.6357593109428179,"1":0.364240689057182}

0	0	{"0":0.6404387449594703,"1":0.3595612550405296}
1	0	{"0":0.6372702905549685,"1":0.36272970944503136}
0	0	{"0":0.635502012172225,"1":0.36449798782777487}
0	0	{"0":0.6262401788033837,"1":0.37375982119661644}

KerasSequential分类训练 (KerasSequentialClassifierTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.KerasSequentialClassifierTrainBatchOp

Python 类名：KerasSequentialClassifierTrainBatchOp

功能介绍

构建一个 Keras 的 Sequential 模型， 训练分类模型。

通过 layers 参数指定构成 Sequential 模型的网络层，Alink 会自动在最开始添加 Input 层，在最后添加 Dense 层和激活层，得到完整的模型用于训练。

指定 layers 参数时，使用的是 Python 语句，例如

```
"Conv1D(256, 5, padding='same', activation='relu')",
"Conv1D(128, 5, padding='same', activation='relu')",
"Dropout(0.1)",
"MaxPooling1D(pool_size=8)",
"Conv1D(128, 5, padding='same', activation='relu')",
"Conv1D(128, 5, padding='same', activation='relu')",
"Flatten()"
```

`tf.keras.layers` 内的网络层已经提前 import，可以直接使用。 使用的 TensorFlow 版本是 2.3.1。

该组件可以接 `KerasSequentialClassifierPredictBatchOp` 或 `KerasSequentialClassifierPredictStreamOp` 进行推理。

参数说明

名称	中文名称	描述	类型	是
labelCol	标签列名	输入表中的标签列名	String	✓
layers	各 layer 的描述	各 layer 的描述，使用 Python 语法，例如 "Conv1D(256, 5, padding='same', activation='relu')"	String[]	✓
tensorCol	tensor列	tensor列	String	✓
batchSize	数据批大小	数据批大小	Integer	

bestMetric	最优指标	判断模型最优时用的指标，仅在总并发度为 1 时起作用。都支持的有：loss； 二分类还支持：auc, precision, recall, binary_accuracy, false_negatives, false_positives, true_negatives, true_positives；多分类还支持：sparse_categorical_accuracy；回归还支持：mean_absolute_error, mean_absolute_percentage_error, mean_squared_error, mean_squared_logarithmic_error, root_mean_squared_error	String	
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estimator` 的 `model_dir` 传入，需要为所有 worker 都能访问到的目录	String	
intraOpParallelism	Op 间并发度	Op 间并发度	Integer	
learningRate	学习率	学习率	Double	
numEpochs	epoch数	epoch数	Integer	
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer	
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer	
optimizer	优化器	优化器，使用 Python 语法，例如 "Adam(learning_rate=0.1)"	String	
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String	

removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只会删除必要的文件	Boolean	
saveBestOnly	是否导出最优的 checkpoint	是否导出最优的 checkpoint，仅在总并发度为 1 时生效	Boolean	
saveCheckpointsEpochs	每隔多少 epochs 保存 checkpoints	每隔多少 epochs 保存 checkpoints	Double	
saveCheckpointsSecs	每隔多少秒保存 checkpoints	每隔多少秒保存 checkpoints	Double	
validationSplit	验证集比例	验证集比例，仅在总并发度为 1 时生效	Double	

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
source = CsvSourceBatchOp() \
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/random_tensor.csv") \
    .setSchemaStr("tensor string, label int")

source = ToTensorBatchOp() \
    .setSelectedCol("tensor") \
    .setTensorDataType("DOUBLE") \
    .setTensorShape([200, 3]) \
    .linkFrom(source)

trainBatchOp = KerasSequentialClassifierTrainBatchOp() \
    .setTensorCol("tensor") \
    .setLabelCol("label") \
    .setLayers([
        "Conv1D(256, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Dropout(0.1)",
        "MaxPooling1D(pool_size=8)",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
```

```
        "Flatten()"
    ]) \
    .setOptimizer("Adam()") \
    .setNumEpochs(1) \
    .linkFrom(source)

predictBatchOp = KerasSequentialClassifierPredictBatchOp() \
    .setPredictionCol("pred") \
    .setPredictionDetailCol("pred_detail") \
    .setReservedCols(["label"]) \
    .linkFrom(trainBatchOp, source)
predictBatchOp.lazyPrint(10)
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.KerasSequentialClassifierPredic
tBatchOp;
import
com.alibaba.alink.operator.batch.classification.KerasSequentialClassifierTrainB
atchOp;
import com.alibaba.alink.operator.batch.dataproc.ToTensorBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class KerasSequentialClassifierTrainBatchOpTest {

    @Test
    public void testKerasSequentialClassifierTrainBatchOp() throws Exception {
        BatchOperator<?> source = new CsvSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/random_tensor.csv")
            .setSchemaStr("tensor string, label int");

        source = new ToTensorBatchOp()
            .setSelectedCol("tensor")
            .setTensorDataType("DOUBLE")
            .setTensorShape(200, 3)
            .linkFrom(source);

        KerasSequentialClassifierTrainBatchOp trainBatchOp = new
KerasSequentialClassifierTrainBatchOp()
            .setTensorCol("tensor")
            .setLabelCol("label")
            .setLayers(new String[] {
```

```
                "Conv1D(256, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Dropout(0.1)",
                "MaxPooling1D(pool_size=8)",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Flatten()"
            })
            .setOptimizer("Adam()")
            .setNumEpochs(1)
            .linkFrom(source);

        KerasSequentialClassifierPredictBatchOp predictBatchOp = new
KerasSequentialClassifierPredictBatchOp()
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail")
            .setReservedCols("label")
            .linkFrom(trainBatchOp, source);
        predictBatchOp.lazyPrint(10);
        BatchOperator.execute();
    }
}
```

运行结果

label	pred	pred_detail
0	0	{"0":0.636155836712713,"1":0.36384416328728697}
1	0	{"0":0.6334926095655181,"1":0.3665073904344819}
1	0	{"0":0.6381823204965642,"1":0.3618176795034358}
1	0	{"0":0.6376416296248051,"1":0.362358370375195}
1	0	{"0":0.6345856985385896,"1":0.36541430146141035}
1	0	{"0":0.6357593109428179,"1":0.364240689057182}
0	0	{"0":0.6404387449594703,"1":0.3595612550405296}
1	0	{"0":0.6372702905549685,"1":0.36272970944503136}
0	0	{"0":0.635502012172225,"1":0.36449798782777487}
0	0	{"0":0.6262401788033837,"1":0.37375982119661644}

最近邻分类预测 (KnnPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.KnnPredictBatchOp

Python 类名：KnnPredictBatchOp

功能介绍

KNN (K Nearest Neighbor）是一种分类算法。 KNN算法的核心思想是如果一个样本在特征空间中的k个最相邻的样本中的大多数属于某一个类别; 则该样本也属于这个类别，并具有这个类别上样本的特性。

相比于Huge版KNN，此KNN的优势在于训练数据（即KNN中的字典表）较小时，速度较快。 此外，KNN的训练与一般机器学习模型的训练过程不同：在KNN训练中我们只进行一些字典表的预处理，而在预测过程中才会进行计算预测每个数据点的类别。 因此，KNN的训练和预测通常同时使用，一般不单独使用。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
k	topK	topK	Integer			10
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1, 0, 0],
    [2, 8, 8],
    [1, 1, 2],
    [2, 9, 10],
    [1, 3, 1],
    [2, 10, 7]
])

dataSourceOp = BatchOperator.fromDataframe(df, schemaStr="label int, f0 int, f1
int")
trainOp = KnnTrainBatchOp().setFeatureCols(["f0",
"f1"]).setLabelCol("label").setDistanceType("EUCLIDEAN").linkFrom(dataSourceOp)
predictOp =
KnnPredictBatchOp().setPredictionCol("pred").setK(4).linkFrom(trainOp,
dataSourceOp)
predictOp.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.KnnPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.KnnTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class KnnPredictBatchOpTest {
    @Test
    public void testKnnPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, 0, 0),
            Row.of(2, 8, 8),
            Row.of(1, 1, 2),
            Row.of(2, 9, 10),
            Row.of(1, 3, 1),
            Row.of(2, 10, 7)
        );
        BatchOperator <?> dataSourceOp = new MemSourceBatchOp(df, "label int,
f0 int, f1 int");
        BatchOperator <?> trainOp = new KnnTrainBatchOp().setFeatureCols("f0",
"f1").setLabelCol("label")
            .setDistanceType("EUCLIDEAN").linkFrom(dataSourceOp);
        BatchOperator <?> predictOp = new
KnnPredictBatchOp().setPredictionCol("pred").setK(4).linkFrom(trainOp,
            dataSourceOp);
        predictOp.print();
    }
}
```

运行结果

label	f0	f1	pred
1	0	0	1
2	8	8	2
1	1	2	1
2	9	10	2
1	3	1	1
2	10	7	2

最近邻分类训练 (KnnTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.KnnTrainBatchOp

Python 类名：KnnTrainBatchOp

功能介绍

KNN (K Nearest Neighbor）是一种分类算法。 KNN算法的核心思想是如果一个样本在特征空间中的k个最相邻的样本中的大多数属于某一个类别; 则该样本也属于这个类别，并具有这个类别上样本的特性。

KNN的训练与一般机器学习模型的训练过程不同：在KNN训练中我们只进行一些字典表的预处理，而在预测过程中才会进行计算预测每个数据点的类别。 因此，KNN的训练和预测通常同时使用，一般不单独使用。

支持 EUCLIDEAN 和 COSINE 两种距离计算方式。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE"	"EUCLIDEAN"
featureCols	特征列名数组	特征列名数组，默认全选	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1, 0, 0],
    [2, 8, 8],
    [1, 1, 2],
    [2, 9, 10],
    [1, 3, 1],
    [2, 10, 7]
])

dataSourceOp = BatchOperator.fromDataframe(df, schemaStr="label int, f0 int, f1
int")

trainOp = KnnTrainBatchOp().setFeatureCols(["f0",
"f1"]).setLabelCol("label").setDistanceType("EUCLIDEAN").linkFrom(dataSourceOp)
predictOp =
KnnPredictBatchOp().setPredictionCol("pred").setK(4).linkFrom(trainOp,
dataSourceOp)
predictOp.print()

trainOp = KnnTrainBatchOp().setFeatureCols(["f0",
"f1"]).setLabelCol("label").setDistanceType("EUCLIDEAN").linkFrom(dataSourceOp)
predictOp =
KnnPredictBatchOp().setPredictionCol("pred").setK(4).linkFrom(trainOp,
dataSourceOp)
predictOp.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.KnnPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.KnnTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KnnTrainBatchOpTest {
    @Test
```

```
    public void testKnnTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, 0, 0),
            Row.of(2, 8, 8),
            Row.of(1, 1, 2),
            Row.of(2, 9, 10),
            Row.of(1, 3, 1),
            Row.of(2, 10, 7)
        );
        BatchOperator <?> dataSourceOp = new MemSourceBatchOp(df, "label int,
f0 int, f1 int");
        BatchOperator <?> trainOp = new KnnTrainBatchOp().setFeatureCols("f0",
"f1").setLabelCol("label")
            .setDistanceType("EUCLIDEAN").linkFrom(dataSourceOp);
        BatchOperator <?> predictOp = new
KnnPredictBatchOp().setPredictionCol("pred").setK(4).linkFrom(trainOp,
            dataSourceOp);
        predictOp.print();
        trainOp = new KnnTrainBatchOp().setFeatureCols("f0",
"f1").setLabelCol("label").setDistanceType("EUCLIDEAN")
            .linkFrom(dataSourceOp);
        predictOp = new
KnnPredictBatchOp().setPredictionCol("pred").setK(4).linkFrom(trainOp,
dataSourceOp);
        predictOp.print();
    }
}
```

运行结果

label	f0	f1	pred
1	0	0	1
2	8	8	2
1	1	2	1
2	9	10	2
1	3	1	1
2	10	7	2

线性支持向量机预测 (LinearSvmPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.LinearSvmPredictBatchOp

Python 类名：LinearSvmPredictBatchOp

功能介绍

线性SVM算法是经典的二分类算法，通过对打标签样本集合训练得到模型，使用模型预测样本的标签。逻辑回归组件支持稀疏、稠密两种数据格式。

算法原理

SVM使用铰链损失函数（hinge loss）计算经验风险（empirical risk）并在求解系统中加入了正则化项以优化结构风险，是一个具有稀疏性 和稳健性的分类器。

算法使用

SVM在各领域的模式识别问题中有应用，包括人像识别、文本分类、手写字符识别、生物信息学等。

文献

[1] Vapnik, V．Statistical learning theory. 1998 (Vol. 3). . New York, NY：Wiley，1998：Chapter 10-11, pp.401-492.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			

reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 2]
])

input = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int, label
int')

dataTest = input
colnames = ["f0","f1"]
svm = LinearSvmTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = input.link(svm)

predictor = LinearSvmPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, dataTest).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.LinearSvmPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.LinearSvmTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LinearSvmPredictBatchOpTest {
    @Test
    public void testLinearSvmPredictBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 2)
        );
        BatchOperator <?> input = new MemSourceBatchOp(df_data, "f0 int, f1
int, label int");
        BatchOperator dataTest = input;
        BatchOperator <?> svm = new
LinearSvmTrainBatchOp().setFeatureCols("f0", "f1").setLabelCol("label");
        BatchOperator model = input.link(svm);
        BatchOperator <?> predictor = new
LinearSvmPredictBatchOp().setPredictionCol("pred");
        predictor.linkFrom(model, dataTest).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2

2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	2	2

线性支持向量机训练 (LinearSvmTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.LinearSvmTrainBatchOp

Python 类名：LinearSvmTrainBatchOp

功能介绍

线性SVM算法是经典的二分类算法，通过对打标签样本集合训练得到模型，使用模型预测样本的标签。逻辑回归组件支持稀疏、稠密两种数据格式。

算法原理

SVM使用铰链损失函数（hinge loss）计算经验风险（empirical risk）并在求解系统中加入了正则化项以优化结构风险，是一个具有稀疏性 和稳健性的分类器。

算法使用

SVM在各领域的模式识别问题中有应用，包括人像识别、文本分类、手写字符识别、生物信息学等。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献

[1] Vapnik, V．Statistical learning theory. 1998 (Vol. 3). . New York, NY：Wiley，1998：Chapter 10-11, pp.401-492.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1.0E-6

featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"	null
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean			true
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 2]
])

input = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int, label
int')

dataTest = input
colnames = ["f0","f1"]
svm = LinearSvmTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = input.link(svm)

predictor = LinearSvmPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, dataTest).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.LinearSvmPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.LinearSvmTrainBatchOp;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LinearSvmTrainBatchOpTest {
    @Test
    public void testLinearSvmTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 2)
        );
        BatchOperator <?> input = new MemSourceBatchOp(df_data, "f0 int, f1
int, label int");
        BatchOperator dataTest = input;
        BatchOperator <?> svm = new
LinearSvmTrainBatchOp().setFeatureCols("f0", "f1").setLabelCol("label");
        BatchOperator model = input.link(svm);
        BatchOperator <?> predictor = new
LinearSvmPredictBatchOp().setPredictionCol("pred");
        predictor.linkFrom(model, dataTest).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	2	2

备注

1. 该组件的输入为训练数据，输出为SVM模型。
2. 参数数据库的使用方式可以覆盖多个参数的使用方式。

逻辑回归预测 (LogisticRegressionPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.LogisticRegressionPredictBatchOp

Python 类名：LogisticRegressionPredictBatchOp

功能介绍

逻辑回归算法是经典的二分类算法，通过对打标签样本集合训练得到模型，使用模型预测样本的标签。逻辑回归组件支持稀疏、稠密两种数据格式。

算法原理

面对二分类问题，建立代价函数，然后通过优化方法迭代求解出最优的模型参数，然后测试验证我们这个求解的模型的好坏。 Logistic回归虽然名字里带“回归”，但是它实际上是一种分类方法，主要用于二分类问题（即输出只有两种，分别代表两个类别）回归模型中，y是一个定性变量，比如y=0或1， logistic方法主要应用于研究某些事件发生的概率。

算法使用

常用于数据挖掘，疾病自动诊断，经济预测等领域。例如，探讨引发疾病的危险因素，并根据危险因素预测疾病发生的概率等。以心脏病病情分析为例，选择两组人群， 一组是心脏病组，一组是非心脏病组，两组人群必定具有不同的属性及身体指标。因此因变量就为是否有心脏病，值为“是”或“否”，自变量就可以包括很多了， 如年龄、性别、最大心跳数、血压、胆固醇、空腹血糖等。自变量既可以是连续的，也可以是分类的。然后通过logistic回归分析，可以得到自变量的权重， 从而可以大致了解到底哪些因素是心脏病的危险因素。同时根据该权值可以根据危险因素预测一个人心脏病的可能性。

文献或出处

[1] Wright, R. E. (1995). Logistic regression. In L. G. Grimm & P. R. Yarnold (Eds.), Reading and understanding multivariate statistics (pp. 217–244). American Psychological Association.

[2] https://baike.baidu.com/item/logistic%E5%9B%9E%E5%BD%92

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		

modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 2]
])

input = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int, label
int')
```

```
# load data
dataTest = input
colnames = ["f0","f1"]
lr =
LogisticRegressionTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = input.link(lr)

predictor = LogisticRegressionPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, dataTest).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.LogisticRegressionPredictBatchO
p;
import
com.alibaba.alink.operator.batch.classification.LogisticRegressionTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LogisticRegressionPredictBatchOpTest {
    @Test
    public void testLogisticRegressionPredictBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 2)
        );
        BatchOperator <?> input = new MemSourceBatchOp(df_data, "f0 int, f1
int, label int");
        BatchOperator dataTest = input;
        BatchOperator <?> lr = new
LogisticRegressionTrainBatchOp().setFeatureCols("f0",
"f1").setLabelCol("label");
        BatchOperator model = input.link(lr);
```

```
        BatchOperator <?> predictor = new
LogisticRegressionPredictBatchOp().setPredictionCol("pred");
        predictor.linkFrom(model, dataTest).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	2	2

逻辑回归训练 (LogisticRegressionTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.LogisticRegressionTrainBatchOp

Python 类名：LogisticRegressionTrainBatchOp

功能介绍

逻辑回归算法是经典的二分类算法，通过对打标签样本集合训练得到模型，使用模型预测样本的标签。逻辑回归组件支持稀疏、稠密两种数据格式。

算法原理

面对二分类问题，建立代价函数，然后通过优化方法迭代求解出最优的模型参数，然后测试验证我们这个求解的模型的好坏。Logistic回归虽然名字里带“回归”， 但是它实际上是一种分类方法，主要用于二分类问题（即输出只有两种，分别代表两个类别）回归模型中，y是一个定性变量，比如y=0或1， logistic方法主要应用于研究某些事件发生的概率。

算法使用

常用于数据挖掘，疾病自动诊断，经济预测等领域。例如，探讨引发疾病的危险因素，并根据危险因素预测疾病发生的概率等。以心脏病病情分析为例，选择两组人群， 一组是心脏病组，一组是非心脏病组，两组人群必定具有不同的属性及身体指标。因此因变量就为是否有心脏病，值为“是”或“否”，自变量就可以包括很多了， 如年龄、性别、最大心跳数、血压、胆固醇、空腹血糖等。自变量既可以是连续的，也可以是分类的。然后通过logistic回归分析，可以得到自变量的权重， 从而可以大致了解到底哪些因素是心脏病的危险因素。同时根据该权值可以根据危险因素预测一个人心脏病的可能性。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Wright, R. E. (1995). Logistic regression. In L. G. Grimm & P. R. Yarnold (Eds.), Reading and understanding multivariate statistics (pp. 217–244). American Psychological Association.

[2] https://baike.baidu.com/item/logistic%E5%9B%9E%E5%BD%92

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

labelCol	标签列名	输入表中的标签列名	String	✓		
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1.0E-6
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"	null
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean			true

vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 2]
])

input = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int, label
int')

# load data
dataTest = input
colnames = ["f0","f1"]
lr =
LogisticRegressionTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
```

```
model = input.link(lr)

predictor = LogisticRegressionPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, dataTest).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.LogisticRegressionPredictBatchO
p;
import
com.alibaba.alink.operator.batch.classification.LogisticRegressionTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LogisticRegressionTrainBatchOpTest {
    @Test
    public void testLogisticRegressionTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 2)
        );
        BatchOperator <?> input = new MemSourceBatchOp(df_data, "f0 int, f1
int, label int");
        BatchOperator dataTest = input;
        BatchOperator <?> lr = new
LogisticRegressionTrainBatchOp().setFeatureCols("f0",
"f1").setLabelCol("label");
        BatchOperator model = input.link(lr);
        BatchOperator <?> predictor = new
LogisticRegressionPredictBatchOp().setPredictionCol("pred");
        predictor.linkFrom(model, dataTest).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	2	2

备注

1. 该组件的输入为训练数据，输出为逻辑回归模型。
2. 参数数据库的使用方式可以覆盖多个参数的使用方式。

多层感知机分类预测 (MultilayerPerceptronPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.MultilayerPerceptronPredictBatchOp

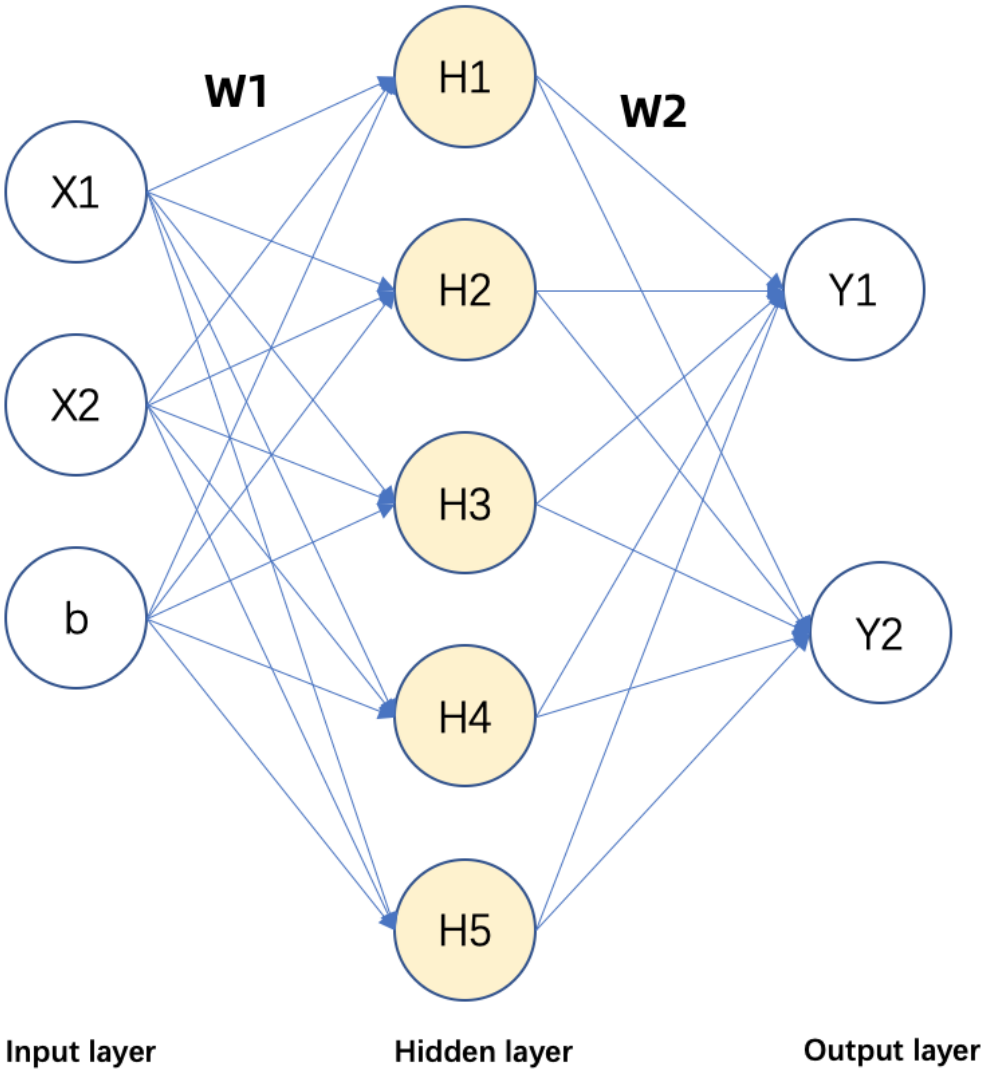
Python 类名：MultilayerPerceptronPredictBatchOp

功能介绍

多层感知机（MLP，Multilayer Perceptron）也被称作人工神经网络（ANN，Artificial Neural Network），经常用来进行多分类问题的训练预测。

算法原理

多层感知机算法除了输入输出层外，它中间可以有多个隐层，最简单的MLP只含一个隐层，即三层的结构，如下图：



从上图可以看到，多层感知机层与层之间是全连接的。多层感知机最左边是输入层，中间是隐藏层，最后是输出层。 其中输出层对应的是各个分类标签，输出层 的每一个节点对应每一个标签的出现的概率。

算法使用

多层感知机主要用于多分类问题，类似文字识别，语音识别，文本分析等问题。

文献

[1]Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences MW Gardner, SR Dorling - Atmospheric environment, 1998 - Elsevier.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [5,2,3.5,1,'Iris-versicolor'],
    [5.1,3.7,1.5,0.4,'Iris-setosa'],
    [6.4,2.8,5.6,2.2,'Iris-virginica'],
    [6,2.9,4.5,1.5,'Iris-versicolor'],
```

```
    [4.9,3,1.4,0.2,'Iris-setosa'],
    [5.7,2.6,3.5,1,'Iris-versicolor'],
    [4.6,3.6,1,0.2,'Iris-setosa'],
    [5.9,3,4.2,1.5,'Iris-versicolor'],
    [6.3,2.8,5.1,1.5,'Iris-virginica'],
    [4.7,3.2,1.3,0.2,'Iris-setosa'],
    [5.1,3.3,1.7,0.5,'Iris-setosa'],
    [5.5,2.4,3.8,1.1,'Iris-versicolor'],
])

data = BatchOperator.fromDataframe(df, schemaStr='sepal_length double,
sepal_width double, petal_length double, petal_width double, category string')

mlpc = MultilayerPerceptronTrainBatchOp() \
  .setFeatureCols(["sepal_length", "sepal_width", "petal_length",
"petal_width"]) \
  .setLabelCol("category") \
  .setLayers([4, 8, 3]) \
  .setMaxIter(10)

model = mlpc.linkFrom(data)

predictor = MultilayerPerceptronPredictBatchOp()\
  .setPredictionCol('p')

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.MultilayerPerceptronPredictBatc
hOp;
import
com.alibaba.alink.operator.batch.classification.MultilayerPerceptronTrainBatchO
p;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MultilayerPerceptronPredictBatchOpTest {
    @Test
    public void testMultilayerPerceptronPredictBatchOp() throws Exception {
```

```
        List <Row> df = Arrays.asList(
            Row.of(5.0, 2.0, 3.5, 1.0, "Iris-versicolor"),
            Row.of(5.1, 3.7, 1.5, 0.4, "Iris-setosa"),
            Row.of(6.4, 2.8, 5.6, 2.2, "Iris-virginica"),
            Row.of(6.0, 2.9, 4.5, 1.5, "Iris-versicolor"),
            Row.of(4.9, 3.0, 1.4, 0.2, "Iris-setosa"),
            Row.of(5.7, 2.6, 3.5, 1.0, "Iris-versicolor"),
            Row.of(4.6, 3.6, 1.0, 0.2, "Iris-setosa"),
            Row.of(5.9, 3.0, 4.2, 1.5, "Iris-versicolor"),
            Row.of(6.3, 2.8, 5.1, 1.5, "Iris-virginica"),
            Row.of(4.7, 3.2, 1.3, 0.2, "Iris-setosa"),
            Row.of(5.1, 3.3, 1.7, 0.5, "Iris-setosa"),
            Row.of(5.5, 2.4, 3.8, 1.1, "Iris-versicolor")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string");
        BatchOperator <?> mlpc = new MultilayerPerceptronTrainBatchOp()
            .setFeatureCols("sepal_length", "sepal_width", "petal_length",
"petal_width")
            .setLabelCol("category")
            .setLayers(new int[] {4, 8, 3})
            .setMaxIter(10);
        BatchOperator model = mlpc.linkFrom(data);
        BatchOperator <?> predictor = new MultilayerPerceptronPredictBatchOp()
            .setPredictionCol("p");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

sepal_length	sepal_width	petal_length	petal_width	category	p
5.0000	2.0000	3.5000	1.0000	Iris-versicolor	Iris-versicolor
5.1000	3.7000	1.5000	0.4000	Iris-setosa	Iris-versicolor
6.4000	2.8000	5.6000	2.2000	Iris-virginica	Iris-versicolor
6.0000	2.9000	4.5000	1.5000	Iris-versicolor	Iris-versicolor
4.9000	3.0000	1.4000	0.2000	Iris-setosa	Iris-versicolor
5.7000	2.6000	3.5000	1.0000	Iris-versicolor	Iris-versicolor
4.6000	3.6000	1.0000	0.2000	Iris-setosa	Iris-setosa
5.9000	3.0000	4.2000	1.5000	Iris-versicolor	Iris-versicolor

6.3000	2.8000	5.1000	1.5000	Iris-virginica	Iris-versicolor
4.7000	3.2000	1.3000	0.2000	Iris-setosa	Iris-versicolor
5.1000	3.3000	1.7000	0.5000	Iris-setosa	Iris-versicolor
5.5000	2.4000	3.8000	1.1000	Iris-versicolor	Iris-versicolor

多层感知机分类训练 (MultilayerPerceptronTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.MultilayerPerceptronTrainBatchOp

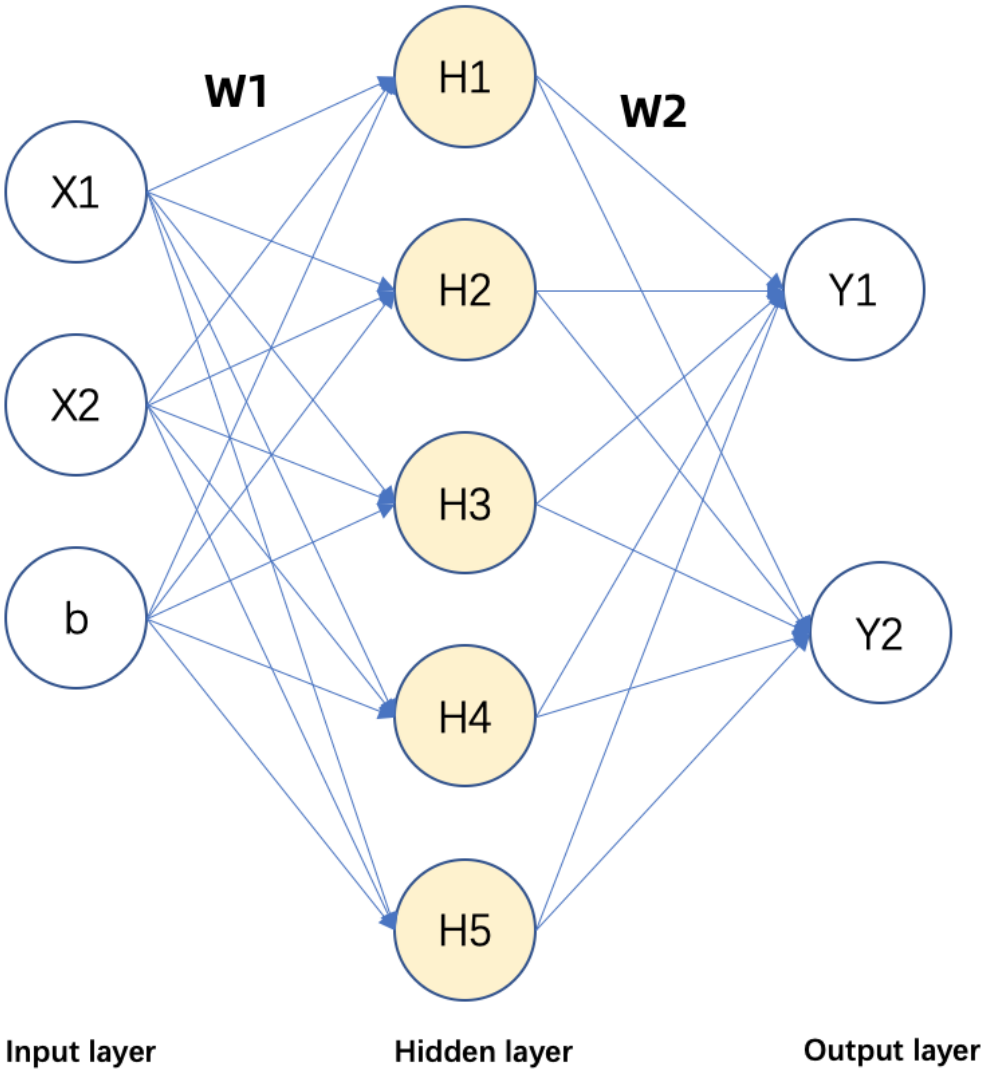
Python 类名：MultilayerPerceptronTrainBatchOp

功能介绍

多层感知机（MLP，Multilayer Perceptron）也被称作人工神经网络（ANN，Artificial Neural Network），经常用来进行多分类问题的训练预测。

算法原理

多层感知机算法除了输入输出层外，它中间可以有多个隐层，最简单的MLP只含一个隐层，即三层的结构，如下图：



从上图可以看到，多层感知机层与层之间是全连接的。多层感知机最左边是输入层，中间是隐藏层，最后是输出层。 其中输出层对应的是各个分类标签，输出层 的每一个节点对应每一个标签的出现的概率。

算法使用

多层感知机主要用于多分类问题，类似文字识别，语音识别，文本分析等问题。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献

[1]Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences MW Gardner, SR Dorling - Atmospheric environment, 1998 - Elsevier.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
layers	神经网络层大小	神经网络层大小	int[]	✓		
blockSize	数据分块大小，默认值64	数据分块大小，默认值64	Integer			64
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为1.0e-6	Double		[0.0, +inf)	1.0E-6
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
initialWeights	初始权重值	初始权重值	DenseVector			null
l1	L1 正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
maxIter	最大迭代步数	最大迭代步数，默认为 100	Integer		[1, +inf)	100

vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [5,2,3.5,1,'Iris-versicolor'],
    [5.1,3.7,1.5,0.4,'Iris-setosa'],
    [6.4,2.8,5.6,2.2,'Iris-virginica'],
    [6,2.9,4.5,1.5,'Iris-versicolor'],
    [4.9,3,1.4,0.2,'Iris-setosa'],
    [5.7,2.6,3.5,1,'Iris-versicolor'],
    [4.6,3.6,1,0.2,'Iris-setosa'],
    [5.9,3,4.2,1.5,'Iris-versicolor'],
    [6.3,2.8,5.1,1.5,'Iris-virginica'],
    [4.7,3.2,1.3,0.2,'Iris-setosa'],
    [5.1,3.3,1.7,0.5,'Iris-setosa'],
    [5.5,2.4,3.8,1.1,'Iris-versicolor'],
])

data = BatchOperator.fromDataframe(df, schemaStr='sepal_length double,
sepal_width double, petal_length double, petal_width double, category string')

mlpc = MultilayerPerceptronTrainBatchOp() \
  .setFeatureCols(["sepal_length", "sepal_width", "petal_length",
"petal_width"]) \
  .setLabelCol("category") \
  .setLayers([4, 8, 3]) \
  .setMaxIter(10)

mlpc.linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;
```

```
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.MultilayerPerceptronTrainBatchO
p;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MultilayerPerceptronTrainBatchOpTest {
    @Test
    public void testMultilayerPerceptronTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(5.0, 2.0, 3.5, 1.0, "Iris-versicolor"),
            Row.of(5.1, 3.7, 1.5, 0.4, "Iris-setosa"),
            Row.of(6.4, 2.8, 5.6, 2.2, "Iris-virginica"),
            Row.of(6.0, 2.9, 4.5, 1.5, "Iris-versicolor"),
            Row.of(4.9, 3.0, 1.4, 0.2, "Iris-setosa"),
            Row.of(5.7, 2.6, 3.5, 1.0, "Iris-versicolor"),
            Row.of(4.6, 3.6, 1.0, 0.2, "Iris-setosa"),
            Row.of(5.9, 3.0, 4.2, 1.5, "Iris-versicolor"),
            Row.of(6.3, 2.8, 5.1, 1.5, "Iris-virginica"),
            Row.of(4.7, 3.2, 1.3, 0.2, "Iris-setosa"),
            Row.of(5.1, 3.3, 1.7, 0.5, "Iris-setosa"),
            Row.of(5.5, 2.4, 3.8, 1.1, "Iris-versicolor")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string");
        BatchOperator <?> mlpc = new MultilayerPerceptronTrainBatchOp()
            .setFeatureCols("sepal_length", "sepal_width", "petal_length",
"petal_width")
            .setLabelCol("category")
            .setLayers(new int[] {4, 8, 3})
            .setMaxIter(10);
        mlpc.linkFrom(data).print();
    }
}
```

运行结果

model_id	
0	{"vectorCol":null,"isVectorInput":"false","layers":"[4,8,3]","featureCols":"[\"sepal_length\

1048576	{"data": [-0.04909618379949584,0.05244036093590636,-0.09152901171897616,-0.1142079 4,-0.059945429358191755,-0.004153281219841397,-0.14924287831591582,0.5654
2251799812636672	null
2251799812636673	null
2251799812636674	null

朴素贝叶斯预测 (NaiveBayesPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.NaiveBayesPredictBatchOp

Python 类名：NaiveBayesPredictBatchOp

功能介绍

使用朴素贝叶斯模型用于多分类任务的预测。

使用方式

该组件是预测组件，需要配合训练组件 NaiveBayesTrainBatchOp 使用。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
       [1.0, 1.0, 0.0, 1.0, 1],
       [1.0, 0.0, 1.0, 1.0, 1],
       [1.0, 0.0, 1.0, 1.0, 1],
```

```
        [0.0, 1.0, 1.0, 0.0, 0],
        [0.0, 1.0, 1.0, 0.0, 0],
        [0.0, 1.0, 1.0, 0.0, 0],
        [0.0, 1.0, 1.0, 0.0, 0],
        [1.0, 1.0, 1.0, 1.0, 1],
        [0.0, 1.0, 1.0, 0.0, 0]
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='f0 double, f1
double, f2 double, f3 double, label int')

colnames = ["f0","f1","f2", "f3"]
ns = NaiveBayesTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = batchData.link(ns)

predictor = NaiveBayesPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.NaiveBayesPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.NaiveBayesTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class NaiveBayesPredictBatchOpTest {
    @Test
    public void testNaiveBayesPredictBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1.0, 1.0, 0.0, 1.0, 1),
            Row.of(1.0, 0.0, 1.0, 1.0, 1),
            Row.of(1.0, 0.0, 1.0, 1.0, 1),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(1.0, 1.0, 1.0, 1.0, 1),
            Row.of(0.0, 1.0, 1.0, 0.0, 0)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data,
```

```
            "f0 double, f1 double, f2 double, f3 double, label int");
        BatchOperator <?> ns = new
NaiveBayesTrainBatchOp().setFeatureCols("f0", "f1", "f2", "f3").setLabelCol(
            "label");
        BatchOperator model = batchData.link(ns);
        BatchOperator <?> predictor = new
NaiveBayesPredictBatchOp().setPredictionCol("pred");
        predictor.linkFrom(model, batchData).print();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred
1.0	1.0	0.0	1.0	1	1
1.0	0.0	1.0	1.0	1	1
1.0	0.0	1.0	1.0	1	1
0.0	1.0	1.0	0.0	0	0
0.0	1.0	1.0	0.0	0	0
0.0	1.0	1.0	0.0	0	0
0.0	1.0	1.0	0.0	0	0
1.0	1.0	1.0	1.0	1	1
0.0	1.0	1.0	0.0	0	0

朴素贝叶斯文本分类预测 (NaiveBayesTextPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.NaiveBayesTextPredictBatchOp

Python 类名：NaiveBayesTextPredictBatchOp

功能介绍

训练一个朴素贝叶斯文本分类模型用于多分类任务。

使用方式

该组件是预测组件，需要配合预测组件 NaiveBayesTextTrainBatchOp 使用。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为 [DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
          ["$31$0:1.0 1:1.0 2:1.0 30:1.0","1.0  1.0  1.0  1.0", '1'],
          ["$31$0:1.0 1:1.0 2:0.0 30:1.0","1.0  1.0  0.0  1.0", '1'],
          ["$31$0:1.0 1:0.0 2:1.0 30:1.0","1.0  0.0  1.0  1.0", '1'],
          ["$31$0:1.0 1:0.0 2:1.0 30:1.0","1.0  0.0  1.0  1.0", '1'],
          ["$31$0:0.0 1:1.0 2:1.0 30:0.0","0.0  1.0  1.0  0.0", '0'],
          ["$31$0:0.0 1:1.0 2:1.0 30:0.0","0.0  1.0  1.0  0.0", '0'],
          ["$31$0:0.0 1:1.0 2:1.0 30:0.0","0.0  1.0  1.0  0.0", '0']
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='sv string, dv
string, label string')

# train op
ns = NaiveBayesTextTrainBatchOp().setVectorCol("sv").setLabelCol("label")
model = batchData.link(ns)
# predict op
predictor =
NaiveBayesTextPredictBatchOp().setVectorCol("sv").setReservedCols(["sv",
"label"]).setPredictionCol("pred")
predictor.linkFrom(model, batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.NaiveBayesTextPredictBatchOp;
import
com.alibaba.alink.operator.batch.classification.NaiveBayesTextTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class NaiveBayesTextPredictBatchOpTest {
    @Test
```

```
    public void testNaiveBayesTextPredictBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("$31$0:1.0 1:1.0 2:1.0 30:1.0", "1.0  1.0  1.0  1.0", "1"),
            Row.of("$31$0:1.0 1:1.0 2:0.0 30:1.0", "1.0  1.0  0.0  1.0", "1"),
            Row.of("$31$0:1.0 1:0.0 2:1.0 30:1.0", "1.0  0.0  1.0  1.0", "1"),
            Row.of("$31$0:1.0 1:0.0 2:1.0 30:1.0", "1.0  0.0  1.0  1.0", "1"),
            Row.of("$31$0:0.0 1:1.0 2:1.0 30:0.0", "0.0  1.0  1.0  0.0", "0"),
            Row.of("$31$0:0.0 1:1.0 2:1.0 30:0.0", "0.0  1.0  1.0  0.0", "0"),
            Row.of("$31$0:0.0 1:1.0 2:1.0 30:0.0", "0.0  1.0  1.0  0.0", "0")
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data, "sv string,
dv string, label string");
        BatchOperator <?> ns = new
NaiveBayesTextTrainBatchOp().setVectorCol("sv").setLabelCol("label");
        BatchOperator model = batchData.link(ns);
        BatchOperator <?> predictor = new
NaiveBayesTextPredictBatchOp().setVectorCol("sv").setReservedCols("sv",
            "label").setPredictionCol("pred");
        predictor.linkFrom(model, batchData).print();
    }
}
```

运行结果

sv	label	pred
"$31$0:1.0 1:1.0 2:1.0 30:1.0"	1	1
"$31$0:1.0 1:1.0 2:0.0 30:1.0"	1	1
"$31$0:1.0 1:0.0 2:1.0 30:1.0"	1	1
"$31$0:1.0 1:0.0 2:1.0 30:1.0"	1	1
"$31$0:0.0 1:1.0 2:1.0 30:0.0"	0	0
"$31$0:0.0 1:1.0 2:1.0 30:0.0"	0	0
"$31$0:0.0 1:1.0 2:1.0 30:0.0"	0	0

朴素贝叶斯文本分类训练 (NaiveBayesTextTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.NaiveBayesTextTrainBatchOp

Python 类名：NaiveBayesTextTrainBatchOp

功能介绍

训练一个朴素贝叶斯文本分类模型用于多分类任务。

算法原理

朴素贝叶斯算法基于贝叶斯定理和一个"朴素"的假设：各特征间两两条件独立。

通过贝叶斯定理可以在给定特征$(x*1,\cdots,x_n)$时计算类别为y的概率：*$P(y|x_1,\cdots,x_n)=\frac{P(y)P(x_1,\cdots,x_n|y)}{P(x_1,\cdots,x_n) }$，而通过特征间两两独立的假设可以将上面公式简化为：$P(y|x_1,\cdots,x_n)=\frac{P(y)\prod{i = 1}^{n}P(x_i|y)}{P(x_1,\cdots,x_n)}$。

在朴素贝叶斯用于文本分类时，文本中的词语（token）对应一个特征。 类别c的概率可以估算为 $\hat P(c)=\frac{Nc}{N}$，*其中 N_c 是类别为c的总文本数，N的总文本数。 词语t_i在类别c所包含的文本出现的频次用 $T*{cti}$表示，那么可以用于估算概率 $\hat P(t|c)= \frac{T*{cti}}{\sum{t'\in V}T_{ct'}}$。

与一般的朴素贝叶斯分类模型类似，可以添加平滑系数来解决T_{ct_i} 为 0 时的问题。

上面描述的是多项分布时的模型，即$T*{ct_i}$ *可以去大于1的值。如果考虑的是二项分布，那么$T*{ct_i}$只能取值0或者1。

使用方式

该组件是训练组件，需要配合预测组件 NaiveBayesTextPredictBatch/StreamOp 使用。

为了训练朴素贝叶斯模型，需要指定参数向量列名（vectorCol）和标签列名（labelCol）。 通过参数模型类型可以设置使用多项分布或者二项分布。 平滑因子可以通过参数 smoothing 指定，默认为不平滑。

组件还支持设置每条样本的权重，通过参数权重列（weightCol）指定。

文献索引

Naive Bayes text classification: https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

labelCol	标签列名	输入表中的标签列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
modelType	模型类型	取值为Multinomial或Bernoulli	String		"Multinomial", "Bernoulli"	"Multinomial"
smoothing	算法参数	光滑因子，默认为1.0	Double		[0.0, +inf)	1.0
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
          ["$31$0:1.0 1:1.0 2:1.0 30:1.0","1.0  1.0  1.0  1.0", '1'],
          ["$31$0:1.0 1:1.0 2:0.0 30:1.0","1.0  1.0  0.0  1.0", '1'],
          ["$31$0:1.0 1:0.0 2:1.0 30:1.0","1.0  0.0  1.0  1.0", '1'],
          ["$31$0:1.0 1:0.0 2:1.0 30:1.0","1.0  0.0  1.0  1.0", '1'],
          ["$31$0:0.0 1:1.0 2:1.0 30:0.0","0.0  1.0  1.0  0.0", '0'],
          ["$31$0:0.0 1:1.0 2:1.0 30:0.0","0.0  1.0  1.0  0.0", '0'],
          ["$31$0:0.0 1:1.0 2:1.0 30:0.0","0.0  1.0  1.0  0.0", '0']
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='sv string, dv
```

```
string, label string')

# train op
ns = NaiveBayesTextTrainBatchOp().setVectorCol("sv").setLabelCol("label")
model = batchData.link(ns)
# predict op
predictor =
NaiveBayesTextPredictBatchOp().setVectorCol("sv").setReservedCols(["sv",
"label"]).setPredictionCol("pred")
predictor.linkFrom(model, batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.NaiveBayesTextPredictBatchOp;
import
com.alibaba.alink.operator.batch.classification.NaiveBayesTextTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class NaiveBayesTextTrainBatchOpTest {
    @Test
    public void testNaiveBayesTextTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("$31$0:1.0 1:1.0 2:1.0 30:1.0", "1.0  1.0  1.0  1.0", "1"),
            Row.of("$31$0:1.0 1:1.0 2:0.0 30:1.0", "1.0  1.0  0.0  1.0", "1"),
            Row.of("$31$0:1.0 1:0.0 2:1.0 30:1.0", "1.0  0.0  1.0  1.0", "1"),
            Row.of("$31$0:1.0 1:0.0 2:1.0 30:1.0", "1.0  0.0  1.0  1.0", "1"),
            Row.of("$31$0:0.0 1:1.0 2:1.0 30:0.0", "0.0  1.0  1.0  0.0", "0"),
            Row.of("$31$0:0.0 1:1.0 2:1.0 30:0.0", "0.0  1.0  1.0  0.0", "0"),
            Row.of("$31$0:0.0 1:1.0 2:1.0 30:0.0", "0.0  1.0  1.0  0.0", "0")
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data, "sv string,
dv string, label string");
        BatchOperator <?> ns = new
NaiveBayesTextTrainBatchOp().setVectorCol("sv").setLabelCol("label");
        BatchOperator model = batchData.link(ns);
        BatchOperator <?> predictor = new
NaiveBayesTextPredictBatchOp().setVectorCol("sv").setReservedCols("sv",
            "label").setPredictionCol("pred");
        predictor.linkFrom(model, batchData).print();
```

```
    }
}
```

运行结果

sv	label	pred
"$31$0:1.0 1:1.0 2:1.0 30:1.0"	1	1
"$31$0:1.0 1:1.0 2:0.0 30:1.0"	1	1
"$31$0:1.0 1:0.0 2:1.0 30:1.0"	1	1
"$31$0:1.0 1:0.0 2:1.0 30:1.0"	1	1
"$31$0:0.0 1:1.0 2:1.0 30:0.0"	0	0
"$31$0:0.0 1:1.0 2:1.0 30:0.0"	0	0
"$31$0:0.0 1:1.0 2:1.0 30:0.0"	0	0

朴素贝叶斯训练 (NaiveBayesTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.NaiveBayesTrainBatchOp

Python 类名：NaiveBayesTrainBatchOp

功能介绍

训练一个朴素贝叶斯模型用于多分类任务。

算法原理

朴素贝叶斯算法基于贝叶斯定理和一个"朴素"的假设：各特征间两两条件独立。

通过贝叶斯定理可以在给定特征$(x*1,\cdots,x_n)$时计算类别为y的概率：*
$P(y|x_1,\cdots,x_n)=\frac{P(y)P(x_1,\cdots,x_n|y)}{P(x_1,\cdots,x_n) }$，而通过特征间两两独立的假设可以将上面公式简化为：$P(y|x_1,\cdots,x_n)=\frac{P(y)\prod{i = 1}^{n}P(x_i|y)}{P(x_1,\cdots,x_n)}$。

对于连续型特征$x*i$，通常假设 $P(x_i|y)$满足高斯分布 (μ_y,σ_y)，参数可以通过对训练数据进行最大似然估计得到。对于离散型特征x_i，$P(x_i|y)=\frac{N{yi} + \alpha}{N{y}+\alpha n}$，其中 $N*{yi}$ 表示类别为 y特征x_i共同出现的样本数， N_y表示类别 y 的样本数，α 是平滑系数。

使用方式

该组件是训练组件，需要配合预测组件 NaiveBayesPredictBatch/StreamOp 使用。

为了训练朴素贝叶斯模型，需要指定参数特征列名（featureCols）和标签列名（labelCol）。 特征列名中，数值类型的列默认看作连续型特征处理，如果需要强制作为离散型特征处理，需要将这些列的列名添加到参数离散特征列名（categoricalCol）中。 平滑因子可以通过参数 smoothing 指定，默认为不平滑。

组件还支持设置每条样本的权重，通过参数权重列（weightCol）指定。

文献索引

H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓		

labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为 [BIGINTEGER, BOOLEAN, INTEGER, LONG, STRING]	
smoothing	算法参数	光滑因子，默认为0.0	Double		[0.0, +inf)	0.0
weightCol	权重列名	权重列对应的列名	String		所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
       [1.0, 1.0, 0.0, 1.0, 1],
       [1.0, 0.0, 1.0, 1.0, 1],
       [1.0, 0.0, 1.0, 1.0, 1],
       [0.0, 1.0, 1.0, 0.0, 0],
       [0.0, 1.0, 1.0, 0.0, 0],
       [0.0, 1.0, 1.0, 0.0, 0],
       [0.0, 1.0, 1.0, 0.0, 0],
       [1.0, 1.0, 1.0, 1.0, 1],
       [0.0, 1.0, 1.0, 0.0, 0]
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='f0 double, f1
double, f2 double, f3 double, label int')
```

```
# train op
colnames = ["f0","f1","f2", "f3"]
ns = NaiveBayesTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = batchData.link(ns)
# predict op
predictor = NaiveBayesPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.NaiveBayesPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.NaiveBayesTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class NaiveBayesTrainBatchOpTest {
    @Test
    public void testNaiveBayesTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1.0, 1.0, 0.0, 1.0, 1),
            Row.of(1.0, 0.0, 1.0, 1.0, 1),
            Row.of(1.0, 0.0, 1.0, 1.0, 1),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(1.0, 1.0, 1.0, 1.0, 1),
            Row.of(0.0, 1.0, 1.0, 0.0, 0)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data,
            "f0 double, f1 double, f2 double, f3 double, label int");
        BatchOperator <?> ns = new
NaiveBayesTrainBatchOp().setFeatureCols("f0", "f1", "f2", "f3").setLabelCol(
            "label");
        BatchOperator model = batchData.link(ns);
        BatchOperator <?> predictor = new
NaiveBayesPredictBatchOp().setPredictionCol("pred");
        predictor.linkFrom(model, batchData).print();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred
1.0	1.0	0.0	1.0	1	1
1.0	0.0	1.0	1.0	1	1
1.0	0.0	1.0	1.0	1	1
0.0	1.0	1.0	0.0	0	0
0.0	1.0	1.0	0.0	0	0
0.0	1.0	1.0	0.0	0	0
0.0	1.0	1.0	0.0	0	0
1.0	1.0	1.0	1.0	1	1
0.0	1.0	1.0	0.0	0	0

感知机预测 (PerceptronPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.PerceptronPredictBatchOp

Python 类名：PerceptronPredictBatchOp

功能介绍

感知机(perceptron)是二类分类的线性分类模型，其输入为实例的特征向量，输出为实例的类别。

算法原理

感知机对应于输入空间中将实例划分为正负两类的分离超平面，通过超平面将正负样本分离，属于判别模型。具体原理可参考文献。

算法使用

该算法经常应用二分类问题中，该算法支持稀疏和稠密两种输入样本。

文献或出处

[1] Gallant, Stephen I. "Perceptron-based learning algorithms." IEEE Transactions on neural networks 1.2 (1990): 179-191.

[2] 李航. 统计学习方法[M]. 北京：清华大学出版社，2012。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			

reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 2]
])

input = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int, label
int')
dataTest = input
colnames = ["f0","f1"]
per = PerceptronTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = input.link(per)

predictor = PerceptronPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, dataTest).print()
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	2	2

感知机训练 (PerceptronTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.PerceptronTrainBatchOp

Python 类名：PerceptronTrainBatchOp

功能介绍

感知机(perceptron)是二类分类的线性分类模型，其输入为实例的特征向量，输出为实例的类别。

算法原理

感知机对应于输入空间中将实例划分为正负两类的分离超平面，通过超平面将正负样本分离，属于判别模型。具体原理可参考文献。

算法使用

该算法经常应用二分类问题中，该算法支持稀疏和稠密两种输入样本。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Gallant, Stephen I. "Perceptron-based learning algorithms." IEEE Transactions on neural networks 1.2 (1990): 179-191.

[2] 李航. 统计学习方法[M]. 北京：清华大学出版社，2012。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1.0E-6

featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"	null
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean			true
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 2]
])

input = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int, label
int')
dataTest = input
colnames = ["f0","f1"]
per = PerceptronTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = input.link(per)

predictor = PerceptronPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, dataTest).print()
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1

4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	2	2

备注

1. 该组件的输入为训练数据，输出为感知机模型。
2. 参数数据库的使用方式可以覆盖多个参数的使用方式。

随机森林预测 (RandomForestPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.RandomForestPredictBatchOp

Python 类名：RandomForestPredictBatchOp

功能介绍

随机森林一种经典的有监督学习非线性决策树模型，可以解决分类，回归和其他的一些决策树模型可以解决的问题，通常可以拿到比单决策树更好的效果。

算法原理

通过 Bagging 的方法组合多棵决策树，生成最终的模型。

算法使用

我们给定 Adult 数据集，在这个场景下介绍随机森林的使用步骤

数据集

Adult

训练集

训练数据集的基本统计结果为

Adult train Summary: | colName|count|missing| sum| mean| variance| min| max| |--------------|-----|-------|----------|----------|---------------|-----|-------| | age|32560| 0| 1256214| 38.5815| 186.0665| 17| 90| | workclass|32560| 1836| NaN| NaN| NaN| NaN| NaN| | fnlwgt|32560| 0|6179243539|189780.207|11141029667.4508|12285|1484705| | education|32560| 0| NaN| NaN| NaN| NaN| NaN| | education_num|32560| 0| 328231| 10.0808| 6.6186| 1| 16| |marital_status|32560| 0| NaN| NaN| NaN| NaN| NaN| | occupation|32560| 1843| NaN| NaN| NaN| NaN| NaN| | relationship|32560| 0| NaN| NaN| NaN| NaN| NaN| | race|32560| 0| NaN| NaN| NaN| NaN| NaN| | sex|32560| 0| NaN| NaN| NaN| NaN| NaN| | capital_gain|32560| 0| 35089324| 1077.6819| 54544178.6998| 0| 99999| | capital_loss|32560| 0| 2842700| 87.3065| 162381.6909| 0| 4356| |hours_per_week|32560| 0| 1316644| 40.4375| 152.4637| 1| 99| |native_country|32560| 583| NaN| NaN| NaN| NaN| NaN| | label|32560| 0| NaN| NaN| NaN| NaN| NaN|

读取数据可以使用如下方法进行：

```
CsvSourceBatchOp trainData = new CsvSourceBatchOp()
    .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv")
    .setIgnoreFirstLine(true)
    .setSchemaStr(schemaStr)
    .lazyPrintStatistics("Adult train");
`
```

上述代码中可以使用

```
lazyPrintStatistics("Adult train");
```

即可拿到数据的统计结果

测试集

测试数据集的基本统计结果为

Adult test Summary: | colName|count|missing| sum| mean| variance| min| max| |--------------|-----|-------|----------|----------|----------------|-----|-------| | age|16280| 0| 631146| 38.7682| 191.8033| 17| 90| | workclass|16280| 963| NaN| NaN| NaN| NaN| NaN| | fnlwgt|16280| 0|3083900756|189428.7934|11175556521.7039|13492|1490400| | education|16280| 0| NaN| NaN| NaN| NaN| NaN| | education_num|16280| 0| 163987| 10.0729| 6.5927| 1| 16| |marital_status|16280| 0| NaN| NaN| NaN| NaN| NaN| | occupation|16280| 966| NaN| NaN| NaN| NaN| NaN| | relationship|16280| 0| NaN| NaN| NaN| NaN| NaN| | race|16280| 0| NaN| NaN| NaN| NaN| NaN| | sex|16280| 0| NaN| NaN| NaN| NaN| NaN| | capital_gain|16280| 0| 17614497| 1081.9716| 57519546.0031| 0| 99999| | capital_loss|16280| 0| 1431088| 87.9047| 162503.3785| 0| 3770| |hours_per_week|16280| 0| 657586| 40.3923| 155.7433| 1| 99| |native_country|16280| 274| NaN| NaN| NaN| NaN| NaN| | label|16280| 0| NaN| NaN| NaN| NaN| NaN|

读取数据可以使用如下方法进行：

```
CsvSourceBatchOp testData = new CsvSourceBatchOp()
    .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_test.csv")
    .setIgnoreFirstLine(true)
    .setSchemaStr(schemaStr)
    .lazyPrintStatistics("Adult test");
```

训练

训练模型可以使用 RandomForestTrainBatchOp， 其中支持一些常用的决策树剪枝参数，可以通过调整这些参数来拿到一些更好的模型，详细可以参考参数说明部分。

```
String[] numericalFeatureColNames = new String[] {"age", "fnlwgt",
"education_num", "capital_gain",
    "capital_loss", "hours_per_week"};

String[] categoryFeatureColNames = new String[] {"workclass", "education",
"marital_status", "occupation",
    "relationship", "race", "sex", "native_country"};

RandomForestTrainBatchOp randomForestBatchOp = new RandomForestTrainBatchOp()
    .setFeatureCols(ArrayUtils.addAll(numericalFeatureColNames,
categoryFeatureColNames))
    .setCategoricalCols(categoryFeatureColNames)
    .setSubsamplingRatio(0.6)
```

```
    .setMaxLeaves(32)
    .setLabelCol("label");
```

预测

```
RandomForestPredictBatchOp prediction = new RandomForestPredictBatchOp()
    .setPredictionCol("prediction")
    .setPredictionDetailCol("prediction_detail");
```

评估

```
EvalBinaryClassBatchOp eval = new EvalBinaryClassBatchOp()
    .setLabelCol("prediction")
    .setPredictionDetailCol("prediction_detail");
```

训练预测流程构建

```
prediction
    .linkFrom(
        randomForestBatchOp
            .linkFrom(trainData)
            .lazyPrintModelInfo("Adult random forest model")
            .lazyCollectModelInfo(new Consumer <RandomForestModelInfo>() {
                @Override
                public void accept(RandomForestModelInfo randomForestModelInfo)
{
                    try {
                        randomForestModelInfo
                            .saveTreeAsImage("/tmp/rf_adult_model.png", 0,
true);
                    } catch (IOException e) {
                        throw new IllegalStateException(e);
                    }
                }
            }),
        testData
    )
    .link(eval)
    .lazyPrintMetrics("Adult random forest evaluation");
```

执行

```
BatchOperator.execute();
```

运行结果

模型信息

Adult random forest model Classification trees modelInfo: Number of trees: 10 Number of features: 14 Number of categorical features: 8 Labels: [<=50K, >50K]

Categorical feature info: | feature|number of categorical value| |----------|-------------------------| | workclass| 8| | education| 16| |marital_status| 7| | ...| ...| | race| 5| | sex| 2| |native_country| 41|

Table of feature importance Top 14: | feature|importance| |----------|----------| | age| 0.1997| | fnlwgt| 0.1992| | capital_gain| 0.1447| |hours_per_week| 0.1091| | education_num| 0.0889| | occupation| 0.0553| | relationship| 0.0423| | capital_loss| 0.0336| | workclass| 0.0306| | sex| 0.0299| | race| 0.0188| |marital_status| 0.0176| |native_country| 0.0158| | education| 0.0144|

Classification trees modelInfo: Number of trees: 10 Number of features: 14 Number of categorical features: 8 Labels: [<=50K, >50K]

Categorical feature info: | feature|number of categorical value| |----------|-------------------------| | workclass| 8| | education| 16| |marital_status| 7| | ...| ...| | race| 5| | sex| 2| |native_country| 41|

Table of feature importance Top 14: | feature|importance| |----------|----------| | fnlwgt| 0.2318| | age| 0.2286| |hours_per_week| 0.1382| | education_num| 0.0706| | occupation| 0.0645| | capital_gain| 0.0568| | workclass| 0.0516| | sex| 0.033| | relationship| 0.0299| | capital_loss| 0.0222| | education| 0.0218| |native_country| 0.0199| | race| 0.0175| |marital_status| 0.0136|

模型信息中包含一些常用的训练输入数据的基本信息，特征的基本信息，模型的基本信息。

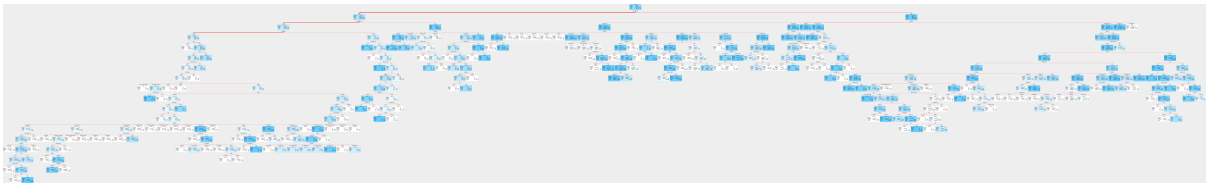
离散特征的一些统计信息，可以通过 Categorical feature info 部分查看。

特征重要性是一类更常用的筛选特征的指标，可以通过 Table of feature importance Top 14 部分查看。

模型可视化

我们也输出了随进森林中第 0 号树的模型结果可视化结果，通过代码中 lazyCollectModelInfo 收集到模型信息之后，通过模型中提供的 saveTreeAsImage ，可以输出模型的图片结果到指定路径。

```
.lazyCollectModelInfo(new Consumer <RandomForestModelInfo>() {
    @Override
    public void accept(RandomForestModelInfo randomForestModelInfo) {
        try {
            randomForestModelInfo
                .saveTreeAsImage("/tmp/rf_adult_model.png", 0, true);
        } catch (IOException e) {
            throw new IllegalStateException(e);
        }
    }
})
```



评估结果

Adult random forest evaluation ---------- Metrics: ---------- Auc:1 Accuracy:0.9995 Precision:0.9965 Recall:1 F1:0.9982 LogLoss:0.2584 |Pred\Real|>50K|<=50K| |---------|----|-----| | >50K|2273| 8| | <=50K| 0|13999|

评估结果中包含一些常用

文献或出处

1. RandomForest
2. weka

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
```

```
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = RandomForestTrainBatchOp()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = RandomForestPredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = RandomForestPredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.RandomForestPredictBatchOp;
import
com.alibaba.alink.operator.batch.classification.RandomForestTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.classification.RandomForestPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RandomForestPredictBatchOpTest {
    @Test
    public void testRandomForestPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
```

```
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new RandomForestTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new RandomForestPredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new
RandomForestPredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

随机森林训练 (RandomForestTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.RandomForestTrainBatchOp

Python 类名：RandomForestTrainBatchOp

功能介绍

随机森林一种经典的有监督学习非线性决策树模型，可以解决分类，回归和其他的一些决策树模型可以解决的问题，通常可以拿到比单决策树更好的效果。

算法原理

通过 Bagging 的方法组合多棵决策树，生成最终的模型。

算法使用

我们给定 Adult 数据集，在这个场景下介绍随机森林的使用步骤

数据集

Adult

训练集

训练数据集的基本统计结果为

Adult train Summary: | colName|count|missing| sum| mean| variance| min| max| |--------------|-----|-------|----------|----------|----------------|-----|-------| | age|32560| 0| 1256214| 38.5815| 186.0665| 17| 90| | workclass|32560| 1836| NaN| NaN| NaN| NaN| NaN| | fnlwgt|32560| 0|6179243539|189780.207|11141029667.4508|12285|1484705| | education|32560| 0| NaN| NaN| NaN| NaN| NaN| | education_num|32560| 0| 328231| 10.0808| 6.6186| 1| 16| |marital_status|32560| 0| NaN| NaN| NaN| NaN| NaN| | occupation|32560| 1843| NaN| NaN| NaN| NaN| NaN| | relationship|32560| 0| NaN| NaN| NaN| NaN| NaN| | race|32560| 0| NaN| NaN| NaN| NaN| NaN| | sex|32560| 0| NaN| NaN| NaN| NaN| NaN| | capital_gain|32560| 0| 35089324| 1077.6819| 54544178.6998| 0| 99999| | capital_loss|32560| 0| 2842700| 87.3065| 162381.6909| 0| 4356| |hours_per_week|32560| 0| 1316644| 40.4375| 152.4637| 1| 99| |native_country|32560| 583| NaN| NaN| NaN| NaN| NaN| | label|32560| 0| NaN| NaN| NaN| NaN| NaN|

读取数据可以使用如下方法进行：

```
CsvSourceBatchOp trainData = new CsvSourceBatchOp()
    .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv")
    .setIgnoreFirstLine(true)
    .setSchemaStr(schemaStr)
    .lazyPrintStatistics("Adult train");
`
```

上述代码中可以使用

```
lazyPrintStatistics("Adult train");
```

即可拿到数据的统计结果

测试集

测试数据集的基本统计结果为

Adult test Summary: | colName|count|missing| sum| mean| variance| min| max| |--------------|-----|-------|----------|----------|----------------|-----|-------| | age|16280| 0| 631146| 38.7682| 191.8033| 17| 90| | workclass|16280| 963| NaN| NaN| NaN| NaN| NaN| | fnlwgt|16280| 0|3083900756|189428.7934|11175556521.7039|13492|1490400| | education|16280| 0| NaN| NaN| NaN| NaN| NaN| | education_num|16280| 0| 163987| 10.0729| 6.5927| 1| 16| |marital_status|16280| 0| NaN| NaN| NaN| NaN| NaN| | occupation|16280| 966| NaN| NaN| NaN| NaN| NaN| | relationship|16280| 0| NaN| NaN| NaN| NaN| NaN| | race|16280| 0| NaN| NaN| NaN| NaN| NaN| | sex|16280| 0| NaN| NaN| NaN| NaN| NaN| | capital_gain|16280| 0| 17614497| 1081.9716| 57519546.0031| 0| 99999| | capital_loss|16280| 0| 1431088| 87.9047| 162503.3785| 0| 3770| |hours_per_week|16280| 0| 657586| 40.3923| 155.7433| 1| 99| |native_country|16280| 274| NaN| NaN| NaN| NaN| NaN| | label|16280| 0| NaN| NaN| NaN| NaN| NaN|

读取数据可以使用如下方法进行：

```
CsvSourceBatchOp testData = new CsvSourceBatchOp()
    .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_test.csv")
    .setIgnoreFirstLine(true)
    .setSchemaStr(schemaStr)
    .lazyPrintStatistics("Adult test");
```

训练

训练模型可以使用 RandomForestTrainBatchOp ， 其中支持一些常用的决策树剪枝参数，可以通过调整这些参数来拿到一些更好的模型，详细可以参考参数说明部分。

```
String[] numericalFeatureColNames = new String[] {"age", "fnlwgt",
"education_num", "capital_gain",
    "capital_loss", "hours_per_week"};

String[] categoryFeatureColNames = new String[] {"workclass", "education",
"marital_status", "occupation",
    "relationship", "race", "sex", "native_country"};

RandomForestTrainBatchOp randomForestBatchOp = new RandomForestTrainBatchOp()
    .setFeatureCols(ArrayUtils.addAll(numericalFeatureColNames,
categoryFeatureColNames))
    .setCategoricalCols(categoryFeatureColNames)
    .setSubsamplingRatio(0.6)
```

```
    .setMaxLeaves(32)
    .setLabelCol("label");
```

预测

```
RandomForestPredictBatchOp prediction = new RandomForestPredictBatchOp()
    .setPredictionCol("prediction")
    .setPredictionDetailCol("prediction_detail");
```

评估

```
EvalBinaryClassBatchOp eval = new EvalBinaryClassBatchOp()
    .setLabelCol("prediction")
    .setPredictionDetailCol("prediction_detail");
```

训练预测流程构建

```
prediction
    .linkFrom(
        randomForestBatchOp
            .linkFrom(trainData)
            .lazyPrintModelInfo("Adult random forest model")
            .lazyCollectModelInfo(new Consumer <RandomForestModelInfo>() {
                @Override
                public void accept(RandomForestModelInfo randomForestModelInfo)
{
                    try {
                        randomForestModelInfo
                            .saveTreeAsImage("/tmp/rf_adult_model.png", 0,
true);
                    } catch (IOException e) {
                        throw new IllegalStateException(e);
                    }
                }
            }),
        testData
    )
    .link(eval)
    .lazyPrintMetrics("Adult random forest evaluation");
```

执行

```
BatchOperator.execute();
```

运行结果

模型信息

Adult random forest model Classification trees modelInfo: Number of trees: 10 Number of features: 14 Number of categorical features: 8 Labels: [<=50K, >50K]

Categorical feature info: | feature|number of categorical value| |----------|-------------------------| | workclass| 8| | education| 16| |marital_status| 7| | ...| ...| | race| 5| | sex| 2| |native_country| 41|

Table of feature importance Top 14: | feature|importance| |----------|----------| | age| 0.1997| | fnlwgt| 0.1992| | capital_gain| 0.1447| |hours_per_week| 0.1091| | education_num| 0.0889| | occupation| 0.0553| | relationship| 0.0423| | capital_loss| 0.0336| | workclass| 0.0306| | sex| 0.0299| | race| 0.0188| |marital_status| 0.0176| |native_country| 0.0158| | education| 0.0144|

Classification trees modelInfo: Number of trees: 10 Number of features: 14 Number of categorical features: 8 Labels: [<=50K, >50K]

Categorical feature info: | feature|number of categorical value| |----------|-------------------------| | workclass| 8| | education| 16| |marital_status| 7| | ...| ...| | race| 5| | sex| 2| |native_country| 41|

Table of feature importance Top 14: | feature|importance| |----------|----------| | fnlwgt| 0.2318| | age| 0.2286| |hours_per_week| 0.1382| | education_num| 0.0706| | occupation| 0.0645| | capital_gain| 0.0568| | workclass| 0.0516| | sex| 0.033| | relationship| 0.0299| | capital_loss| 0.0222| | education| 0.0218| |native_country| 0.0199| | race| 0.0175| |marital_status| 0.0136|

模型信息中包含一些常用的训练输入数据的基本信息，特征的基本信息，模型的基本信息。

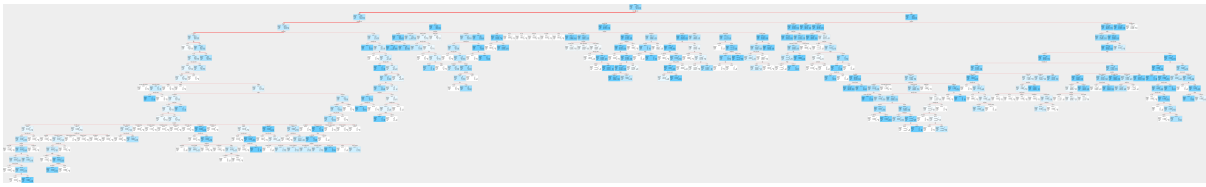
离散特征的一些统计信息，可以通过 Categorical feature info 部分查看。

特征重要性是一类更常用的筛选特征的指标，可以通过 Table of feature importance Top 14 部分查看。

模型可视化

我们也输出了随进森林中第 0 号树的模型结果可视化结果，通过代码中 lazyCollectModelInfo 收集到模型信息之后，通过模型中提供的 saveTreeAsImage ，可以输出模型的图片结果到指定路径。

```
.lazyCollectModelInfo(new Consumer <RandomForestModelInfo>() {
    @Override
    public void accept(RandomForestModelInfo randomForestModelInfo) {
        try {
            randomForestModelInfo
                .saveTreeAsImage("/tmp/rf_adult_model.png", 0, true);
        } catch (IOException e) {
            throw new IllegalStateException(e);
        }
    }
})
```



评估结果

Adult random forest evaluation ---------- Metrics: ---------- Auc:1 Accuracy:0.9995 Precision:0.9965 Recall:1 F1:0.9982 LogLoss:0.2584 |Pred\Real|>50K|<=50K| |---------|----|-----| | >50K|2273| 8| | <=50K| 0|13999|

评估结果中包含一些常用

文献或出处

1. RandomForest
2. weka

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
labelCol	标签列名	输入表中的标签列名	String	✓		

categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String			"series"
featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double			0.2
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			128
maxDepth	树的深度限制	树的深度限制	Integer			2147483647
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2147483647

maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer			64
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.0
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.0
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			2
numSubsetFeatures	每棵树的特征采样数目	每棵树的特征采样数目	Integer			2147483647
numTrees	模型中树的棵数	模型中树的棵数	Integer		[1, +inf)	10
numTreesOfGini	模型中Cart树的棵数	模型中Cart树的棵数	Integer			null

numTreesOfInfoGain	模型中Id3树的棵数	模型中Id3树的棵数	Integer			null
numTreesOfInfoGainRatio	模型中C4.5树的棵数	模型中C4.5树的棵数	Integer			null
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double			100000.0
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
```

```
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

trainOp = RandomForestTrainBatchOp()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = RandomForestPredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')
predictStreamOp = RandomForestPredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.classification.RandomForestPredictBatchOp;
import
com.alibaba.alink.operator.batch.classification.RandomForestTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.classification.RandomForestPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RandomForestTrainBatchOpTest {
    @Test
    public void testRandomForestTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
```

```
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new RandomForestTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new RandomForestPredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new
RandomForestPredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

Softmax预测 (SoftmaxPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.SoftmaxPredictBatchOp

Python 类名：SoftmaxPredictBatchOp

功能介绍

Softmax算法是Logistic回归算法的推广，Logistic回归主要是用来处理二分类问题，而Softmax回归则是处理多分类问题。

算法原理

面对多分类问题，建立代价函数，然后通过优化方法迭代求解出最优的模型参数。具体原理可参考文献。

算法使用

该算法经常应用到多分类问题中，类似情感分析、手写字识别等问题都可以使用Softmax算法，该算法支持稀疏和稠密两种输入样本。

文献或出处

[1] Brown, Peter F., et al. "Class-based n-gram models of natural language." Computational linguistics 18.4 (1992): 467-480.

[2] Goodman, Joshua. "Classes for fast maximum entropy training." 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221). Vol. 1. IEEE, 2001.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			

reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df_data = pd.DataFrame([
       [2, 1, 1],
       [3, 2, 1],
       [4, 3, 2],
       [2, 4, 1],
       [2, 2, 1],
       [4, 3, 2],
       [1, 2, 1],
       [5, 3, 3]
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int,
label int')
dataTest = batchData

colnames = ["f0","f1"]
lr = SoftmaxTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = batchData.link(lr)

predictor = SoftmaxPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, dataTest).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
```

```
import com.alibaba.alink.operator.batch.classification.SoftmaxPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.SoftmaxTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SoftmaxPredictBatchOpTest {
    @Test
    public void testSoftmaxPredictBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 3)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data, "f0 int, f1
int, label int");
        BatchOperator dataTest = batchData;
        BatchOperator <?> lr = new SoftmaxTrainBatchOp().setFeatureCols("f0",
"f1").setLabelCol("label");
        BatchOperator model = batchData.link(lr);
        BatchOperator <?> predictor = new
SoftmaxPredictBatchOp().setPredictionCol("pred");
        predictor.linkFrom(model, dataTest).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2

1	2	1	1
5	3	3	3

Softmax训练 (SoftmaxTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.SoftmaxTrainBatchOp

Python 类名：SoftmaxTrainBatchOp

功能介绍

Softmax算法是Logistic回归算法的推广，Logistic回归主要是用来处理二分类问题，而Softmax回归则是处理多分类问题。

算法原理

面对多分类问题，建立代价函数，然后通过优化方法迭代求解出最优的模型参数。具体原理可参考文献。

算法使用

该算法经常应用到多分类问题中，类似情感分析、手写字识别等问题都可以使用Softmax算法，该算法支持稀疏和稠密两种输入样本。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Brown, Peter F., et al. "Class-based n-gram models of natural language." Computational linguistics 18.4 (1992): 467-480.

[2] Goodman, Joshua. "Classes for fast maximum entropy training." 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221). Vol. 1. IEEE, 2001.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1.0E-6

featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"	null
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean			true
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
       [2, 1, 1],
       [3, 2, 1],
       [4, 3, 2],
       [2, 4, 1],
       [2, 2, 1],
       [4, 3, 2],
       [1, 2, 1],
       [5, 3, 3]
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int,
label int')
dataTest = batchData

colnames = ["f0","f1"]
lr = SoftmaxTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = batchData.link(lr)

predictor = SoftmaxPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, dataTest).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.SoftmaxPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.SoftmaxTrainBatchOp;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SoftmaxTrainBatchOpTest {
    @Test
    public void testSoftmaxTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 3)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data, "f0 int, f1
int, label int");
        BatchOperator dataTest = batchData;
        BatchOperator <?> lr = new SoftmaxTrainBatchOp().setFeatureCols("f0",
"f1").setLabelCol("label");
        BatchOperator model = batchData.link(lr);
        BatchOperator <?> predictor = new
SoftmaxPredictBatchOp().setPredictionCol("pred");
        predictor.linkFrom(model, dataTest).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	3	3

备注

1. 该组件的输入为训练数据，输出为Softmax模型。
2. 参数数据库的使用方式可以覆盖多个参数的使用方式。

XGBoost二分类预测 (XGBoostPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.XGBoostPredictBatchOp

Python 类名：XGBoostPredictBatchOp

功能介绍

XGBoost 组件是在开源社区的基础上进行包装，使功能和 PAI 更兼容，更易用。 XGBoost 算法在 Boosting 算法的基础上进行了扩展和升级，具有较好的易用性和鲁棒性，被广泛用在各种机器学习生产系统和竞赛领域。 当前支持分类，回归和排序。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
pluginVersion	插件版本号	插件版本号	String			"1.5.1"
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    [0, 1, 1.1, 1.0],
    [1, -2, 0.9, 2.0],
    [0, 100, -0.01, 3.0],
    [1, -99, 0.1, 4.0],
    [0, 1, 1.1, 5.0],
    [1, -2, 0.9, 6.0]
])
```

```
batchSource = BatchOperator.fromDataframe(
    df, schemaStr='y int, x1 int, x2 double, x3 double'
)

streamSource = StreamOperator.fromDataframe(
    df, schemaStr='y int, x1 int, x2 double, x3 double'
)

trainOp = XGBoostTrainBatchOp()\
    .setNumRound(1)\
    .setPluginVersion('1.5.1')\
    .setLabelCol('y')\
    .linkFrom(batchSource)

predictBatchOp = XGBoostPredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setPluginVersion('1.5.1')

predictStreamOp = XGBoostPredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setPluginVersion('1.5.1')

predictBatchOp.linkFrom(trainOp, batchSource).print()

predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.XGBoostPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.XGBoostTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.classification.XGBoostPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class XGBoostTrainBatchOpTest {

    @Test
    public void testXGBoostTrainBatchOp() throws Exception {
        List <Row> data = Arrays.asList(
            Row.of(0, 1, 1.1, 1.0),
            Row.of(1, -2, 0.9, 2.0),
            Row.of(0, 100, -0.01, 3.0),
            Row.of(1, -99, 0.1, 4.0),
            Row.of(0, 1, 1.1, 5.0),
            Row.of(1, -2, 0.9, 6.0)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(data, "y int, x1
int, x2 double, x3 double");
        StreamOperator <?> streamSource = new MemSourceStreamOp(data, "y int,
x1 int, x2 double, x3 double");
        BatchOperator <?> trainOp = new XGBoostTrainBatchOp()
            .setNumRound(1)
            .setPluginVersion("1.5.1")
            .setLabelCol("y")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new XGBoostPredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setPluginVersion("1.5.1");
        StreamOperator <?> predictStreamOp = new
XGBoostPredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setPluginVersion("1.5.1");

        predictBatchOp.linkFrom(trainOp, batchSource).print();

        predictStreamOp.linkFrom(streamSource).print();

        StreamOperator.execute();
    }
}
```

运行结果

XGBoost二分类训练 (XGBoostTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.classification.XGBoostTrainBatchOp

Python 类名：XGBoostTrainBatchOp

功能介绍

XGBoost 组件是在开源社区的基础上进行包装，使功能和 PAI 更兼容，更易用。 XGBoost 算法在 Boosting 算法的基础上进行了扩展和升级，具有较好的易用性和鲁棒性，被广泛用在各种机器学习生产系统和竞赛领域。 当前支持分类，回归和排序。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范
labelCol	标签列名	输入表中的标签列名	String	✓	
numRound	树的棵树	树的棵树	Integer	✓	
alpha	L1 正则项	L1 正则项	Double		
baseScore	Base score	Base score	Double		
colSampleByLevel	每个树列采样	每个树列采样	Double		
colSampleByNode	每个结点列采样	每个结点采样	Double		
colSampleByTree	每个树列采样	每个树列采样	Double		
eta	学习率	学习率	Double		
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为 [BIGDECIMA BIGINTEGER DOUBLE, FL INTEGER, LO SHORT]
gamma	结点分裂最小损失变化	节点分裂最小损失变化	Double		
growPolicy	GrowPolicy	GrowPolicy	String		"DEPTH_WIS "LOSS_GUID

interactionConstraints	interaction constraints	interaction constraints	String		
lambda	L2 正则项	L2 正则项	Double		
maxBin	最大结点个数	最大结点个数	Integer		
maxDeltaStep	Delta step	Delta step	Double		
maxDepth	最大深度	最大深度	Integer		
maxLeaves	最大结点个数	最大结点个数	Integer		
minChildWeight	结点的最小权重	结点的最小权重	Double		
monotoneConstraints	monotone constraints	monotone constraints	String		
numClass	标签类别个数	标签类别个数，多分类时有效	Integer		
objective	objective	objective	String		"BINARY_LO "BINARY_LO "BINARY_HI "MULTI_SOF "MULTI_SOF
pluginVersion	插件版本号	插件版本号	String		
processType	ProcessType	ProcessType	String		"DEFAULT", "
refreshLeaf	RefreshLeaf	RefreshLeaf	Integer		
runningMode	运行模式	XGBoost的运行模型，ICQ速度快，但使用内存多，TRIAVIAL速度略慢，但是节省内存，按照流式方式处理。由于训练数据本身在XGBoost运行时已经被缓存进内存，所以存两份和存一份数据的资源消耗和速度对比，还需要进一步的测试。	String		"ICQ", "TRIVI
samplingMethod	采样方法	采样方法	String		"UNIFORM", "GRADIENT_

scalePosWeight	ScalePosWeight	ScalePosWeight	Double		
singlePrecisionHistogram	single precision histogram	single precision histogram	Boolean		
sketchEps	SketchEps	SketchEps	Double		
subSample	样本采样比例	样本采样比例	Double		
treeMethod	构建树的方法	构建树的方法	String		"AUTO", "EX "APPROX", "I
updater	Updater	Updater	String		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为 [DENSE_VEC SPARSE_VE STRING, VEC

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    [0, 1, 1.1, 1.0],
    [1, -2, 0.9, 2.0],
    [0, 100, -0.01, 3.0],
    [1, -99, 0.1, 4.0],
    [0, 1, 1.1, 5.0],
    [1, -2, 0.9, 6.0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='y int, x1 int, x2 double, x3 double'
)

streamSource = StreamOperator.fromDataframe(
    df, schemaStr='y int, x1 int, x2 double, x3 double'
)

trainOp = XGBoostTrainBatchOp()\
    .setNumRound(1)\
    .setPluginVersion('1.5.1')\
    .setLabelCol('y')\
    .linkFrom(batchSource)
```

```
predictBatchOp = XGBoostPredictBatchOp()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setPluginVersion('1.5.1')

predictStreamOp = XGBoostPredictStreamOp(trainOp)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setPluginVersion('1.5.1')

predictBatchOp.linkFrom(trainOp, batchSource).print()

predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.classification.XGBoostPredictBatchOp;
import com.alibaba.alink.operator.batch.classification.XGBoostTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.classification.XGBoostPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class XGBoostTrainBatchOpTest {

    @Test
    public void testXGBoostTrainBatchOp() throws Exception {
        List <Row> data = Arrays.asList(
            Row.of(0, 1, 1.1, 1.0),
            Row.of(1, -2, 0.9, 2.0),
            Row.of(0, 100, -0.01, 3.0),
            Row.of(1, -99, 0.1, 4.0),
            Row.of(0, 1, 1.1, 5.0),
            Row.of(1, -2, 0.9, 6.0)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(data, "y int, x1
int, x2 double, x3 double");
```

```
        StreamOperator <?> streamSource = new MemSourceStreamOp(data, "y int,
x1 int, x2 double, x3 double");
        BatchOperator <?> trainOp = new XGBoostTrainBatchOp()
            .setNumRound(1)
            .setPluginVersion("1.5.1")
            .setLabelCol("y")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new XGBoostPredictBatchOp()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setPluginVersion("1.5.1");
        StreamOperator <?> predictStreamOp = new
XGBoostPredictStreamOp(trainOp)
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setPluginVersion("1.5.1");

        predictBatchOp.linkFrom(trainOp, batchSource).print();

        predictStreamOp.linkFrom(streamSource).print();

        StreamOperator.execute();
    }
}
```

运行结果

生存回归预测 (AftSurvivalRegPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.AftSurvivalRegPredictBatchOp

Python 类名：AftSurvivalRegPredictBatchOp

功能介绍

在生存分析领域，加速失效时间模型(accelerated failure time model,AFT 模型)可以作为比例风险模型的替代模型。生存回归组件支持稀疏、稠密两种数据格式。

算法原理

AFT模型将线性回归模型的建模方法引人到生存分析的领域， 将生存时间的对数作为反应变量，研究多协变量与对数生存时间之间的回归关系，在形式上，模型与一般的线性回归模型相似。对回归系数的解释也与一般的线性回归模型相似，较之Cox模型， AFT模型对分析结果的解释更加简单、直观且易于理解，并且可以预测个体的生存时间。

算法使用

生存回归分析是研究特定事件的发生与时间的关系的回归。这里特定事件可以是：病人死亡、病人康复、用户流失、商品下架等。

文献或出处

[1] Wei, Lee-Jen. "The accelerated failure time model: a useful alternative to the Cox regression model in survival analysis." Statistics in medicine 11.14　15 (1992): 1871-1879.

[2] https://spark.apache.org/docs/latest/ml-classification-regression.html#survival-regression

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默
predictionCol	预测结果列名	预测结果列名	String	✓		

modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
quantileProbabilities	分位数概率数组	分位数概率数组	double[]			[0.01,0.05,0.1,0.25,0
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.218, 1.0, "1.560,-0.605"],
    [2.949, 0.0, "0.346,2.158"],
    [3.627, 0.0, "1.380,0.231"],
    [0.273, 1.0, "0.520,1.151"],
    [4.199, 0.0, "0.795,-0.226"]
])

data = BatchOperator.fromDataframe(df, schemaStr="label double, censor double,
features string")

trainOp = AftSurvivalRegTrainBatchOp()\
            .setVectorCol("features")\
            .setLabelCol("label")\
            .setCensorCol("censor")

model = trainOp.linkFrom(data)

predictOp = AftSurvivalRegPredictBatchOp()\
            .setPredictionCol("pred")

predictOp.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.regression.AftSurvivalRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.AftSurvivalRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AftSurvivalRegPredictBatchOpTest {
    @Test
    public void testAftSurvivalRegPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.218, 1.0, "1.560,-0.605"),
            Row.of(2.949, 0.0, "0.346,2.158"),
            Row.of(3.627, 0.0, "1.380,0.231"),
            Row.of(0.273, 1.0, "0.520,1.151"),
            Row.of(4.199, 0.0, "0.795,-0.226")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "label double, censor
double, features string");
        BatchOperator <?> trainOp = new AftSurvivalRegTrainBatchOp()
            .setVectorCol("features")
            .setLabelCol("label")
            .setCensorCol("censor");
        BatchOperator model = trainOp.linkFrom(data);
        BatchOperator <?> predictOp = new AftSurvivalRegPredictBatchOp()
            .setPredictionCol("pred");
        predictOp.linkFrom(model, data).print();
    }
}
```

运行结果

模型结果

model_id	model_info
0	{"hasInterceptItem":"true","vectorCol":"\"features\"","modelName":"\"AFTSurvivalRegTrainBatch(
1048576	{"featureColNames":null,"featureColTypes":null,"coefVector":{"data": [2.6373721387804276,-0.49591581739360013,0.19847648151323818,1.5469720551612485]},

预测结果

label	censor	features	pred
0.273	1.0	0.520,1.151	13.571097451777327
1.218	1.0	1.560,-0.605	5.718263596902868
3.627	0.0	1.380,0.231	7.380610641992667
4.199	0.0	0.795,-0.226	9.009354073821902
2.949	0.0	0.346,2.158	18.067188679653064

生存回归训练 (AftSurvivalRegTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.AftSurvivalRegTrainBatchOp

Python 类名：AftSurvivalRegTrainBatchOp

功能介绍

在生存分析领域，加速失效时间模型(accelerated failure time model,AFT 模型)可以作为比例风险模型的替代模型。生存回归组件支持稀疏、稠密两种数据格式。

算法原理

AFT模型将线性回归模型的建模方法引人到生存分析的领域， 将生存时间的对数作为反应变量，研究多协变量与对数生存时间之间的回归关系，在形式上，模型与一般的线性回归模型相似。对回归系数的解释也与一般的线性回归模型相似，较之Cox模型， AFT模型对分析结果的解释更加简单、直观且易于理解，并且可以预测个体的生存时间。

算法使用

生存回归分析是研究特定事件的发生与时间的关系的回归。这里特定事件可以是：病人死亡、病人康复、用户流失、商品下架等。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Wei, Lee-Jen. "The accelerated failure time model: a useful alternative to the Cox regression model in survival analysis." Statistics in medicine 11.14　15 (1992): 1871-1879.

[2] https://spark.apache.org/docs/latest/ml-classification-regression.html#survival-regression

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
censorCol	生存列名	生存列名	String	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	

labelCol	标签列名	输入表中的标签列名	String	✓		
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为1.0e-6	Double		[0.0, +inf)	1.0E-6
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.218, 1.0, "1.560,-0.605"],
    [2.949, 0.0, "0.346,2.158"],
    [3.627, 0.0, "1.380,0.231"],
    [0.273, 1.0, "0.520,1.151"],
    [4.199, 0.0, "0.795,-0.226"]
])

data = BatchOperator.fromDataframe(df, schemaStr="label double, censor double,
features string")

trainOp = AftSurvivalRegTrainBatchOp()\
            .setVectorCol("features")\
            .setLabelCol("label")\
            .setCensorCol("censor")

model = trainOp.linkFrom(data)

predictOp = AftSurvivalRegPredictBatchOp()\
            .setPredictionCol("pred")

predictOp.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.regression.AftSurvivalRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.AftSurvivalRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class AftSurvivalRegTrainBatchOpTest {
    @Test
    public void testAftSurvivalRegTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.218, 1.0, "1.560,-0.605"),
            Row.of(2.949, 0.0, "0.346,2.158"),
            Row.of(3.627, 0.0, "1.380,0.231"),
            Row.of(0.273, 1.0, "0.520,1.151"),
            Row.of(4.199, 0.0, "0.795,-0.226")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "label double, censor
double, features string");
        BatchOperator <?> trainOp = new AftSurvivalRegTrainBatchOp()
            .setVectorCol("features")
            .setLabelCol("label")
            .setCensorCol("censor");
        BatchOperator model = trainOp.linkFrom(data);
        BatchOperator <?> predictOp = new AftSurvivalRegPredictBatchOp()
            .setPredictionCol("pred");
        predictOp.linkFrom(model, data).print();
    }
}
```

运行结果

模型结果

model_id	model_info
0	{"hasInterceptItem":"true","vectorCol":"\"features\"","modelName":"\"AFTSurvivalRegTrainBatch(
1048576	{"featureColNames":null,"featureColTypes":null,"coefVector":{"data": [2.6373721387804276,-0.49591581739360013,0.19847648151323818,1.5469720551612485]},

预测结果

label	censor	features	pred
0.273	1.0	0.520,1.151	13.571097451777327
1.218	1.0	1.560,-0.605	5.718263596902868
3.627	0.0	1.380,0.231	7.380610641992667
4.199	0.0	0.795,-0.226	9.009354073821902
2.949	0.0	0.346,2.158	18.067188679653064

Bert文本对回归预测 (BertTextPairRegressorPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.BertTextPairRegressorPredictBatchOp

Python 类名：BertTextPairRegressorPredictBatchOp

功能介绍

与 BERT 文本对回归训练组件对应的预测组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
inferBatchSize	推理数据批大小	推理数据批大小	Integer			256
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv"
schemaStr = "f_quality double, f_id_1 string, f_id_2 string, f_string_1 string,
f_string_2 string"
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schemaStr) \
    .setFieldDelimiter("\t") \
    .setIgnoreFirstLine(True) \
    .setQuoteChar(None)
data = data.firstN(300)
model = CsvSourceBatchOp() \
    .setFilePath("http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/bert_text_pair_regressor_model.csv") \
    .setSchemaStr("model_id bigint, model_info string, label_value double")
```

```
predict = BertTextPairRegressorPredictBatchOp() \
    .setPredictionCol("pred") \
    .setReservedCols(["f_quality"]) \
    .linkFrom(model, data)
predict.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.regression.BertTextPairRegressorPredictBatchOp
;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class BertTextPairRegressorPredictBatchOpTest {

    @Test
    public void test() throws Exception {
        String url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv";
        String schemaStr = "f_quality double, f_id_1 string, f_id_2 string,
f_string_1 string, f_string_2 string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schemaStr)
            .setFieldDelimiter("\t")
            .setIgnoreFirstLine(true)
            .setQuoteChar(null);
        data = data.firstN(300);
        BatchOperator <?> model = new CsvSourceBatchOp()
            .setFilePath("http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/bert_text_pair_regressor_model.csv")
            .setSchemaStr("model_id bigint, model_info string, label_value
double");
        BertTextPairRegressorPredictBatchOp predict = new
BertTextPairRegressorPredictBatchOp()
            .setPredictionCol("pred")
            .setReservedCols("f_quality")
            .linkFrom(model, data);
        predict.print();
    }
}
```

运行结果

f_quality	pred
0.0	1.404307
0.0	1.404307
1.0	1.404307
0.0	1.404307
1.0	1.404307
...	...
0.0	1.404392
1.0	1.404392
0.0	1.404392
1.0	1.404392
1.0	1.404392

Bert文本对回归训练 (BertTextPairRegressorTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.BertTextPairRegressorTrainBatchOp

Python 类名：BertTextPairRegressorTrainBatchOp

功能介绍

在预训练的 BERT 模型的基础上增加一个全连接层，用于进行文本对回归任务。

参数说明

名称	中文名称	描述
labelCol	标签列名	输入表中的标签列名
textCol	文本列	文本列
textPairCol	文本对列	文本对列
batchSize	数据批大小	数据批大小
bertModelName	BERT模型名字	BERT模型名字： Base-Chinese,Base-Multilingual-Case
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estima` 为所有 worker 都能访问到的目录
customConfigJson	自定义参数	对应 https://github.com/alibaba/EasyTransfer/blob/master/eas 中的config_json
intraOpParallelism	Op 间并发度	Op 间并发度
learningRate	学习率	学习率

maxSeqLength	句子截断长度	句子截断长度
numEpochs	epoch 数	epoch 数
numFineTunedLayers	微调层数	微调层数
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设（需要取整），否则为总并发度减去 Worker 角色数。
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设（需要取整），否则为总并发度减去 PS 角色数。
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文且目录名与压缩文件主文件名一致，可以使用 http://, http是目录，那么只能使用本地路径，即 file://。
removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv"
schemaStr = "f_quality double, f_id_1 string, f_id_2 string, f_string_1 string,
f_string_2 string"
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schemaStr) \
    .setFieldDelimiter("\t") \
    .setIgnoreFirstLine(True) \
    .setQuoteChar(None)
data = ShuffleBatchOp().linkFrom(data)

train = BertTextPairRegressorTrainBatchOp() \

.setTextCol("f_string_1").setTextPairCol("f_string_2").setLabelCol("f_quality")
\
    .setNumEpochs(0.1) \
    .setMaxSeqLength(32) \
    .setNumFineTunedLayers(1) \
    .setBertModelName("Base-Uncased") \
```

```
    .linkFrom(data)

AkSinkBatchOp() \
    .setFilePath("/tmp/bert_text_pair_regressor_model.ak") \
    .setOverwriteSink(True) \
    .linkFrom(train)
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ShuffleBatchOp;
import
com.alibaba.alink.operator.batch.regression.BertTextPairRegressorTrainBatchOp;
import com.alibaba.alink.operator.batch.sink.AkSinkBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class BertTextPairRegressorTrainBatchOpTest {

    @Test
    public void testBertTextPairRegressorTrainBatchOp() throws Exception {
        String url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv";
        String schemaStr = "f_quality double, f_id_1 string, f_id_2 string,
f_string_1 string, f_string_2 string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schemaStr)
            .setFieldDelimiter("\t")
            .setIgnoreFirstLine(true)
            .setQuoteChar(null);
        data = new ShuffleBatchOp().linkFrom(data);

        BertTextPairRegressorTrainBatchOp train = new
BertTextPairRegressorTrainBatchOp()

.setTextCol("f_string_1").setTextPairCol("f_string_2").setLabelCol("f_quality")
            .setNumEpochs(0.1)
            .setMaxSeqLength(32)
            .setNumFineTunedLayers(1)
            .setBertModelName("Base-Uncased")
            .linkFrom(data);

        new AkSinkBatchOp()
            .setFilePath("/tmp/bert_text_pair_regressor_model.ak")
            .setOverwriteSink(true)
```

```
            .linkFrom(train);
        BatchOperator.execute();
    }
}
```

Bert文本回归预测 (BertTextRegressorPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.BertTextRegressorPredictBatchOp

Python 类名：BertTextRegressorPredictBatchOp

功能介绍

与 BERT 文本回归训练组件对应的预测组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
inferBatchSize	推理数据批大小	推理数据批大小	Integer			256
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv"
schema = "label double, review string"
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schema) \
    .setIgnoreFirstLine(True)
data = data.where("review is not null")
data = data.firstN(300)
model = CsvSourceBatchOp() \
    .setFilePath("http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/bert_text_regressor_model.csv") \
    .setSchemaStr("model_id bigint, model_info string, label_value double")
predict = BertTextRegressorPredictBatchOp() \
    .setPredictionCol("pred") \
```

```
    .setReservedCols(["label"]) \
    .linkFrom(model, data)
predict.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.regression.BertTextRegressorPredictBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class BertTextRegressorPredictBatchOpTest {

    @Test
    public void testBertTextRegressorPredictBatchOp() throws Exception {
        String url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv";
        String schema = "label double, review string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schema)
            .setIgnoreFirstLine(true);
        data = data.where("review is not null");
        data = data.firstN(300);
        BatchOperator <?> model = new CsvSourceBatchOp()
            .setFilePath("http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/bert_text_regressor_model.csv")
            .setSchemaStr("model_id bigint, model_info string, label_value
double");
        BertTextRegressorPredictBatchOp predict = new
BertTextRegressorPredictBatchOp()
            .setPredictionCol("pred")
            .setReservedCols("label")
            .linkFrom(model, data);
        predict.print();
    }
}
```

运行结果

label	pred
1.0	5.004022
1.0	5.004022

1.0	5.004022
1.0	5.004022
1.0	5.004022
...	...
0.0	5.004022
0.0	5.004022
0.0	5.004022
0.0	5.004022
0.0	5.004022

Bert文本回归训练 (BertTextRegressorTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.BertTextRegressorTrainBatchOp

Python 类名：BertTextRegressorTrainBatchOp

功能介绍

在预训练的 BERT 模型的基础上增加一个全连接层，用于进行文本回归任务。

参数说明

名称	中文名称	描述
labelCol	标签列名	输入表中的标签列名
textCol	文本列	文本列
batchSize	数据批大小	数据批大小
bertModelName	BERT模型名字	BERT模型名字： Base-Chinese,Base-Multilingual-Case
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estima` 为所有 worker 都能访问到的目录
customConfigJson	自定义参数	对应 https://github.com/alibaba/EasyTransfer/blob/master/eas 中的config_json
intraOpParallelism	Op 间并发度	Op 间并发度
learningRate	学习率	学习率
maxSeqLength	句子截断长度	句子截断长度
numEpochs	epoch 数	epoch 数
numFineTunedLayers	微调层数	微调层数

numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设 （需要取整），否则为总并发度减去 Worker 角色数。
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设 （需要取整），否则为总并发度减去 PS 角色数。
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文 且目录名与压缩文件主文件名一致，可以使用 http://, http 是目录，那么只能使用本地路径，即 file://。
removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv";
schema = "label double, review string"
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schema) \
    .setIgnoreFirstLine(True)
data = data.where("review is not null")
data = ShuffleBatchOp().linkFrom(data)

train = BertTextRegressorTrainBatchOp() \
    .setTextCol("review") \
    .setLabelCol("label") \
    .setNumEpochs(0.05) \
    .setNumFineTunedLayers(1) \
    .setMaxSeqLength(128) \
    .setBertModelName("Base-Chinese") \
    .linkFrom(data)

AkSinkBatchOp() \
    .setFilePath("/tmp/bert_text_regressor_model.ak") \
    .setOverwriteSink(True) \
    .linkFrom(train)

BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.common.utils.JsonConverter;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ShuffleBatchOp;
import
com.alibaba.alink.operator.batch.regression.BertTextRegressorTrainBatchOp;
import com.alibaba.alink.operator.batch.sink.AkSinkBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.google.common.collect.ImmutableMap;
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

public class BertTextRegressorTrainBatchOpTest {

    @Test
    public void testBertTextRegressorTrainBatchOpTest() throws Exception {
        String url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv";
        String schema = "label double, review string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schema)
            .setIgnoreFirstLine(true);
        data = data.where("review is not null");
        data = new ShuffleBatchOp().linkFrom(data);

        Map <String, Map <String, Object>> customConfig = new HashMap <>();
        customConfig.put("train_config", ImmutableMap.of("optimizer_config",
ImmutableMap.of("learning_rate", 0.01)));

        BertTextRegressorTrainBatchOp train = new
BertTextRegressorTrainBatchOp()
            .setTextCol("review")
            .setLabelCol("label")
            .setNumEpochs(0.05)
            .setNumFineTunedLayers(1)
            .setMaxSeqLength(128)
            .setBertModelName("Base-Chinese")
            .setCustomJsonJson(JsonConverter.toJson(customConfig))
            .linkFrom(data);

        new AkSinkBatchOp()
            .setFilePath("/tmp/bert_text_regressor_model.ak")
            .setOverwriteSink(true)
            .linkFrom(train);
```

```
        BatchOperator.execute();
    }
}
```

CART决策树回归预测 (CartRegPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.CartRegPredictBatchOp

Python 类名：CartRegPredictBatchOp

功能介绍

- cart回归是一种常用的树模型
- cart回归组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
```

```
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

trainOp = CartRegTrainBatchOp()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = CartRegPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = CartRegPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.CartRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.CartRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.regression.CartRegPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CartRegPredictBatchOpTest {
    @Test
    public void testCartRegPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, "f0 double, f1
string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new CartRegTrainBatchOp()
```

```
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new CartRegPredictBatchOp()
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new
CartRegPredictStreamOp(trainOp)
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred
3.0000	C	2	2	1	1.0000
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
4.0000	D	3	3	1	1.0000

CART决策树回归训练 (CartRegTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.CartRegTrainBatchOp

Python 类名：CartRegTrainBatchOp

功能介绍

- cart回归是一种常用的树模型
- cart回归组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	

createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String			"series"
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			128
maxDepth	树的深度限制	树的深度限制	Integer			2147483647
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2147483647
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer			64
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.0

minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.0
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			2
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

trainOp = CartRegTrainBatchOp()\
    .setLabelCol('label')\
```

```
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = CartRegPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = CartRegPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.CartRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.CartRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.regression.CartRegPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CartRegTrainBatchOpTest {
    @Test
    public void testCartRegTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new CartRegTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new CartRegPredictBatchOp()
            .setPredictionCol("pred");
```

```
        StreamOperator <?> predictStreamOp = new
CartRegPredictStreamOp(trainOp)
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
4.0000	D	3	3	1	1.0000
3.0000	C	2	2	1	1.0000
2.0000	B	1	1	0	0.0000

决策树回归预测 (DecisionTreeRegPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.DecisionTreeRegPredictBatchOp

Python 类名：DecisionTreeRegPredictBatchOp

功能介绍

- 决策树回归组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
```

```
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

trainOp = DecisionTreeRegTrainBatchOp()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = DecisionTreeRegPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = DecisionTreeRegPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.regression.DecisionTreeRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.DecisionTreeRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.regression.DecisionTreeRegPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DecisionTreeRegPredictBatchOpTest {
    @Test
    public void testDecisionTreeRegPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
```

```
        BatchOperator <?> trainOp = new DecisionTreeRegTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new DecisionTreeRegPredictBatchOp()
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new
DecisionTreeRegPredictStreamOp(trainOp)
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred
4.0000	D	3	3	1	1.0000
3.0000	C	2	2	1	1.0000
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000

决策树回归训练 (DecisionTreeRegTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.DecisionTreeRegTrainBatchOp

Python 类名：DecisionTreeRegTrainBatchOp

功能介绍

- 决策树回归组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	

createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String			"series"
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			128
maxDepth	树的深度限制	树的深度限制	Integer			2147483647
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2147483647
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer			64
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.0

minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.0
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			2
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

trainOp = DecisionTreeRegTrainBatchOp()\
    .setLabelCol('label')\
```

```
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = DecisionTreeRegPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = DecisionTreeRegPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.regression.DecisionTreeRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.DecisionTreeRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.regression.DecisionTreeRegPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DecisionTreeRegTrainBatchOpTest {
    @Test
    public void testDecisionTreeRegTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new DecisionTreeRegTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
```

```
        BatchOperator <?> predictBatchOp = new DecisionTreeRegPredictBatchOp()
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new
DecisionTreeRegPredictStreamOp(trainOp)
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
4.0000	D	3	3	1	1.0000
3.0000	C	2	2	1	1.0000

FM回归预测 (FmRegressorPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.FmRegressorPredictBatchOp

Python 类名：FmRegressorPredictBatchOp

功能介绍

FM即因子分解机（Factor Machine），它的特点是考虑了特征之间的相互作用，是一种非线性模型。该组件使用FM模型解决回归问题。

算法原理

FM模型是线性模型的升级，是在线性表达式后面加入了新的交叉项特征及对应的权值，FM模型的表达式如下所示：

$$y = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

这里我们使用 Adagrad 优化算法求解该模型。算法原理细节可以参考文献[1]。

算法使用

FM算法是推荐领域被验证的效果较好的推荐方案之一，在电商、广告、视频、信息流、游戏的推荐领域有广泛应用。

文献

[1] S. Rendle, "Factorization Machines," 2010 IEEE International Conference on Data Mining, 2010, pp. 995-1000, doi: 10.1109/ICDM.2010.127.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		

modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1:1.1 3:2.0", 1.0],
    ["2:2.1 10:3.1", 1.0],
    ["1:1.2 5:3.2", 0.0],
    ["3:1.2 7:4.2", 0.0]
])

input = BatchOperator.fromDataframe(df, schemaStr='kv string, label double')

fm = FmRegressorTrainBatchOp()\
        .setVectorCol("kv")\
        .setLabelCol("label")
model = input.link(fm)
```

```
predictor = FmRegressorPredictBatchOp()\
        .setPredictionCol("pred")

predictor.linkFrom(model, input).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.FmRegressorPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.FmRegressorTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmRegressorPredictBatchOpTest {
    @Test
    public void testFmRegressorPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:1.1 3:2.0", 1.0),
            Row.of("2:2.1 10:3.1", 1.0),
            Row.of("1:1.2 5:3.2", 0.0),
            Row.of("3:1.2 7:4.2", 0.0)
        );
        BatchOperator <?> input = new MemSourceBatchOp(df, "kv string, label
double");
        BatchOperator <?> fm = new FmRegressorTrainBatchOp()
            .setVectorCol("kv")
            .setLabelCol("label");
        BatchOperator model = input.link(fm);
        BatchOperator <?> predictor = new FmRegressorPredictBatchOp()
            .setPredictionCol("pred");
        predictor.linkFrom(model, input).print();
    }
}
```

运行结果

kv	label	pred
1:1.1 3:2.0	1.0	0.473600
2:2.1 10:3.1	1.0	0.755115

1:1.2 5:3.2	0.0	0.005875
3:1.2 7:4.2	0.0	0.004641

FM回归训练 (FmRegressorTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.FmRegressorTrainBatchOp

Python 类名：FmRegressorTrainBatchOp

功能介绍

FM即因子分解机（Factor Machine），它的特点是考虑了特征之间的相互作用，是一种非线性模型。该组件使用FM模型解决回归问题。

算法原理

FM模型是线性模型的升级，是在线性表达式后面加入了新的交叉项特征及对应的权值，FM模型的表达式如下所示：

$$y = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

这里我们使用 Adagrad 优化算法求解该模型。算法原理细节可以参考文献[1]。

算法使用

FM算法是推荐领域被验证的效果较好的推荐方案之一，在电商、广告、视频、信息流、游戏的推荐领域有广泛应用。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献

[1] S. Rendle, "Factorization Machines," 2010 IEEE International Conference on Data Mining, 2010, pp. 995-1000, doi: 10.1109/ICDM.2010.127.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		

batchSize	迭代数据batch size	数据batch size	Integer			-1
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1.0E-6
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
initStdev	初始化参数的标准差	初始化参数的标准差	Double			0.05
lambda0	常数项正则化系数	常数项正则化系数	Double			0.0
lambda1	线性项正则化系数	线性项正则化系数	Double			0.0
lambda2	二次项正则化系数	二次项正则化系数	Double			0.0
learnRate	学习率	学习率	Double			0.01
numEpochs	epoch数	epoch数	Integer			10
numFactor	因子数	因子数	Integer			10
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true
withLinearItem	是否含有线性项	是否含有线性项	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1:1.1 3:2.0", 1.0],
    ["2:2.1 10:3.1", 1.0],
    ["1:1.2 5:3.2", 0.0],
    ["3:1.2 7:4.2", 0.0]
])

input = BatchOperator.fromDataframe(df, schemaStr='kv string, label double')

fm = FmRegressorTrainBatchOp()\
        .setVectorCol("kv")\
        .setLabelCol("label")
model = input.link(fm)

predictor = FmRegressorPredictBatchOp()\
        .setPredictionCol("pred")

predictor.linkFrom(model, input).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.FmRegressorPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.FmRegressorTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmRegressorTrainBatchOpTest {
    @Test
    public void testFmRegressorTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:1.1 3:2.0", 1.0),
            Row.of("2:2.1 10:3.1", 1.0),
            Row.of("1:1.2 5:3.2", 0.0),
            Row.of("3:1.2 7:4.2", 0.0)
        );
        BatchOperator <?> input = new MemSourceBatchOp(df, "kv string, label
double");
        BatchOperator <?> fm = new FmRegressorTrainBatchOp()
            .setVectorCol("kv")
            .setLabelCol("label");
        BatchOperator model = input.link(fm);
        BatchOperator <?> predictor = new FmRegressorPredictBatchOp()
            .setPredictionCol("pred");
        predictor.linkFrom(model, input).print();
    }
}
```

运行结果

kv	label	pred
1:1.1 3:2.0	1.0	0.473600
2:2.1 10:3.1	1.0	0.755115
1:1.2 5:3.2	0.0	0.005875
3:1.2 7:4.2	0.0	0.004641

备注

该组件的输入为训练数据，输出为Fm回归模型。

GBRank训练 (GBRankBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.GBRankBatchOp

Python 类名：GBRankBatchOp

功能介绍

- 支持连续特征和离散特征
- 支持数据采样和特征采样
- 需要指定分组列，模型会在每一组内根据label列拟合排序结果。和原始论文一样，label列表示“相关度”，越高越好，在计算得分时，会使用2^label -1 来计算。例如可以使用0,1,2,3,....来依次表示很不相关、有点相关、比较相关、相关等等

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
criteria	树分裂的策略	树分裂的策略，可以为PAI, XGBOOST	String		"PAI", "XGBOOST"	"F
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	nu
featureImportanceType	特征重要性类型	特征重要性类型（默认为GAIN）	String		"WEIGHT", "GAIN", "COVER"	"C

featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double			1.
gamma	xgboost中的l2正则项	xgboost中的l2正则项	Double			0.
groupCol	分组单列名	分组单列名，可选	String			nu
lambda	xgboost中的l1正则项	xgboost中的l1正则项	Double			0.
learningRate	学习率	学习率（默认为0.3）	Double			0.
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			12
maxDepth	树的深度限制	树的深度限制	Integer			6
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2(
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			1(
minSumHessianPerLeaf	叶子节点最小Hessian值	叶子节点最小Hessian值（默认为0）	Double			0.

newtonStep	是否使用二阶梯度	是否使用二阶梯度	Boolean			tru
numTrees	模型中树的棵数	模型中树的棵数	Integer			10
p	p	the reference will be pow by p.	Double			1.
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double			1.
tau	tau	the coef of label diff.	Double			0.
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	nu
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	nu

参数建议

对于训练效果来说，比较重要的参数是 树的棵树+学习率、叶子节点最小样本数、单颗树最大深度、特征采样比例。

单个离散特征的取值种类数不能超过256，否则会出错。

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 0, 0],
    [3, 2, 0, 2],
    [4, 3, 0, 1],
    [2, 4, 0, 0],
    [2, 2, 1, 0],
    [4, 3, 1, 2],
    [1, 2, 1, 1],
    [5, 3, 1, 0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')

trainOp = GBRankBatchOp()\
    .setFeatureCols(['f0', 'f1'])\
    .setLabelCol("label")\
    .setGroupCol("group")\
    .setNumTrees(3)\
    .linkFrom(batchSource)

predictBatchOp = GBRankPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = GBRankPredictStreamOp(trainOp)\
        .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

运行结果

批预测结果

```
    f0   f1  group  label      pred
0  2.0  1.0      0    0.0 -1.826941
1  3.0  2.0      0    2.0  2.091384
2  4.0  3.0      0    1.0  1.154667
3  2.0  4.0      0    0.0 -1.822394
4  2.0  2.0      1    0.0 -1.852800
5  4.0  3.0      1    2.0  1.154667
6  1.0  2.0      1    1.0  0.306810
7  5.0  3.0      1    0.0 -1.892832
```

流预测结果

```
f0    f1    group    label    pred
0    2.0    1.0    0    0.0    -1.826941
1    3.0    2.0    0    2.0    2.091384
2    4.0    3.0    0    1.0    1.154667
3    2.0    4.0    0    0.0    -1.822394
4    2.0    2.0    1    0.0    -1.852800
5    4.0    3.0    1    2.0    1.154667
6    1.0    2.0    1    1.0    0.306810
7    5.0    3.0    1    0.0    -1.892832
```

GBRank预测 (GBRankPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.GBRankPredictBatchOp

Python 类名：GBRankPredictBatchOp

功能介绍

- 支持连续特征和离散特征
- 支持数据采样和特征采样
- 需要指定分组列，模型会在每一组内根据label列拟合排序结果。和原始论文一样，label列表示“相关度”，越高越好，在计算得分时，会使用2^label -1 来计算。例如可以使用0,1,2,3,....来依次表示很不相关、有点相关、比较相关、相关等等

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

参数建议

对于训练效果来说，比较重要的参数是 树的棵树+学习率、叶子节点最小样本数、单颗树最大深度、特征采样比例。

单个离散特征的取值种类数不能超过256，否则会出错。

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 0, 0],
    [3, 2, 0, 2],
    [4, 3, 0, 1],
    [2, 4, 0, 0],
    [2, 2, 1, 0],
    [4, 3, 1, 2],
    [1, 2, 1, 1],
    [5, 3, 1, 0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')

trainOp = GBRankBatchOp()\
    .setFeatureCols(['f0', 'f1'])\
    .setLabelCol("label")\
    .setGroupCol("group")\
    .setNumTrees(3)\
    .linkFrom(batchSource)

predictBatchOp = GBRankPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = GBRankPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

运行结果

```
    f0   f1  group  label       pred
0  2.0  1.0      0    0.0 -10.667265
```

```
 1  3.0  2.0      0    2.0  12.336884
 2  4.0  3.0      0    1.0   4.926796
 3  2.0  4.0      0    0.0 -10.816632
 4  2.0  2.0      1    0.0  -8.938445
 5  4.0  3.0      1    2.0   4.926796
 6  1.0  2.0      1    1.0  -1.877382
 7  5.0  3.0      1    0.0 -11.284907
```

GBDT回归预测 (GbdtRegPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.GbdtRegPredictBatchOp

Python 类名：GbdtRegPredictBatchOp

功能介绍

- gbdt(Gradient Boosting Decision Trees)回归，是经典的基于boosting的有监督学习模型，可以用来解决回归问题
- 支持连续特征和离散特征
- 支持数据采样和特征采样

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

trainOp = GbdtRegTrainBatchOp()\
    .setLearningRate(1.0)\
    .setNumTrees(3)\
    .setMinSamplesPerLeaf(1)\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = GbdtRegPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = GbdtRegPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.GbdtRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.GbdtRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.regression.GbdtRegPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class GbdtRegPredictBatchOpTest {
    @Test
    public void testGbdtRegPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new GbdtRegTrainBatchOp()
            .setLearningRate(1.0)
            .setNumTrees(3)
            .setMinSamplesPerLeaf(1)
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new GbdtRegPredictBatchOp()
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new
GbdtRegPredictStreamOp(trainOp)
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
4.0000	D	3	3	1	1.0000
3.0000	C	2	2	1	1.0000
2.0000	B	1	1	0	0.0000

GBDT回归训练 (GbdtRegTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.GbdtRegTrainBatchOp

Python 类名：GbdtRegTrainBatchOp

功能介绍

- gbdt(Gradient Boosting Decision Trees)回归，是经典的基于boosting的有监督学习模型，可以用来解决回归问题
- 支持连续特征和离散特征
- 支持数据采样和特征采样

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
criteria	树分裂的策略	树分裂的策略，可以为PAI, XGBOOST	String		"PAI", "XGBOOST"	"F
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	nu
featureImportanceType	特征重要性类型	特征重要性类型（默认为GAIN）	String		"WEIGHT", "GAIN", "COVER"	"G

featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double			1.
gamma	xgboost中的l2正则项	xgboost中的l2正则项	Double			0.
lambda	xgboost中的l1正则项	xgboost中的l1正则项	Double			0.
learningRate	学习率	学习率（默认为0.3）	Double			0.
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			1
maxDepth	树的深度限制	树的深度限制	Integer			6
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			1
minSumHessianPerLeaf	叶子节点最小Hessian值	叶子节点最小Hessian值（默认为0）	Double			0.

newtonStep	是否使用二阶梯度	是否使用二阶梯度	Boolean			tru
numTrees	模型中树的棵数	模型中树的棵数	Integer			10
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double			1.
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	nu
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	nu

参数建议

对于训练效果来说，比较重要的参数是 树的棵树+学习率、叶子节点最小样本数、单颗树最大深度、特征采样比例。

单个离散特征的取值种类数不能超过256，否则会出错。

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
```

```
])
batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

trainOp = GbdtRegTrainBatchOp()\
    .setLearningRate(1.0)\
    .setNumTrees(3)\
    .setMinSamplesPerLeaf(1)\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .linkFrom(batchSource)
predictBatchOp = GbdtRegPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = GbdtRegPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.GbdtRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.GbdtRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.regression.GbdtRegPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GbdtRegTrainBatchOpTest {
    @Test
    public void testGbdtRegTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
```

```
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, "f0 double, f1 string, f2 int, f3 int, label int");
        BatchOperator <?> trainOp = new GbdtRegTrainBatchOp()
            .setLearningRate(1.0)
            .setNumTrees(3)
            .setMinSamplesPerLeaf(1)
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new GbdtRegPredictBatchOp()
            .setPredictionCol("pred");
        StreamOperator <?> predictStreamOp = new
GbdtRegPredictStreamOp(trainOp)
            .setPredictionCol("pred");
        predictBatchOp.linkFrom(trainOp, batchSource).print();
        predictStreamOp.linkFrom(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
4.0000	D	3	3	1	1.0000
3.0000	C	2	2	1	1.0000
2.0000	B	1	1	0	0.0000

广义线性回归评估 (GlmEvaluationBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.GlmEvaluationBatchOp

Python 类名：GlmEvaluationBatchOp

功能介绍

GLM(Generalized Linear Model)又称为广义线性回归模型，是一种常用的统计模型，也是一种非线性模型族，许多常用的模型都属于广义线性回归。

它描述了响应和预测因子之间的非线性关系。广义线性回归模型具有线性回归模型的广义特征。响应变量遵循正态、二项式、泊松分布、伽马分布或逆高斯分布，链接函数f定义了μ和预测值的线性组合之间的关系。

GLM功能包括GLM训练，GLM预测(批和流)和GLM评估, 其中训练使用迭代最小二乘方法。

算法使用

分布	连接函数	对应算法
二项分布	Logit	逻辑回归
多项分布	Logit	softmax
高斯分布	Identity	线性回归
Poisson分布	Log	Possion回归

文献或出处

[1] https://en.wikipedia.org/wiki/Generalized_linear_model

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.6094,118.0000,69.0000,1.0000,2.0000],
    [2.3026,58.0000,35.0000,1.0000,2.0000],
    [2.7081,42.0000,26.0000,1.0000,2.0000],
    [2.9957,35.0000,21.0000,1.0000,2.0000],
```

```
    [3.4012,27.0000,18.0000,1.0000,2.0000],
    [3.6889,25.0000,16.0000,1.0000,2.0000],
    [4.0943,21.0000,13.0000,1.0000,2.0000],
    [4.3820,19.0000,12.0000,1.0000,2.0000],
    [4.6052,18.0000,12.0000,1.0000,2.0000]
])

source = BatchOperator.fromDataframe(df, schemaStr='u double, lot1 double, lot2
double, offset double, weights double')

featureColNames = ["lot1", "lot2"]
labelColName = "u"

# train
train = GlmTrainBatchOp()\
                .setFamily("gamma")\
                .setLink("Log")\
                .setRegParam(0.3)\
                .setMaxIter(5)\
                .setFeatureCols(featureColNames)\
                .setLabelCol(labelColName)

source.link(train)

# predict
predict =  GlmPredictBatchOp()\
                .setPredictionCol("pred")

predict.linkFrom(train, source)


# eval
eval =  GlmEvaluationBatchOp()\
                .setFamily("gamma")\
                .setLink("Log")\
                .setRegParam(0.3)\
                .setMaxIter(5)\
                .setFeatureCols(featureColNames)\
                .setLabelCol(labelColName)

eval.linkFrom(train, source)

predict.lazyPrint(10)
eval.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.GlmEvaluationBatchOp;
import com.alibaba.alink.operator.batch.regression.GlmPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.GlmTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GlmEvaluationBatchOpTest {
    @Test
    public void testGlmEvaluationBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.6094, 118.0000, 69.0000, 1.0000, 2.0000),
            Row.of(2.3026, 58.0000, 35.0000, 1.0000, 2.0000),
            Row.of(2.7081, 42.0000, 26.0000, 1.0000, 2.0000),
            Row.of(2.9957, 35.0000, 21.0000, 1.0000, 2.0000),
            Row.of(3.4012, 27.0000, 18.0000, 1.0000, 2.0000),
            Row.of(3.6889, 25.0000, 16.0000, 1.0000, 2.0000),
            Row.of(4.0943, 21.0000, 13.0000, 1.0000, 2.0000),
            Row.of(4.3820, 19.0000, 12.0000, 1.0000, 2.0000),
            Row.of(4.6052, 18.0000, 12.0000, 1.0000, 2.0000)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df,
            "u double, lot1 double, lot2 double, offset double, weights
double");
        String[] featureColNames = new String[] {"lot1", "lot2"};
        String labelColName = "u";
        BatchOperator <?> train = new GlmTrainBatchOp()
            .setFamily("gamma")
            .setLink("Log")
            .setRegParam(0.3)
            .setMaxIter(5)
            .setFeatureCols(featureColNames)
            .setLabelCol(labelColName);
        source.link(train);
        BatchOperator <?> predict = new GlmPredictBatchOp()
            .setPredictionCol("pred");
        predict.linkFrom(train, source);
        BatchOperator <?> eval = new GlmEvaluationBatchOp()
            .setFamily("gamma")
            .setLink("Log")
            .setRegParam(0.3)
            .setMaxIter(5)
            .setFeatureCols(featureColNames)
```

```
            .setLabelCol(labelColName);
        eval.linkFrom(train, source);
        predict.lazyPrint(10);
        eval.print();
    }
}
```

运行结果

预测结果

u	lot1	lot2	offset	weights	pred	
0	1.6094	118.0	69.0	1.0	2.0	0.378525
1	2.3026	58.0	35.0	1.0	2.0	0.970639
2	2.7081	42.0	26.0	1.0	2.0	1.126458
3	2.9957	35.0	21.0	1.0	2.0	1.227753
4	3.4012	27.0	18.0	1.0	2.0	1.258898
5	3.6889	25.0	16.0	1.0	2.0	1.305654
6	4.0943	21.0	13.0	1.0	2.0	1.367991
7	4.3820	19.0	12.0	1.0	2.0	1.383571
8	4.6052	18.0	12.0	1.0	2.0	1.375774

评估结果

{"rank":3,"degreeOfFreedom":6,"residualDegreeOfFreeDom":6,"residualDegreeOfFreedomNull":8,"aic":9702.08
[0.007797743508551773,-0.031175844426501245],"intercept":1.6095243247335171,"coefficientStandardError
[0.2566303869744822,-0.5880323136508093,14.715031444760513],"pValues":[0.8060371545111102,0.57795

广义线性回归预测 (GlmPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.GlmPredictBatchOp

Python 类名：GlmPredictBatchOp

功能介绍

GLM(Generalized Linear Model)又称为广义线性回归模型，是一种常用的统计模型，也是一种非线性模型族，许多常用的模型都属于广义线性回归。

它描述了响应和预测因子之间的非线性关系。广义线性回归模型具有线性回归模型的广义特征。响应变量遵循正态、二项式、泊松分布、伽马分布或逆高斯分布，链接函数f定义了μ和预测值的线性组合之间的关系。

GLM功能包括GLM训练，GLM预测(批和流)和GLM评估, 其中训练使用迭代最小二乘方法。

算法使用

分布	连接函数	对应算法
二项分布	Logit	逻辑回归
多项分布	Logit	softmax
高斯分布	Identity	线性回归
Poisson分布	Log	Possion回归

文献或出处

[1] https://en.wikipedia.org/wiki/Generalized_linear_model

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
linkPredResultCol	连接函数结果的列名	连接函数结果的列名	String			null
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.6094,118.0000,69.0000,1.0000,2.0000],
    [2.3026,58.0000,35.0000,1.0000,2.0000],
    [2.7081,42.0000,26.0000,1.0000,2.0000],
    [2.9957,35.0000,21.0000,1.0000,2.0000],
    [3.4012,27.0000,18.0000,1.0000,2.0000],
    [3.6889,25.0000,16.0000,1.0000,2.0000],
    [4.0943,21.0000,13.0000,1.0000,2.0000],
    [4.3820,19.0000,12.0000,1.0000,2.0000],
    [4.6052,18.0000,12.0000,1.0000,2.0000]
])

source = BatchOperator.fromDataframe(df, schemaStr='u double, lot1 double, lot2
double, offset double, weights double')

featureColNames = ["lot1", "lot2"]
labelColName = "u"

# train
train = GlmTrainBatchOp()\
                .setFamily("gamma")\
                .setLink("Log")\
                .setRegParam(0.3)\
                .setMaxIter(5)\
                .setFeatureCols(featureColNames)\
                .setLabelCol(labelColName)

source.link(train)

# predict
predict =  GlmPredictBatchOp()\
                .setPredictionCol("pred")

predict.linkFrom(train, source)


# eval
eval =  GlmEvaluationBatchOp()\
```

```
                .setFamily("gamma")\
                .setLink("Log")\
                .setRegParam(0.3)\
                .setMaxIter(5)\
                .setFeatureCols(featureColNames)\
                .setLabelCol(labelColName)

eval.linkFrom(train, source)

predict.lazyPrint(10)
eval.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.GlmEvaluationBatchOp;
import com.alibaba.alink.operator.batch.regression.GlmPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.GlmTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GlmPredictBatchOpTest {
    @Test
    public void testGlmPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.6094, 118.0000, 69.0000, 1.0000, 2.0000),
            Row.of(2.3026, 58.0000, 35.0000, 1.0000, 2.0000),
            Row.of(2.7081, 42.0000, 26.0000, 1.0000, 2.0000),
            Row.of(2.9957, 35.0000, 21.0000, 1.0000, 2.0000),
            Row.of(3.4012, 27.0000, 18.0000, 1.0000, 2.0000),
            Row.of(3.6889, 25.0000, 16.0000, 1.0000, 2.0000),
            Row.of(4.0943, 21.0000, 13.0000, 1.0000, 2.0000),
            Row.of(4.3820, 19.0000, 12.0000, 1.0000, 2.0000),
            Row.of(4.6052, 18.0000, 12.0000, 1.0000, 2.0000)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df,
            "u double, lot1 double, lot2 double, offset double, weights
double");
        String[] featureColNames = new String[] {"lot1", "lot2"};
        String labelColName = "u";
        BatchOperator <?> train = new GlmTrainBatchOp()
            .setFamily("gamma")
```

```
            .setLink("Log")
            .setRegParam(0.3)
            .setMaxIter(5)
            .setFeatureCols(featureColNames)
            .setLabelCol(labelColName);
        source.link(train);
        BatchOperator <?> predict = new GlmPredictBatchOp()
            .setPredictionCol("pred");
        predict.linkFrom(train, source);
        BatchOperator <?> eval = new GlmEvaluationBatchOp()
            .setFamily("gamma")
            .setLink("Log")
            .setRegParam(0.3)
            .setMaxIter(5)
            .setFeatureCols(featureColNames)
            .setLabelCol(labelColName);
        eval.linkFrom(train, source);
        predict.lazyPrint(10);
        eval.print();
    }
}
```

运行结果

预测结果

u	lot1	lot2	offset	weights	pred	
0	1.6094	118.0	69.0	1.0	2.0	0.378525
1	2.3026	58.0	35.0	1.0	2.0	0.970639
2	2.7081	42.0	26.0	1.0	2.0	1.126458
3	2.9957	35.0	21.0	1.0	2.0	1.227753
4	3.4012	27.0	18.0	1.0	2.0	1.258898
5	3.6889	25.0	16.0	1.0	2.0	1.305654
6	4.0943	21.0	13.0	1.0	2.0	1.367991
7	4.3820	19.0	12.0	1.0	2.0	1.383571
8	4.6052	18.0	12.0	1.0	2.0	1.375774

评估结果

{"rank":3,"degreeOfFreedom":6,"residualDegreeOfFreeDom":6,"residualDegreeOfFreedomNull":8,"aic":9702.08
[0.007797743508551773,-0.031175844426501245],"intercept":1.6095243247335171,"coefficientStandardError
[0.2566303869744822,-0.5880323136508093,14.715031444760513],"pValues":[0.8060371545111102,0.57795

广义线性回归训练 (GlmTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.GlmTrainBatchOp

Python 类名：GlmTrainBatchOp

功能介绍

GLM(Generalized Linear Model)又称为广义线性回归模型，是一种常用的统计模型，也是一种非线性模型族，许多常用的模型都属于广义线性回归。

它描述了响应和预测因子之间的非线性关系。广义线性回归模型具有线性回归模型的广义特征。响应变量遵循正态、二项式、泊松分布、伽马分布或逆高斯分布，链接函数f定义了μ和预测值的线性组合之间的关系。

GLM功能包括GLM训练，GLM预测(批和流)和GLM评估, 其中训练使用迭代最小二乘方法。

算法使用

分布	连接函数	对应算法
二项分布	Logit	逻辑回归
多项分布	Logit	softmax
高斯分布	Identity	线性回归
Poisson分布	Log	Possion回归

文献或出处

[1] https://en.wikipedia.org/wiki/Generalized_linear_model

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
labelCol	标签列名	输入表中的标签列名	String	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
epsilon	收敛精度	收敛精度	Double			1.0E-5
family	分布族	分布族，包含gaussian, Binomial, Poisson, Gamma and Tweedie，默认值gaussian。	String		"Gamma", "Binomial", "Gaussian", "Poisson", "Tweedie"	"Gaussian"
fitIntercept	是否拟合常数项	是否拟合常数项，默认是拟合	Boolean			true
link	连接函数	连接函数，包含cloglog, Identity, Inverse, log, logit, power, probit和sqrt，默认值是指数分布族对应的连接函数。	String		"CLogLog", "Identity", "Inverse", "Log", "Logit", "Power", "Probit", "Sqrt"	null

linkPower	连接函数的超参	连接函数的超参	Double			1.0
maxIter	最大迭代步数	最大迭代步数，默认为 10。	Integer			10
offsetCol	偏移列	偏移列	String			null
regParam	l2正则系数	l2正则系数	Double			0.0
variancePower	分布族的超参	分布族的超参，默认值是0.0	Double			0.0
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.6094,118.0000,69.0000,1.0000,2.0000],
    [2.3026,58.0000,35.0000,1.0000,2.0000],
    [2.7081,42.0000,26.0000,1.0000,2.0000],
    [2.9957,35.0000,21.0000,1.0000,2.0000],
    [3.4012,27.0000,18.0000,1.0000,2.0000],
    [3.6889,25.0000,16.0000,1.0000,2.0000],
    [4.0943,21.0000,13.0000,1.0000,2.0000],
    [4.3820,19.0000,12.0000,1.0000,2.0000],
    [4.6052,18.0000,12.0000,1.0000,2.0000]
])

source = BatchOperator.fromDataframe(df, schemaStr='u double, lot1 double, lot2
double, offset double, weights double')

featureColNames = ["lot1", "lot2"]
labelColName = "u"

# train
train = GlmTrainBatchOp()\
                .setFamily("gamma")\
                .setLink("Log")\
                .setRegParam(0.3)\
                .setMaxIter(5)\
                .setFeatureCols(featureColNames)\
                .setLabelCol(labelColName)

source.link(train)

# predict
predict =  GlmPredictBatchOp()\
                .setPredictionCol("pred")

predict.linkFrom(train, source)


# eval
eval =  GlmEvaluationBatchOp()\
                .setFamily("gamma")\
                .setLink("Log")\
                .setRegParam(0.3)\
                .setMaxIter(5)\
                .setFeatureCols(featureColNames)\
                .setLabelCol(labelColName)
```

```
eval.linkFrom(train, source)

predict.lazyPrint(10)
eval.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.GlmEvaluationBatchOp;
import com.alibaba.alink.operator.batch.regression.GlmPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.GlmTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GlmTrainBatchOpTest {
    @Test
    public void testGlmTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.6094, 118.0000, 69.0000, 1.0000, 2.0000),
            Row.of(2.3026, 58.0000, 35.0000, 1.0000, 2.0000),
            Row.of(2.7081, 42.0000, 26.0000, 1.0000, 2.0000),
            Row.of(2.9957, 35.0000, 21.0000, 1.0000, 2.0000),
            Row.of(3.4012, 27.0000, 18.0000, 1.0000, 2.0000),
            Row.of(3.6889, 25.0000, 16.0000, 1.0000, 2.0000),
            Row.of(4.0943, 21.0000, 13.0000, 1.0000, 2.0000),
            Row.of(4.3820, 19.0000, 12.0000, 1.0000, 2.0000),
            Row.of(4.6052, 18.0000, 12.0000, 1.0000, 2.0000)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df,
            "u double, lot1 double, lot2 double, offset double, weights
double");
        String[] featureColNames = new String[] {"lot1", "lot2"};
        String labelColName = "u";
        BatchOperator <?> train = new GlmTrainBatchOp()
            .setFamily("gamma")
            .setLink("Log")
            .setRegParam(0.3)
            .setMaxIter(5)
            .setFeatureCols(featureColNames)
            .setLabelCol(labelColName);
        source.link(train);
```

```
        BatchOperator <?> predict = new GlmPredictBatchOp()
            .setPredictionCol("pred");
        predict.linkFrom(train, source);
        BatchOperator <?> eval = new GlmEvaluationBatchOp()
            .setFamily("gamma")
            .setLink("Log")
            .setRegParam(0.3)
            .setMaxIter(5)
            .setFeatureCols(featureColNames)
            .setLabelCol(labelColName);
        eval.linkFrom(train, source);
        predict.lazyPrint(10);
        eval.print();
    }
}
```

运行结果

预测结果

u	lot1	lot2	offset	weights	pred	
0	1.6094	118.0	69.0	1.0	2.0	0.378525
1	2.3026	58.0	35.0	1.0	2.0	0.970639
2	2.7081	42.0	26.0	1.0	2.0	1.126458
3	2.9957	35.0	21.0	1.0	2.0	1.227753
4	3.4012	27.0	18.0	1.0	2.0	1.258898
5	3.6889	25.0	16.0	1.0	2.0	1.305654
6	4.0943	21.0	13.0	1.0	2.0	1.367991
7	4.3820	19.0	12.0	1.0	2.0	1.383571
8	4.6052	18.0	12.0	1.0	2.0	1.375774

评估结果

{"rank":3,"degreeOfFreedom":6,"residualDegreeOfFreeDom":6,"residualDegreeOfFreedomNull":8,"aic":9702.08
[0.007797743508551773,-0.031175844426501245],"intercept":1.6095243247335171,"coefficientStandardError
[0.2566303869744822,-0.5880323136508093,14.715031444760513],"pValues":[0.8060371545111102,0.57795

保序回归预测 (IsotonicRegPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.IsotonicRegPredictBatchOp

Python 类名：IsotonicRegPredictBatchOp

功能介绍

保序回归在观念上是寻找一组非递减的片段连续线性函数（piecewise linear continuous functions），即保序函数，使其与样本尽可能的接近。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0.35, 1],
    [0.6, 1],
    [0.55, 1],
    [0.5, 1],
    [0.18, 0],
    [0.1, 1],
    [0.8, 1],
    [0.45, 0],
    [0.4, 1],
    [0.7, 0],
    [0.02, 1],
```

```
    [0.3, 0],
    [0.27, 1],
    [0.2, 0],
    [0.9, 1]
])

data = BatchOperator.fromDataframe(df, schemaStr="feature double, label
double")

trainOp = IsotonicRegTrainBatchOp()\
            .setFeatureCol("feature")\
            .setLabelCol("label")

model = trainOp.linkFrom(data)

predictOp = IsotonicRegPredictBatchOp()\
            .setPredictionCol("result")

predictOp.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.IsotonicRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.IsotonicRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class IsotonicRegPredictBatchOpTest {
    @Test
    public void testIsotonicRegPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0.35, 1.0),
            Row.of(0.6, 1.0),
            Row.of(0.55, 1.0),
            Row.of(0.5, 1.0),
            Row.of(0.18, 0.0),
            Row.of(0.1, 1.0),
            Row.of(0.8, 1.0),
            Row.of(0.45, 0.0),
            Row.of(0.4, 1.0),
            Row.of(0.7, 0.0),
```

```
            Row.of(0.02, 1.0),
            Row.of(0.3, 0.0),
            Row.of(0.27, 1.0),
            Row.of(0.2, 0.0),
            Row.of(0.9, 1.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "feature double,
label double");
        BatchOperator <?> trainOp = new IsotonicRegTrainBatchOp()
            .setFeatureCol("feature")
            .setLabelCol("label");
        BatchOperator model = trainOp.linkFrom(data);
        BatchOperator <?> predictOp = new IsotonicRegPredictBatchOp()
            .setPredictionCol("result");
        predictOp.linkFrom(model, data).print();
    }
}
```

运行结果

模型结果

model_id	model_info
0	{"vectorCol":"\"col2\"","featureIndex":"0","featureCol":null}
1048576	[0.02,0.3,0.35,0.45,0.5,0.7]
2097152	[0.5,0.5,0.6666666865348816,0.6666666865348816,0.75,0.75]

预测结果

col1	col2	col3	pred
1.0	0.9	1.0	0.75
0.0	0.7	1.0	0.75
1.0	0.35	1.0	0.6666666865348816
1.0	0.02	1.0	0.5
1.0	0.27	1.0	0.5
1.0	0.5	1.0	0.75
0.0	0.18	1.0	0.5
0.0	0.45	1.0	0.6666666865348816

1.0	0.8	1.0	0.75
1.0	0.6	1.0	0.75
1.0	0.4	1.0	0.6666666865348816
0.0	0.3	1.0	0.5
1.0	0.55	1.0	0.75
0.0	0.2	1.0	0.5
1.0	0.1	1.0	0.5

保序回归训练 (IsotonicRegTrainBatchOp)

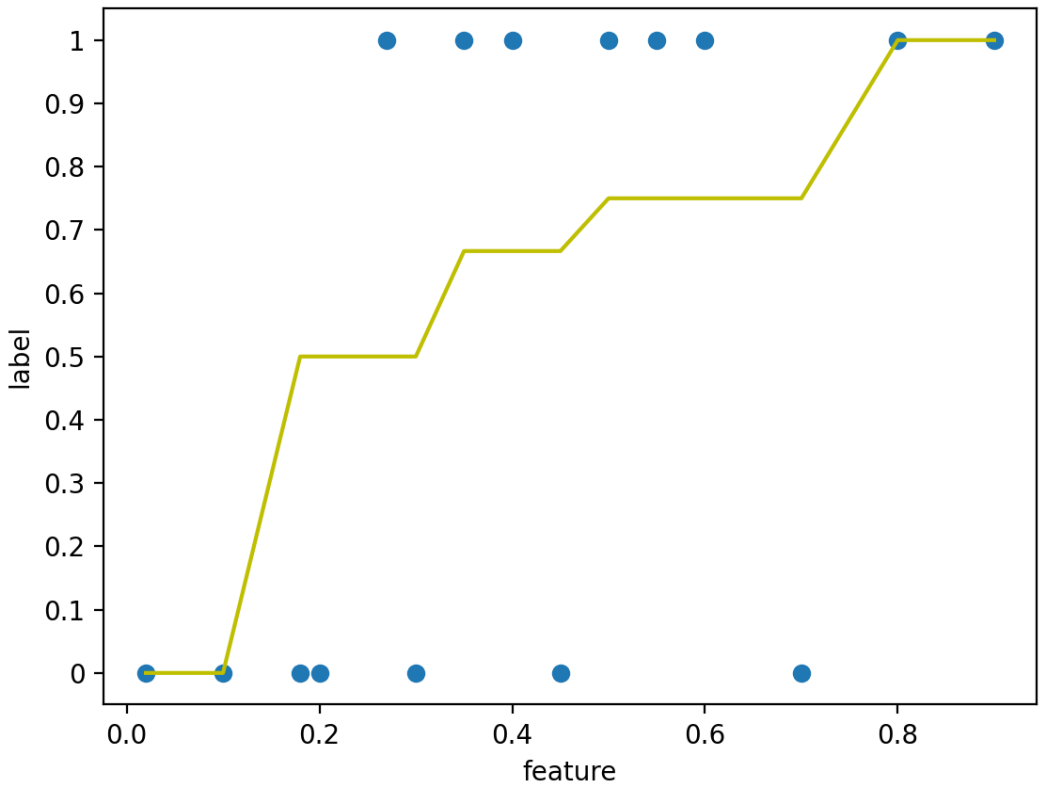
Java 类名：com.alibaba.alink.operator.batch.regression.IsotonicRegTrainBatchOp

Python 类名：IsotonicRegTrainBatchOp

功能介绍

保序回归在观念上是寻找一组非递减的片段连续线性函数（piecewise linear continuous functions），即保序函数，使其与样本尽可能的接近。

保序回归的输入在Alink中称分别为特征（feature）、标签（label）和权重（weight），特征可以是数值或向量，如果是向量还需要设定特征索引（feature index），组件将使用该维进行计算。保序回归的目标是求解一个能使$\textstyle \sum_i{w_i(y_i-\hat{y}_i)^2}$最小的序列$\hat{y}$，若选择保增序，该序列还应满足$X_i<X_j$时$\hat{y}_i\le\hat{y}_j$，若选择保降序满足$X_i<X_j$时$\hat{y}_i\ge\hat{y}_j$。下图中，散点图是训练数据，折线图是得到的保序回归模型，对于训练数据中没有的特征，使用线性插值得到其标签。对应训练和预测代码见示例。



参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
featureCol	特征列名	特征列的名称	String		所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
featureIndex	训练特征所在维度	训练特征在输入向量的维度索引	Integer		[0, +inf)	0
isotonic	输出序列是否	输出序列是否递增	Boolean			true
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为 [DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0.35, 1],
    [0.6, 1],
    [0.55, 1],
    [0.5, 1],
```

```
    [0.18, 0],
    [0.1, 1],
    [0.8, 1],
    [0.45, 0],
    [0.4, 1],
    [0.7, 0],
    [0.02, 1],
    [0.3, 0],
    [0.27, 1],
    [0.2, 0],
    [0.9, 1]
])

data = BatchOperator.fromDataframe(df, schemaStr="feature double, label
double")

trainOp = IsotonicRegTrainBatchOp()\
            .setFeatureCol("feature")\
            .setLabelCol("label")

model = trainOp.linkFrom(data)

predictOp = IsotonicRegPredictBatchOp()\
            .setPredictionCol("result")

predictOp.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.IsotonicRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.IsotonicRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class IsotonicRegTrainBatchOpTest {
    @Test
    public void testIsotonicRegTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0.35, 1.0),
            Row.of(0.6, 1.0),
            Row.of(0.55, 1.0),
```

```
            Row.of(0.5, 1.0),
            Row.of(0.18, 0.0),
            Row.of(0.1, 1.0),
            Row.of(0.8, 1.0),
            Row.of(0.45, 0.0),
            Row.of(0.4, 1.0),
            Row.of(0.7, 0.0),
            Row.of(0.02, 1.0),
            Row.of(0.3, 0.0),
            Row.of(0.27, 1.0),
            Row.of(0.2, 0.0),
            Row.of(0.9, 1.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "feature double,
label double");
        BatchOperator <?> trainOp = new IsotonicRegTrainBatchOp()
            .setFeatureCol("feature")
            .setLabelCol("label");
        BatchOperator model = trainOp.linkFrom(data);
        BatchOperator <?> predictOp = new IsotonicRegPredictBatchOp()
            .setPredictionCol("result");
        predictOp.linkFrom(model, data).print();
    }
}
```

运行结果

模型结果

model_id	model_info
0	{"vectorCol":"\"col2\"","featureIndex":"0","featureCol":null}
1048576	[0.02,0.3,0.35,0.45,0.5,0.7]
2097152	[0.5,0.5,0.6666666865348816,0.6666666865348816,0.75,0.75]

预测结果

col1	col2	col3	pred
1.0	0.9	1.0	0.75
0.0	0.7	1.0	0.75
1.0	0.35	1.0	0.6666666865348816
1.0	0.02	1.0	0.5

1.0	0.27	1.0	0.5
1.0	0.5	1.0	0.75
0.0	0.18	1.0	0.5
0.0	0.45	1.0	0.6666666865348816
1.0	0.8	1.0	0.75
1.0	0.6	1.0	0.75
1.0	0.4	1.0	0.6666666865348816
0.0	0.3	1.0	0.5
1.0	0.55	1.0	0.75
0.0	0.2	1.0	0.5
1.0	0.1	1.0	0.5

KerasSequential回归预测 (KerasSequentialRegressorPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.KerasSequentialRegressorPredictBatchOp

Python 类名：KerasSequentialRegressorPredictBatchOp

功能介绍

与 KerasSequential 回归训练组件对应的预测组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
inferBatchSize	推理数据批大小	推理数据批大小	Integer			256
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
source = CsvSourceBatchOp() \
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/random_tensor.csv") \
    .setSchemaStr("tensor string, label double")

source = ToTensorBatchOp() \
    .setSelectedCol("tensor") \
    .setTensorDataType("DOUBLE") \
    .setTensorShape([200, 3]) \
    .linkFrom(source)

trainBatchOp = KerasSequentialRegressorTrainBatchOp() \
    .setTensorCol("tensor") \
    .setLabelCol("label") \
    .setLayers([
```

```
        "Conv1D(256, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Dropout(0.1)",
        "MaxPooling1D(pool_size=8)",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Flatten()"
    ]) \
    .setOptimizer("Adam()") \
    .setNumEpochs(1) \
    .linkFrom(source)

predictBatchOp = KerasSequentialRegressorPredictBatchOp() \
    .setPredictionCol("pred") \
    .setReservedCols(["label"]) \
    .linkFrom(trainBatchOp, source)
predictBatchOp.lazyPrint(10)
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ToTensorBatchOp;
import
com.alibaba.alink.operator.batch.regression.KerasSequentialRegressorPredictBatc
hOp;
import
com.alibaba.alink.operator.batch.regression.KerasSequentialRegressorTrainBatchO
p;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class KerasSequentialRegressorTrainBatchOpTest {

    @Test
    public void testKerasSequentialRegressorTrainBatchOp() throws Exception {
        BatchOperator<?> source = new CsvSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/random_tensor.csv")
            .setSchemaStr("tensor string, label double");

        source = new ToTensorBatchOp()
            .setSelectedCol("tensor")
            .setTensorDataType("DOUBLE")
            .setTensorShape(200, 3)
            .linkFrom(source);
```

```
        KerasSequentialRegressorTrainBatchOp trainBatchOp = new
KerasSequentialRegressorTrainBatchOp()
            .setTensorCol("tensor")
            .setLabelCol("label")
            .setLayers(new String[] {
                "Conv1D(256, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Dropout(0.1)",
                "MaxPooling1D(pool_size=8)",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Flatten()"
            })
            .setOptimizer("Adam()")
            .setNumEpochs(1)
            .linkFrom(source);

        KerasSequentialRegressorPredictBatchOp predictBatchOp = new
KerasSequentialRegressorPredictBatchOp()
            .setPredictionCol("pred")
            .setReservedCols("label")
            .linkFrom(trainBatchOp, source);
        predictBatchOp.lazyPrint(10);
        BatchOperator.execute();
    }
}
```

运行结果

label	pred
1.0000	0.4822
0.0000	0.4826
0.0000	0.4752
0.0000	0.4702
1.0000	0.4907
1.0000	0.4992
0.0000	0.4866
1.0000	0.5045
0.0000	0.4994
1.0000	0.4837

KerasSequential回归训练 (KerasSequentialRegressorTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.KerasSequentialRegressorTrainBatchOp

Python 类名：KerasSequentialRegressorTrainBatchOp

功能介绍

构建一个 Keras 的 Sequential 模型， 训练回归模型。

通过 layers 参数指定构成 Sequential 模型的网络层，Alink 会自动在最开始添加 Input 层，在最后添加 Dense 层和激活层，得到完整的模型用于训练。

指定 layers 参数时，使用的是 Python 语句，例如

```
"Conv1D(256, 5, padding='same', activation='relu')",
"Conv1D(128, 5, padding='same', activation='relu')",
"Dropout(0.1)",
"MaxPooling1D(pool_size=8)",
"Conv1D(128, 5, padding='same', activation='relu')",
"Conv1D(128, 5, padding='same', activation='relu')",
"Flatten()"
```

`tf.keras.layers` 内的网络层已经提前 import，可以直接使用。 使用的 TensorFlow 版本是 2.3.1。

该组件可以接 `KerasSequentialRegressorPredictBatchOp` 或 `KerasSequentialRegressorPredictStreamOp` 进行推理。

参数说明

名称	中文名称	描述	类型	是
labelCol	标签列名	输入表中的标签列名	String	✓
layers	各 layer 的描述	各 layer 的描述，使用 Python 语法，例如 "Conv1D(256, 5, padding='same', activation='relu')"	String[]	✓
tensorCol	tensor列	tensor列	String	✓
batchSize	数据批大小	数据批大小	Integer	

bestMetric	最优指标	判断模型最优时用的指标，仅在总并发度为 1 时起作用。都支持的有：loss； 二分类还支持：auc, precision, recall, binary_accuracy, false_negatives, false_positives, true_negatives, true_positives；多分类还支持：sparse_categorical_accuracy；回归还支持：mean_absolute_error, mean_absolute_percentage_error, mean_squared_error, mean_squared_logarithmic_error, root_mean_squared_error	String	
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estimator` 的 `model_dir` 传入，需要为所有 worker 都能访问到的目录	String	
intraOpParallelism	Op 间并发度	Op 间并发度	Integer	
learningRate	学习率	学习率	Double	
numEpochs	epoch数	epoch数	Integer	
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer	
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer	
optimizer	优化器	优化器，使用 Python 语法，例如 "Adam(learning_rate=0.1)"	String	
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String	

removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只会删除必要的文件	Boolean	
saveBestOnly	是否导出最优的 checkpoint	是否导出最优的 checkpoint，仅在总并发度为 1 时生效	Boolean	
saveCheckpointsEpochs	每隔多少 epochs 保存 checkpoints	每隔多少 epochs 保存 checkpoints	Double	
saveCheckpointsSecs	每隔多少秒保存 checkpoints	每隔多少秒保存 checkpoints	Double	
validationSplit	验证集比例	验证集比例，仅在总并发度为 1 时生效	Double	

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
source = CsvSourceBatchOp() \
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/random_tensor.csv") \
    .setSchemaStr("tensor string, label double")

source = ToTensorBatchOp() \
    .setSelectedCol("tensor") \
    .setTensorDataType("DOUBLE") \
    .setTensorShape([200, 3]) \
    .linkFrom(source)

trainBatchOp = KerasSequentialRegressorTrainBatchOp() \
    .setTensorCol("tensor") \
    .setLabelCol("label") \
    .setLayers([
        "Conv1D(256, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Dropout(0.1)",
        "MaxPooling1D(pool_size=8)",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
```

```
        "Flatten()"
    ]) \
    .setOptimizer("Adam()") \
    .setNumEpochs(1) \
    .linkFrom(source)

predictBatchOp = KerasSequentialRegressorPredictBatchOp() \
    .setPredictionCol("pred") \
    .setReservedCols(["label"]) \
    .linkFrom(trainBatchOp, source)
predictBatchOp.lazyPrint(10)
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ToTensorBatchOp;
import
com.alibaba.alink.operator.batch.regression.KerasSequentialRegressorPredictBatc
hOp;
import
com.alibaba.alink.operator.batch.regression.KerasSequentialRegressorTrainBatchO
p;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import org.junit.Test;

public class KerasSequentialRegressorTrainBatchOpTest {

    @Test
    public void testKerasSequentialRegressorTrainBatchOp() throws Exception {
        BatchOperator<?> source = new CsvSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/random_tensor.csv")
            .setSchemaStr("tensor string, label double");

        source = new ToTensorBatchOp()
            .setSelectedCol("tensor")
            .setTensorDataType("DOUBLE")
            .setTensorShape(200, 3)
            .linkFrom(source);

        KerasSequentialRegressorTrainBatchOp trainBatchOp = new
KerasSequentialRegressorTrainBatchOp()
            .setTensorCol("tensor")
            .setLabelCol("label")
            .setLayers(new String[] {
                "Conv1D(256, 5, padding='same', activation='relu')",
```

```
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Dropout(0.1)",
                "MaxPooling1D(pool_size=8)",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Flatten()"
            })
            .setOptimizer("Adam()")
            .setNumEpochs(1)
            .linkFrom(source);

        KerasSequentialRegressorPredictBatchOp predictBatchOp = new
KerasSequentialRegressorPredictBatchOp()
            .setPredictionCol("pred")
            .setReservedCols("label")
            .linkFrom(trainBatchOp, source);
        predictBatchOp.lazyPrint(10);
        BatchOperator.execute();
    }
}
```

运行结果

label	pred
1.0000	0.4822
0.0000	0.4826
0.0000	0.4752
0.0000	0.4702
1.0000	0.4907
1.0000	0.4992
0.0000	0.4866
1.0000	0.5045
0.0000	0.4994
1.0000	0.4837

LambdaMart DCG训练 (LambdaMartDcgBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LambdaMartDcgBatchOp

Python 类名：LambdaMartDcgBatchOp

功能介绍

- LambdaMART，是经典的基于gbdt的learning to rank模型。模型细节可参考 Burges, Christopher JC. "From ranknet to lambdarank to lambdamart: An overview." Learning 11.23-581 (2010): 81.
- 支持连续特征和离散特征
- 支持数据采样和特征采样
- 需要指定分组列，模型会在每一组内根据label列拟合排序结果。和原始论文一样，label列表示“相关度”，越高越好，在计算得分时，会使用2^label -1 来计算。例如可以使用0,1,2,3,....来依次表示很不相关、有点相关、比较相关、相关等等

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
criteria	树分裂的策略	树分裂的策略，可以为PAI, XGBOOST	String		"PAI", "XGBOOST"	"F
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	nu

featureImportanceType	特征重要性类型	特征重要性类型（默认为GAIN）	String		"WEIGHT", "GAIN", "COVER"	"G
featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double			1.
gamma	xgboost中的l2正则项	xgboost中的l2正则项	Double			0.
groupCol	分组单列名	分组单列名，可选	String			nu
lambda	xgboost中的l1正则项	xgboost中的l1正则项	Double			0.
learningRate	学习率	学习率（默认为0.3）	Double			0.
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			12
maxDepth	树的深度限制	树的深度限制	Integer			6
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			1(

minSumHessianPerLeaf	叶子节点最小Hessian值	叶子节点最小Hessian值（默认为0）	Double			0.
newtonStep	是否使用二阶梯度	是否使用二阶梯度	Boolean			tru
numTrees	模型中树的棵数	模型中树的棵数	Integer			10
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double			1.
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	nu
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	nu

参数建议

对于训练效果来说，比较重要的参数是 树的棵树+学习率、叶子节点最小样本数、单颗树最大深度、特征采样比例。

单个离散特征的取值种类数不能超过256，否则会出错。

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
```

```
    [2, 1, 0, 0],
    [3, 2, 0, 2],
    [4, 3, 0, 1],
    [2, 4, 0, 0],
    [2, 2, 1, 0],
    [4, 3, 1, 2],
    [1, 2, 1, 1],
    [5, 3, 1, 0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')

trainOp = LambdaMartDcgBatchOp()\
    .setFeatureCols(['f0', 'f1'])\
    .setLabelCol("label")\
    .setGroupCol("group")\
    .setNumTrees(3)\
    .linkFrom(batchSource)

predictBatchOp = LambdaMartDcgPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = LambdaMartDcgPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

运行结果

批预测结果

```
    f0   f1  group  label      pred
0  2.0  1.0      0    0.0 -0.964759
1  3.0  2.0      0    2.0  0.619588
2  4.0  3.0      0    1.0  0.619588
3  2.0  4.0      0    0.0 -0.964759
4  2.0  2.0      1    0.0 -0.964759
5  4.0  3.0      1    2.0  0.619588
6  1.0  2.0      1    1.0 -0.964759
7  5.0  3.0      1    0.0  0.619588
```

流预测结果

```
    f0    f1    group    label    pred
0    2.0    1.0    0    0.0    -0.964759
1    3.0    2.0    0    2.0    0.619588
2    4.0    3.0    0    1.0    0.619588
3    2.0    4.0    0    0.0    -0.964759
4    2.0    2.0    1    0.0    -0.964759
5    4.0    3.0    1    2.0    0.619588
6    1.0    2.0    1    1.0    -0.964759
7    5.0    3.0    1    0.0    0.619588
```

LambdaMart DCG预测 (LambdaMartDcgPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LambdaMartDcgPredictBatchOp

Python 类名：LambdaMartDcgPredictBatchOp

功能介绍

- LambdaMART，是经典的基于gbdt的learning to rank模型。模型细节可参考 Burges, Christopher JC. "From ranknet to lambdarank to lambdamart: An overview." Learning 11.23-581 (2010): 81.
- 支持连续特征和离散特征
- 支持数据采样和特征采样
- 需要指定分组列，模型会在每一组内根据label列拟合排序结果。和原始论文一样，label列表示“相关度”，越高越好，在计算得分时，会使用2^label -1 来计算。例如可以使用0,1,2,3,....来依次表示很不相关、有点相关、比较相关、相关等等

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

参数建议

对于训练效果来说，比较重要的参数是 树的棵树+学习率、叶子节点最小样本数、单颗树最大深度、特征采样比例。

单个离散特征的取值种类数不能超过256，否则会出错。

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 0, 0],
    [3, 2, 0, 2],
    [4, 3, 0, 1],
    [2, 4, 0, 0],
    [2, 2, 1, 0],
    [4, 3, 1, 2],
    [1, 2, 1, 1],
    [5, 3, 1, 0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')

trainOp = LambdaMartDcgBatchOp()\
    .setFeatureCols(['f0', 'f1'])\
    .setLabelCol("label")\
    .setGroupCol("group")\
    .setNumTrees(3)\
    .linkFrom(batchSource)

predictBatchOp = LambdaMartDcgPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = LambdaMartDcgPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

运行结果

	f0	f1	group	label	pred
0	2.0	1.0	0	0.0	-0.964759
1	3.0	2.0	0	2.0	0.619588
2	4.0	3.0	0	1.0	0.619588
3	2.0	4.0	0	0.0	-0.964759
4	2.0	2.0	1	0.0	-0.964759
5	4.0	3.0	1	2.0	0.619588
6	1.0	2.0	1	1.0	-0.964759
7	5.0	3.0	1	0.0	0.619588

LambdaMart NDCG训练 (LambdaMartNdcgBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LambdaMartNdcgBatchOp

Python 类名：LambdaMartNdcgBatchOp

功能介绍

- LambdaMART，是经典的基于gbdt的learning to rank模型。模型细节可参考 Burges, Christopher JC. "From ranknet to lambdarank to lambdamart: An overview." Learning 11.23-581 (2010): 81.
- 支持连续特征和离散特征
- 支持数据采样和特征采样
- 需要指定分组列，模型会在每一组内根据label列拟合排序结果。和原始论文一样，label列表示“相关度”，越高越好，在计算得分时，会使用2^label -1 来计算。例如可以使用0,1,2,3,....来依次表示很不相关、有点相关、比较相关、相关等等

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
criteria	树分裂的策略	树分裂的策略，可以为PAI, XGBOOST	String		"PAI", "XGBOOST"	"F
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	nu

featureImportanceType	特征重要性类型	特征重要性类型（默认为GAIN）	String		"WEIGHT", "GAIN", "COVER"	"G
featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double			1.
gamma	xgboost中的l2正则项	xgboost中的l2正则项	Double			0.
groupCol	分组单列名	分组单列名，可选	String			nu
lambda	xgboost中的l1正则项	xgboost中的l1正则项	Double			0.
learningRate	学习率	学习率（默认为0.3）	Double			0.
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			12
maxDepth	树的深度限制	树的深度限制	Integer			6
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			1(

minSumHessianPerLeaf	叶子节点最小Hessian值	叶子节点最小Hessian值（默认为0）	Double			0.
newtonStep	是否使用二阶梯度	是否使用二阶梯度	Boolean			tru
numTrees	模型中树的棵数	模型中树的棵数	Integer			10
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double			1.
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	nu
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	nu

参数建议

对于训练效果来说，比较重要的参数是 树的棵树+学习率、叶子节点最小样本数、单颗树最大深度、特征采样比例。

单个离散特征的取值种类数不能超过256，否则会出错。

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
```

```
    [2, 1, 0, 0],
    [3, 2, 0, 2],
    [4, 3, 0, 1],
    [2, 4, 0, 0],
    [2, 2, 1, 0],
    [4, 3, 1, 2],
    [1, 2, 1, 1],
    [5, 3, 1, 0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')

trainOp = LambdaMartNdcgBatchOp()\
    .setFeatureCols(['f0', 'f1'])\
    .setLabelCol("label")\
    .setGroupCol("group")\
    .setNumTrees(3)\
    .linkFrom(batchSource)

predictBatchOp = LambdaMartNdcgPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = LambdaMartNdcgPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

运行结果

批预测结果

```
    f0   f1  group  label      pred
0  2.0  1.0      0    0.0 -0.964759
1  3.0  2.0      0    2.0  0.619588
2  4.0  3.0      0    1.0  0.619588
3  2.0  4.0      0    0.0 -0.964759
4  2.0  2.0      1    0.0 -0.964759
5  4.0  3.0      1    2.0  0.619588
6  1.0  2.0      1    1.0 -0.964759
7  5.0  3.0      1    0.0  0.619588
```

流预测结果

```
f0    f1    group    label    pred
0    2.0    1.0    0    0.0    -0.964759
1    3.0    2.0    0    2.0    0.619588
2    4.0    3.0    0    1.0    0.619588
3    2.0    4.0    0    0.0    -0.964759
4    2.0    2.0    1    0.0    -0.964759
5    4.0    3.0    1    2.0    0.619588
6    1.0    2.0    1    1.0    -0.964759
7    5.0    3.0    1    0.0    0.619588
```

LambdaMart NDCG预测 (LambdaMartNdcgPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LambdaMartNdcgPredictBatchOp

Python 类名：LambdaMartNdcgPredictBatchOp

功能介绍

- LambdaMART，是经典的基于gbdt的learning to rank模型。模型细节可参考 Burges, Christopher JC. "From ranknet to lambdarank to lambdamart: An overview." Learning 11.23-581 (2010): 81.
- 支持连续特征和离散特征
- 支持数据采样和特征采样
- 需要指定分组列，模型会在每一组内根据label列拟合排序结果。和原始论文一样，label列表示“相关度”，越高越好，在计算得分时，会使用2^label -1 来计算。例如可以使用0,1,2,3,....来依次表示很不相关、有点相关、比较相关、相关等等

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

参数建议

对于训练效果来说，比较重要的参数是 树的棵树+学习率、叶子节点最小样本数、单颗树最大深度、特征采样比例。

单个离散特征的取值种类数不能超过256，否则会出错。

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 0, 0],
    [3, 2, 0, 2],
    [4, 3, 0, 1],
    [2, 4, 0, 0],
    [2, 2, 1, 0],
    [4, 3, 1, 2],
    [1, 2, 1, 1],
    [5, 3, 1, 0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 double, group long, label double')

trainOp = LambdaMartNdcgBatchOp()\
    .setFeatureCols(['f0', 'f1'])\
    .setLabelCol("label")\
    .setGroupCol("group")\
    .setNumTrees(3)\
    .linkFrom(batchSource)

predictBatchOp = LambdaMartNdcgPredictBatchOp()\
    .setPredictionCol('pred')
predictStreamOp = LambdaMartNdcgPredictStreamOp(trainOp)\
    .setPredictionCol('pred')

predictBatchOp.linkFrom(trainOp, batchSource).print()
predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

运行结果

```
    f0   f1  group  label      pred
0  2.0  1.0      0    0.0 -0.964759
1  3.0  2.0      0    2.0  0.619588
2  4.0  3.0      0    1.0  0.619588
3  2.0  4.0      0    0.0 -0.964759
4  2.0  2.0      1    0.0 -0.964759
5  4.0  3.0      1    2.0  0.619588
6  1.0  2.0      1    1.0 -0.964759
7  5.0  3.0      1    0.0  0.619588
```

Lasso回归预测 (LassoRegPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LassoRegPredictBatchOp

Python 类名：LassoRegPredictBatchOp

功能介绍

Lasso回归算法是由1996年Robert Tibshirani首次提出。是一种经典的回归算法。Lasso回归组件支持稀疏、稠密两种数据格式，并且支持带权重样本训练。

算法原理

Lasso回归算法通过构造一个惩罚函数得到一个较为精炼的模型，使得它压缩一些回归系数，即强制系数绝对值之和小于某个固定值；同时设定一些回归系数为零。因此保留了子集收缩的优点，是一种处理具有复共线性数据的有偏估计。

算法使用

Lasso回归模型应用领域和线性回归类似，经常被用来做一些数值型变量的预测，类似房价预测、销售量预测、贷款额度预测、温度预测、适度预测等。

文献或出处

[1] Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society: Series B (Methodological) 58.1 (1996): 267-288.

[2] https://baike.baidu.com/item/LASSO/20366865?fr=aladdin

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null

vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]
])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')

lasso = LassoRegTrainBatchOp()\
        .setLambda(0.1)\
        .setFeatureCols(["f0","f1"])\
        .setLabelCol("label")

model = batchData.link(lasso)

predictor = LassoRegPredictBatchOp()\
        .setPredictionCol("pred")

predictor.linkFrom(model, batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.LassoRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.LassoRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LassoRegPredictBatchOpTest {
    @Test
    public void testLassoRegPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 3)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        BatchOperator <?> lasso = new LassoRegTrainBatchOp()
            .setLambda(0.1)
            .setFeatureCols("f0", "f1")
            .setLabelCol("label");
        BatchOperator model = batchData.link(lasso);
        BatchOperator <?> predictor = new LassoRegPredictBatchOp()
            .setPredictionCol("pred");
        predictor.linkFrom(model, batchData).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	0.830304
3	2	1	1.377312
4	3	2	1.924320
2	4	1	1.159119

2	2	1	0.939909
4	3	2	1.924320
1	2	1	0.502506
5	3	3	2.361724

Lasso回归训练 (LassoRegTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LassoRegTrainBatchOp

Python 类名：LassoRegTrainBatchOp

功能介绍

Lasso回归算法是由1996年Robert Tibshirani首次提出。是一种经典的回归算法。Lasso回归组件支持稀疏、稠密两种数据格式，并且支持带权重样本训练。

算法原理

Lasso回归算法通过构造一个惩罚函数得到一个较为精炼的模型，使得它压缩一些回归系数，即强制系数绝对值之和小于某个固定值；同时设定一些回归系数为零。因此保留了子集收缩的优点，是一种处理具有复共线性数据的有偏估计。

算法使用

Lasso回归模型应用领域和线性回归类似，经常被用来做一些数值型变量的预测，类似房价预测、销售量预测、贷款额度预测、温度预测、适度预测等。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society: Series B (Methodological) 58.1 (1996): 267-288.

[2] https://baike.baidu.com/item/LASSO/20366865?fr=aladdin

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
lambda	希腊字母：lambda	惩罚因子，必选	Double	✓		

epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为1.0e-6	Double		[0.0, +inf)	1.0E-6
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
maxIter	最大迭代步数	最大迭代步数，默认为 100	Integer		[1, +inf)	100
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"	null
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean			true
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]
])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')

lasso = LassoRegTrainBatchOp()\
        .setLambda(0.1)\
        .setFeatureCols(["f0","f1"])\
        .setLabelCol("label")

model = batchData.link(lasso)

predictor = LassoRegPredictBatchOp()\
        .setPredictionCol("pred")

predictor.linkFrom(model, batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.LassoRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.LassoRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LassoRegTrainBatchOpTest {
    @Test
    public void testLassoRegTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
```

```
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 3)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        BatchOperator <?> lasso = new LassoRegTrainBatchOp()
            .setLambda(0.1)
            .setFeatureCols("f0", "f1")
            .setLabelCol("label");
        BatchOperator model = batchData.link(lasso);
        BatchOperator <?> predictor = new LassoRegPredictBatchOp()
            .setPredictionCol("pred");
        predictor.linkFrom(model, batchData).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	0.830304
3	2	1	1.377312
4	3	2	1.924320
2	4	1	1.159119
2	2	1	0.939909
4	3	2	1.924320
1	2	1	0.502506
5	3	3	2.361724

线性回归预测 (LinearRegPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LinearRegPredictBatchOp

Python 类名：LinearRegPredictBatchOp

功能介绍

线性回归算法是经典的回归算法，通过对带有回归值的样本集合训练得到回归模型，使用模型预测样本的回归值。线性回归组件支持稀疏、稠密两种数据格式，并且支持带权重样本训练。

算法原理

面对回归类问题，线性回归利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。

算法使用

线性回归模型经常被用来做一些数值型变量的预测，类似房价预测、销售量预测、贷款额度预测、温度预测、适度预测等。

文献或出处

[1] Seber, George AF, and Alan J. Lee. Linear regression analysis. John Wiley & Sons, 2012.

[2] https://baike.baidu.com/item/%E7%BA%BF%E6%80%A7%E5%9B%9E%E5%BD%92/8190345?fr=aladdin

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]
])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')

lr = LinearRegTrainBatchOp()\
        .setFeatureCols(["f0","f1"])\
        .setLabelCol("label")

model = batchData.link(lr)

predictor = LinearRegPredictBatchOp()\
    .setPredictionCol("pred")

predictor.linkFrom(model, batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.LinearRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.LinearRegTrainBatchOp;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LinearRegTrainBatchOpTest {
    @Test
    public void testLinearRegTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 3)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        BatchOperator <?> lr = new LinearRegTrainBatchOp()
            .setFeatureCols("f0", "f1")
            .setLabelCol("label");
        BatchOperator model = batchData.link(lr);
        BatchOperator <?> predictor = new LinearRegPredictBatchOp()
            .setPredictionCol("pred");
        predictor.linkFrom(model, batchData).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1.000014
3	2	1	1.538474
4	3	2	2.076934
2	4	1	1.138446
2	2	1	1.046158
4	3	2	2.076934
1	2	1	0.553842
5	3	3	2.569250

线性回归Stepwise预测 (LinearRegStepwisePredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LinearRegStepwisePredictBatchOp

Python 类名：LinearRegStepwisePredictBatchOp

功能介绍

- Stepwise回归是一个回归算法
- Stepwise回归组件仅支持稠密数据格式

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [16.3, 1.1, 1.1],
    [16.8, 1.4, 1.5],
    [19.2, 1.7, 1.8],
    [18.0, 1.7, 1.7],
    [19.5, 1.8, 1.9],
    [20.9, 1.8, 1.8],
    [21.1, 1.9, 1.8],
```

```
    [20.9, 2.0, 2.1],
    [20.3, 2.3, 2.4],
    [22.0, 2.4, 2.5]
])

batchData = BatchOperator.fromDataframe(df, schemaStr='y double, x1 double, x2
double')

lrs = LinearRegStepwiseTrainBatchOp()\
    .setFeatureCols(["x1", "x2"])\
    .setLabelCol("y")\
    .setMethod("Forward")

model = batchData.link(lrs)

predictor = LinearRegStepwisePredictBatchOp()\
    .setPredictionCol("pred")

predictor.linkFrom(model, batchData).print()
```

运行结果

y	x1	x2	pred
16.3	1.1	1.1	16.380060195635785
16.8	1.4	1.5	17.698344620015032
19.2	1.7	1.8	19.01662904439428
18.0	1.7	1.7	19.01662904439428
19.5	1.8	1.9	19.456057185854025
20.9	1.8	1.8	19.456057185854025
21.1	1.9	1.8	19.89548532731377
20.9	2.0	2.1	20.33491346877352
20.3	2.3	2.4	21.653197893152765
22.0	2.4	2.5	22.092626034612515

线性回归Stepwise训练 (LinearRegStepwiseTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LinearRegStepwiseTrainBatchOp

Python 类名：LinearRegStepwiseTrainBatchOp

功能介绍

- Stepwise回归是一个回归算法
- Stepwise回归组件仅支持稠密数据格式

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
labelCol	标签列名	输入表中的标签列名	String	✓		
method	回归统计的方法	可取值包括：stepwise，forward，backward	String		"Stepwise", "Forward", "Backward"	"Forward"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [16.3, 1.1, 1.1],
    [16.8, 1.4, 1.5],
    [19.2, 1.7, 1.8],
    [18.0, 1.7, 1.7],
    [19.5, 1.8, 1.9],
    [20.9, 1.8, 1.8],
    [21.1, 1.9, 1.8],
    [20.9, 2.0, 2.1],
    [20.3, 2.3, 2.4],
    [22.0, 2.4, 2.5]
])

batchData = BatchOperator.fromDataframe(df, schemaStr='y double, x1 double, x2
double')

lrs = LinearRegStepwiseTrainBatchOp()\
    .setFeatureCols(["x1", "x2"])\
    .setLabelCol("y")\
    .setMethod("Forward")

model = batchData.link(lrs)

predictor = LinearRegStepwisePredictBatchOp()\
    .setPredictionCol("pred")

predictor.linkFrom(model, batchData).print()
```

运行结果

y	x1	x2	pred
16.3	1.1	1.1	16.380060195635785
16.8	1.4	1.5	17.698344620015032
19.2	1.7	1.8	19.01662904439428
18.0	1.7	1.7	19.01662904439428
19.5	1.8	1.9	19.456057185854025
20.9	1.8	1.8	19.456057185854025

21.1	1.9	1.8	19.89548532731377
20.9	2.0	2.1	20.33491346877352
20.3	2.3	2.4	21.653197893152765
22.0	2.4	2.5	22.092626034612515

线性回归训练 (LinearRegTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LinearRegTrainBatchOp

Python 类名：LinearRegTrainBatchOp

功能介绍

线性回归算法是经典的回归算法，通过对带有回归值的样本集合训练得到回归模型，使用模型预测样本的回归值。线性回归组件支持稀疏、稠密两种数据格式，并且支持带权重样本训练。

算法原理

面对回归类问题，线性回归利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。

算法使用

线性回归模型经常被用来做一些数值型变量的预测，类似房价预测、销售量预测、贷款额度预测、温度预测、适度预测等。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Seber, George AF, and Alan J. Lee. Linear regression analysis. John Wiley & Sons, 2012.

[2] https://baike.baidu.com/item/%E7%BA%BF%E6%80%A7%E5%9B%9E%E5%BD%92/8190345?fr=aladdin

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1.0E-6

featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"	null
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean			true
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]
])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')

lr = LinearRegTrainBatchOp()\
        .setFeatureCols(["f0","f1"])\
        .setLabelCol("label")

model = batchData.link(lr)

predictor = LinearRegPredictBatchOp()\
    .setPredictionCol("pred")

predictor.linkFrom(model, batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;
```

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.LinearRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.LinearRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LinearRegTrainBatchOpTest {
    @Test
    public void testLinearRegTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 3)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        BatchOperator <?> lr = new LinearRegTrainBatchOp()
            .setFeatureCols("f0", "f1")
            .setLabelCol("label");
        BatchOperator model = batchData.link(lr);
        BatchOperator <?> predictor = new LinearRegPredictBatchOp()
            .setPredictionCol("pred");
        predictor.linkFrom(model, batchData).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1.000014
3	2	1	1.538474
4	3	2	2.076934
2	4	1	1.138446
2	2	1	1.046158
4	3	2	2.076934

1	2	1	0.553842
5	3	3	2.569250

线性SVR预测 (LinearSvrPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LinearSvrPredictBatchOp

Python 类名：LinearSvrPredictBatchOp

功能介绍

- 线性SVR是一个回归算法
- 线性SVR组件支持稀疏、稠密两种数据格式
- 线性SVR组件支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [16.3, 1.1, 1.1],
    [16.8, 1.4, 1.5],
    [19.2, 1.7, 1.8],
    [18.0, 1.7, 1.7],
    [19.5, 1.8, 1.9],
    [20.9, 1.8, 1.8],
    [21.1, 1.9, 1.8],
    [20.9, 2.0, 2.1],
    [20.3, 2.3, 2.4],
    [22.0, 2.4, 2.5]
])

batchSource = BatchOperator.fromDataframe(df, schemaStr=' y double, x1 double,
x2 double')

lsvr = LinearSvrTrainBatchOp()\
    .setFeatureCols(["x1", "x2"])\
    .setLabelCol("y")\
    .setC(1.0)\
    .setTau(0.01)

model = batchSource.link(lsvr)

predictor = LinearSvrPredictBatchOp()\
    .setPredictionCol("pred")

predictor.linkFrom(model, batchSource).print()
```

运行结果

y	x1	x2	pred
16.3	1.1	1.1	16.48073043727051
16.8	1.4	1.5	17.236649847389877
19.2	1.7	1.8	18.651637270539197
18.0	1.7	1.7	19.31070528356914
19.5	1.8	1.9	19.123299744922306
20.9	1.8	1.8	19.78236775795225
21.1	1.9	1.8	20.913098245365298
20.9	2.0	2.1	20.066624693688514

20.3	2.3	2.4	21.481612116837834
22.0	2.4	2.5	21.953274591220936

线性SVR训练 (LinearSvrTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.LinearSvrTrainBatchOp

Python 类名：LinearSvrTrainBatchOp

功能介绍

- 线性SVR是一个回归算法
- 线性SVR组件支持稀疏、稠密两种数据格式
- 线性SVR组件支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
C	算法参数	支撑向量回归参数	Double	✓		
labelCol	标签列名	输入表中的标签列名	String	✓		
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1.0E-6
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"	null
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean			true
tau	算法参数	支撑向量回归参数	Double			0.1
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [16.3, 1.1, 1.1],
    [16.8, 1.4, 1.5],
    [19.2, 1.7, 1.8],
    [18.0, 1.7, 1.7],
    [19.5, 1.8, 1.9],
    [20.9, 1.8, 1.8],
    [21.1, 1.9, 1.8],
    [20.9, 2.0, 2.1],
    [20.3, 2.3, 2.4],
    [22.0, 2.4, 2.5]
])

batchSource = BatchOperator.fromDataframe(df, schemaStr=' y double, x1 double,
x2 double')

lsvr = LinearSvrTrainBatchOp()\
    .setFeatureCols(["x1", "x2"])\
    .setLabelCol("y")\
    .setC(1.0)\
    .setTau(0.01)

model = batchSource.link(lsvr)

predictor = LinearSvrPredictBatchOp()\
    .setPredictionCol("pred")

predictor.linkFrom(model, batchSource).print()
```

运行结果

y	x1	x2	pred
16.3	1.1	1.1	16.48073043727051
16.8	1.4	1.5	17.236649847389877
19.2	1.7	1.8	18.651637270539197
18.0	1.7	1.7	19.31070528356914
19.5	1.8	1.9	19.123299744922306
20.9	1.8	1.8	19.78236775795225
21.1	1.9	1.8	20.913098245365298

20.9	2.0	2.1	20.066624693688514
20.3	2.3	2.4	21.481612116837834
22.0	2.4	2.5	21.953274591220936

随机森林回归预测 (RandomForestRegPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.RandomForestRegPredictBatchOp

Python 类名：RandomForestRegPredictBatchOp

功能介绍

- 随机森林回归是一种常用的树模型，由于bagging的过程，可以避免过拟合
- 随机森林回归组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        [1.0, "A", 0, 0, 0],
        [2.0, "B", 1, 1, 0],
        [3.0, "C", 2, 2, 1],
        [4.0, "D", 3, 3, 1]
    ])
```

```
batchSource = BatchOperator.fromDataframe(df, schemaStr=' f0 double, f1 string,
f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(df, schemaStr=' f0 double, f1
string, f2 int, f3 int, label int')

trainOp = RandomForestRegTrainBatchOp()\
            .setLabelCol('label')\
            .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
            .linkFrom(batchSource)

RandomForestRegPredictBatchOp()\
     .setPredictionCol('pred')\
     .linkFrom(trainOp, batchSource).print()

RandomForestRegPredictStreamOp(trainOp)\
    .setPredictionCol('pred')\
    .linkFrom(streamSource)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.regression.RandomForestRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.RandomForestRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.regression.RandomForestRegPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RandomForestRegPredictBatchOpTest {
    @Test
    public void testRandomForestRegPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
```

```
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, " f0
double, f1 string, f2 int, f3 int, label "
            + "int");
        BatchOperator <?> trainOp = new RandomForestRegTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        new RandomForestRegPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(trainOp, batchSource).print();
        new RandomForestRegPredictStreamOp(trainOp)
            .setPredictionCol("pred")
            .linkFrom(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
4.0000	D	3	3	1	1.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000

随机森林回归训练 (RandomForestRegTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.RandomForestRegTrainBatchOp

Python 类名：RandomForestRegTrainBatchOp

功能介绍

- 随机森林回归是一种常用的树模型，由于bagging的过程，可以避免过拟合
- 随机森林回归组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	
labelCol	标签列名	输入表中的标签列名	String	✓		
categoricalCols	离散特征列名	离散特征列名	String[]		所选列类型为[BOOLEAN, DATE, DOUBLE, FLOAT, INTEGER, LONG, SHORT, STRING, TIME, TIMESTAMP]	

createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String			"series"
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer			128
maxDepth	树的深度限制	树的深度限制	Integer			2147483647
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer			2147483647
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer			64
minInfoGain	分裂的最小增益	分裂的最小增益	Double			0.0

minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double			0.0
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer			2
numSubsetFeatures	每棵树的特征采样数目	每棵树的特征采样数目	Integer			2147483647
numTrees	模型中树的棵数	模型中树的棵数	Integer		[1, +inf)	10
seed	采样种子	采样种子	Long			0
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double			100000.0
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        [1.0, "A", 0, 0, 0],
        [2.0, "B", 1, 1, 0],
        [3.0, "C", 2, 2, 1],
        [4.0, "D", 3, 3, 1]
    ])

batchSource = BatchOperator.fromDataframe(df, schemaStr=' f0 double, f1 string,
f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(df, schemaStr=' f0 double, f1
string, f2 int, f3 int, label int')

trainOp = RandomForestRegTrainBatchOp()\
            .setLabelCol('label')\
            .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
            .linkFrom(batchSource)

RandomForestRegPredictBatchOp()\
     .setPredictionCol('pred')\
     .linkFrom(trainOp, batchSource).print()

RandomForestRegPredictStreamOp(trainOp)\
    .setPredictionCol('pred')\
    .linkFrom(streamSource)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.regression.RandomForestRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.RandomForestRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
```

```
com.alibaba.alink.operator.stream.regression.RandomForestRegPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RandomForestRegTrainBatchOpTest {
    @Test
    public void testRandomForestRegTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, " f0
double, f1 string, f2 int, f3 int, label "
            + "int");
        BatchOperator <?> trainOp = new RandomForestRegTrainBatchOp()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .linkFrom(batchSource);
        new RandomForestRegPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(trainOp, batchSource).print();
        new RandomForestRegPredictStreamOp(trainOp)
            .setPredictionCol("pred")
            .linkFrom(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
4.0000	D	3	3	1	1.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000

岭回归预测 (RidgeRegPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.RidgeRegPredictBatchOp

Python 类名：RidgeRegPredictBatchOp

功能介绍

岭回归(Ridge regression)算法是一种经典的回归算法。岭回归组件支持稀疏、稠密两种数据格式，并且支持带权重样本训练。

算法原理

岭回归是一种专用于共线性数据分析的有偏估计回归方法，实质上是一种改良的最小二乘估计法，通过放弃最小二乘法的无偏性，以损失部分信息、降低精度为代价获得回归系数更为符合实际、更可靠的回归方法，对病态数据的拟合要强于最小二乘法。

算法使用

岭回归模型应用领域和线性回归类似，经常被用来做一些数值型变量的预测，类似房价预测、销售量预测、贷款额度预测、温度预测、适度预测等。

文献或出处

[1] Hoerl, Arthur E., and Robert W. Kennard. "Ridge regression: Biased estimation for nonorthogonal problems." Technometrics 12.1 (1970): 55-67.

[2] https://baike.baidu.com/item/%E5%B2%AD%E5%9B%9E%E5%BD%92/554917?fr=aladdin

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null

vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')
ridge = RidgeRegTrainBatchOp()\
    .setLambda(0.1)\
    .setFeatureCols(["f0","f1"])\
    .setLabelCol("label")
model = batchData.link(ridge)

predictor = RidgeRegPredictBatchOp()\
    .setPredictionCol("pred")
predictor.linkFrom(model, batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
```

```
import com.alibaba.alink.operator.batch.regression.RidgeRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.RidgeRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RidgeRegPredictBatchOpTest {
    @Test
    public void testRidgeRegPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        BatchOperator <?> ridge = new RidgeRegTrainBatchOp()
            .setLambda(0.1)
            .setFeatureCols("f0", "f1")
            .setLabelCol("label");
        BatchOperator model = batchData.link(ridge);
        BatchOperator <?> predictor = new RidgeRegPredictBatchOp()
            .setPredictionCol("pred");
        predictor.linkFrom(model, batchData).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	0.830304
3	2	1	1.377312
4	3	2	1.924320
2	4	1	1.159119
2	2	1	0.939909
4	3	2	1.924320

1	2	1	0.502506
5	3	3	2.361724

岭回归训练 (RidgeRegTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.RidgeRegTrainBatchOp

Python 类名：RidgeRegTrainBatchOp

功能介绍

岭回归(Ridge regression)算法是一种经典的回归算法。岭回归组件支持稀疏、稠密两种数据格式，并且支持带权重样本训练。

算法原理

岭回归是一种专用于共线性数据分析的有偏估计回归方法，实质上是一种改良的最小二乘估计法，通过放弃最小二乘法的无偏性，以损失部分信息、降低精度为代价获得回归系数更为符合实际、更可靠的回归方法，对病态数据的拟合要强于最小二乘法。

算法使用

岭回归模型应用领域和线性回归类似，经常被用来做一些数值型变量的预测，类似房价预测、销售量预测、贷款额度预测、温度预测、适度预测等。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Hoerl, Arthur E., and Robert W. Kennard. "Ridge regression: Biased estimation for nonorthogonal problems." Technometrics 12.1 (1970): 55-67.

[2] https://baike.baidu.com/item/%E5%B2%AD%E5%9B%9E%E5%BD%92/554917?fr=aladdin

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
lambda	希腊字母：lambda	惩罚因子，必选	Double	✓		

epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为1.0e-6	Double		[0.0, +inf)	1.0E-6
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
maxIter	最大迭代步数	最大迭代步数，默认为 100	Integer		[1, +inf)	100
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"	null
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean			true
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')
ridge = RidgeRegTrainBatchOp()\
    .setLambda(0.1)\
    .setFeatureCols(["f0","f1"])\
    .setLabelCol("label")
model = batchData.link(ridge)

predictor = RidgeRegPredictBatchOp()\
    .setPredictionCol("pred")
predictor.linkFrom(model, batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.RidgeRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.RidgeRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RidgeRegTrainBatchOpTest {
    @Test
    public void testRidgeRegTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
```

```
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        BatchOperator <?> ridge = new RidgeRegTrainBatchOp()
            .setLambda(0.1)
            .setFeatureCols("f0", "f1")
            .setLabelCol("label");
        BatchOperator model = batchData.link(ridge);
        BatchOperator <?> predictor = new RidgeRegPredictBatchOp()
            .setPredictionCol("pred");
        predictor.linkFrom(model, batchData).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	0.830304
3	2	1	1.377312
4	3	2	1.924320
2	4	1	1.159119
2	2	1	0.939909
4	3	2	1.924320
1	2	1	0.502506
5	3	3	2.361724

XGBoost 回归预测 (XGBoostRegPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.XGBoostRegPredictBatchOp

Python 类名：XGBoostRegPredictBatchOp

功能介绍

XGBoost 组件是在开源社区的基础上进行包装，使功能和 PAI 更兼容，更易用。 XGBoost 算法在 Boosting 算法的基础上进行了扩展和升级，具有较好的易用性和鲁棒性，被广泛用在各种机器学习生产系统和竞赛领域。 当前支持分类，回归和排序。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
pluginVersion	插件版本号	插件版本号	String			"1.5.1"
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    [0, 1, 1.1, 1.0],
    [1, -2, 0.9, 2.0],
    [0, 100, -0.01, 3.0],
    [1, -99, 0.1, 4.0],
    [0, 1, 1.1, 5.0],
    [1, -2, 0.9, 6.0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='y int, x1 double, x2 double, x3 double'
```

```
)

streamSource = StreamOperator.fromDataframe(
    df, schemaStr='y int, x1 double, x2 double, x3 double'
)

trainOp = XGBoostRegTrainBatchOp()\
    .setNumRound(1)\
    .setPluginVersion('1.5.1')\
    .setLabelCol('y')\
    .linkFrom(batchSource)

predictBatchOp = XGBoostRegPredictBatchOp()\
    .setPredictionCol('pred')\
    .setPluginVersion('1.5.1')

predictStreamOp = XGBoostRegPredictStreamOp(trainOp)\
    .setPredictionCol('pred')\
    .setPluginVersion('1.5.1')

predictBatchOp.linkFrom(trainOp, batchSource).print()

predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.XGBoostRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.XGBoostRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.regression.XGBoostRegPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class XGBoostRegTrainBatchOpTest {

    @Test
    public void testXGBoostTrainBatchOp() throws Exception {
        List <Row> data = Arrays.asList(
```

```
            Row.of(0, 1, 1.1, 1.0),
            Row.of(1, -2, 0.9, 2.0),
            Row.of(0, 100, -0.01, 3.0),
            Row.of(1, -99, 0.1, 4.0),
            Row.of(0, 1, 1.1, 5.0),
            Row.of(1, -2, 0.9, 6.0)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(data, "y int, x1
int, x2 double, x3 double");
        StreamOperator <?> streamSource = new MemSourceStreamOp(data, "y int,
x1 int, x2 double, x3 double");
        BatchOperator <?> trainOp = new XGBoostRegTrainBatchOp()
            .setNumRound(1)
            .setPluginVersion("1.5.1")
            .setLabelCol("y")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new XGBoostRegPredictBatchOp()
            .setPredictionCol("pred")
            .setPluginVersion("1.5.1");
        StreamOperator <?> predictStreamOp = new
XGBoostRegPredictStreamOp(trainOp)
            .setPredictionCol("pred")
            .setPluginVersion("1.5.1");

        predictBatchOp.linkFrom(trainOp, batchSource).print();

        predictStreamOp.linkFrom(streamSource).print();

        StreamOperator.execute();
    }
}
```

运行结果

XGBoost 回归训练 (XGBoostRegTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.regression.XGBoostRegTrainBatchOp

Python 类名：XGBoostRegTrainBatchOp

功能介绍

XGBoost 组件是在开源社区的基础上进行包装，使功能和 PAI 更兼容，更易用。 XGBoost 算法在 Boosting 算法的基础上进行了扩展和升级，具有较好的易用性和鲁棒性，被广泛用在各种机器学习生产系统和竞赛领域。 当前支持分类，回归和排序。

参数说明

名称	中文名称	描述	类型	是否必须?	取
labelCol	标签列名	输入表中的标签列名	String	✓	
numRound	树的棵树	树的棵树	Integer	✓	
alpha	L1 正则项	L1 正则项	Double		
baseScore	Base score	Base score	Double		
colSampleByLevel	每个树列采样	每个树列采样	Double		
colSampleByNode	每个结点列采样	每个结点采样	Double		
colSampleByTree	每个树列采样	每个树列采样	Double		
eta	学习率	学习率	Double		
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为 BIGINTEGER FLOAT, INTE SHORT]
gamma	结点分裂最小损失变化	节点分裂最小损失变化	Double		
growPolicy	GrowPolicy	GrowPolicy	String		"DEPTH_WIS
interactionConstraints	interaction constraints	interaction constraints	String		
lambda	L2 正则项	L2 正则项	Double		

maxBin	最大结点个数	最大结点个数	Integer		
maxDeltaStep	Delta step	Delta step	Double		
maxDepth	最大深度	最大深度	Integer		
maxLeaves	最大结点个数	最大结点个数	Integer		
minChildWeight	结点的最小权重	结点的最小权重	Double		
monotoneConstraints	monotone constraints	monotone constraints	String		
numClass	标签类别个数	标签类别个数，多分类时有效	Integer		
objective	objective	objective	String		"REG_SQUA "REG_SQUA "REG_LOGIS "REG_PSEUI "REG_GAMM "REG_TWEE
pluginVersion	插件版本号	插件版本号	String		
processType	ProcessType	ProcessType	String		"DEFAULT", "
refreshLeaf	RefreshLeaf	RefreshLeaf	Integer		
runningMode	运行模式	XGBoost的运行模型，ICQ速度快，但使用内存多，TRIAVIAL速度略慢，但是节省内存，按照流式方式处理。由于训练数据本身在XGBoost运行时已经被缓存进内存，所以存两份和存一份数据的资源消耗和速度对比，还需要进一步的测试。	String		"ICQ", "TRIVI
samplingMethod	采样方法	采样方法	String		"UNIFORM", "GRADIENT_
scalePosWeight	ScalePosWeight	ScalePosWeight	Double		

singlePrecisionHistogram	single precision histogram	single precision histogram	Boolean		
sketchEps	SketchEps	SketchEps	Double		
subSample	样本采样比例	样本采样比例	Double		
treeMethod	构建树的方法	构建树的方法	String		"AUTO", "EXA "HIST"
tweedieVariancePower	学习率	学习率	Double		
updater	Updater	Updater	String		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为SPARSE_VE VECTOR]

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    [0, 1, 1.1, 1.0],
    [1, -2, 0.9, 2.0],
    [0, 100, -0.01, 3.0],
    [1, -99, 0.1, 4.0],
    [0, 1, 1.1, 5.0],
    [1, -2, 0.9, 6.0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='y int, x1 double, x2 double, x3 double'
)

streamSource = StreamOperator.fromDataframe(
    df, schemaStr='y int, x1 double, x2 double, x3 double'
)

trainOp = XGBoostRegTrainBatchOp()\
    .setNumRound(1)\
    .setPluginVersion('1.5.1')\
    .setLabelCol('y')\
    .linkFrom(batchSource)

predictBatchOp = XGBoostRegPredictBatchOp()\
```

```
    .setPredictionCol('pred')\
    .setPluginVersion('1.5.1')

predictStreamOp = XGBoostRegPredictStreamOp(trainOp)\
    .setPredictionCol('pred')\
    .setPluginVersion('1.5.1')

predictBatchOp.linkFrom(trainOp, batchSource).print()

predictStreamOp.linkFrom(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.regression.XGBoostRegPredictBatchOp;
import com.alibaba.alink.operator.batch.regression.XGBoostRegTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.regression.XGBoostRegPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class XGBoostRegTrainBatchOpTest {

    @Test
    public void testXGBoostTrainBatchOp() throws Exception {
        List <Row> data = Arrays.asList(
            Row.of(0, 1, 1.1, 1.0),
            Row.of(1, -2, 0.9, 2.0),
            Row.of(0, 100, -0.01, 3.0),
            Row.of(1, -99, 0.1, 4.0),
            Row.of(0, 1, 1.1, 5.0),
            Row.of(1, -2, 0.9, 6.0)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(data, "y int, x1
int, x2 double, x3 double");
        StreamOperator <?> streamSource = new MemSourceStreamOp(data, "y int,
x1 int, x2 double, x3 double");
        BatchOperator <?> trainOp = new XGBoostRegTrainBatchOp()
```

```
            .setNumRound(1)
            .setPluginVersion("1.5.1")
            .setLabelCol("y")
            .linkFrom(batchSource);
        BatchOperator <?> predictBatchOp = new XGBoostRegPredictBatchOp()
            .setPredictionCol("pred")
            .setPluginVersion("1.5.1");
        StreamOperator <?> predictStreamOp = new
XGBoostRegPredictStreamOp(trainOp)
            .setPredictionCol("pred")
            .setPluginVersion("1.5.1");

        predictBatchOp.linkFrom(trainOp, batchSource).print();

        predictStreamOp.linkFrom(streamSource).print();

        StreamOperator.execute();
    }
}
```

运行结果

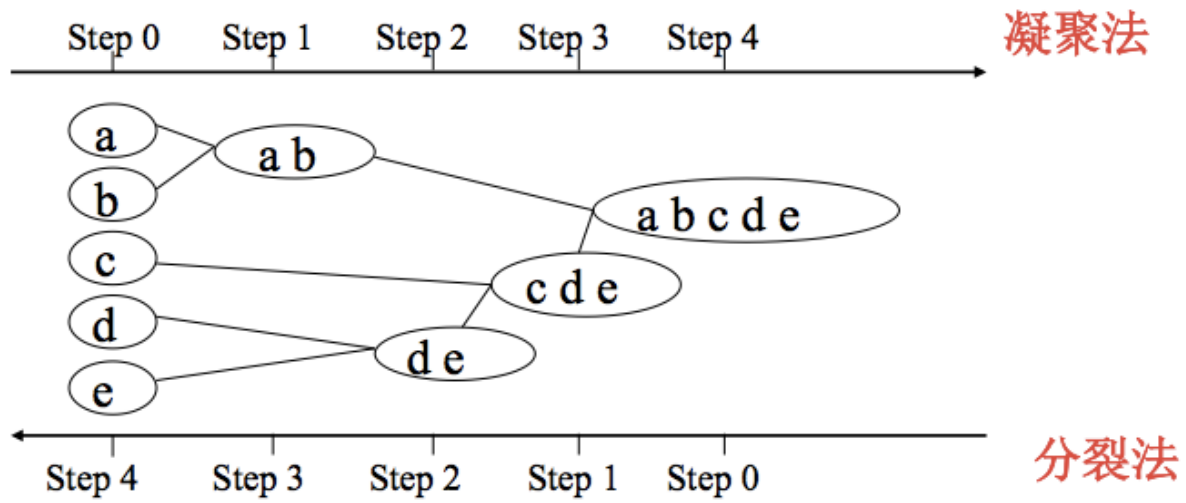
Agnes (AgnesBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.AgnesBatchOp

Python 类名：AgnesBatchOp

功能介绍

AGNES，AGglomerative NESting，是一个比较有代表性的 凝聚 的层次聚类。最开始的时候将所有数据点本身作为簇，然后找出距离最近的两个簇将它们合为一个，不断重复以上步骤直到达到预设的簇的个数或者簇之间距离大于一个距离阈值。



距离度量方式

参数名称	参数描述	说明
EUCLIDEAN		欧式距离
COSINE		夹角余弦距离
CITYBLOCK		城市街区距离，也称曼哈顿距离

时间复杂度

假定在开始的时候有N个簇，在结束的时候有K个簇，因此在主循环中有(N-K)次迭代，在第i次迭代中，我们必须在n−i+1个簇中找到最靠近的两个进行合并。另外算法必须计算所有对象两两之间的距离，如果使用簇间距离度量方式为COSIN，那这个算法的复杂度为 O(N^2 *(N-K)* D) ，该算法对于较大的情况是不适用的。此外，目前AGNES中只支持串行执行。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
idCol	id列名	id列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
distanceThreshold	距离阈值	距离阈值	Double			1.7976931348623157E3
distanceType	距离度量方式	距离类型	String		"EUCLIDEAN", "COSINE", "CITYBLOCK"	"EUCLIDEAN"
k	聚类中心点数量	聚类中心点数量	Integer			2

linkage	类的聚合方式	类的聚合方式	String		"MIN", "MAX", "MEAN", "AVERAGE"	"MIN"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ["id_1", "2.0,3.0"],
        ["id_2", "2.1,3.1"],
        ["id_3", "200.1,300.1"],
        ["id_4", "200.2,300.2"],
        ["id_5", "200.3,300.3"],
        ["id_6", "200.4,300.4"],
        ["id_7", "200.5,300.5"],
        ["id_8", "200.6,300.6"],
        ["id_9", "2.1,3.1"],
        ["id_10", "2.1,3.1"],
        ["id_11", "2.1,3.1"],
        ["id_12", "2.1,3.1"],
        ["id_16", "300.,3.2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id string, vec string')

agnes = AgnesBatchOp()\
    .setIdCol("id")\
    .setVectorCol("vec")\
    .setPredictionCol("pred")\
    .linkFrom(inOp)

agnes.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.clustering.AgnesBatchOp;

import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AgnesBatchOpTest {

    @Test
    public void testAgnesBatchOp() throws Exception {
        List<Row> trainData = Arrays.asList(
            Row.of("id_1", "2.0,3.0"),
            Row.of("id_2", "2.1,3.1"),
               Row.of("id_3", "200.1,300.1"),
            Row.of("id_4", "200.2,300.2"),
            Row.of("id_5", "200.3,300.3"),
            Row.of("id_6", "200.4,300.4"),
            Row.of("id_7", "200.5,300.5"),
            Row.of("id_8", "200.6,300.6"),
            Row.of("id_9", "2.1,3.1"),
            Row.of("id_10", "2.1,3.1"),
            Row.of("id_11", "2.1,3.1"),
            Row.of("id_12", "2.1,3.1"),
            Row.of("id_16", "300.,3.2")
        );

        MemSourceBatchOp inputOp = new MemSourceBatchOp(trainData,
            new String[] {"id", "vec"});
        AgnesBatchOp op = new AgnesBatchOp()
            .setIdCol("id")
            .setVectorCol("vec")
            .setPredictionCol("pred")
            .linkFrom(inputOp);
        op.print();
    }
}
```

运行结果

id	pred
id_1	0

id_2	0
id_9	0
id_10	0
id_11	0
id_12	0
id_16	0
id_3	1
id_4	1
id_5	1
id_6	1
id_7	1
id_8	1

二分K均值聚类预测 (BisectingKMeansPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.BisectingKMeansPredictBatchOp

Python 类名：BisectingKMeansPredictBatchOp

功能介绍

二分k均值算法（BisectingKmeansTrainBatchOp）对应的预测组件，基于训练好的二分k均值模型进行聚类预测。

参数说明

训练

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "0.1,0.1,0.1"],
    [2, "0.2,0.2,0.2"],
    [3, "9 9 9"],
```

```
    [4, "9.1 9.1 9.1"],
    [5, "9.2 9.2 9.2"]
])

inBatch = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
inStream = StreamOperator.fromDataframe(df, schemaStr='id int, vec string')

kmeansTrain = BisectingKMeansTrainBatchOp()\
    .setVectorCol("vec")\
    .setK(2)\
    .linkFrom(inBatch)
kmeansTrain.lazyPrint(10)

predictBatch = BisectingKMeansPredictBatchOp()\
    .setPredictionCol("pred")\
    .linkFrom(kmeansTrain, inBatch)
predictBatch.print()

predictStream = BisectingKMeansPredictStreamOp(kmeansTrain)\
    .setPredictionCol("pred")\
    .linkFrom(inStream)
predictStream.print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.clustering.BisectingKMeansPredictBatchOp;
import com.alibaba.alink.operator.batch.clustering.BisectingKMeansTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.clustering.BisectingKMeansPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class BisectingKMeansPredictBatchOpTest {
    @Test
    public void testBisectingKMeansPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
```

```
            Row.of(0, "0 0 0"),
            Row.of(1, "0.1,0.1,0.1"),
            Row.of(2, "0.2,0.2,0.2"),
            Row.of(3, "9 9 9"),
            Row.of(4, "9.1 9.1 9.1"),
            Row.of(5, "9.2 9.2 9.2")
        );
        BatchOperator <?> inBatch = new MemSourceBatchOp(df, "id int, vec
string");
        StreamOperator <?> inStream = new MemSourceStreamOp(df, "id int, vec
string");
        BatchOperator <?> kmeansTrain = new BisectingKMeansTrainBatchOp()
            .setVectorCol("vec")
            .setK(2)
            .linkFrom(inBatch);
        kmeansTrain.lazyPrint(10);

        BatchOperator <?> predictBatch = new BisectingKMeansPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(kmeansTrain, inBatch);
        predictBatch.print();

        StreamOperator <?> predictStream = new
BisectingKMeansPredictStreamOp(kmeansTrain)
            .setPredictionCol("pred")
            .linkFrom(inStream);
        predictStream.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型结果

model_id	model_info
0	{"vectorCol":"\"vec\"","distanceType":"\"EUCLIDEAN\"","k":"2","vectorSize":"3"}
1048576	{"clusterId":1,"size":6,"center":{"data":[4.6,4.6,4.6]},"cost":364.61999999999995}
2097152	{"clusterId":2,"size":3,"center":{"data":[0.1,0.1,0.1]},"cost":0.06}
3145728	{"clusterId":3,"size":3,"center":{"data":[9.099999999999998,9.099999999999998,9.099999999999998]},"cost":0.06000000000017280

预测结果

id	vec	pred
0	0 0 0	0
1	0.1,0.1,0.1	0
2	0.2,0.2,0.2	0
3	9 9 9	1
4	9.1 9.1 9.1	1
5	9.2 9.2 9.2	1

二分K均值聚类训练 (BisectingKMeansTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.BisectingKMeansTrainBatchOp

Python 类名：BisectingKMeansTrainBatchOp

功能介绍

二分k均值算法是k-means聚类算法的一个变体，主要是为了改进k-means算法随机选择初始质心的随机性造成聚类结果不确定性的问题。

二分k均值方法是一种自顶向下的聚类。初始阶段该算法将所有节点视为属于同一个cluster。在每次迭代中，它选择一个类别，然后使用Kmeans算法对该类中的所有数据点进行二分类。当没有类可拆分时或者达到最大聚类个数时，算法终止。

参数说明

训练

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE"	"EUCLIDEAN"
k	聚类中心点数目	聚类中心点数目	Integer			4

maxIter	最大迭代步数	最大迭代步数，默认为10。	Integer			10
minDivisibleClusterSize	最小可分裂的聚类大小	最小可分裂的聚类大小	Integer			1
randomSeed	随机数种子	随机数种子	Integer			0

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "0.1,0.1,0.1"],
    [2, "0.2,0.2,0.2"],
    [3, "9 9 9"],
    [4, "9.1 9.1 9.1"],
    [5, "9.2 9.2 9.2"]
])

inBatch = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
inStream = StreamOperator.fromDataframe(df, schemaStr='id int, vec string')
```

```
kmeansTrain = BisectingKMeansTrainBatchOp()\
    .setVectorCol("vec")\
    .setK(2)\
    .linkFrom(inBatch)
kmeansTrain.lazyPrint(10)

predictBatch = BisectingKMeansPredictBatchOp()\
    .setPredictionCol("pred")\
    .linkFrom(kmeansTrain, inBatch)
predictBatch.print()

predictStream = BisectingKMeansPredictStreamOp(kmeansTrain)\
    .setPredictionCol("pred")\
    .linkFrom(inStream)
predictStream.print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.clustering.BisectingKMeansPredictBatchOp;
import com.alibaba.alink.operator.batch.clustering.BisectingKMeansTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.clustering.BisectingKMeansPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class BisectingKMeansTrainBatchOpTest {
    @Test
    public void testBisectingKMeansTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "0.1,0.1,0.1"),
            Row.of(2, "0.2,0.2,0.2"),
            Row.of(3, "9 9 9"),
            Row.of(4, "9.1 9.1 9.1"),
            Row.of(5, "9.2 9.2 9.2")
        );
```

```
        BatchOperator <?> inBatch = new MemSourceBatchOp(df, "id int, vec
string");
        StreamOperator <?> inStream = new MemSourceStreamOp(df, "id int, vec
string");
        BatchOperator <?> kmeansTrain = new BisectingKMeansTrainBatchOp()
            .setVectorCol("vec")
            .setK(2)
            .linkFrom(inBatch);
        kmeansTrain.lazyPrint(10);

        BatchOperator <?> predictBatch = new BisectingKMeansPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(kmeansTrain, inBatch);
        predictBatch.print();

        StreamOperator <?> predictStream = new
BisectingKMeansPredictStreamOp(kmeansTrain)
            .setPredictionCol("pred")
            .linkFrom(inStream);
        predictStream.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型结果

model_id	model_info
0	{"vectorCol":"\"vec\"","distanceType":"\"EUCLIDEAN\"","k":"2","vectorSize":"3"}
1048576	{"clusterId":1,"size":6,"center":{"data":[4.6,4.6,4.6]},"cost":364.61999999999995}
2097152	{"clusterId":2,"size":3,"center":{"data":[0.1,0.1,0.1]},"cost":0.06}
3145728	{"clusterId":3,"size":3,"center":{"data":[9.099999999999998,9.099999999999998,9.099999999999998]},"cost":0.06000000000017280

预测结果

id	vec	pred
0	0 0 0	0
1	0.1,0.1,0.1	0
2	0.2,0.2,0.2	0

3	9 9 9	1
4	9.1 9.1 9.1	1
5	9.2 9.2 9.2	1

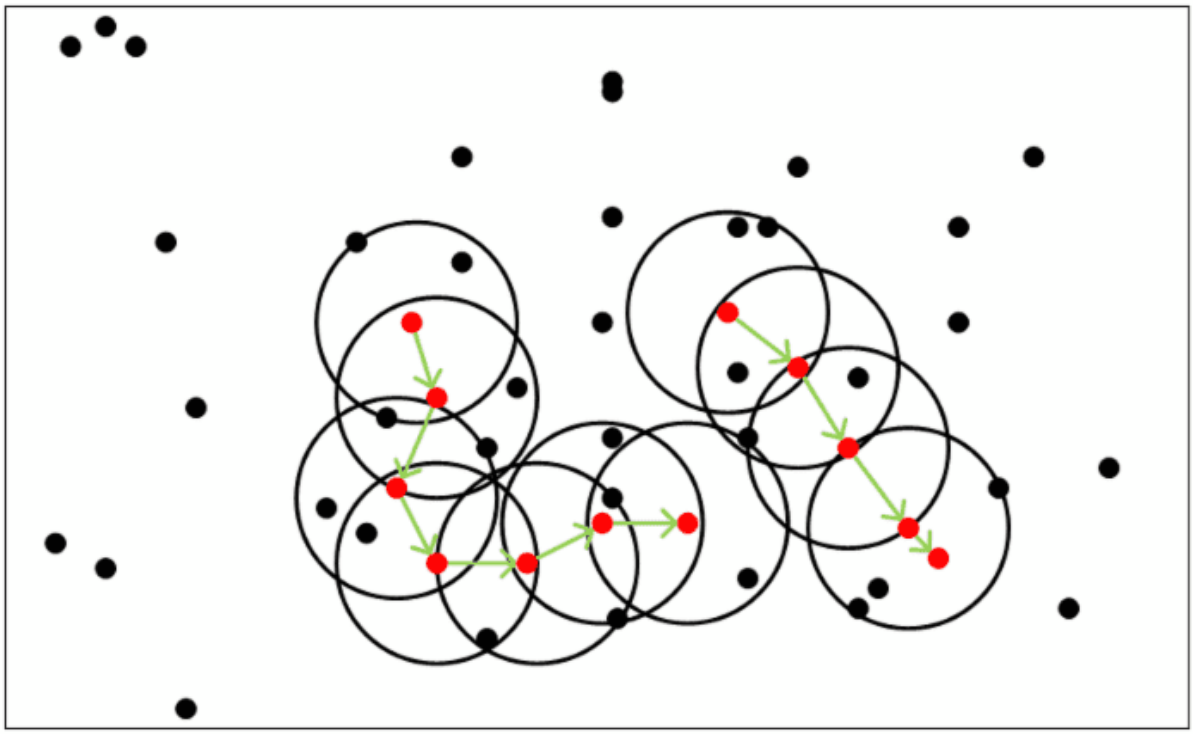
Dbscan (DbscanBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.DbscanBatchOp

Python 类名：DbscanBatchOp

功能介绍

DBSCAN，Density-Based Spatial Clustering of Applications with Noise，是一个比较有代表性的基于密度的聚类算法。与划分和层次聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在噪声的空间数据库中发现任意形状的聚类。



距离度量方式

参数名称	参数描述	说明
EUCLIDEAN	$d(x-c)=(x-c)(x-c)'$	欧式距离
COSINE	$d(x-c)=0.5-0.5*\frac{xc'}{\sqrt{xx'}\sqrt{cc'}}$	夹角余弦距离
CITYBLOCK		街区距离

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
epsilon	临域距离阈值	临域距离阈值	Double	✓		
idCol	ID列名	ID列对应的列名	String	✓		
minPoints	临域中样本个数的阈值	临域中样本个数的阈值	Integer	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
distanceType	距离度量方式	距离类型	String		"EUCLIDEAN", "COSINE", "CITYBLOCK"	"EUCLIDEAN"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

data = pd.DataFrame([
        ["id_1", "2.0,3.0"],
        ["id_2", "2.1,3.1"],
        ["id_3", "200.1,300.1"],
```

```
        ["id_4", "200.2,300.2"],
        ["id_5", "200.3,300.3"],
        ["id_6", "200.4,300.4"],
        ["id_7", "200.5,300.5"],
        ["id_8", "200.6,300.6"],
        ["id_9", "2.1,3.1"],
        ["id_10", "2.1,3.1"],
        ["id_11", "2.1,3.1"],
        ["id_12", "2.1,3.1"],
        ["id_16", "300.,3.2"]
])

inOp1 = BatchOperator.fromDataframe(data, schemaStr='id string, vec string')
inOp2 = StreamOperator.fromDataframe(data, schemaStr='id string, vec string')

dbscan = DbscanBatchOp()\
    .setIdCol("id")\
    .setVectorCol("vec")\
    .setMinPoints(3)\
    .setEpsilon(0.5)\
    .setPredictionCol("pred")\
    .linkFrom(inOp1)

dbscan.print()

predict = DbscanPredictBatchOp()\
    .setPredictionCol("pred")\
    .linkFrom(dbscan.getSideOutput(0), inOp1)

predict.print()

predict = DbscanPredictStreamOp(dbscan.getSideOutput(0))\
    .setPredictionCol("pred")\
    .linkFrom(inOp2)

predict.print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.clustering.DbscanPredictStreamOp;
import com.alibaba.alink.operator.batch.clustering.DbscanPredictBatchOp;
```

```
import com.alibaba.alink.operator.batch.clustering.DbscanBatchOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;

import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DbscanBatchOpTest {

    @Test
    public void testDbscanBatchop() throws Exception {
        List <Row> dataPoints = Arrays.asList(
            Row.of("id_1", "2.0,3.0"),
            Row.of("id_2", "2.1,3.1"),
            Row.of("id_3", "200.1,300.1"),
            Row.of("id_4", "200.2,300.2"),
            Row.of("id_5", "200.3,300.3"),
            Row.of("id_6", "200.4,300.4"),
            Row.of("id_7", "200.5,300.5"),
            Row.of("id_8", "200.6,300.6"),
            Row.of("id_9", "2.1,3.1"),
            Row.of("id_10", "2.1,3.1"),
            Row.of("id_11", "2.1,3.1"),
            Row.of("id_12", "2.1,3.1"),
            Row.of("id_16", "300.,3.2"));

        MemSourceBatchOp inOp1 = new MemSourceBatchOp(dataPoints, "id string,
vec string");
        MemSourceStreamOp inOp2 = new MemSourceStreamOp(dataPoints, "id string,
vec string");

        DbscanBatchOp dbscanBatchOp = new DbscanBatchOp()
            .setIdCol("id")
            .setVectorCol("vec")
            .setMinPoints(3)
            .setEpsilon(0.5)
            .setPredictionCol("pred")
            .linkFrom(inOp1);
        dbscanBatchOp.print();

        DbscanPredictBatchOp dbscanPredictBatchOp = new DbscanPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(dbscanBatchOp.getSideOutput(0), inOp1);
        dbscanPredictBatchOp.print();

        DbscanPredictStreamOp dbscanPredictStreamOp = new
DbscanPredictStreamOp(dbscanBatchOp.getSideOutput(0))
```

```
            .setPredictionCol("pred")
            .linkFrom(inOp2);
        dbscanPredictStreamOp.print();
        StreamOperator.execute();
    }
}
```

运行结果

训练结果

id	type	pred
id_4	CORE	1
id_8	CORE	1
id_2	CORE	0
id_6	CORE	1
id_16	NOISE	-2147483648
id_7	CORE	1
id_12	CORE	0
id_5	CORE	1
id_1	CORE	0
id_3	CORE	1
id_9	CORE	0
id_10	CORE	0
id_11	CORE	0

批式预测结果

id	vec	pred
id_1	2.0,3.0	0
id_2	2.1,3.1	0
id_3	200.1,300.1	1
id_4	200.2,300.2	1
id_5	200.3,300.3	1

id_6	200.4,300.4	1
id_7	200.5,300.5	1
id_8	200.6,300.6	1
id_9	2.1,3.1	0
id_10	2.1,3.1	0
id_11	2.1,3.1	0
id_12	2.1,3.1	0
id_16	300.,3.2	-2147483648

流式预测结果

id	vec	pred
id_11	2.1,3.1	0
id_1	2.0,3.0	0
id_16	300.,3.2	-2147483648
id_12	2.1,3.1	0
id_6	200.4,300.4	1
id_3	200.1,300.1	1
id_7	200.5,300.5	1
id_9	2.1,3.1	0
id_2	2.1,3.1	0
id_10	2.1,3.1	0
id_4	200.2,300.2	1
id_5	200.3,300.3	1
id_8	200.6,300.6	1

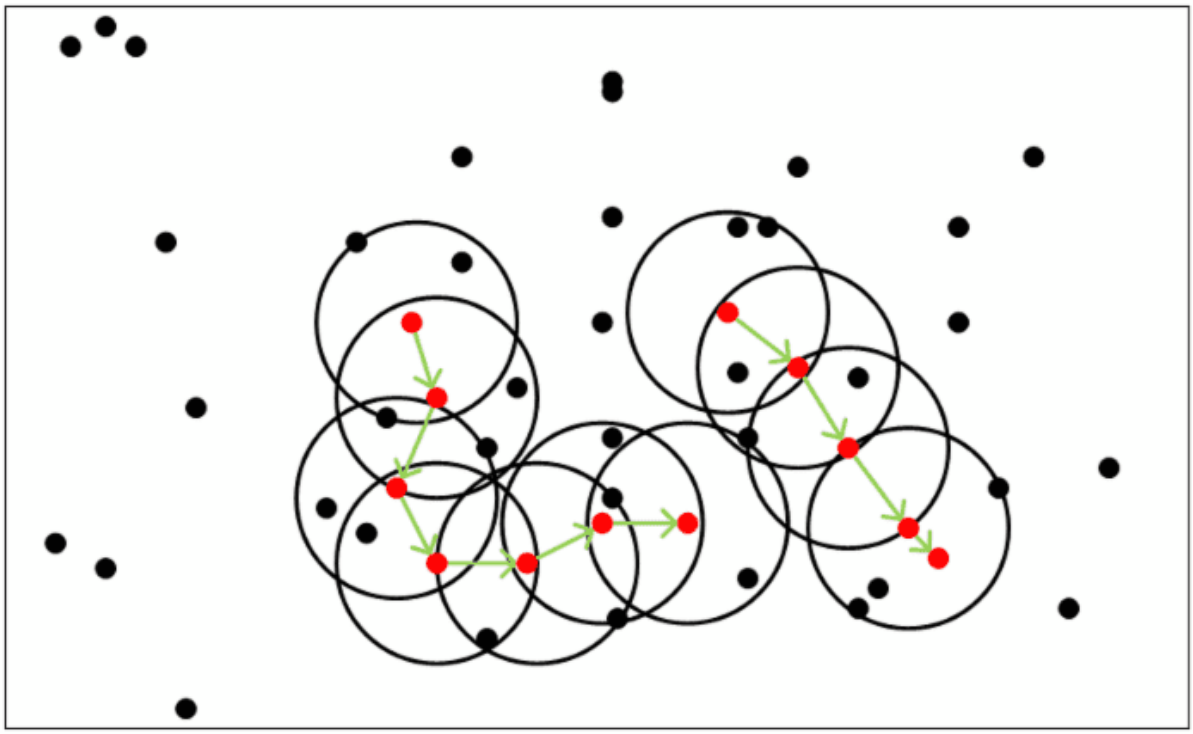
Dbscan预测 (DbscanPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.DbscanPredictBatchOp

Python 类名：DbscanPredictBatchOp

功能介绍

DBSCAN，Density-Based Spatial Clustering of Applications with Noise，是一个比较有代表性的基于密度的聚类算法。与划分和层次聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在噪声的空间数据库中发现任意形状的聚类。



Alink上DBSCAN算法括[DBSCAN]，[DBSCAN批量预测], [DBSCAN流式预测]。

距离度量方式

参数名称	参数描述	说明
EUCLIDEAN	$d(x-c)=(x-c)(x-c)^{'}$	欧式距离
COSINE	$d(x-c)=0.5-0.5*\frac{xc^{'}}{\sqrt{xx^{'}}\sqrt{cc^{'}}}$	夹角余弦距离
CITYBLOCK		街区距离

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ["id_1", "2.0,3.0"],
        ["id_2", "2.1,3.1"],
        ["id_3", "200.1,300.1"],
        ["id_4", "200.2,300.2"],
        ["id_5", "200.3,300.3"],
        ["id_6", "200.4,300.4"],
        ["id_7", "200.5,300.5"],
        ["id_8", "200.6,300.6"],
        ["id_9", "2.1,3.1"],
        ["id_10", "2.1,3.1"],
        ["id_11", "2.1,3.1"],
        ["id_12", "2.1,3.1"],
        ["id_16", "300.,3.2"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id string, vec string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id string, vec string')

dbscan = DbscanBatchOp()\
    .setIdCol("id")\
```

```
    .setVectorCol("vec")\
    .setMinPoints(3)\
    .setEpsilon(0.5)\
    .setPredictionCol("pred")\
    .linkFrom(inOp1)

dbscan.print()

predict = DbscanPredictBatchOp()\
    .setPredictionCol("pred")\
    .linkFrom(dbscan.getSideOutput(0), inOp1)

predict.print()

predict = DbscanPredictStreamOp(dbscan.getSideOutput(0))\
    .setPredictionCol("pred")\
    .linkFrom(inOp2)

predict.print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.clustering.DbscanPredictStreamOp;
import com.alibaba.alink.operator.batch.clustering.DbscanPredictBatchOp;
import com.alibaba.alink.operator.batch.clustering.DbscanBatchOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;

import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DbscanPredictBatchOpTest {

    @Test
    public void testDbscanPredictBatchop() throws Exception {
        List <Row> dataPoints = Arrays.asList(
            Row.of("id_1", "2.0,3.0"),
            Row.of("id_2", "2.1,3.1"),
            Row.of("id_3", "200.1,300.1"),
            Row.of("id_4", "200.2,300.2"),
```

```
            Row.of("id_5", "200.3,300.3"),
            Row.of("id_6", "200.4,300.4"),
            Row.of("id_7", "200.5,300.5"),
            Row.of("id_8", "200.6,300.6"),
            Row.of("id_9", "2.1,3.1"),
            Row.of("id_10", "2.1,3.1"),
            Row.of("id_11", "2.1,3.1"),
            Row.of("id_12", "2.1,3.1"),
            Row.of("id_16", "300.,3.2"));

        MemSourceBatchOp inOp1 = new MemSourceBatchOp(dataPoints, "id string,
vec string");
        MemSourceStreamOp inOp2 = new MemSourceStreamOp(dataPoints, "id string,
vec string");

        DbscanBatchOp dbscanBatchOp = new DbscanBatchOp()
            .setIdCol("id")
            .setVectorCol("vec")
            .setMinPoints(3)
            .setEpsilon(0.5)
            .setPredictionCol("pred")
            .linkFrom(inOp1);
        dbscanBatchOp.print();

        DbscanPredictBatchOp dbscanPredictBatchOp = new DbscanPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(dbscanBatchOp.getSideOutput(0), inOp1);
        dbscanPredictBatchOp.print();

        DbscanPredictStreamOp dbscanPredictStreamOp = new
DbscanPredictStreamOp(dbscanBatchOp.getSideOutput(0))
            .setPredictionCol("pred")
            .linkFrom(inOp2);
        dbscanPredictStreamOp.print();
        StreamOperator.execute();
    }
}
```

运行结果

训练结果

id	type	pred
id_4	CORE	1
id_8	CORE	1

id_2	CORE	0
id_6	CORE	1
id_16	NOISE	-2147483648
id_7	CORE	1
id_12	CORE	0
id_5	CORE	1
id_1	CORE	0
id_3	CORE	1
id_9	CORE	0
id_10	CORE	0
id_11	CORE	0

批式预测结果

id	vec	pred
id_1	2.0,3.0	0
id_2	2.1,3.1	0
id_3	200.1,300.1	1
id_4	200.2,300.2	1
id_5	200.3,300.3	1
id_6	200.4,300.4	1
id_7	200.5,300.5	1
id_8	200.6,300.6	1
id_9	2.1,3.1	0
id_10	2.1,3.1	0
id_11	2.1,3.1	0
id_12	2.1,3.1	0
id_16	300.,3.2	-2147483648

流式预测结果

id	vec	pred
id_11	2.1,3.1	0
id_1	2.0,3.0	0
id_16	300.,3.2	-2147483648
id_12	2.1,3.1	0
id_6	200.4,300.4	1
id_3	200.1,300.1	1
id_7	200.5,300.5	1
id_9	2.1,3.1	0
id_2	2.1,3.1	0
id_10	2.1,3.1	0
id_4	200.2,300.2	1
id_5	200.3,300.3	1
id_8	200.6,300.6	1

经纬度K均值聚类预测 (GeoKMeansPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.GeoKMeansPredictBatchOp

Python 类名：GeoKMeansPredictBatchOp

功能介绍

KMeans 是一个经典的聚类算法。

基本思想是：以空间中k个点为中心进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。

本组件主要针对经纬度距离做Kmeans聚类，包括经纬度KMeans，经纬度KMeans预测, 经纬度KMeans流式预测。

经纬度距离（https://en.wikipedia.org/wiki/Haversine_formula)

$$d = 2r\arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)}\right)$$

$$\operatorname{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2}$$

输入数据中的经度和纬度使用 `度数` 表示，得到的距离单位为千米(km)。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
predictionDistanceCol	预测距离列名	预测距离列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, 0],
    [8, 8],
    [1, 2],
    [9, 10],
    [3, 1],
    [10, 7]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='f0 long, f1 long')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='f0 long, f1 long')

kmeans = GeoKMeansTrainBatchOp()\
                .setLongitudeCol("f0")\
                .setLatitudeCol("f1")\
                .setK(2)\
                .linkFrom(inOp1)
kmeans.print()

predict = GeoKMeansPredictBatchOp()\
                .setPredictionCol("pred")\
                .linkFrom(kmeans, inOp1)
predict.print()

predict = GeoKMeansPredictStreamOp(kmeans)\
                .setPredictionCol("pred")\
                .linkFrom(inOp2)
predict.print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.clustering.KMeans4LongiLatitudePredictBatchOp;
import
com.alibaba.alink.operator.batch.clustering.KMeans4LongiLatitudeTrainBatchOp;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.clustering.KMeans4LongiLatitudePredictStreamO
p;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GeoKMeansPredictBatchOpTest {
    @Test
    public void testGeoKMeansPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, 0),
            Row.of(8, 8),
            Row.of(1, 2),
            Row.of(9, 10),
            Row.of(3, 1),
            Row.of(10, 7)
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "f0 int, f1 int");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "f0 int, f1 int");
        BatchOperator <?> kmeans = new GeoKMeansTrainBatchOp()
            .setLongitudeCol("f0")
            .setLatitudeCol("f1")
            .setK(2)
            .linkFrom(inOp1);
        kmeans.print();
        BatchOperator <?> result = new GeoKMeansPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(kmeans, inOp1);
        result.print();
        StreamOperator <?> predict = new GeoKMeansPredictStreamOp(kmeans)
            .setPredictionCol("pred")
            .linkFrom(inOp2);
        predict.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型数据

model_id	model_info

0	{"vectorCol":null,"latitudeCol":"\"f1\"","longitudeCol":"\"f0\"","distanceType":"\"HAVERSINE\"","k":"
1048576	{"clusterId":0,"weight":3.0,"center":"[8.333333333333332, 9.0]","vec":null}
2097152	{"clusterId":1,"weight":3.0,"center":"[1.0, 1.3333333333333333]","vec":null}

预测输出

f0	f1	pred
0	0	1
8	8	0
1	2	1
9	10	0
3	1	1
10	7	0

经纬度K均值聚类训练 (GeoKMeansTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.GeoKMeansTrainBatchOp

Python 类名：GeoKMeansTrainBatchOp

功能介绍

KMeans 是一个经典的聚类算法。

基本思想是：以空间中k个点为中心进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。

本组件主要针对经纬度距离做Kmeans聚类，包括经纬度KMeans，经纬度KMeans预测, 经纬度KMeans流式预测。

经纬度距离（https://en.wikipedia.org/wiki/Haversine_formula）

$$d = 2r \arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)}\right)$$

$$\operatorname{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

输入数据中的经度和纬度使用 `度数` 表示，得到的距离单位为千米(km)。

参数说明

名称	中文名称	描述	类型	是否必须?	
latitudeCol	经度列名	经度列名	String	✓	所选列[BIGDEBIGINTDOUBLINTEGISHORT
longitudeCol	纬度列名	纬度列名	String	✓	所选列[BIGDEBIGINTDOUBLINTEGISHORT
epsilon	收敛阈值	当两轮迭代的中心点距离小于epsilon时，算法收敛。	Double		

initMode	中心点初始化方法	初始化中心点的方法，支持"K_MEANS_PARALLEL"和"RANDOM"	String		"RAND "K_ME
initSteps	k-means++初始化迭代步数	k-means初始化中心点时迭代的步数	Integer		
k	聚类中心点数量	聚类中心点数量	Integer		
maxIter	最大迭代步数	最大迭代步数，默认为 50。	Integer		
randomSeed	随机数种子	随机数种子	Integer		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, 0],
    [8, 8],
    [1, 2],
    [9, 10],
    [3, 1],
    [10, 7]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='f0 long, f1 long')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='f0 long, f1 long')

kmeans = GeoKMeansTrainBatchOp()\
                .setLongitudeCol("f0")\
                .setLatitudeCol("f1")\
                .setK(2)\
                .linkFrom(inOp1)
kmeans.print()

predict = GeoKMeansPredictBatchOp()\
                .setPredictionCol("pred")\
                .linkFrom(kmeans, inOp1)
```

```
predict.print()

predict = GeoKMeansPredictStreamOp(kmeans)\
                .setPredictionCol("pred")\
                .linkFrom(inOp2)
predict.print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.clustering.KMeans4LongiLatitudePredictBatchOp;
import
com.alibaba.alink.operator.batch.clustering.KMeans4LongiLatitudeTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.clustering.KMeans4LongiLatitudePredictStreamO
p;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GeoKMeansTrainBatchOpTest {
    @Test
    public void testGeoKMeansTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, 0),
            Row.of(8, 8),
            Row.of(1, 2),
            Row.of(9, 10),
            Row.of(3, 1),
            Row.of(10, 7)
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "f0 int, f1 int");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "f0 int, f1 int");
        BatchOperator <?> kmeans = new GeoKMeansTrainBatchOp()
            .setLongitudeCol("f0")
            .setLatitudeCol("f1")
            .setK(2)
            .linkFrom(inOp1);
        kmeans.print();
```

```
        BatchOperator <?> result = new GeoKMeansPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(kmeans, inOp1);
        result.print();
        StreamOperator <?> predict = new GeoKMeansPredictStreamOp(kmeans)
            .setPredictionCol("pred")
            .linkFrom(inOp2);
        predict.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型数据

model_id	model_info
0	{"vectorCol":null,"latitudeCol":"\"f1\"","longitudeCol":"\"f0\"","distanceType":"\"HAVERSINE\"","k":"
1048576	{"clusterId":0,"weight":3.0,"center":"[8.333333333333332, 9.0]","vec":null}
2097152	{"clusterId":1,"weight":3.0,"center":"[1.0, 1.3333333333333333]","vec":null}

预测输出

f0	f1	pred
0	0	1
8	8	0
1	2	1
9	10	0
3	1	1
10	7	0

高斯混合模型预测 (GmmPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.GmmPredictBatchOp

Python 类名：GmmPredictBatchOp

功能介绍

高斯混合模型对应的预测组件，基于训练好的高斯混合模型（Gaussian Mixture Model）进行聚类预测。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ["-0.6264538 0.1836433"],
    ["-0.8356286 1.5952808"],
    ["0.3295078 -0.8204684"],
    ["0.4874291 0.7383247"],
    ["0.5757814 -0.3053884"],
    ["1.5117812 0.3898432"],
    ["-0.6212406 -2.2146999"],
    ["11.1249309 9.9550664"],
    ["9.9838097 10.9438362"],
    ["10.8212212 10.5939013"],
    ["10.9189774 10.7821363"],
    ["10.0745650 8.0106483"],
    ["10.6198257 9.9438713"],
    ["9.8442045 8.5292476"],
    ["9.5218499 10.4179416"],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='features string')
dataStream = StreamOperator.fromDataframe(df_data, schemaStr='features string')

gmm = GmmTrainBatchOp() \
    .setVectorCol("features") \
    .setEpsilon(0.)

model = gmm.linkFrom(data)

predictor = GmmPredictBatchOp() \
    .setPredictionCol("cluster_id") \
    .setVectorCol("features") \
    .setPredictionDetailCol("cluster_detail")

predictor.linkFrom(model, data).print()

predictorStream = GmmPredictStreamOp(model) \
    .setPredictionCol("cluster_id") \
    .setVectorCol("features") \
    .setPredictionDetailCol("cluster_detail")

predictorStream.linkFrom(dataStream).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.clustering.GmmPredictBatchOp;
import com.alibaba.alink.operator.batch.clustering.GmmTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.clustering.GmmPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GmmPredictBatchOpTest {
    @Test
    public void testGmmPredictBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("-0.6264538 0.1836433"),
            Row.of("-0.8356286 1.5952808"),
            Row.of("0.3295078 -0.8204684"),
            Row.of("0.4874291 0.7383247"),
            Row.of("0.5757814 -0.3053884"),
            Row.of("1.5117812 0.3898432"),
            Row.of("-0.6212406 -2.2146999"),
            Row.of("11.1249309 9.9550664"),
            Row.of("9.9838097 10.9438362"),
            Row.of("10.8212212 10.5939013"),
            Row.of("10.9189774 10.7821363"),
            Row.of("10.0745650 8.0106483"),
            Row.of("10.6198257 9.9438713"),
            Row.of("9.8442045 8.5292476"),
            Row.of("9.5218499 10.4179416")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "features
string");
        StreamOperator <?> dataStream = new MemSourceStreamOp(df_data,
"features string");
        BatchOperator <?> gmm = new GmmTrainBatchOp()
            .setVectorCol("features")
            .setEpsilon(0.);
        BatchOperator <?> model = gmm.linkFrom(data);
        BatchOperator <?> predictor = new GmmPredictBatchOp()
            .setPredictionCol("cluster_id")
            .setVectorCol("features")
            .setPredictionDetailCol("cluster_detail");
        predictor.linkFrom(model, data).print();
        StreamOperator <?> predictorStream = new GmmPredictStreamOp(model)
```

```
            .setPredictionCol("cluster_id")
            .setVectorCol("features")
            .setPredictionDetailCol("cluster_detail");
        predictorStream.linkFrom(dataStream).print();
        StreamOperator.execute();
    }
}
```

运行结果

features	cluster_id	cluster_detail
-0.6264538 0.1836433	0	1.0 4.275273913994647E-92
-0.8356286 1.5952808	0	1.0 1.0260377730322135E-92
0.3295078 -0.8204684	0	1.0 1.0970173367582936E-80
0.4874291 0.7383247	0	1.0 3.30217313232611E-75
0.5757814 -0.3053884	0	1.0 3.163811360527691E-76
1.5117812 0.3898432	0	1.0 2.1018052308786076E-62
-0.6212406 -2.2146999	0	1.0 6.772270268625197E-97
11.1249309 9.9550664	1	3.1567838012477083E-56 1.0
9.9838097 10.9438362	1	1.9024447346702333E-51 1.0
10.8212212 10.5939013	1	2.8009730987296404E-56 1.0
10.9189774 10.7821363	1	1.7209132744891575E-57 1.0
10.0745650 8.0106483	1	2.864269663513225E-43 1.0
10.6198257 9.9438713	1	5.77327399194046E-53 1.0
9.8442045 8.5292476	1	2.5273123050926845E-43 1.0
9.5218499 10.4179416	1	1.7314580596765865E-46 1.0

高斯混合模型训练 (GmmTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.GmmTrainBatchOp

Python 类名：GmmTrainBatchOp

功能介绍

混合模型（Mixture Model）是一个可以用来表示在总体分布中含有K个子分布的概率模型。换句话说，混合模型表示了观测数据在总体中的概率分布，它是一个由K个子分布组成的混合分布。 而高斯混合模型（Gaussian Mixture Model, GMM）可以用来表示在总体分布中含有K个高斯子分布的概率模型。它通常可以被用作分类模型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
epsilon	收敛阈值	当两轮迭代的中心点距离小于epsilon时，算法收敛。	Double			1.0E-4
k	聚类中心点数量	聚类中心点数量	Integer			2
maxIter	最大迭代步数	最大迭代步数，默认为 100	Integer		[1, +inf)	100

randomSeed	随机数种子	随机数种子	Integer			0

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ["-0.6264538 0.1836433"],
    ["-0.8356286 1.5952808"],
    ["0.3295078 -0.8204684"],
    ["0.4874291 0.7383247"],
    ["0.5757814 -0.3053884"],
    ["1.5117812 0.3898432"],
    ["-0.6212406 -2.2146999"],
    ["11.1249309 9.9550664"],
    ["9.9838097 10.9438362"],
    ["10.8212212 10.5939013"],
    ["10.9189774 10.7821363"],
    ["10.0745650 8.0106483"],
    ["10.6198257 9.9438713"],
    ["9.8442045 8.5292476"],
    ["9.5218499 10.4179416"],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='features string')
dataStream = StreamOperator.fromDataframe(df_data, schemaStr='features string')

gmm = GmmTrainBatchOp() \
    .setVectorCol("features") \
    .setEpsilon(0.)

model = gmm.linkFrom(data)

predictor = GmmPredictBatchOp() \
    .setPredictionCol("cluster_id") \
    .setVectorCol("features") \
```

```
    .setPredictionDetailCol("cluster_detail")

predictor.linkFrom(model, data).print()

predictorStream = GmmPredictStreamOp(model) \
    .setPredictionCol("cluster_id") \
    .setVectorCol("features") \
    .setPredictionDetailCol("cluster_detail")

predictorStream.linkFrom(dataStream).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.clustering.GmmPredictBatchOp;
import com.alibaba.alink.operator.batch.clustering.GmmTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.clustering.GmmPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GmmTrainBatchOpTest {
    @Test
    public void testGmmTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("-0.6264538 0.1836433"),
            Row.of("-0.8356286 1.5952808"),
            Row.of("0.3295078 -0.8204684"),
            Row.of("0.4874291 0.7383247"),
            Row.of("0.5757814 -0.3053884"),
            Row.of("1.5117812 0.3898432"),
            Row.of("-0.6212406 -2.2146999"),
            Row.of("11.1249309 9.9550664"),
            Row.of("9.9838097 10.9438362"),
            Row.of("10.8212212 10.5939013"),
            Row.of("10.9189774 10.7821363"),
            Row.of("10.0745650 8.0106483"),
            Row.of("10.6198257 9.9438713"),
            Row.of("9.8442045 8.5292476"),
```

```
            Row.of("9.5218499 10.4179416")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "features
string");
        StreamOperator <?> dataStream = new MemSourceStreamOp(df_data,
"features string");
        BatchOperator <?> gmm = new GmmTrainBatchOp()
            .setVectorCol("features")
            .setEpsilon(0.);
        BatchOperator <?> model = gmm.linkFrom(data);
        BatchOperator <?> predictor = new GmmPredictBatchOp()
            .setPredictionCol("cluster_id")
            .setVectorCol("features")
            .setPredictionDetailCol("cluster_detail");
        predictor.linkFrom(model, data).print();
        StreamOperator <?> predictorStream = new GmmPredictStreamOp(model)
            .setPredictionCol("cluster_id")
            .setVectorCol("features")
            .setPredictionDetailCol("cluster_detail");
        predictorStream.linkFrom(dataStream).print();
        StreamOperator.execute();
    }
}
```

运行结果

features	cluster_id	cluster_detail
-0.6264538 0.1836433	1	4.275273913968281E-92 1.0
-0.8356286 1.5952808	1	1.0260377730239899E-92 1.0
0.3295078 -0.8204684	1	1.0970173367545207E-80 1.0
0.4874291 0.7383247	1	3.302173132311E-75 1.0
0.5757814 -0.3053884	1	3.1638113605165424E-76 1.0
1.5117812 0.3898432	1	2.101805230873173E-62 1.0
-0.6212406 -2.2146999	1	6.772270268600749E-97 1.0
11.1249309 9.9550664	0	1.0 3.156783801247968E-56
9.9838097 10.9438362	0	1.0 1.9024447346702425E-51
10.8212212 10.5939013	0	1.0 2.800973098729604E-56
10.9189774 10.7821363	0	1.0 1.7209132744891298E-57
10.0745650 8.0106483	0	1.0 2.8642696635130495E-43

10.6198257 9.9438713	0	1.0 5.773273991940433E-53
9.8442045 8.5292476	0	1.0 2.5273123050925483E-43
9.5218499 10.4179416	0	1.0 1.7314580596767853E-46

分组Dbscan (GroupDbscanBatchOp)

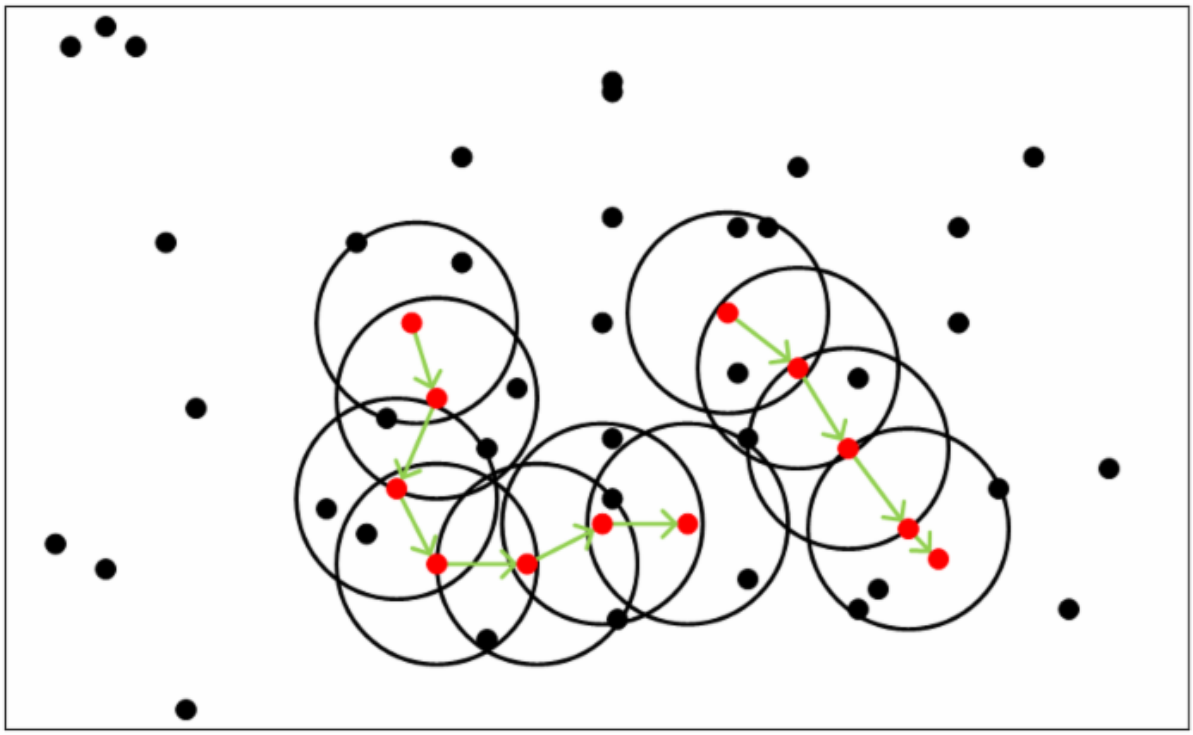
Java 类名：com.alibaba.alink.operator.batch.clustering.GroupDbscanBatchOp

Python 类名：GroupDbscanBatchOp

功能介绍

DBSCAN，Density-Based Spatial Clustering of Applications with Noise，是一个比较有代表性的基于密度的聚类算法。与划分和层次聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在噪声的空间数据库中发现任意形状的聚类。

分组DBSCAN算法根据用户指定的"分组列"将输入数据分为多个组，然后在每个组内部进行DBSCAN聚类算法。



距离度量方式

参数名称	参数描述	说明
EUCLIDEAN		欧式距离
COSINE		夹角余弦距离
CITYBLOCK		城市街区距离，也称曼哈顿距离

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
epsilon	临域距离阈值	临域距离阈值	Double	✓		
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
groupCols	分组列名，多列	分组列名，多列，必选	String[]	✓		
idCol	id列名	id列名	String	✓		
minPoints	临域中样本个数的阈值	临域中样本个数的阈值	Integer	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE", "CITYBLOCK", "HAVERSINE", "JACCARD"	"EUCLIDEAN"
groupMaxSamples	每个分组的最大样本数	每个分组的最大样本数	Integer			2147483647

isOutputVector	输出是否为向量格式	输出是否为向量格式	Boolean			false
skip	每个分组超过最大样本数时，是否跳过	每个分组超过最大样本数时，是否跳过	Boolean			false

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "id_1", 2.0, 3.0],
    [0, "id_2", 2.1, 3.1],
    [0, "id_18", 2.4, 3.2],
    [0, "id_15", 2.8, 3.2],
    [0, "id_12", 2.1, 3.1],
    [0, "id_3", 200.1, 300.1],
    [0, "id_4", 200.2, 300.2],
    [0, "id_8", 200.6, 300.6],

    [1, "id_5", 200.3, 300.3],
    [1, "id_6", 200.4, 300.4],
    [1, "id_7", 200.5, 300.5],
    [1, "id_16", 300., 300.2],
    [1, "id_9", 2.1, 3.1],
    [1, "id_10", 2.2, 3.2],
    [1, "id_11", 2.3, 3.3],
    [1, "id_13", 2.4, 3.4],
    [1, "id_14", 2.5, 3.5],
    [1, "id_17", 2.6, 3.6],
    [1, "id_19", 2.7, 3.7],
```

```
    [1, "id_20", 2.8, 3.8],
    [1, "id_21", 2.9, 3.9],

    [2, "id_20", 2.8, 3.8]])

source = BatchOperator.fromDataframe(df, schemaStr='group string, id string, c1
double, c2 double')

groupDbscan = GroupDbscanBatchOp()\
    .setIdCol("id")\
    .setGroupCols(["group"])\
    .setFeatureCols(["c1", "c2"])\
    .setMinPoints(4)\
    .setEpsilon(0.6)\
    .linkFrom(source)

groupDbscan.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.clustering.GroupDbscanBatchOp;

import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GroupDbscanBatchOpTest {

    @Test
    public void testGroupDbscanBatchOp() throws Exception {
        List<Row> trainData = Arrays.asList(
            Row.of(0, "id_1", 2.0, 3.0),
            Row.of(0, "id_2", 2.1, 3.1),
            Row.of(0, "id_18", 2.4, 3.2),
            Row.of(0, "id_15", 2.8, 3.2),
            Row.of(0, "id_12", 2.1, 3.1),
            Row.of(0, "id_3", 200.1, 300.1),
            Row.of(0, "id_4", 200.2, 300.2),
            Row.of(0, "id_8", 200.6, 300.6),

            Row.of(1, "id_5", 200.3, 300.3),
            Row.of(1, "id_6", 200.4, 300.4),
```

```
            Row.of(1, "id_7", 200.5, 300.5),
            Row.of(1, "id_16", 300., 300.2),
            Row.of(1, "id_9", 2.1, 3.1),
            Row.of(1, "id_10", 2.2, 3.2),
            Row.of(1, "id_11", 2.3, 3.3),
            Row.of(1, "id_13", 2.4, 3.4),
            Row.of(1, "id_14", 2.5, 3.5),
            Row.of(1, "id_17", 2.6, 3.6),
            Row.of(1, "id_19", 2.7, 3.7),
            Row.of(1, "id_20", 2.8, 3.8),
            Row.of(1, "id_21", 2.9, 3.9),

            Row.of(2, "id_20", 2.8, 3.8)
        );

        MemSourceBatchOp inputOp = new MemSourceBatchOp(trainData,
            new String[] {"group", "id", "c1", "c2"});
        GroupDbscanBatchOp op = new GroupDbscanBatchOp()
            .setIdCol("id")
            .setGroupCols("group")
            .setFeatureCols("c1", "c2")
            .setMinPoints(4)
            .setEpsilon(0.6)
            .linkFrom(inputOp);
        op.print();
    }
}
```

运行结果

```
group|id|type|cluster_id|c1|c2
-----|---|----|----------|---|---
1|id_5|NOISE|-2147483648|200.3000|300.3000
1|id_6|NOISE|-2147483648|200.4000|300.4000
1|id_7|NOISE|-2147483648|200.5000|300.5000
1|id_16|NOISE|-2147483648|300.0000|300.2000
1|id_9|CORE|0|2.1000|3.1000
1|id_10|CORE|0|2.2000|3.2000
1|id_11|CORE|0|2.3000|3.3000
1|id_13|CORE|0|2.4000|3.4000
1|id_14|CORE|0|2.5000|3.5000
1|id_17|CORE|0|2.6000|3.6000
 ......
1|id_21|CORE|0|2.9000|3.9000
0|id_1|CORE|0|2.0000|3.0000
0|id_2|CORE|0|2.1000|3.1000
0|id_18|CORE|0|2.4000|3.2000
```

```
0|id_15|LINKED|0|2.8000|3.2000
0|id_12|CORE|0|2.1000|3.1000
0|id_3|NOISE|-2147483648|200.1000|300.1000
0|id_4|NOISE|-2147483648|200.2000|300.2000
0|id_8|NOISE|-2147483648|200.6000|300.6000
2|id_20|NOISE|-2147483648|2.8000|3.8000
```

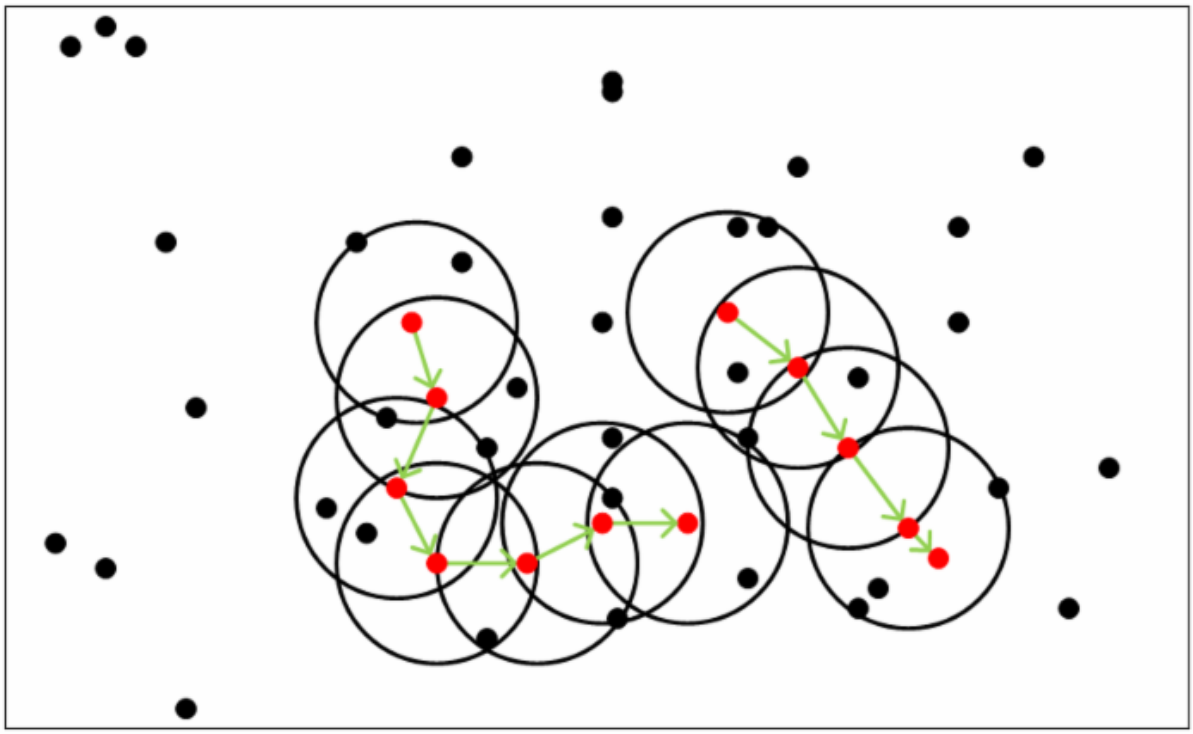
分组Dbscan模型 (GroupDbscanModelBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.GroupDbscanModelBatchOp

Python 类名：GroupDbscanModelBatchOp

功能介绍

DBSCAN，Density-Based Spatial Clustering of Applications with Noise，是一个比较有代表性的基于密度的聚类算法。与划分和层次聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在噪声的空间数据库中发现任意形状的聚类。



距离度量方式

参数名称	参数描述	说明
EUCLIDEAN		欧式距离
COSINE		夹角余弦距离
CITYBLOCK		城市街区距离，也称曼哈顿距离

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
epsilon	临域距离阈值	临域距离阈值	Double	✓		
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
groupCols	分组列名，多列	分组列名，多列，必选	String[]	✓		
minPoints	临域中样本个数的阈值	临域中样本个数的阈值	Integer	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE", "CITYBLOCK", "HAVERSINE", "JACCARD"	"EUCLIDEAN"
groupMaxSamples	每个分组的最大样本数	每个分组的最大样本数	Integer			2147483647

skip	每个分组超过最大样本数时，是否跳过	每个分组超过最大样本数时，是否跳过	Boolean			false

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "id_1", 2.0, 3.0],
    [0, "id_2", 2.1, 3.1],
    [0, "id_18", 2.4, 3.2],
    [0, "id_15", 2.8, 3.2],
    [0, "id_12", 2.1, 3.1],
    [0, "id_3", 200.1, 300.1],
    [0, "id_4", 200.2, 300.2],
    [0, "id_8", 200.6, 300.6],

    [1, "id_5", 200.3, 300.3],
    [1, "id_6", 200.4, 300.4],
    [1, "id_7", 200.5, 300.5],
    [1, "id_16", 300., 300.2],
    [1, "id_9", 2.1, 3.1],
    [1, "id_10", 2.2, 3.2],
    [1, "id_11", 2.3, 3.3],
    [1, "id_13", 2.4, 3.4],
    [1, "id_14", 2.5, 3.5],
    [1, "id_17", 2.6, 3.6],
    [1, "id_19", 2.7, 3.7],
    [1, "id_20", 2.8, 3.8],
    [1, "id_21", 2.9, 3.9],

    [2, "id_20", 2.8, 3.8]])
```

```
source = BatchOperator.fromDataframe(df, schemaStr='group string, id string, c1
double, c2 double')

groupDbscan = GroupDbscanModelBatchOp()\
    .setGroupCols(["group"])\
    .setFeatureCols(["c1", "c2"])\
    .setMinPoints(4)\
    .setEpsilon(0.6)\
    .linkFrom(source)

groupDbscan.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.clustering.GroupDbscanModelBatchOp;

import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GroupDbscanModelBatchOpTest {

    @Test
    public void testGroupDbscanModelBatchOp() throws Exception {
        List<Row> trainData = Arrays.asList(
            Row.of(0, "id_1", 2.0, 3.0),
            Row.of(0, "id_2", 2.1, 3.1),
            Row.of(0, "id_18", 2.4, 3.2),
            Row.of(0, "id_15", 2.8, 3.2),
            Row.of(0, "id_12", 2.1, 3.1),
            Row.of(0, "id_3", 200.1, 300.1),
            Row.of(0, "id_4", 200.2, 300.2),
            Row.of(0, "id_8", 200.6, 300.6),

            Row.of(1, "id_5", 200.3, 300.3),
            Row.of(1, "id_6", 200.4, 300.4),
            Row.of(1, "id_7", 200.5, 300.5),
            Row.of(1, "id_16", 300., 300.2),
            Row.of(1, "id_9", 2.1, 3.1),
            Row.of(1, "id_10", 2.2, 3.2),
            Row.of(1, "id_11", 2.3, 3.3),
            Row.of(1, "id_13", 2.4, 3.4),
```

```
            Row.of(1, "id_14", 2.5, 3.5),
            Row.of(1, "id_17", 2.6, 3.6),
            Row.of(1, "id_19", 2.7, 3.7),
            Row.of(1, "id_20", 2.8, 3.8),
            Row.of(1, "id_21", 2.9, 3.9),

            Row.of(2, "id_20", 2.8, 3.8)
        );

        MemSourceBatchOp inputOp = new MemSourceBatchOp(trainData,
            new String[] {"group", "id", "c1", "c2"});
        GroupDbscanModelBatchOp op = new GroupDbscanModelBatchOp()
            .setGroupCols("group")
            .setFeatureCols("c1", "c2")
            .setMinPoints(4)
            .setEpsilon(0.6)
            .linkFrom(inputOp);
        op.print();
    }
}
```

运行结果

group	cluster_id	count	c1	c2
1	0	9	2.5000	3.5000
0	0	5	2.2800	3.1200

分组Kmeans (GroupKMeansBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.GroupKMeansBatchOp

Python 类名：GroupKMeansBatchOp

功能介绍

分组Kmeans聚类算法。本算法按照用户指定的分组列（groupCol）将数据分成很多个组，然后分别对这些组进行Kmeans聚类。算法的输出值是每个数据点的所属类别。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
groupCols	分组列名，多列	分组列名，多列，必选	String[]	✓		
idCol	id列名	id列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
distanceType	距离度量方式	距离类型	String		"EUCLIDEAN", "COSINE", "CITYBLOCK"	"EUCLIDEAN"
epsilon	收敛阈值	当两轮迭代的中心点距离小于epsilon时，算法收敛。	Double			1.0E-4

k	聚类中心点数量	聚类中心点数量	Integer			2
maxIter	最大迭代步数	最大迭代步数，默认为10。	Integer			10

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)


df = pd.DataFrame([
    [0, "id_1", 2.0, 3.0],
    [0, "id_2", 2.1, 3.1],
    [0, "id_18", 2.4, 3.2],
    [0, "id_15", 2.8, 3.2],
    [0, "id_12", 2.1, 3.1],
    [0, "id_3", 200.1, 300.1],
    [0, "id_4", 200.2, 300.2],
    [0, "id_8", 200.6, 300.6],

    [1, "id_5", 200.3, 300.3],
    [1, "id_6", 200.4, 300.4],
    [1, "id_7", 200.5, 300.5],
    [1, "id_16", 300., 300.2],
    [1, "id_9", 2.1, 3.1],
    [1, "id_10", 2.2, 3.2],
    [1, "id_11", 2.3, 3.3],
    [1, "id_13", 2.4, 3.4],
    [1, "id_14", 2.5, 3.5],
    [1, "id_17", 2.6, 3.6],
    [1, "id_19", 2.7, 3.7],
    [1, "id_20", 2.8, 3.8],
    [1, "id_21", 2.9, 3.9],

    [2, "id_20", 2.8, 3.8]])

source = BatchOperator.fromDataframe(df, schemaStr='group string, id string, c1
```

```
double, c2 double')

groupKmeans = GroupKMeansBatchOp()\
    .setGroupCols(["group"])\
    .setK(2)\
    .setMaxIter(50)\
    .setPredictionCol("pred")\
    .setEpsilon(1e-8)\
    .setFeatureCols(["c1", "c2"])\
    .setIdCol("id")\
    .linkFrom(source)

groupKmeans.print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.clustering;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.clustering.GroupKMeansBatchOp;

import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GroupKmeansBatchOpTest {

    @Test
    public void testGroupKmeansBatchOp() throws Exception {
        List<Row> trainData = Arrays.asList(
            Row.of(0, "id_1", 2.0, 3.0),
            Row.of(0, "id_2", 2.1, 3.1),
            Row.of(0, "id_18", 2.4, 3.2),
            Row.of(0, "id_15", 2.8, 3.2),
            Row.of(0, "id_12", 2.1, 3.1),
            Row.of(0, "id_3", 200.1, 300.1),
            Row.of(0, "id_4", 200.2, 300.2),
            Row.of(0, "id_8", 200.6, 300.6),

            Row.of(1, "id_5", 200.3, 300.3),
            Row.of(1, "id_6", 200.4, 300.4),
            Row.of(1, "id_7", 200.5, 300.5),
            Row.of(1, "id_16", 300., 300.2),
```

```
            Row.of(1, "id_9", 2.1, 3.1),
            Row.of(1, "id_10", 2.2, 3.2),
            Row.of(1, "id_11", 2.3, 3.3),
            Row.of(1, "id_13", 2.4, 3.4),
            Row.of(1, "id_14", 2.5, 3.5),
            Row.of(1, "id_17", 2.6, 3.6),
            Row.of(1, "id_19", 2.7, 3.7),
            Row.of(1, "id_20", 2.8, 3.8),
            Row.of(1, "id_21", 2.9, 3.9),

            Row.of(2, "id_20", 2.8, 3.8)
        );

        MemSourceBatchOp inputOp = new MemSourceBatchOp(trainData,
            new String[] {"group", "id", "c1", "c2"});
        GroupKMeansBatchOp op = new GroupKMeansBatchOp()
            .setGroupCols(new String[] {"group"})
            .setK(2)
            .setMaxIter(50)
            .setPredictionCol("pred")
            .setEpsilon(1e-8)
            .setFeatureCols(new String[] {"c1", "c2"})
            .setIdCol("id")
            .linkFrom(inputOp);
        op.print();
    }
}
```

运行结果

预测结果

```
group|id|pred
-----|---|----
1|id_10|0
1|id_17|0
1|id_13|0
1|id_6|1
1|id_9|0
1|id_11|0
1|id_14|0
1|id_20|0
1|id_7|1
1|id_16|1
 ......
1|id_21|0
0|id_2|0
```

```
0|id_3|1
0|id_15|0
0|id_1|0
0|id_18|0
0|id_12|0
0|id_8|1
0|id_4|1
2|id_20|0
```

K均值聚类预测 (KMeansPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.KMeansPredictBatchOp

Python 类名：KMeansPredictBatchOp

功能介绍

KMeans 是一个经典的聚类算法。

基本思想是：以空间中k个点为中心进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。

Alink上KMeans算法包括KMeans，KMeans批量预测, KMeans流式预测。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
predictionDistanceCol	预测距离列名	预测距离列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "0.1,0.1,0.1"],
    [2, "0.2,0.2,0.2"],
    [3, "9 9 9"],
    [4, "9.1 9.1 9.1"],
    [5, "9.2 9.2 9.2"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id int, vec string')

kmeans = KMeansTrainBatchOp()\
    .setVectorCol("vec")\
    .setK(2)\
    .linkFrom(inOp1)
kmeans.lazyPrint(10)

predictBatch = KMeansPredictBatchOp()\
    .setPredictionCol("pred")\
    .linkFrom(kmeans, inOp1)
predictBatch.print()

predictStream = KMeansPredictStreamOp(kmeans)\
    .setPredictionCol("pred")\
    .linkFrom(inOp2)
predictStream.print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.clustering.KMeansPredictBatchOp;
import com.alibaba.alink.operator.batch.clustering.KMeansTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.clustering.KMeansPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KMeansPredictBatchOpTest {
```

```
    @Test
    public void testKMeansPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "0.1,0.1,0.1"),
            Row.of(2, "0.2,0.2,0.2"),
            Row.of(3, "9 9 9"),
            Row.of(4, "9.1 9.1 9.1"),
            Row.of(5, "9.2 9.2 9.2")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, vec
string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, vec
string");
        BatchOperator <?> kmeans = new KMeansTrainBatchOp()
            .setVectorCol("vec")
            .setK(2)
            .linkFrom(inOp1);
        kmeans.lazyPrint(10);

        BatchOperator <?> predictBatch = new KMeansPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(kmeans, inOp1);
        predictBatch.print();

        StreamOperator <?> predictStream = new KMeansPredictStreamOp(kmeans)
            .setPredictionCol("pred")
            .linkFrom(inOp2);
        predictStream.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型结果

model_id	model_info
0	{"vectorCol":"\"vec\"","latitudeCol":null,"longitudeCol":null,"distanceType":"\"EUCLIDEAN\"","k":"2
1048576	{"clusterId":0,"weight":3.0,"vec":{"data":[9.099999999999998,9.099999999999998,9.099999999
2097152	{"clusterId":1,"weight":3.0,"vec":{"data":[0.1,0.1,0.1]}}

预测结果

id	vec	pred

0	0 0 0	1
1	0.1,0.1,0.1	1
2	0.2,0.2,0.2	1
3	9 9 9	0
4	9.1 9.1 9.1	0
5	9.2 9.2 9.2	0

K均值聚类训练 (KMeansTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.KMeansTrainBatchOp

Python 类名：KMeansTrainBatchOp

功能介绍

Kmeans算法的训练组件。KMeans是一个经典的聚类算法。该算法的基本思想是：以空间中k个点为中心进行聚类，对最靠近它们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。

距离度量方式

参数名称	参数描述	说明
EUCLIDEAN	$d(x-c)=(x-c)(x-c)^{'}$	欧式距离
COSINE	$d(x-c)=0.5-0.5*\frac{xc^{'}}{\sqrt{xx^{'}}\sqrt{cc^{'}}}$	夹角余弦距离

参数说明

名称	中文名称	描述	类型	是否必须?	
vectorCol	向量列名	向量列对应的列名	String	✓	所选列[DENS SPARS STRIN
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCL "COSI
epsilon	收敛阈值	当两轮迭代的中心点距离小于epsilon时，算法收敛。	Double		
initMode	中心点初始化方法	初始化中心点的方法，支持"K_MEANS_PARALLEL"和"RANDOM"	String		"RAND "K_ME
initSteps	k-means++初始化迭代步数	k-means初始化中心点时迭代的步数	Integer		

k	聚类中心点数量	聚类中心点数量	Integer		
maxIter	最大迭代步数	最大迭代步数，默认为 50。	Integer		
randomSeed	随机数种子	随机数种子	Integer		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "0.1,0.1,0.1"],
    [2, "0.2,0.2,0.2"],
    [3, "9 9 9"],
    [4, "9.1 9.1 9.1"],
    [5, "9.2 9.2 9.2"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id int, vec string')

kmeans = KMeansTrainBatchOp()\
    .setVectorCol("vec")\
    .setK(2)\
    .linkFrom(inOp1)
kmeans.lazyPrint(10)

predictBatch = KMeansPredictBatchOp()\
    .setPredictionCol("pred")\
    .linkFrom(kmeans, inOp1)
predictBatch.print()

predictStream = KMeansPredictStreamOp(kmeans)\
    .setPredictionCol("pred")\
    .linkFrom(inOp2)
predictStream.print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.clustering.KMeansPredictBatchOp;
import com.alibaba.alink.operator.batch.clustering.KMeansTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.clustering.KMeansPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KMeansTrainBatchOpTest {
    @Test
    public void testKMeansTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "0.1,0.1,0.1"),
            Row.of(2, "0.2,0.2,0.2"),
            Row.of(3, "9 9 9"),
            Row.of(4, "9.1 9.1 9.1"),
            Row.of(5, "9.2 9.2 9.2")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, vec
string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, vec
string");
        BatchOperator <?> kmeans = new KMeansTrainBatchOp()
            .setVectorCol("vec")
            .setK(2)
            .linkFrom(inOp1);
        kmeans.lazyPrint(10);

        BatchOperator <?> predictBatch = new KMeansPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(kmeans, inOp1);
        predictBatch.print();

        StreamOperator <?> predictStream = new KMeansPredictStreamOp(kmeans)
            .setPredictionCol("pred")
            .linkFrom(inOp2);
        predictStream.print();
        StreamOperator.execute();
```

```
    }
}
```

运行结果

模型结果

model_id	model_info
0	{"vectorCol":"\"vec\"","latitudeCol":null,"longitudeCol":null,"distanceType":"\"EUCLIDEAN\"","k":"2
1048576	{"clusterId":0,"weight":3.0,"vec":{"data":[9.099999999999998,9.099999999999998,9.099999999
2097152	{"clusterId":1,"weight":3.0,"vec":{"data":[0.1,0.1,0.1]}}

预测结果

id	vec	pred
0	0 0 0	1
1	0.1,0.1,0.1	1
2	0.2,0.2,0.2	1
3	9 9 9	0
4	9.1 9.1 9.1	0
5	9.2 9.2 9.2	0

Kmodes预测 (KModesPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.KModesPredictBatchOp

Python 类名：KModesPredictBatchOp

功能介绍

KModes是一种用于离散数据/分类数据(categorical data)的聚类算法。 基本思想是：把n个对象分为k个簇，使簇内具有较小的的相异度(或者称距离)。 距离计算方法：两个字符串比较，相同为0，不同为1。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["pc", "Hp.com"],
    ["camera", "Hp.com"],
    ["digital camera", "Hp.com"],
    ["camera", "BestBuy.com"],
    ["digital camera", "BestBuy.com"],
    ["tv", "BestBuy.com"],
```

```
    ["flower", "Teleflora.com"],
    ["flower", "Orchids.com"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='f0 string, f1 string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='f0 string, f1 string')

kmodes = KModesTrainBatchOp()\
    .setFeatureCols(["f0", "f1"])\
    .setK(2)\
    .linkFrom(inOp1)

predict = KModesPredictBatchOp()\
    .setPredictionCol("pred")\
    .linkFrom(kmodes, inOp1)

kmodes.lazyPrint(10)
predict.print()

predict = KModesPredictStreamOp(kmodes)\
    .setPredictionCol("pred")\
    .linkFrom(inOp2)

predict.print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.clustering.KModesPredictStreamOp;
import com.alibaba.alink.operator.batch.clustering.KModesPredictBatchOp;
import com.alibaba.alink.operator.batch.clustering.KModesTrainBatchOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;

import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KModesPredictBatchOpTest {

    @Test
    public void testKModesPredictBatchOp() throws Exception {
```

```
        List <Row> dataPoints = Arrays.asList(
            Row.of("pc", "Hp.com"),
            Row.of("camera", "Hp.com"),
            Row.of("digital camera", "Hp.com"),
            Row.of("camera", "BestBuy.com"),
            Row.of("digital camera", "BestBuy.com"),
            Row.of("tv", "BestBuy.com"),
            Row.of("flower", "Teleflora.com"),
            Row.of("flower", "Orchids.com")
        );

        MemSourceBatchOp inOp1 = new MemSourceBatchOp(dataPoints, "f0 string,
f1 string");
        MemSourceStreamOp inOp2 = new MemSourceStreamOp(dataPoints, "f0 string,
f1 string");

        KModesTrainBatchOp kmodes = new KModesTrainBatchOp()
            .setFeatureCols(new String[]{"f0", "f1"})
            .setK(2)
            .linkFrom(inOp1);
        KModesPredictBatchOp kModesPredictBatchOp = new KModesPredictBatchOp()
            .setPredictionCol("pred")
            .linkFrom(kmodes, inOp1);

        kmodes.lazyPrint(10);
        kModesPredictBatchOp.print();

        KModesPredictStreamOp kModesPredictStreamOp = new
KModesPredictStreamOp(kmodes)
            .setPredictionCol("pred")
            .linkFrom(inOp2);
        kModesPredictStreamOp.print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	pred
pc	Hp.com	1
flower	Teleflora.com	0
digital camera	BestBuy.com	1
digital camera	Hp.com	1
flower	Orchids.com	0

tv	BestBuy.com	0
camera	BestBuy.com	0
camera	Hp.com	1

Kmodes训练 (KModesTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.KModesTrainBatchOp

Python 类名：KModesTrainBatchOp

功能介绍

KModes是一种用于离散数据/分类数据(categorical data)的聚类算法。 基本思想是：把n个对象分为k个簇，使簇内具有较小的的相异度(或者称距离)。 距离计算方法：两个字符串比较，相同为0，不同为1。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
k	聚类中心点数量	聚类中心点数量	Integer			2
numIter	迭代次数	迭代次数，默认为10	Integer			10

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["pc", "Hp.com", 1],
    ["camera", "Hp.com", 1],
    ["digital camera", "Hp.com", 1],
    ["camera", "BestBuy.com", 1],
```

```
    ["digital camera", "BestBuy.com", 1],
    ["tv", "BestBuy.com", 1],
    ["flower", "Teleflora.com", 1],
    ["flower", "Orchids.com", 1]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='f0 string, f1 string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='f0 string, f1 string')

kmodes = KModesTrainBatchOp()\
    .setFeatureCols(["f0", "f1"])\
    .setK(2)\
    .linkFrom(inOp1)

predict = KModesPredictBatchOp()\
    .setPredictionCol("pred")\
    .linkFrom(kmodes, inOp1)

kmodes.lazyPrint(10)
predict.print()

predict = KModesPredictStreamOp(kmodes)\
    .setPredictionCol("pred")\
    .linkFrom(inOp2)

predict.print()

StreamOperator.execute()
```

运行结果

模型结果

```
model_id                                           model_info
0         0                   {"featureCols":"[\"f0\",\"f1\"]"}
1   1048576  {"center":"[\"camera\",\"BestBuy.com\"]","clus...
2   2097152  {"center":"[\"flower\",\"Hp.com\"]","clusterId...
```

预测结果

```
               f0             f1  pred
0              pc         Hp.com     1
1          camera         Hp.com     1
2  digital camera         Hp.com     1
3          camera    BestBuy.com     0
4  digital camera    BestBuy.com     0
5              tv    BestBuy.com     0
```

```
6          flower  Teleflora.com     0
7          flower    Orchids.com     0
```

LDA预测 (LdaPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.LdaPredictBatchOp

Python 类名：LdaPredictBatchOp

功能介绍

LDA(Latent Dirichlet allocation)是一种主题模型。LDA是一种非监督机器学习技术，可以用来识别大规模文档集（document collection）或语料库（corpus）中潜藏的主题信息。它采用了词袋（bag of words）的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

它将文档集中每篇文档的主题按照概率分布的形式给出，同时它是一种无监督学习算法，在训练时不需要手工标注的训练集，需要的仅仅是文档集以及指定主题的数量k即可。

LDA功能包含LDA训练和LDA预测(批和流)以及pipeline。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a b b c c c c c e e f f f g h k k k"],
    ["a b b b d e e e h h k"],
    ["a b b b b c f f f f g g g g g g g g i j j"],
    ["a a b d d d g g g g g i i j j j k k k k k k k k"],
    ["a a a b c d d d d d d d d d e e e g g j k k k"],
    ["a a a a b b d d d e e e e f f f f f g h i j j j j"],
    ["a a b d d d g g g g g i i j j k k k k k k k k k"],
    ["a b c d d d d d d d d d e e f g g j k k k"],
    ["a a a a b b b b d d d e e e e f f g h h h"],
    ["a a b b b b b b b b c c e e e g g i i j j j j j j j k k"],
    ["a b c d d d d d d d d d f f g g j j j k k k"],
    ["a a a a b e e e e f f f f f g h h h j"],
])

inOp = BatchOperator.fromDataframe(df, schemaStr="doc string")
inOp2 = StreamOperator.fromDataframe(df, schemaStr="doc string")

ldaTrain = LdaTrainBatchOp()\
            .setSelectedCol("doc")\
            .setTopicNum(6)\
            .setMethod("online")\
            .setSubsamplingRate(1.0)\
            .setOptimizeDocConcentration(True)\
            .setNumIter(20)

ldaPredict = LdaPredictBatchOp()\
    .setPredictionCol("pred")\
    .setSelectedCol("doc")

model = ldaTrain.linkFrom(inOp)
ldaPredict.linkFrom(model, inOp)

model.lazyPrint(10)
ldaPredict.print()

ldaPredictS = LdaPredictStreamOp(model)\
    .setPredictionCol("pred")\
```

```
    .setSelectedCol("doc")\
    .linkFrom(inOp2)

ldaPredictS.print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.clustering.LdaPredictBatchOp;
import com.alibaba.alink.operator.batch.clustering.LdaTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.clustering.LdaPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LdaPredictBatchOpTest {
    @Test
    public void testLdaPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a b b c c c c c c e e f f f g h k k k"),
            Row.of("a b b b d e e e h h k"),
            Row.of("a b b b b c f f f f g g g g g g g g i j j"),
            Row.of("a a b d d d g g g g g i i j j j k k k k k k k k k"),
            Row.of("a a a b c d d d d d d d d d e e e g g j k k k"),
            Row.of("a a a a b b d d d e e e e f f f f f g h i j j j j"),
            Row.of("a a b d d d g g g g g i i j j k k k k k k k k k"),
            Row.of("a b c d d d d d d d d d e e f g g j k k k"),
            Row.of("a a a a b b b b d d d e e e e f f g h h h"),
            Row.of("a a b b b b b b b b c c e e e g g i i j j j j j j j k k"),
            Row.of("a b c d d d d d d d d d f f g g j j j k k k"),
            Row.of("a a a a b e e e e f f f f f g h h h j")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "doc string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "doc string");
        BatchOperator <?> ldaTrain = new LdaTrainBatchOp()
            .setSelectedCol("doc")
            .setTopicNum(6)
            .setMethod("online")
            .setSubsamplingRate(1.0)
```

```
            .setOptimizeDocConcentration(true)
            .setNumIter(20);
        BatchOperator <?> ldaPredict = new LdaPredictBatchOp()
            .setPredictionCol("pred")
            .setSelectedCol("doc");
        BatchOperator <?> model = ldaTrain.linkFrom(inOp);
        ldaPredict.linkFrom(model, inOp);
        model.lazyPrint(10);
        ldaPredict.print();
        StreamOperator <?> ldaPredictS = new LdaPredictStreamOp(model)
            .setPredictionCol("pred")
            .setSelectedCol("doc")
            .linkFrom(inOp2);
        ldaPredictS.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型结果

model_id	
0	{"logPerplexity":"3.7090449161397796","betaArray":"[0.16666666666666666,0.1666666666666
1048576	{"m":6,"n":11,"data": [6125.275647735944,5541.830400832857,5277.404107556518,5575.307666756267,5738.822
2097152	{"f0":"d","f1":0.36772478012531734,"f2":0}
3145728	{"f0":"k","f1":0.36772478012531734,"f2":1}
4194304	{"f0":"f","f1":0.4855078157817008,"f2":7}
5242880	{"f0":"c","f1":0.6190392084062235,"f2":8}
6291456	{"f0":"h","f1":0.7731898882334817,"f2":9}
7340032	{"f0":"i","f1":0.7731898882334817,"f2":10}
8388608	{"f0":"g","f1":0.08004270767353636,"f2":2}
9437184	{"f0":"b","f1":0.0,"f2":3}
10485760	{"f0":"a","f1":0.0,"f2":4}
11534336	{"f0":"e","f1":0.36772478012531734,"f2":5}
12582912	{"f0":"j","f1":0.26236426446749106,"f2":6}

预测结果

id	libsvm	pred
0	a b b c c c c c c e e f f f g h k k k	0
1	a b b b d e e e h h k	4
2	a b b b b c f f f f g g g g g g g g g i j j	5
3	a a b d d d g g g g g i i j j j k k k k k k k k k	1
4	a a a b c d d d d d d d d d e e e g g j k k k	1
5	a a a a b b d d d e e e e f f f f f g h i j j j j	2
6	a a b d d d g g g g g i i j j k k k k k k k k k	1
7	a b c d d d d d d d d d e e f g g j k k k	0
8	a a a a b b b b d d d e e e e e f f g h h h	4
9	a a b b b b b b b b c c e e e e g g i i j j j j j j j k k	4
10	a b c d d d d d d d d d f f g g j j j k k k	0
11	a a a a b e e e e e f f f f f g h h h j	1

LDA训练 (LdaTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.clustering.LdaTrainBatchOp

Python 类名：LdaTrainBatchOp

功能介绍

LDA(Latent Dirichlet allocation)是一种主题模型。LDA是一种非监督机器学习技术，可以用来识别大规模文档集（document collection）或语料库（corpus）中潜藏的主题信息。它采用了词袋（bag of words）的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

它将文档集中每篇文档的主题按照概率分布的形式给出，同时它是一种无监督学习算法，在训练时不需要手工标注的训练集，需要的仅仅是文档集以及指定主题的数量k即可。

LDA功能包含LDA训练和LDA预测(批和流)以及pipeline。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]
topicNum	主题个数	主题个数	Integer	✓	
alpha	文章的超参	文章的超参	Double		
beta	词的超参	词的超参	Double		
learningDecay	衰减率	衰减率	Double		
method	优化方法	优化方法, 包含"em"和"online"两种。	String		"Online", "EM"

numIter	迭代次数	迭代次数，默认为10	Integer		
onlineLearningOffset	偏移量	偏移量	Double		
optimizeDocConcentration	是否优化alpha	是否优化alpha	Boolean		
randomSeed	随机数种子	随机数种子	Integer		
subsamplingRate	采样率	采样率	Double		
vocabSize	字典库大小	字典库大小，如果总词数目大于这个值，那个文档频率低的词会被过滤掉。	Integer		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a b b c c c c c c e e f f f g h k k k"],
    ["a b b b d e e e h h k"],
    ["a b b b b c f f f f g g g g g g g g i j j"],
    ["a a b d d d g g g g g i i j j j k k k k k k k k"],
    ["a a a b c d d d d d d d d d e e e g g j k k k"],
    ["a a a a b b d d d e e e e f f f f f g h i j j j j"],
    ["a a b d d d g g g g g i i j j k k k k k k k k k"],
    ["a b c d d d d d d d d d e e f g g j k k k"],
    ["a a a a b b b b d d d e e e e f f g h h h"],
    ["a a b b b b b b b b c c e e e g g i i j j j j j j j k k"],
    ["a b c d d d d d d d d d f f g g j j j k k k"],
    ["a a a a b e e e e f f f f f g h h h j"],
])
```

```
inOp = BatchOperator.fromDataframe(df, schemaStr="doc string")
inOp2 = StreamOperator.fromDataframe(df, schemaStr="doc string")

ldaTrain = LdaTrainBatchOp()\
            .setSelectedCol("doc")\
            .setTopicNum(6)\
            .setMethod("online")\
            .setSubsamplingRate(1.0)\
            .setOptimizeDocConcentration(True)\
            .setNumIter(20)

ldaPredict = LdaPredictBatchOp()\
    .setPredictionCol("pred")\
    .setSelectedCol("doc")

model = ldaTrain.linkFrom(inOp)
ldaPredict.linkFrom(model, inOp)

model.lazyPrint(10)
ldaPredict.print()

ldaPredictS = LdaPredictStreamOp(model)\
    .setPredictionCol("pred")\
    .setSelectedCol("doc")\
    .linkFrom(inOp2)

ldaPredictS.print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.clustering.LdaPredictBatchOp;
import com.alibaba.alink.operator.batch.clustering.LdaTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.clustering.LdaPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LdaTrainBatchOpTest {
```

```
    @Test
    public void testLdaTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a b b c c c c c c e e f f f g h k k k"),
            Row.of("a b b b d e e e h h k"),
            Row.of("a b b b b c f f f f g g g g g g g g i j j"),
            Row.of("a a b d d d g g g g g i i j j j k k k k k k k k"),
            Row.of("a a a b c d d d d d d d d d e e e g g j k k k"),
            Row.of("a a a a b b d d d e e e e f f f f f g h i j j j j"),
            Row.of("a a b d d d g g g g g i i j j k k k k k k k k k"),
            Row.of("a b c d d d d d d d d d e e f g g j k k k"),
            Row.of("a a a a b b b b d d d e e e e f f g h h h"),
            Row.of("a a b b b b b b b b c c e e e g g i i j j j j j j j k k"),
            Row.of("a b c d d d d d d d d d f f g g j j j k k k"),
            Row.of("a a a a b e e e e f f f f f g h h h j")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "doc string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "doc string");
        BatchOperator <?> ldaTrain = new LdaTrainBatchOp()
            .setSelectedCol("doc")
            .setTopicNum(6)
            .setMethod("online")
            .setSubsamplingRate(1.0)
            .setOptimizeDocConcentration(true)
            .setNumIter(20);
        BatchOperator <?> ldaPredict = new LdaPredictBatchOp()
            .setPredictionCol("pred")
            .setSelectedCol("doc");
        BatchOperator <?> model = ldaTrain.linkFrom(inOp);
        ldaPredict.linkFrom(model, inOp);
        model.lazyPrint(10);
        ldaPredict.print();
        StreamOperator <?> ldaPredictS = new LdaPredictStreamOp(model)
            .setPredictionCol("pred")
            .setSelectedCol("doc")
            .linkFrom(inOp2);
        ldaPredictS.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型结果

model_id	

0	{"logPerplexity":"3.7090449161397796","betaArray":"[0.16666666666666666,0.1666666666666
1048576	{"m":6,"n":11,"data": [6125.275647735944,5541.830400832857,5277.404107556518,5575.307666756267,5738.822
2097152	{"f0":"d","f1":0.36772478012531734,"f2":0}
3145728	{"f0":"k","f1":0.36772478012531734,"f2":1}
4194304	{"f0":"f","f1":0.4855078157817008,"f2":7}
5242880	{"f0":"c","f1":0.6190392084062235,"f2":8}
6291456	{"f0":"h","f1":0.7731898882334817,"f2":9}
7340032	{"f0":"i","f1":0.7731898882334817,"f2":10}
8388608	{"f0":"g","f1":0.08004270767353636,"f2":2}
9437184	{"f0":"b","f1":0.0,"f2":3}
10485760	{"f0":"a","f1":0.0,"f2":4}
11534336	{"f0":"e","f1":0.36772478012531734,"f2":5}
12582912	{"f0":"j","f1":0.26236426446749106,"f2":6}

预测结果

id	libsvm	pred
0	a b b c c c c c c e e f f f g h k k k	0
1	a b b b d e e e h h k	4
2	a b b b b c f f f f g g g g g g g g g i j j	5
3	a a b d d d g g g g g i i j j j k k k k k k k k k	1
4	a a a b c d d d d d d d d d e e e g g j k k k	1
5	a a a a b b d d d e e e e f f f f f g h i j j j j	2
6	a a b d d d g g g g g i i j j k k k k k k k k k	1
7	a b c d d d d d d d d d e e f g g j k k k	0
8	a a a a b b b b d d d e e e e e f f g h h h	4
9	a a b b b b b b b b c c e e e e g g i i j j j j j j j k k	4
10	a b c d d d d d d d d d f f g g j j j k k k	0
11	a a a a b e e e e f f f f f g h h h j	1

关联规则预测 (ApplyAssociationRuleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.associationrule.ApplyAssociationRuleBatchOp

Python 类名：ApplyAssociationRuleBatchOp

功能介绍

- 应用关联规则，计算命中的关联规则的输出。
- 模型由FpGrowthBatchOp训练得出，通过getSideOutputAssociationRules()函数获取模型

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["A,B,C,D"],
    ["B,C,E"],
```

```
    ["A,B,C,E"],
    ["B,D,E"],
    ["A,B,C,D"],
])

data = BatchOperator.fromDataframe(df, schemaStr='items string')

fpGrowth = FpGrowthBatchOp() \
    .setItemsCol("items") \
    .setMinSupportPercent(0.4) \
    .setMinConfidence(0.6)

fpGrowth.linkFrom(data)

fpGrowth.print()
fpGrowth.getSideOutputAssociationRules().print()

ApplyAssociationRuleBatchOp()\
    .setSelectedCol("items") \
    .setOutputCol("result") \
    .linkFrom(fpGrowth.getSideOutputAssociationRules(), data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ApplyAssociationRuleBatchOpTest {
    @Test
    public void testFpGrowth() throws Exception {
        List <Row> rows = Arrays.asList(
            Row.of("A,B,C,D"),
            Row.of("B,C,E"),
            Row.of("A,B,C,E"),
            Row.of("B,D,E"),
            Row.of("A,B,C,D")
            );

        BatchOperator data = new MemSourceBatchOp(rows, "items string");
```

```
        FpGrowthBatchOp fpGrowth = new FpGrowthBatchOp()
            .setItemsCol("items")
            .setMinSupportPercent(0.4)
            .setMinConfidence(0.6);

        fpGrowth.linkFrom(data);
        fpGrowth.print();
        fpGrowth.getSideOutputAssociationRules().print();

        ApplyAssociationRuleBatchOp op = new ApplyAssociationRuleBatchOp()
            .setSelectedCol("items")
            .setOutputCol("result")
            .linkFrom(fpGrowth.getSideOutputAssociationRules(), data);
        op.print();
    }
}
```

运行结果

频繁项集输出：

itemset	supportcount	itemcount
E	3	1
B,E	3	2
C,E	2	2
B,C,E	2	3
D	3	1
B,D	3	2
C,D	2	2
B,C,D	2	3
A,D	2	2
B,A,D	2	3
C,A,D	2	3
B,C,A,D	2	4
A	3	1
B,A	3	2
C,A	3	2

B,C,A	3	3
C	4	1
B,C	4	2
B	5	1

关联规则输出：

rule	itemcount	lift	support_percent	confidence_percent	transaction_cou
D=>B	2	1.0000	0.6000	1.0000	3
D=>A	2	1.1111	0.4000	0.6667	2
C,D=>B	3	1.0000	0.4000	1.0000	2
A,D=>B	3	1.0000	0.4000	1.0000	2
B,D=>A	3	1.1111	0.4000	0.6667	2
A,D=>C	3	1.2500	0.4000	1.0000	2
C,D=>A	3	1.6667	0.4000	1.0000	2
C,A,D=>B	4	1.0000	0.4000	1.0000	2
B,A,D=>C	4	1.2500	0.4000	1.0000	2
B,C,D=>A	4	1.6667	0.4000	1.0000	2
C=>A	2	1.2500	0.6000	0.7500	3
C=>B	2	1.0000	0.8000	1.0000	4
B,C=>A	3	1.2500	0.6000	0.7500	3
E=>B	2	1.0000	0.6000	1.0000	3
C,E=>B	3	1.0000	0.4000	1.0000	2
B=>E	2	1.0000	0.6000	0.6000	3
B=>D	2	1.0000	0.6000	0.6000	3
B=>A	2	1.0000	0.6000	0.6000	3
B=>C	2	1.0000	0.8000	0.8000	4
A=>D	2	1.1111	0.4000	0.6667	2
A=>B	2	1.0000	0.6000	1.0000	3
A=>C	2	1.2500	0.6000	1.0000	3

B,A=>D	3	1.1111	0.4000	0.6667	2
C,A=>D	3	1.1111	0.4000	0.6667	2
C,A=>B	3	1.0000	0.6000	1.0000	3
B,A=>C	3	1.2500	0.6000	1.0000	3
B,C,A=>D	4	1.1111	0.4000	0.6667	2

关联规则预测输出

items	result
A,B,C,D	E
B,C,E	A,D
A,B,C,E	D
B,D,E	A,C
A,B,C,D	E

序列规则预测 (ApplySequenceRuleBatchOp)

Java 类名：com.alibaba.alink.operator.batch.associationrule.ApplySequenceRuleBatchOp

Python 类名：ApplySequenceRuleBatchOp

功能介绍

- 应用频繁子序列关联规则，计算命中的频繁子序列。
- 模型由PrefixSpanBatchOp训练，通过getSideOutputAssociationRules()获取。

输入说明：一个sequence由多个element组成，element之间用分号分隔；一个element由多个item组成，item间用逗号分隔。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
```

```
    ["a;a,b,c;a,c;d;c,f"],
    ["a,d;c;b,c;a,e"],
    ["e,f;a,b;d,f;c;b"],
    ["e;g;a,f;c;b;c"],
])

data = BatchOperator.fromDataframe(df, schemaStr='sequence string')

prefixSpan = PrefixSpanBatchOp() \
    .setItemsCol("sequence") \
    .setMinSupportCount(3)

prefixSpan.linkFrom(data)

prefixSpan.print()
prefixSpan.getSideOutputAssociationRules().print()

ApplySequenceRuleBatchOp()\
    .setSelectedCol("sequence")\
    .setOutputCol("result")\
    .linkFrom(prefixSpan.getSideOutputAssociationRules(), data)\
    .print()
```

Java代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ApplySequenceRuleBatchOpTest {
    @Test
    public void testPrefixSpan() throws Exception {
        List <Row> rows = Arrays.asList(
            Row.of("a;a,b,c;a,c;d;c,f"),
            Row.of("a,d;c;b,c;a,e"),
            Row.of("e,f;a,b;d,f;c;b"),
            Row.of("e;g;a,f;c;b;c")
        );

        BatchOperator data = new MemSourceBatchOp(rows, "sequence string");
        PrefixSpanBatchOp prefixSpan = new PrefixSpanBatchOp()
```

```
            .setItemsCol("sequence")
            .setMinSupportCount(3);

        prefixSpan.linkFrom(data);

        ApplySequenceRuleBatchOp op = new ApplySequenceRuleBatchOp()
            .setSelectedCol("sequence")
            .setOutputCol("result")
            .linkFrom(prefixSpan.getSideOutputAssociationRules(), data);
        op.print();
    }
}
```

运行结果

频繁项集输出：

itemset	supportcount	itemcount
e	3	1
f	3	1
a	4	1
a;c	4	2
a;c;c	3	3
a;c;b	3	3
a;b	4	2
b	4	1
b;c	3	2
c	4	1
c;c	3	2
c;b	3	2
d	3	1
d;c	3	2

关联规则输出：

rule	chain_length	support	confidence	transaction_count
c=>c	2	0.7500	0.7500	3

c=>b	2	0.7500	0.7500	3
d=>c	2	0.7500	1.0000	3
b=>c	2	0.7500	0.7500	3
a=>c	2	1.0000	1.0000	4
a;c=>c	3	0.7500	0.7500	3
a;c=>b	3	0.7500	0.7500	3
a=>b	2	1.0000	1.0000	4

预测结果输出

sequence	result
a;a,b,c;a,c;d;c,f	b
a,d;c;b,c;a,e	c
e,f;a,b;d,f;c;b	c
e;g;a,f;c;b;c	

FpGrowth (FpGrowthBatchOp)

Java 类名：com.alibaba.alink.operator.batch.associationrule.FpGrowthBatchOp

Python 类名：FpGrowthBatchOp

功能介绍

FP Growth(Frequent Pattern growth)算法是一种非时序的关联分析算法. 它利用FP tree生成频繁项集和规则,效率优于传统的Apriori算法。

该算法只需要扫描数据2次。其中第1次扫描获得当个项目的频率，去掉不符合支持度要求的项，并对剩下的项按照置信度降序排序。第2遍扫描是建立一棵频繁项树FP-Tree。

算法生成频繁项集分为两个过程：（1）构建FP树；(2)从FP树中挖掘频繁项集。

论文：Han et al.,《Mining frequent patterns without candidate generation》

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemsCol	项集列名	项集列名	String	✓	所选列类型为[STRING]	
maxConsequentLength	最大关联规则后继长度	最大关联规则后继(consequent)长度	Integer			1

maxPatternLength	最大频繁项集长度	最大频繁项集长度	Integer			10
minConfidence	最小置信度	最小置信度，同时包含X和Y的样本与包含X的样本之比，反映了当样本中包含项集X时，项集Y同时出现的概率。	Double			0.05
minLift	最小提升度	最小提升度，提升度是用来衡量A出现的情况下，是否会对B出现的概率有所提升。	Double			1.0
minSupportCount	最小支持度数目	最小支持度目，当取值大于或等于0时起作用，当小于0时参数minSupportPercent起作用	Integer			-1
minSupportPercent	最小支持度占比	最小支持度占比，当minSupportCount取值小于0时起作用，当minSupportCount大于或等于0时该参数不起作用	Double			0.02

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
```

```
    ["A,B,C,D"],
    ["B,C,E"],
    ["A,B,C,E"],
    ["B,D,E"],
    ["A,B,C,D"],
])

data = BatchOperator.fromDataframe(df, schemaStr='items string')

fpGrowth = FpGrowthBatchOp() \
    .setItemsCol("items") \
    .setMinSupportPercent(0.4) \
    .setMinConfidence(0.6)

fpGrowth.linkFrom(data)

fpGrowth.print()
fpGrowth.getSideOutput(0).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.associationrule.FpGrowthBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FpGrowthBatchOpTest {
    @Test
    public void testFpGrowthBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("A,B,C,D"),
            Row.of("B,C,E"),
            Row.of("A,B,C,E"),
            Row.of("B,D,E"),
            Row.of("A,B,C,D")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "items string");
        BatchOperator <?> fpGrowth = new FpGrowthBatchOp()
            .setItemsCol("items")
            .setMinSupportPercent(0.4)
            .setMinConfidence(0.6);
        fpGrowth.linkFrom(data);
```

```
        fpGrowth.print();
        fpGrowth.getSideOutput(0).print();
    }
}
```

二元组作为输入

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

public class FpGrowthBatchOpTest {
    @Test
    public void testFpGrowth() throws Exception {
        Row[] rows = new Row[] {
            Row.of(1, "A"),
            Row.of(1, "B"),
            Row.of(1, "C"),
            Row.of(1, "D"),
            Row.of(2, "B"),
            Row.of(2, "C"),
            Row.of(2, "E"),
            Row.of(3, "A"),
            Row.of(3, "B"),
            Row.of(3, "C"),
            Row.of(3, "E"),
            Row.of(4, "B"),
            Row.of(4, "D"),
            Row.of(4, "E"),
            Row.of(5, "A"),
            Row.of(5, "B"),
            Row.of(5, "C"),
            Row.of(5, "D"),
        };

        BatchOperator data = new MemSourceBatchOp(rows, "id int, item string")
            .groupBy("id", "id,CONCAT_AGG(item) AS items");

        FpGrowthBatchOp fpGrowth = new FpGrowthBatchOp()
            .setItemsCol("items")
            .setMinSupportPercent(0.4)
            .setMinConfidence(0.6);

        fpGrowth.linkFrom(data);
        fpGrowth.print();
```

```
        fpGrowth.getSideOutputAssociationRules().print();
    }
}
```

运行结果

频繁项集输出：

itemset	supportcount	itemcount
E	3	1
B,E	3	2
C,E	2	2
B,C,E	2	3
D	3	1
B,D	3	2
C,D	2	2
B,C,D	2	3
A,D	2	2
B,A,D	2	3
C,A,D	2	3
B,C,A,D	2	4
A	3	1
B,A	3	2
C,A	3	2
B,C,A	3	3
C	4	1
B,C	4	2
B	5	1

关联规则输出：

rule	itemcount	lift	support_percent	confidence_percent	transaction_cou
D=>B	2	1.0000	0.6000	1.0000	3
D=>A	2	1.1111	0.4000	0.6667	2

C,D=>B	3	1.0000	0.4000	1.0000	2
A,D=>B	3	1.0000	0.4000	1.0000	2
B,D=>A	3	1.1111	0.4000	0.6667	2
A,D=>C	3	1.2500	0.4000	1.0000	2
C,D=>A	3	1.6667	0.4000	1.0000	2
C,A,D=>B	4	1.0000	0.4000	1.0000	2
B,A,D=>C	4	1.2500	0.4000	1.0000	2
B,C,D=>A	4	1.6667	0.4000	1.0000	2
C=>A	2	1.2500	0.6000	0.7500	3
C=>B	2	1.0000	0.8000	1.0000	4
B,C=>A	3	1.2500	0.6000	0.7500	3
E=>B	2	1.0000	0.6000	1.0000	3
C,E=>B	3	1.0000	0.4000	1.0000	2
B=>E	2	1.0000	0.6000	0.6000	3
B=>D	2	1.0000	0.6000	0.6000	3
B=>A	2	1.0000	0.6000	0.6000	3
B=>C	2	1.0000	0.8000	0.8000	4
A=>D	2	1.1111	0.4000	0.6667	2
A=>B	2	1.0000	0.6000	1.0000	3
A=>C	2	1.2500	0.6000	1.0000	3
B,A=>D	3	1.1111	0.4000	0.6667	2
C,A=>D	3	1.1111	0.4000	0.6667	2
C,A=>B	3	1.0000	0.6000	1.0000	3
B,A=>C	3	1.2500	0.6000	1.0000	3
B,C,A=>D	4	1.1111	0.4000	0.6667	2

分组FP增长训练 (GroupedFpGrowthBatchOp)

Java 类名：com.alibaba.alink.operator.batch.associationrule.GroupedFpGrowthBatchOp

Python 类名：GroupedFpGrowthBatchOp

功能介绍

分组FpGrowth组件按照指定的分组列，在每个分组内使用FpGrowth算法进行频繁项集挖掘。

FpGrowth算法详见FpGrowthBatchOp

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemsCol	项集列名	项集列名	String	✓	所选列类型为[STRING]	
groupCol	分组单列名	分组单列名，可选	String			null
maxConsequentLength	最大关联规则后继长度	最大关联规则后继(consequent)长度	Integer			1

maxPatternLength	最大频繁项集长度	最大频繁项集长度	Integer			10
minConfidence	最小置信度	最小置信度，同时包含X和Y的样本与包含X的样本之比，反映了当样本中包含项集X时，项集Y同时出现的概率。	Double			0.05
minLift	最小提升度	最小提升度，提升度是用来衡量A出现的情况下，是否会对B出现的概率有所提升。	Double			1.0
minSupportCount	最小支持度数目	最小支持度目，当取值大于或等于0时起作用，当小于0时参数minSupportPercent起作用	Integer			-1
minSupportPercent	最小支持度占比	最小支持度占比，当minSupportCount取值小于0时起作用，当minSupportCount大于或等于0时该参数不起作用	Double			0.02

<!--

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    ["changjiang", "A,B,C,D"],
    ["changjiang", "B,C,E"],
    ["huanghe", "A,B,C,E"],
    ["huanghe", "B,D,E"],
    ["huanghe", "A,B,C,D"],
])

data = BatchOperator.fromDataframe(df, schemaStr='group string, items string')

fpGrowth = GroupedFpGrowthBatchOp() \
    .setItemsCol("items").setGroupCol("group").setMinSupportCount(2)

fpGrowth.linkFrom(data)

fpGrowth.print()
```

Java代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;

public class GroupedFpGrowthBatchOpTest {
    @Test
    public void test() throws Exception {
        Row[] rows = new Row[] {
            Row.of("changjiang", "A,B,C,D"),
            Row.of("changjiang", "B,C,E"),
            Row.of("huanghe", "A,B,C,E"),
            Row.of("huanghe", "B,D,E"),
            Row.of("huanghe", "A,B,C,D"),
        };

        BatchOperator data = new MemSourceBatchOp(Arrays.asList(rows), "group
string, items string");

        BatchOperator fpgrowth = new GroupedFpGrowthBatchOp()
            .setGroupCol("group")
            .setItemsCol("items")
            .setMinSupportCount(2);
```

```
        fpgrowth.linkFrom(data);
        fpgrowth.print();
    }
}
```

三元组输入

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;

public class GroupedFpGrowthBatchOpTest {
    @Test
    public void test() throws Exception {
        Row[] rows = new Row[] {
            Row.of("changjiang", 1, "A"),
            Row.of("changjiang", 1, "B"),
            Row.of("changjiang", 1, "C"),
            Row.of("changjiang", 1, "D"),
            Row.of("changjiang", 2, "B"),
            Row.of("changjiang", 2, "C"),
            Row.of("changjiang", 2, "E"),
            Row.of("huanghe", 3, "A"),
            Row.of("huanghe", 3, "B"),
            Row.of("huanghe", 3, "C"),
            Row.of("huanghe", 3, "E"),
            Row.of("huanghe", 4, "B"),
            Row.of("huanghe", 4, "D"),
            Row.of("huanghe", 4, "E"),
            Row.of("huanghe", 5, "A"),
            Row.of("huanghe", 5, "B"),
            Row.of("huanghe", 5, "C"),
            Row.of("huanghe", 5, "D"),
        };

        BatchOperator data = new MemSourceBatchOp(Arrays.asList(rows), "groupid
string, id int, item string")
            .groupBy("groupid,id", "groupid,id,CONCAT_AGG(item) AS items");

        BatchOperator fpgrowth = new GroupedFpGrowthBatchOp()
            .setGroupCol("groupid")
            .setItemsCol("items")
            .setMinSupportCount(2);
```

```
        fpgrowth.linkFrom(data);
        fpgrowth.print();
    }
}
```

运行结果

group	itemset	supportcount	itemcount
changjiang	B	2	1
changjiang	C	2	1
changjiang	B,C	2	2
huanghe	A	2	1
huanghe	B,A	2	2
huanghe	B	3	1
huanghe	C	2	1
huanghe	B,C	2	2
huanghe	A,C	2	2
huanghe	B,A,C	2	3
huanghe	D	2	1
huanghe	B,D	2	2
huanghe	E	2	1
huanghe	B,E	2	2

PrefixSpan (PrefixSpanBatchOp)

Java 类名：com.alibaba.alink.operator.batch.associationrule.PrefixSpanBatchOp

Python 类名：PrefixSpanBatchOp

功能介绍

PrefixSpan算法的全称是Prefix-Projected Pattern Growth，即前缀投影的模式挖掘。

与关联规则挖掘不同的是，频繁序列模式挖掘的对象和结果都是有序的， 即数据集中的项在时间和空间上是有序排列的，输出的结果也是有序的。 比如用户多次购物的购买情况，不同时间点的交易记录就构成了一个购买序列，用户在第一次购买了商品A，第二次购买了商品B和C；那的购物序列. 当N个用户的购买序列就形成了一个规模为N的数据集。可能会发现存在因果关系的规律。 因此序列模式挖掘相对于关联规则挖掘可以挖掘出更加深刻的知识。

PrefixSpan是从输入序列中选取所有满足支持度的频繁子序列。

算法的目标是挖掘出满足最小支持度的频繁序列。从长度为1的前缀开始挖掘序列模式，搜索对应的投影数据库得到长度为1的前缀对应的频繁序列 ，然后递归的挖掘长度为2的前缀所对应的频繁序列，。。。以此类推，一直递归到不能挖掘到更长的前缀挖掘为止。

算法经常用于推荐系统，如电商中的商品推荐、社交媒体中的好友推荐等。

论文《Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach》

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemsCol	项集列名	项集列名	String	✓	所选列类型为[STRING]	
maxPatternLength	最大频繁项集长度	最大频繁项集长度	Integer			10

minConfidence	最小置信度	最小置信度，同时包含X和Y的样本与包含X的样本之比，反映了当样本中包含项集X时，项集Y同时出现的概率。	Double			0.05
minSupportCount	最小支持度数目	最小支持度目，当取值大于或等于0时起作用，当小于0时参数minSupportPercent起作用	Integer			-1
minSupportPercent	最小支持度占比	最小支持度占比，当minSupportCount取值小于0时起作用，当minSupportCount大于或等于0时该参数不起作用	Double			0.02

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a;a,b,c;a,c;d;c,f"],
    ["a,d;c;b,c;a,e"],
    ["e,f;a,b;d,f;c;b"],
    ["e;g;a,f;c;b;c"],
])

data = BatchOperator.fromDataframe(df, schemaStr='sequence string')

prefixSpan = PrefixSpanBatchOp() \
    .setItemsCol("sequence") \
    .setMinSupportCount(3)

prefixSpan.linkFrom(data)
```

```
prefixSpan.print()
prefixSpan.getSideOutput(0).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.associationrule.PrefixSpanBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class PrefixSpanBatchOpTest {
    @Test
    public void testPrefixSpanBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a;a,b,c;a,c;d;c,f"),
            Row.of("a,d;c;b,c;a,e"),
            Row.of("e,f;a,b;d,f;c;b"),
            Row.of("e;g;a,f;c;b;c")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "sequence string");
        BatchOperator <?> prefixSpan = new PrefixSpanBatchOp()
            .setItemsCol("sequence")
            .setMinSupportCount(3);
        prefixSpan.linkFrom(data);
        prefixSpan.print();
        prefixSpan.getSideOutput(0).print();
    }
}
```

其他输入格式示例

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;

public class PrefixSpanBatchOpTest {
    @Test
    public void testPrefixSpan() throws Exception {
```

```
        Row[] rows = new Row[] {
            Row.of("user_a", "1", "2022-03-11 10:12:10", "a"),
            Row.of("user_a", "2", "2022-03-12 10:12:10", "a"),
            Row.of("user_a", "2", "2022-03-12 10:12:10", "b"),
            Row.of("user_a", "2", "2022-03-12 10:12:10", "c"),
            Row.of("user_a", "3", "2022-03-13 10:12:10", "a"),
            Row.of("user_a", "3", "2022-03-13 10:12:10", "c"),
            Row.of("user_a", "4", "2022-03-14 10:12:10", "d"),
            Row.of("user_a", "5", "2022-03-15 10:12:10", "c"),
            Row.of("user_a", "5", "2022-03-15 10:12:10", "f"),
            Row.of("user_b", "1", "2022-03-11 10:12:10", "a"),
            Row.of("user_b", "1", "2022-03-11 10:12:10", "d"),
            Row.of("user_b", "2", "2022-03-12 10:12:10", "c"),
            Row.of("user_b", "3", "2022-03-13 10:12:10", "b"),
            Row.of("user_b", "3", "2022-03-13 10:12:10", "c"),
            Row.of("user_b", "4", "2022-03-14 10:12:10", "a"),
            Row.of("user_b", "4", "2022-03-14 10:12:10", "e"),
            Row.of("user_c", "1", "2022-03-11 10:12:10", "e"),
            Row.of("user_c", "1", "2022-03-11 10:12:10", "f"),
            Row.of("user_c", "2", "2022-03-12 10:12:10", "a"),
            Row.of("user_c", "2", "2022-03-12 10:12:10", "b"),
            Row.of("user_c", "3", "2022-03-13 10:12:10", "d"),
            Row.of("user_c", "3", "2022-03-13 10:12:10", "f"),
            Row.of("user_c", "4", "2022-03-14 10:12:10", "c"),
            Row.of("user_c", "5", "2022-03-15 10:12:10", "b"),
            Row.of("user_d", "1", "2022-03-11 10:12:10", "e"),
            Row.of("user_d", "2", "2022-03-12 10:12:10", "g"),
            Row.of("user_d", "3", "2022-03-13 10:12:10", "a"),
            Row.of("user_d", "3", "2022-03-13 10:12:10", "f"),
            Row.of("user_d", "4", "2022-03-14 10:12:10", "c"),
            Row.of("user_d", "5", "2022-03-15 10:12:10", "b"),
            Row.of("user_d", "6", "2022-03-16 10:12:10", "c")
        };

        BatchOperator op = new MemSourceBatchOp(Arrays.asList(rows), "uid
string,order_id string,occur_time string,item string")
            .groupBy("uid,order_id", "uid,order_id,CONCAT_AGG(item) AS items")
            .orderBy("uid,order_id", -1)
            .groupBy("uid", "uid,CONCAT_AGG(items, ';') AS sequence");

        PrefixSpanBatchOp prefixSpan = new PrefixSpanBatchOp()
            .setItemsCol("sequence")
            .setMinSupportCount(3);

        prefixSpan.linkFrom(op).print();
    }
}
```

输入说明：一个sequence由多个element组成，element之间用分号分隔；一个element由多个item组成，item间用逗号分隔。 由于sequence有次序关系，需要保持原有的顺序，因此在中间组合时加入orderBy操作。

运行结果

itemset	supportcount	itemcount
e	3	1
f	3	1
a	4	1
a;c	4	2
a;c;c	3	3
a;c;b	3	3
a;b	4	2
b	4	1
b;c	3	2
c	4	1
c;c	3	2
c;b	3	2
d	3	1
d;c	3	2

关联规则输出：

rule	chain_length	support	confidence	transaction_count
c=>c	2	0.7500	0.7500	3
c=>b	2	0.7500	0.7500	3
d=>c	2	0.7500	1.0000	3
b=>c	2	0.7500	0.7500	3
a=>c	2	1.0000	1.0000	4
a;c=>c	3	0.7500	0.7500	3
a;c=>b	3	0.7500	0.7500	3
a=>b	2	1.0000	1.0000	4

ALS隐式训练 (AlsImplicitTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.AlsImplicitTrainBatchOp

Python 类名：AlsImplicitTrainBatchOp

功能介绍

隐式反馈对应的ALS算法即：ALS-WR（Alternating Least Squares With Weighted-λ -regularization）。 用于对评分矩阵进行因子分解，然后预测user对item的评分。 它通过观察到的所有用户给产品的打分，来推断每个用户的喜好并向用户推荐适合的产品。

算法原理

很多情况下，用户并没有明确反馈对物品的偏好，需要通过用户的相关行为去推测其对物品的偏好，比如在电商网站中，用户是否点击物品，点击的话在一定程度上表示喜欢， 未点击的话可能是不喜欢，也可能是没有看到该物品。这种形式下的反馈就被称为隐式反馈。即矩阵R为隐式反馈矩阵，引入变量p_ij表示用户u_i对物品v_j的置信度， 如果隐式反馈大于0，置信度为，反之置信度为0。

$$p_{ij} = \begin{cases} 1 & r_{ij} > 0 \\ 0 & r_{ij} = 0 \end{cases}$$

上文也提到了，隐式反馈为0，不代表用户完全不喜欢，也可能是用户没有看到该物品。另外用户点击一个物品，也不代表是喜欢他， 可能是误点，所以需要一个信任等级来显示用户喜欢某个物品，一般情况下，r_ij越大(用户行为的次数)，越能暗示用户喜欢某个物品，因此引入变量c_ij，来衡量p_ij的信任度。

$$c_{ij} = 1 + \alpha r_{ij}$$

α 为置信度系数，那么代价函数变为如下形式：

$$J(U,V) = \sum_{i}^{m}\sum_{j}^{n}[c_{ij}(p_{ij} - u_i v_j^T)^2 + \lambda(||u_i||^2 + ||v_j||^2)]$$

其中：λ 为正则项系数。我们需要找出代价函数最小的U和V。常规的梯度下降算法不能求解。但是先固定U求V，再固定V求U， 如此迭代下去，问题就可以解决了。

算法使用

隐式ALS算法和ALS算法相同，支持输出item或者user的隐向量，我们可以计算出用户或者物品的相似度，继而进行排序得到用户或者item的top N相似user或者item。 这样在数据进行召回时便可以进行召回了。 比如根据用户用行为的物品召回，当用户浏览了若干了item时， 便将这些item相似的item加入到召回池中，进行排序。

参考文献：

1. implicit feedback: Collaborative Filtering for Implicit Feedback Datasets, 2008

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
rateCol	打分列列名	打分列列名	String	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
userCol	User列列名	User列列名	String	✓		
alpha	隐式偏好模型系数alpha	隐式偏好模型系数alpha	Double			40.0
lambda	正则化系数	正则化系数	Double			0.1
nonnegative	是否约束因子非负	是否约束因子非负	Boolean			false
numBlocks	分块数目	分块数目	Integer			1
numIter	迭代次数	迭代次数，默认为10	Integer			10
rank	因子数	因子数	Integer			10

代码示例

Python 代码

```
df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

als =
AlsImplicitTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rat
ing") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)
model.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.AlsImplicitTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsImplicitTrainBatchOpTest {
    @Test
    public void testAlsImplicitTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> als = new
AlsImplicitTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol(
                "rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
```

```
        model.print();
    }
}
```

运行结果

id	p0	p1	p2
-1	{"f0":[["user","factors"],["item","factors"],["user","item"]],"f1":[[2,1],[2,1],[2,3]]}	null	null
2	null	1	1
2	null	2	2
2	null	2	3
2	null	4	1
2	null	4	2
2	null	4	3
1	0.7067459225654602 0.47736793756484985 0.14002709090709686 0.6041285991668701 0.5332542657852173 0.6160406470298767 0.5020460486412048 0.48263704776763916 0.5093641877174377 0.5752177834510803	3	null
0	0.1763894259929657 -0.1366494745016098 0.6774582266807556 -0.4786214232444763 0.4608040153980255 0.4111763834953308 -0.49566400051116943 -0.25755634903907776 -0.17400074005126953 0.17524860799312592	1	null
0	0.2749805450439453 0.14517369866371155 0.15636394917964935 0.13525108993053436 0.25944361090660095 0.2805081903934479 0.0968703031539917 0.12784309685230255 0.15043362975120544 0.22882965207099915	4	null
1	0.7121766805648804 0.30997803807258606 0.5707741379737854 0.1878654509782791 0.7565042972564697 0.7929229140281677 0.09063886851072311 0.23355203866958618 0.3119025230407715 0.6008232831954956	1	null
1	0.7061372399330139 0.4771113395690918 0.1395183950662613 0.603988528251648 0.5325971245765686 0.615354597568512 0.5019887685775757 0.4824497699737549 0.5091073513031006 0.5747033953666687	2	null
0	0.10347702354192734 0.2801591753959656 -0.5076550841331482 0.6058341264724731 -0.19131097197532654 -0.12141165882349014 0.5839534997940063 0.38140058517456055 0.3221108913421631 0.05817580968141556	2	null

ALS：ItemsPerUser推荐 (AlsItemsPerUserRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.AlsItemsPerUserRecommBatchOp

Python 类名：AlsItemsPerUserRecommBatchOp

功能介绍

使用ALS (Alternating Lease Square）训练的模型为 user 推荐 items。这里的ALS模型可以是隐式模型，也可以是显式模型，输出格式是MTable。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean			false
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)
predictor = AlsItemsPerUserRecommBatchOp() \
    .setUserCol("user").setRecommCol("rec").setK(1).setReservedCols(["user"])

predictor.linkFrom(model, data).print();
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.AlsItemsPerUserRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsItemsPerUserRecommBatchOpTest {
    @Test
    public void testAlsItemsPerUserRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
```

```
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> als = new
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
        BatchOperator <?> predictor = new AlsItemsPerUserRecommBatchOp()

.setUserCol("user").setRecommCol("rec").setK(1).setReservedCols("user");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	rec
1	{"object":"[1]","rate":"[0.5796224474906921]"}
2	{"object":"[2]","rate":"[0.7668506503105164]"}
2	{"object":"[2]","rate":"[0.7668506503105164]"}
4	{"object":"[1]","rate":"[0.5744813084602356]"}
4	{"object":"[1]","rate":"[0.5744813084602356]"}
4	{"object":"[1]","rate":"[0.5744813084602356]"}

ALS：打分推荐推荐 (AlsRateRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.AlsRateRecommBatchOp

Python 类名：AlsRateRecommBatchOp

功能介绍

ALS打分预测，可对每一个（user，item）输入对进行评分预测。这里的ALS模型可以是隐式模型，也可以是显式模型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])
```

```
data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)
predictor = AlsRateRecommBatchOp()\
    .setUserCol("user").setItemCol("item").setRecommCol("predicted_rating")

model = als.linkFrom(data)
predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.AlsRateRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsRateRecommBatchOpTest {
    @Test
    public void testAlsRateRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> als = new
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator <?> predictor = new AlsRateRecommBatchOp()

.setUserCol("user").setItemCol("item").setRecommCol("predicted_rating");
        BatchOperator model = als.linkFrom(data);
        predictor.linkFrom(model, data).print();
```

```
    }
}
```

运行结果

user	item	rating	predicted_rating
1	1	0.6000	0.5810
2	2	0.8000	0.7669
2	3	0.6000	0.5809
4	1	0.6000	0.5753
4	2	0.3000	0.2989
4	3	0.4000	0.3833

ALS：相似items推荐 (AlsSimilarItemsRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.AlsSimilarItemsRecommBatchOp

Python 类名：AlsSimilarItemsRecommBatchOp

功能介绍

使用ALS (Alternating Lease Square）model 对相似的item的进行推荐。这里的ALS模型可以是隐式模型，也可以是显式模型，输出格式是MTable。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)
predictor = AlsSimilarItemsRecommBatchOp() \
    .setItemCol("item").setRecommCol("rec").setK(1).setReservedCols(["item"])

predictor.linkFrom(model, data).print();
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.AlsSimilarItemsRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsSimilarItemsRecommBatchOpTest {
    @Test
    public void testAlsSimilarItemsRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> als = new
```

```
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
        BatchOperator <?> predictor = new AlsSimilarItemsRecommBatchOp()

.setItemCol("item").setRecommCol("rec").setK(1).setReservedCols("item");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

item	rec
1	{"object":"[3]","score":"[0.8821980357170105]"}
2	{"object":"[3]","score":"[0.9917739629745483]"}
3	{"object":"[2]","score":"[0.9917739629745483]"}
1	{"object":"[3]","score":"[0.8821980357170105]"}
2	{"object":"[3]","score":"[0.9917739629745483]"}
3	{"object":"[2]","score":"[0.9917739629745483]"}

ALS：相似users推荐 (AlsSimilarUsersRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.AlsSimilarUsersRecommBatchOp

Python 类名：AlsSimilarUsersRecommBatchOp

功能介绍

使用ALS (Alternating Lease Square）model 对相似的user的进行推荐。这里的ALS模型可以是隐式模型，也可以是显式模型，输出格式是MTable。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)
predictor = AlsSimilarUsersRecommBatchOp() \
    .setUserCol("user").setRecommCol("rec").setK(1).setReservedCols(["user"])

predictor.linkFrom(model, data).print();
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.AlsSimilarUsersRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsSimilarUsersRecommBatchOpTest {
    @Test
    public void testAlsSimilarUsersRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> als = new
```

```
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
        BatchOperator <?> predictor = new AlsSimilarUsersRecommBatchOp()

.setUserCol("user").setRecommCol("rec").setK(1).setReservedCols("user");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	rec
1	{"object":"[4]","score":"[0.2515771985054016]"}
2	{"object":"[1]","score":"[0.17212671041488647]"}
2	{"object":"[1]","score":"[0.17212671041488647]"}
4	{"object":"[1]","score":"[0.2515771985054016]"}
4	{"object":"[1]","score":"[0.2515771985054016]"}
4	{"object":"[1]","score":"[0.2515771985054016]"}

ALS训练 (AlsTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp

Python 类名：AlsTrainBatchOp

功能介绍

ALS (Alternating Lease Square）交替最小二乘法是一种model based的协同过滤算法， 用于对评分矩阵进行因子分解，然后预测user对item的评分。 它通过观察到的所有用户给产品的打分，来推断每个用户的喜好并向用户推荐适合的产品。

算法原理

推荐所使用的数据可以抽象成一个[m,n]的矩阵R，R的每一行代表m个用户对所有电影的评分， n列代表每部电影对应的得分。R是个稀疏矩阵，一个用户只是对所有电影中的一小部分看过， 有评分。通过矩阵分解方法，我可以把这个低秩的矩阵，分解成两个小矩阵的点乘。公式如下：

$$R_{(m,n)} = U_{(m,k)} * V^T_{(n,k)}$$

有了这个矩阵分解公式，我们可以定义代价函数：

$$J(U,V) = \sum_{i}^{m}\sum_{j}^{n}[(r_{ij} - u_i v_j^T)^2 + \lambda(||u_i||^2 + ||v_j||^2)]$$

其中：λ 为正则项系数。我们需要找出代价函数最小的U和V。常规的梯度下降算法不能求解。但是先固定U求V，再固定V求U， 如此迭代下去，问题就可以解决了。

算法使用

ALS算法支持输出item或者user的隐向量，我们可以计算出用户或者物品的相似度，继而进行排序得到用户或者item的top N相似user或者item。这样在数据进行召回时便可以进行召回了。 比如根据用户用行为的物品召回，当用户浏览了若干了item时， 便将这些item相似的item加入到召回池中，进行排序。

参考文献：

1. explicit feedback: Large-scale Parallel Collaborative Filtering for the Netflix Prize, 2007

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
rateCol	打分列列名	打分列列名	String	✓	所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
userCol	User列列名	User列列名	String	✓		
lambda	正则化系数	正则化系数	Double			0.1
nonnegative	是否约束因子非负	是否约束因子非负	Boolean			false
numBlocks	分块数目	分块数目	Integer			1
numIter	迭代次数	迭代次数，默认为10	Integer			10
rank	因子数	因子数	Integer			10

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)
model.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsTrainBatchOpTest {
    @Test
    public void testAlsTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> als = new
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
        model.print();
```

```
    }
}
```

运行结果

id	p0	p1	p2
-1	{"f0":[["user","factors"],["item","factors"],["user","item"]],"f1":[[2,1],[2,1],[2,3]]}	null	null
2	null	1	1
2	null	2	2
2	null	2	3
2	null	4	1
2	null	4	2
2	null	4	3
0	0.15332114696502686 0.2907584309577942 0.08023820072412491 0.12368439137935638 0.16154064238071442 -0.0482657290995121 0.12886907160282135 -0.104159414768219 0.15081298351287842 0.22745263576507568	4	null
1	0.4621395170688629 0.16237080097198486 0.07094596326351166 0.38359203934669495 0.3225877583026886 0.5486094355583191 0.2962109446525574 0.46471306681632996 0.29413706064224243 0.29865172505378723	2	null
1	0.38991987705230713 0.3296423852443695 0.10596991330385208 0.3207378387451172 0.3165116012096405 0.2756109833717346 0.2748037576675415 0.18200615048408508 0.29145893454551697 0.35637563467025757	3	null
0	0.1556907445192337 0.2930602431297302 0.08095365017652512 0.12562905251979828 0.16353283822536469 -0.046881016343832016 0.13057765364646912 -0.10337890684604645 0.15265130996704102 0.2297801822423935	1	null
1	0.3398374617099762 0.6423624157905579 0.17734453082084656 0.2741791605949402 0.35757142305374146 -0.10493437945842743 0.28536728024482727 -0.22857370972633362 0.3338049352169037 0.5030093193054199	1	null
0	0.276202917098999 0.08905492722988129 0.04048972949385643 0.22937875986099243 0.19095991551876068 0.3356474041938782 0.1760021150112152 0.28645196557044983 0.17399948835372925 0.17416398227214813	2	null

ALS：UsersPerItem推荐 (AlsUsersPerItemRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.AlsUsersPerItemRecommBatchOp

Python 类名：AlsUsersPerItemRecommBatchOp

功能介绍

使用ALS (Alternating Lease Square）model 为item 推荐users。这里的ALS模型可以是隐式模型，也可以是显式模型，输出格式是MTable。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean			false
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)
predictor = AlsUsersPerItemRecommBatchOp() \
    .setItemCol("item").setRecommCol("rec").setK(1).setReservedCols(["item"])

predictor.linkFrom(model, data).print();
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import
com.alibaba.alink.operator.batch.recommendation.AlsUsersPerItemRecommBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsUsersPerItemRecommBatchOpTest {
    @Test
    public void testAlsUsersPerItemRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
```

```
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> als = new
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
        BatchOperator <?> predictor = new AlsUsersPerItemRecommBatchOp()

.setItemCol("item").setRecommCol("rec").setK(1).setReservedCols("item");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	rec
1	{"object":"[1]","rate":"[0.5796224474906921]"}
2	{"object":"[2]","rate":"[0.7668506503105164]"}
3	{"object":"[2]","rate":"[0.5810791850090027]"}
1	{"object":"[1]","rate":"[0.5796224474906921]"}
2	{"object":"[2]","rate":"[0.7668506503105164]"}
3	{"object":"[2]","rate":"[0.5810791850090027]"}

展开KObject (FlattenKObjectBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.FlattenKObjectBatchOp

Python 类名：FlattenKObjectBatchOp

功能介绍

将推荐结果从json序列化格式转为table格式。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCols	输出结果列列名数组	输出结果列列名数组，必选	String[]	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
outputColTypes	输出结果列列类型数组	输出结果列类型数组	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])
```

```
data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

jsonData = Zipped2KObjectBatchOp()\
            .setGroupCol("user")\
            .setObjectCol("item")\
            .setInfoCols(["rating"])\
            .setOutputCol("recomm")\
            .linkFrom(data)\
            .lazyPrint(-1);
recList = FlattenKObjectBatchOp()\
            .setSelectedCol("recomm")\
            .setOutputCols(["item", "rating"])\
            .setOutputColTypes(["long", "double"])\
            .setReservedCols(["user"])\
            .linkFrom(jsonData)\
            .lazyPrint(-1);
BatchOperator.execute();
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.FlattenKObjectBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.recommendation.Zipped2KObjectBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FlattenKObjectBatchOpTest {
    @Test
    public void testFlattenKObjectBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> jsonData = new Zipped2KObjectBatchOp()
```

```
            .setGroupCol("user")
            .setObjectCol("item")
            .setInfoCols("rating")
            .setOutputCol("recomm")
            .linkFrom(data)
            .lazyPrint(-1);
        BatchOperator <?> recList = new FlattenKObjectBatchOp()
            .setSelectedCol("recomm")
            .setOutputCols("item", "rating")
            .setOutputColTypes("long", "double")
            .setReservedCols("user")
            .linkFrom(jsonData)
            .lazyPrint(-1);
        BatchOperator.execute();
    }
}
```

运行结果

user	recomm
1	{"item":"[1]","rating":"[0.6]"}
4	{"item":"[1,2,3]","rating":"[0.6,0.3,0.4]"}
2	{"item":"[2,3]","rating":"[0.8,0.6]"}

user	item	rating
1	1	0.6000
4	1	0.6000
4	2	0.3000
4	3	0.4000
2	2	0.8000
2	3	0.6000

FM：ItemsPerUser推荐 (FmItemsPerUserRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.FmItemsPerUserRecommBatchOp

Python 类名：FmItemsPerUserRecommBatchOp

功能介绍

使用Fm推荐模型，为user推荐item list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean			false
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = FmRecommTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setNumFactor(20)\
    .setRateCol("rating").linkFrom(data);

predictor = FmItemsPerUserRecommBatchOp()\
    .setUserCol("user")\
    .setK(1).setReservedCols(["user"])\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.FmItemsPerUserRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.FmRecommTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmItemsPerUserRecommBatchOpTest {
    @Test
```

```
    public void testFmItemsPerUserRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new FmRecommTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setNumFactor(20)
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new FmItemsPerUserRecommBatchOp()
            .setUserCol("user")
            .setK(1).setReservedCols("user")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	prediction_result
1	{"object":"[1]","rate":"[0.5829579830169678]"}
2	{"object":"[2]","rate":"[0.576914370059967]"}
2	{"object":"[2]","rate":"[0.576914370059967]"}
4	{"object":"[1]","rate":"[0.5055253505706787]"}
4	{"object":"[1]","rate":"[0.5055253505706787]"}
4	{"object":"[1]","rate":"[0.5055253505706787]"}

FM：打分推荐 (FmRateRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.FmRateRecommBatchOp

Python 类名：FmRateRecommBatchOp

功能介绍

Fm 推荐打分是使用Fm推荐模型，预测user对item的评分。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])
```

```
data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = FmRecommTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setNumFactor(20)\
    .setRateCol("rating").linkFrom(data);

predictor = FmRateRecommBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.FmRateRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.FmRecommTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmRateRecommBatchOpTest {
    @Test
    public void testFmRateRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new FmRecommTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setNumFactor(20)
            .setRateCol("rating").linkFrom(data);
```

```
        BatchOperator <?> predictor = new FmRateRecommBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	item	rating	prediction_result
1	1	0.6	0.582958
2	2	0.8	0.576914
2	3	0.6	0.508942
4	1	0.6	0.505525
4	2	0.3	0.372908
4	3	0.4	0.347927

FM二分类隐式训练 (FmRecommBinaryImplicitTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.FmRecommBinaryImplicitTrainBatchOp

Python 类名：FmRecommBinaryImplicitTrainBatchOp

功能介绍

Fm 隐式推荐是使用Fm算法在推荐场景的一种扩展，用给定user-item pair 及user和item的特征信息，训练一个推荐专用的Fm模型， 用于预测user对item的评分、对user推荐itemlist，或者对item推荐userlist。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
userCol	User列列名	User列列名	String	✓		
initStdev	初始化参数的标准差	初始化参数的标准差	Double			0.05
itemCategoricalFeatureCols	item离散值列名字数组	item离散值列名字数组	String[]			[]
itemFeatureCols	item特征列名字数组	item特征列名字数组	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	[]

lambda0	常数项正则化系数	常数项正则化系数	Double			0.0
lambda1	线性项正则化系数	线性项正则化系数	Double			0.0
lambda2	二次项正则化系数	二次项正则化系数	Double			0.0
learnRate	学习率	学习率	Double			0.01
numEpochs	epoch数	epoch数	Integer			10
numFactor	因子数	因子数	Integer			10
rateCol	打分列列名	打分列列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
userCategoricalFeatureCols	用户离散值列名字数组	用户离散值列名字数组	String[]			[]
userFeatureCols	用户特征列名字数组	用户特征列名字数组	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	[]
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

withLinearItem	是否含有线性项	是否含有线性项	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = FmRecommBinaryImplicitTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setNumFactor(20).linkFrom(data);

predictor = FmUsersPerItemRecommBatchOp()\
    .setItemCol("user")\
    .setK(2).setReservedCols(["item"])\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.FmRecommBinaryImplicitTrainBatc
```

```
hOp;
import
com.alibaba.alink.operator.batch.recommendation.FmUsersPerItemRecommBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmRecommBinaryImplicitTrainBatchOpTest {
    @Test
    public void testFmRecommBinaryImplicitTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new FmRecommBinaryImplicitTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setNumFactor(20).linkFrom(data);
        BatchOperator <?> predictor = new FmUsersPerItemRecommBatchOp()
            .setItemCol("user")
            .setK(2).setReservedCols("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

item	prediction_result
1	{"object":"[1]","rate":"[0.6802429556846619]"}
2	{"object":"[2]","rate":"[0.6637783646583557]"}
3	{"object":"[2]","rate":"[0.6637783646583557]"}
1	{"object":"[]","rate":"[]"}
2	{"object":"[]","rate":"[]"}
3	{"object":"[]","rate":"[]"}

FM推荐训练 (FmRecommTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.FmRecommTrainBatchOp

Python 类名：FmRecommTrainBatchOp

功能介绍

Fm 推荐是使用Fm算法在推荐场景的一种扩展，用给定打分数据及user和item的特征信息，训练一个推荐专用的Fm模型， 用于预测user对item的评分、对user推荐itemlist，或者对item推荐userlist。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
rateCol	打分列列名	打分列列名	String	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
userCol	User列列名	User列列名	String	✓		
initStdev	初始化参数的标准差	初始化参数的标准差	Double			0.05
itemCategoricalFeatureCols	item离散值列名字数组	item离散值列名字数组	String[]			[]

itemFeatureCols	item特征列名字数组	item特征列名字数组	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	[]
lambda0	常数项正则化系数	常数项正则化系数	Double			0.0
lambda1	线性项正则化系数	线性项正则化系数	Double			0.0
lambda2	二次项正则化系数	二次项正则化系数	Double			0.0
learnRate	学习率	学习率	Double			0.01
numEpochs	epoch数	epoch数	Integer			10
numFactor	因子数	因子数	Integer			10
userCategoricalFeatureCols	用户离散值列名字数组	用户离散值列名字数组	String[]			[]
userFeatureCols	用户特征列名字数组	用户特征列名字数组	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	[]
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

withLinearItem	是否含有线性项	是否含有线性项	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = FmRecommTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setNumFactor(20)\
    .setRateCol("rating").linkFrom(data);

predictor = FmRateRecommBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.FmRateRecommBatchOp;
```

```
import com.alibaba.alink.operator.batch.recommendation.FmRecommTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmRecommTrainBatchOpTest {
    @Test
    public void testFmRecommTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new FmRecommTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setNumFactor(20)
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new FmRateRecommBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	item	rating	prediction_result
1	1	0.6	0.582958
2	2	0.8	0.576914
2	3	0.6	0.508942
4	1	0.6	0.505525
4	2	0.3	0.372908
4	3	0.4	0.347927

FM：UsersPerItem推荐 (FmUsersPerItemRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.FmUsersPerItemRecommBatchOp

Python 类名：FmUsersPerItemRecommBatchOp

功能介绍

使用Fm推荐模型，为item推荐user list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean			false
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = FmRecommTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setNumFactor(20)\
    .setRateCol("rating").linkFrom(data);

predictor = FmRateRecommBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()

model = FmRecommTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setNumFactor(20)\
    .setRateCol("rating").linkFrom(data);

predictor = FmUsersPerItemRecommBatchOp()\
    .setItemCol("user")\
    .setK(1).setReservedCols(["item"])\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.FmRateRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.FmRecommTrainBatchOp;
import
com.alibaba.alink.operator.batch.recommendation.FmUsersPerItemRecommBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmUsersPerItemRecommBatchOpTest {
    @Test
    public void testFmUsersPerItemRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new FmRecommTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setNumFactor(20)
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new FmRateRecommBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
        model = new FmRecommTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setNumFactor(20)
            .setRateCol("rating").linkFrom(data);
        predictor = new FmUsersPerItemRecommBatchOp()
            .setItemCol("user")
            .setK(1).setReservedCols("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

item	prediction_result
1	{"object":"[1]","rate":"[0.5829579830169678]"}
2	{"object":"[2]","rate":"[0.576914370059967]"}
3	{"object":"[1]","rate":"[0.5055253505706787]"}
1	{"object":"[1]","rate":"[0.5829579830169678]"}
2	{"object":"[2]","rate":"[0.576914370059967]"}
3	{"object":"[1]","rate":"[0.5055253505706787]"}

ItemCf：ItemsPerUser推荐 (ItemCfItemsPerUserRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.ItemCfItemsPerUserRecommBatchOp

Python 类名：ItemCfItemsPerUserRecommBatchOp

功能介绍

用ItemCF模型 为user推荐item list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean			false
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = ItemCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = ItemCfItemsPerUserRecommBatchOp()\
    .setUserCol("user")\
    .setReservedCols(["user"])\
    .setK(1)\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.ItemCfItemsPerUserRecommBatchOp
;
import com.alibaba.alink.operator.batch.recommendation.ItemCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ItemCfItemsPerUserRecommBatchOpTest {
```

```
    @Test
    public void testItemCfItemsPerUserRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new ItemCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new ItemCfItemsPerUserRecommBatchOp()
            .setUserCol("user")
            .setReservedCols("user")
            .setK(1)
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	prediction_result
1	MTable(1,2)(item,score) 3 | 0.2353
2	MTable(1,2)(item,score) 3 | 0.3895
2	MTable(1,2)(item,score) 3 | 0.3895
4	MTable(1,2)(item,score) 2 | 0.1795
4	MTable(1,2)(item,score) 2 | 0.1795
4	MTable(1,2)(item,score) 2 | 0.1795

ItemCf：打分推荐 (ItemCfRateRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.ItemCfRateRecommBatchOp

Python 类名：ItemCfRateRecommBatchOp

功能介绍

ItemCF 打分是使用ItemCF模型，于预测user对item的评分。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])
```

```
data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = ItemCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = ItemCfRateRecommBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.ItemCfRateRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.ItemCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ItemCfRateRecommBatchOpTest {
    @Test
    public void testItemCfRateRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new ItemCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new ItemCfRateRecommBatchOp()
            .setUserCol("user")
```

```
            .setItemCol("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	item	rating	prediction_result
1	1	0.6000	0.0000
2	2	0.8000	0.6000
2	3	0.6000	0.8000
4	1	0.6000	0.3612
4	2	0.3000	0.4406
4	3	0.4000	0.3861

ItemCf：相似items推荐 (ItemCfSimilarItemsRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.ItemCfSimilarItemsRecommBatchOp

Python 类名：ItemCfSimilarItemsRecommBatchOp

功能介绍

用itemCF模型为item推荐相似的item list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
```

```
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = ItemCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = ItemCfSimilarItemsRecommBatchOp()\
    .setItemCol("item")\
    .setReservedCols(["item"])\
    .setK(1)\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.ItemCfSimilarItemsRecommBatchOp
;
import com.alibaba.alink.operator.batch.recommendation.ItemCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ItemCfSimilarItemsRecommBatchOpTest {
    @Test
    public void testItemCfSimilarItemsRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new ItemCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new ItemCfSimilarItemsRecommBatchOp()
            .setItemCol("item")
            .setReservedCols("item")
            .setK(1)
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

item	prediction_result
1	{"item":"[3]","similarities":"[0.3922322702763681]"}
2	{"item":"[3]","similarities":"[0.9738412097417931]"}
3	{"item":"[2]","similarities":"[0.9738412097417931]"}
1	{"item":"[3]","similarities":"[0.3922322702763681]"}
2	{"item":"[3]","similarities":"[0.9738412097417931]"}
3	{"item":"[2]","similarities":"[0.9738412097417931]"}

ItemCf训练 (ItemCfTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.ItemCfTrainBatchOp

Python 类名：ItemCfTrainBatchOp

功能介绍

ItemCF 是一种被广泛使用的协同过滤算法，用给定打分数据训练一个推荐模型， 用于预测user对item的评分、对user喜欢的itemlist，或者对item推荐可能的userlist等。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
userCol	User列列名	User列列名	String	✓		
maxNeighborNumber	保存相似item的数目	保存相似item的数目，该参数设置后将降低内存使用量，同时可能会降低训练速度	Integer			64
rateCol	打分列列名	打分列列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

similarityThreshold	相似阈值	只有大于该阈值的Object才会被计算	Double			1.0E-4
similarityType	距离度量方式	聚类使用的距离类型	String		"COSINE", "JACCARD", "PEARSON"	"COSINE"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = ItemCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = ItemCfRateRecommBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.ItemCfRateRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.ItemCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ItemCfTrainBatchOpTest {
    @Test
    public void testItemCfTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new ItemCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new ItemCfRateRecommBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	item	rating	prediction_result
1	1	0.6000	0.0000
2	2	0.8000	0.6000
2	3	0.6000	0.8000
4	1	0.6000	0.3612

4	2	0.3000	0.4406
4	3	0.4000	0.3861

ItemCf：UsersPerItem推荐 (ItemCfUsersPerItemRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.ItemCfUsersPerItemRecommBatchOp

Python 类名：ItemCfUsersPerItemRecommBatchOp

功能介绍

用ItemCF模型 为item推荐user list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean			false
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = ItemCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = ItemCfUsersPerItemRecommBatchOp()\
    .setItemCol("item")\
    .setReservedCols(["item"])\
    .setK(1)\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.ItemCfTrainBatchOp;
import
com.alibaba.alink.operator.batch.recommendation.ItemCfUsersPerItemRecommBatchOp
;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ItemCfUsersPerItemRecommBatchOpTest {
```

```
    @Test
    public void testItemCfUsersPerItemRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new ItemCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new ItemCfUsersPerItemRecommBatchOp()
            .setItemCol("item")
            .setReservedCols("item")
            .setK(1)
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

item	prediction_result
1	{"user":"[2]","score":"[0.21698238771519462]"}
2	{"user":"[2]","score":"[0.29215236292253793]"}
3	{"user":"[2]","score":"[0.38953648389671724]"}
1	{"user":"[2]","score":"[0.21698238771519462]"}
2	{"user":"[2]","score":"[0.29215236292253793]"}
3	{"user":"[2]","score":"[0.38953648389671724]"}

推荐结果采样处理 (LeaveKObjectOutBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.LeaveKObjectOutBatchOp

Python 类名：LeaveKObjectOutBatchOp

功能介绍

将推荐结果分成两个表。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
groupCol	分组列	分组单列名，必选	String	✓		
objectCol	Object列列名	Object列列名	String	✓		
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
fraction	拆分到测试集最大数据比例	拆分到测试集最大数据比例	Double		[0.0, 1.0]	1.0
k	推荐TOP数量	推荐TOP数量	Integer			10

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 0, 0.6],
    [6, 4, 0.3],
    [4, 7, 0.4],
```

```
    [2, 6, 0.6],
    [4, 5, 0.6],
    [4, 6, 0.3],
    [4, 3, 0.4]
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

spliter = LeaveKObjectOutBatchOp()\
            .setK(2)\
            .setGroupCol("user")\
            .setObjectCol("item")\
            .setOutputCol("label")
spliter.linkFrom(data).print()
spliter.getSideOutput(0).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.LeaveKObjectOutBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LeaveKObjectOutBatchOpTest {
    @Test
    public void testLeaveKObjectOutBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 0, 0.6),
            Row.of(6, 4, 0.3),
            Row.of(4, 7, 0.4),
            Row.of(2, 6, 0.6),
            Row.of(4, 5, 0.6),
            Row.of(4, 6, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> spliter = new LeaveKObjectOutBatchOp()
```

```
            .setK(2)
            .setGroupCol("user")
            .setObjectCol("item")
            .setOutputCol("label");
        spliter.linkFrom(data).print();
        spliter.getSideOutput(0).print();
    }
}
```

运行结果

user	label
1	{"item":"[1]"}
6	{"item":"[4]"}
4	{"item":"[7,3]"}
2	{"item":"[2,6]"}

user	item	rating
4	5	0.6000
4	3	0.4000
4	0	0.6000
2	3	0.6000

推荐结果TopK采样处理 (LeaveTopKObjectOutBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.LeaveTopKObjectOutBatchOp

Python 类名：LeaveTopKObjectOutBatchOp

功能介绍

将推荐结果按取topK部分作为一个输出表。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
groupCol	分组列	分组单列名，必选	String	✓		
objectCol	Object列列名	Object列列名	String	✓		
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
rateCol	打分列列名	打分列列名	String	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
fraction	拆分到测试集最大数据比例	拆分到测试集最大数据比例	Double		[0.0, 1.0]	1.0
k	推荐TOP数量	推荐TOP数量	Integer			10
rateThreshold	打分阈值	打分阈值	Double			-Infinity

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 0, 0.6],
    [6, 4, 0.3],
    [4, 7, 0.4],
    [2, 6, 0.6],
    [4, 5, 0.6],
    [4, 6, 0.3],
    [4, 3, 0.4]
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

spliter = LeaveTopKObjectOutBatchOp()\
            .setK(2)\
            .setGroupCol("user")\
            .setObjectCol("item")\
            .setOutputCol("label")\
            .setRateCol("rating")
spliter.linkFrom(data).print()
spliter.getSideOutput(0).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.LeaveTopKObjectOutBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class LeaveTopKObjectOutBatchOpTest {
    @Test
    public void testLeaveTopKObjectOutBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 0, 0.6),
            Row.of(6, 4, 0.3),
            Row.of(4, 7, 0.4),
            Row.of(2, 6, 0.6),
            Row.of(4, 5, 0.6),
            Row.of(4, 6, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> spliter = new LeaveTopKObjectOutBatchOp()
            .setK(2)
            .setGroupCol("user")
            .setObjectCol("item")
            .setOutputCol("label")
            .setRateCol("rating");
        spliter.linkFrom(data).print();
        spliter.getSideOutput(0).print();
    }
}
```

运行结果

user	label
1	{"item":"[1]","rating":"[0.6]"}
6	{"item":"[4]","rating":"[0.3]"}
4	{"item":"[0,5]","rating":"[0.6,0.6]"}
2	{"item":"[2,3]","rating":"[0.8,0.6]"}

user	item	rating
4	7	0.4000
4	3	0.4000
4	6	0.3000
2	6	0.6000

推荐负采样 (NegativeItemSamplingBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.NegativeItemSamplingBatchOp

Python 类名：NegativeItemSamplingBatchOp

功能介绍

当给定user-item pair数据的时候，为数据生成若干负样本数据，构成训练数据。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
samplingFactor	采样因子	采样因子	Integer			3

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
      [1, 1],
      [2, 2],
      [2, 3],
      [4, 1],
      [4, 2],
      [4, 3],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint')

NegativeItemSamplingBatchOp().linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.NegativeItemSamplingBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class NegativeItemSamplingBatchOpTest {
    @Test
    public void testNegativeItemSamplingBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1),
            Row.of(2, 2),
            Row.of(2, 3),
            Row.of(4, 1),
            Row.of(4, 2),
            Row.of(4, 3)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int");
        new NegativeItemSamplingBatchOp().linkFrom(data).print();
    }
}
```

运行结果

user	item	label
2	1	0
1	3	0
4	1	1
4	2	1
1	3	0
2	1	0
2	1	0
4	3	1
2	2	1

2	3	1
2	1	0
1	1	1
2	1	0
1	3	0
2	1	0

推荐组件：精排 (RecommendationRankingBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.RecommendationRankingBatchOp

Python 类名：RecommendationRankingBatchOp

功能介绍

该组件功能是对召回的结果进行排序，并输出排序后的TopK个object，此处排序算法用户可以通过创建PipelineModel的方式定制，具体使用方式参见代码示例。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
mTableCol	MTable 列名	召回列表列	String	✓	所选列类型为[M_TABLE]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
rankingCol	用来排序的得分列	用来排序的得分列	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
topN	前N的数据	挑选最近的N个数据	Integer		[1, +inf)	10

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import pandas as pd
```

```
data = pd.DataFrame([["u6", "0.0 1.0", 0.0, 1.0, 1, "{\"data\":{\"iid\":
[18,19,88]},\"schema\":\"iid INT\"}"]])
predData = BatchOperator.fromDataframe(data, schemaStr='uid string, uf string,
f0 double, f1 double, labels int, ilist string')
predData = predData.link(ToMTableBatchOp().setSelectedCol("ilist"))
data = pd.DataFrame([
            ["u0", "1.0 1.0", 1.0, 1.0, 1, 18],
            ["u1", "1.0 1.0", 1.0, 1.0, 0, 19],
            ["u2", "1.0 0.0", 1.0, 0.0, 1, 88],
            ["u3", "1.0 0.0", 1.0, 0.0, 0, 18],
            ["u4", "0.0 1.0", 0.0, 1.0, 1, 88],
            ["u5", "0.0 1.0", 0.0, 1.0, 0, 19],
            ["u6", "0.0 1.0", 0.0, 1.0, 1, 88]]);
trainData = BatchOperator.fromDataframe(data, schemaStr='uid string, uf string,
f0 double, f1 double, labels int, iid string')
oneHotCols = ["uid", "f0", "f1", "iid"]
multiHotCols = ["uf"]
pipe = Pipeline() \
    .add( \
        OneHotEncoder() \
            .setSelectedCols(oneHotCols) \
            .setOutputCols(["ovec"])) \
    .add( \
        MultiHotEncoder().setDelimiter(" ") \
            .setSelectedCols(multiHotCols) \
            .setOutputCols(["mvec"])) \
    .add( \
        VectorAssembler() \
            .setSelectedCols(["ovec", "mvec"]) \
            .setOutputCol("vec")) \
    .add(
        LogisticRegression() \
            .setVectorCol("vec") \
            .setLabelCol("labels") \
            .setReservedCols(["uid", "iid"]) \
            .setPredictionDetailCol("detail") \
            .setPredictionCol("pred")) \
    .add( \
        JsonValue() \
            .setSelectedCol("detail") \
            .setJsonPath(["$.1"]) \
            .setOutputCols(["score"]))
lrModel = pipe.fit(trainData)
rank = RecommendationRankingBatchOp()\
            .setMTableCol("ilist")\
            .setOutputCol("il")\
            .setTopN(2)\
```

```
            .setRankingCol("score")\
            .setReservedCols(["uid", "labels"])
rank.linkFrom(lrModel.save(), predData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.Pipeline;
import com.alibaba.alink.pipeline.classification.LogisticRegression;
import com.alibaba.alink.pipeline.dataproc.JsonValue;
import com.alibaba.alink.pipeline.dataproc.vector.VectorAssembler;
import com.alibaba.alink.pipeline.feature.MultiHotEncoder;
import com.alibaba.alink.pipeline.feature.OneHotEncoder;
import org.junit.Test;

import java.util.Arrays;

public class RecommendationRankingTest {

    @Test
    public void test() throws Exception {

        Row[] predArray = new Row[] {
            Row.of("u6", "0.0 1.0", 0.0, 1.0, 1, "{\"data\":{\"iid\":
[18,19,88]},"
                + "\"schema\":\"iid INT\"}")
        };

        Row[] trainArray = new Row[] {
            Row.of("u0", "1.0 1.0", 1.0, 1.0, 1, 18),
            Row.of("u1", "1.0 1.0", 1.0, 1.0, 0, 19),
            Row.of("u2", "1.0 0.0", 1.0, 0.0, 1, 88),
            Row.of("u3", "1.0 0.0", 1.0, 0.0, 1, 18),
            Row.of("u4", "0.0 1.0", 0.0, 1.0, 1, 88),
            Row.of("u5", "0.0 1.0", 0.0, 1.0, 1, 19),
            Row.of("u6", "0.0 1.0", 0.0, 1.0, 1, 88)
        };
        BatchOperator <?> trainData = new
MemSourceBatchOp(Arrays.asList(trainArray),
                new String[] {"uid", "uf", "f0", "f1", "labels", "iid"});
        BatchOperator <?> predData =  new
MemSourceBatchOp(Arrays.asList(predArray),
                new String[] {"uid", "uf", "f0", "f1", "labels", "ilist"});
```

```
        String[] oneHotCols = new String[] {"uid", "f0", "f1", "iid"};
        String[] multiHotCols = new String[] {"uf"};

        Pipeline pipe = new Pipeline()
            .add(
                new OneHotEncoder()
                    .setSelectedCols(oneHotCols)
                    .setOutputCols("ovec"))
            .add(
                new MultiHotEncoder().setDelimiter(" ")
                    .setSelectedCols(multiHotCols)
                    .setOutputCols("mvec"))
            .add(
                new VectorAssembler()
                    .setSelectedCols("ovec", "mvec")
                    .setOutputCol("vec"))
            .add(
                new LogisticRegression()
                    .setVectorCol("vec")
                    .setLabelCol("labels")
                    .setReservedCols("uid", "iid")
                    .setPredictionDetailCol("detail")
                    .setPredictionCol("pred"))
            .add(
                new JsonValue()
                    .setSelectedCol("detail")
                    .setJsonPath("$.1")
                    .setOutputCols("score"));
        RecommendationRankingBatchOp rank = new RecommendationRankingBatchOp()
            .setMTableCol("ilist")
            .setOutputCol("ilist")
            .setTopN(2)
            .setRankingCol("score")
            .setReservedCols("uid", "labels");
        rank.linkFrom(pipe.fit(trainData).save(), predData).print();
    }
}
```

运行结果

uid	uf	f0	f1	labels	ilist
u6	0.0 1.0	0.0000	1.0000	1	{"data":{"iid":[18,88],"score": [0.9999999999999553,0.9999999999999472]},"schema":" INT,score DOUBLE"}

swing推荐 (SwingRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.SwingRecommBatchOp

Python 类名：SwingRecommBatchOp

功能介绍

Swing 是一种被广泛使用的item召回算法，算法详细介绍可以参考SwingTrainBatchOp组件。

该组件为Swing的批处理预测组件，输入为 SwingTrainBatchOp 输出的模型和要预测的item列。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
      ["a1", "11L", 2.2],
      ["a1", "12L", 2.0],
      ["a2", "11L", 2.0],
```

```
       ["a2", "12L", 2.0],
       ["a3", "12L", 2.0],
       ["a3", "13L", 2.0],
       ["a4", "13L", 2.0],
       ["a4", "14L", 2.0],
       ["a5", "14L", 2.0],
       ["a5", "15L", 2.0],
       ["a6", "15L", 2.0],
       ["a6", "16L", 2.0],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user string, item
string, rating double')

model = SwingTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .linkFrom(data)

predictor = SwingRecommBatchOp()\
    .setItemCol("item")\
    .setRecommCol("prediction_result")

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.SwingRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.SwingTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SwingRecommBatchOpTest {
    @Test
    public void testSwingRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("a1", "11L", 2.2),
            Row.of("a1", "12L", 2.0),
            Row.of("a2", "11L", 2.0),
            Row.of("a2", "12L", 2.0),
            Row.of("a3", "12L", 2.0),
```

```
            Row.of("a3", "13L", 2.0),
            Row.of("a4", "13L", 2.0),
            Row.of("a4", "14L", 2.0),
            Row.of("a5", "14L", 2.0),
            Row.of("a5", "15L", 2.0),
            Row.of("a6", "15L", 2.0),
            Row.of("a6", "16L", 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user string,
item string, rating double");
        BatchOperator <?> model = new SwingTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .linkFrom(data);
        BatchOperator <?> predictor = new SwingRecommBatchOp()
            .setItemCol("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	item	rating	prediction_result
a1	11L	2.2000	{"item":"[\"12L\"]","score":"[0.12805642187595367]"}
a1	12L	2.0000	{"item":"[\"11L\"]","score":"[0.11662912368774414]"}
a2	11L	2.0000	{"item":"[\"12L\"]","score":"[0.12805642187595367]"}
a2	12L	2.0000	{"item":"[\"11L\"]","score":"[0.11662912368774414]"}
a3	12L	2.0000	{"item":"[\"11L\"]","score":"[0.11662912368774414]"}
a3	13L	2.0000	null
a4	13L	2.0000	null
a4	14L	2.0000	null
a5	14L	2.0000	null
a5	15L	2.0000	null
a6	15L	2.0000	null
a6	16L	2.0000	null

swing训练 (SwingTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.SwingTrainBatchOp

Python 类名：SwingTrainBatchOp

功能介绍

Swing 是一种被广泛使用的item召回算法，输入数据只包含user和item即可，使用简单。

Swing算法原理比较简单，是阿里最早使用到的一种召回算法，在阿里多个业务被验证过非常有效的一种召回方式。 它认为 user-item-user 的结构比 itemCF 的单边结构更稳定。

Swing指的是秋千，例如用户u和用户v，都购买过同一件商品i，则三者之间会构成一个类似秋千的关系图。 若用户u和用户v之间除了购买过i外，还购买过商品j，则认为两件商品是具有某种程度上的相似的。 也就是说，商品与商品之间的相似关系，是通过用户关系来传递的。为了衡量物品i和j的相似性，考察都购买了物品i和j的用户u和用户v， 如果这两个用户共同购买的物品越少，则物品i和j的相似性越高。

Swing算法的表达式如下：

用户权重 w_u = 1 / (user_click_num + userAlpha) ^ userBeta

score(item_i, item_j) = sum(wu * wv / (alpha + common_items(wu, wv)))

其中userAlpha、userBeta和alpha，用户可以根据自己的数据情况设置。

此外，可以通过设置maxUserItems和minUserItems，过滤掉关联item太多或者太少的用户信息。 对于出现次数过多的item，设置maxItemNumber参数，当出现次数大于maxItemNumber时，算法从所有的user中随机抽取maxItemNumber个用户，计算结果，因此该算法两次执行的结果不一定相同。

在电商场景中，user和item的关系可以为点击、购买、收藏等。在Feed流场景关系可以为点击、点赞、评论、收藏等，item可以为文章、博主、话题等。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
userCol	User列列名	User列列名	String	✓		
alpha	alpha参数	alpha参数，默认1.0	Float			1.0

maxItemNumber	item参与计算的人数最大值	如果item出现次数大于该次数，会随机选择该次数的用户数据，默认1000	Integer			1000
maxUserItems	用户互动的最大Item数量	如果用户互动Item数量大于该次数，该用户数据不参与计算过程，默认1000	Integer			1000
minUserItems	用户互动的最小Item数量	如果用户互动Item数量小于该次数，该用户数据不参与计算过程，默认10	Integer			10
resultNormalize	结果是否归一化	是否归一化，默认False	Boolean			false
userAlpha	用户alpha参数	用户alpha参数，默认5.0, user weight = 1.0/(userAlpha + userClickCount)^userBeta	Float			5.0
userBeta	用户beta参数	用户beta参数，默认-0.35, user weight = 1.0/(userAlpha + userClickCount)^userBeta	Float			-0.35

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ["a1", "11L", 2.2],
    ["a1", "12L", 2.0],
    ["a2", "11L", 2.0],
    ["a2", "12L", 2.0],
    ["a3", "12L", 2.0],
    ["a3", "13L", 2.0],
    ["a4", "13L", 2.0],
    ["a4", "14L", 2.0],
    ["a5", "14L", 2.0],
    ["a5", "15L", 2.0],
```

```
    ["a6", "15L", 2.0],
    ["a6", "16L", 2.0],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user string, item
string, rating double')


model = SwingTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setMinUserItems(1)\
    .linkFrom(data)

model.print()

predictor = SwingRecommBatchOp()\
    .setItemCol("item")\
    .setRecommCol("prediction_result")

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.SwingRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.SwingTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SwingTrainBatchOpTest {
    @Test
    public void testSwingTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("a1", "11L", 2.2),
            Row.of("a1", "12L", 2.0),
            Row.of("a2", "11L", 2.0),
            Row.of("a2", "12L", 2.0),
            Row.of("a3", "12L", 2.0),
            Row.of("a3", "13L", 2.0),
            Row.of("a4", "13L", 2.0),
            Row.of("a4", "14L", 2.0),
```

```
            Row.of("a5", "14L", 2.0),
            Row.of("a5", "15L", 2.0),
            Row.of("a6", "15L", 2.0),
            Row.of("a6", "16L", 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user string,
item string, rating double");
        BatchOperator <?> model = new SwingTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setMinUserItems(1)
            .linkFrom(data);
        model.print();
        BatchOperator <?> predictor = new SwingRecommBatchOp()
            .setItemCol("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

mainItems	recommItemAndSimilarity
11L	{"object":["12L"],"score":[1.3015095],"itemCol":"item"}
14L	{"object":[],"score":[],"itemCol":"item"}
16L	{"object":[],"score":[],"itemCol":"item"}
15L	{"object":[],"score":[],"itemCol":"item"}
13L	{"object":[],"score":[],"itemCol":"item"}
12L	{"object":["11L"],"score":[1.3015095],"itemCol":"item"}

user	item	rating	prediction_result
a1	11L	2.2000	{"item":"[\"12L\"]","score":"[1.3015094995498657]"}
a1	12L	2.0000	{"item":"[\"11L\"]","score":"[1.3015094995498657]"}
a2	11L	2.0000	{"item":"[\"12L\"]","score":"[1.3015094995498657]"}
a2	12L	2.0000	{"item":"[\"11L\"]","score":"[1.3015094995498657]"}
a3	12L	2.0000	{"item":"[\"11L\"]","score":"[1.3015094995498657]"}
a3	13L	2.0000	{"item":"[]","score":"[]"}
a4	13L	2.0000	{"item":"[]","score":"[]"}

a4	14L	2.0000	{"item":"[]","score":"[]"}
a5	14L	2.0000	{"item":"[]","score":"[]"}
a5	15L	2.0000	{"item":"[]","score":"[]"}
a6	15L	2.0000	{"item":"[]","score":"[]"}
a6	16L	2.0000	{"item":"[]","score":"[]"}

UserCf：ItemsPerUser推荐 (UserCfItemsPerUserRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.UserCfItemsPerUserRecommBatchOp

Python 类名：UserCfItemsPerUserRecommBatchOp

功能介绍

用UserCF模型 为user推荐item list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean			false
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = UserCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = UserCfItemsPerUserRecommBatchOp()\
    .setUserCol("user")\
    .setReservedCols(["user"])\
    .setK(1)\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.UserCfItemsPerUserRecommBatchOp
;
import com.alibaba.alink.operator.batch.recommendation.UserCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class UserCfItemsPerUserRecommBatchOpTest {
```

```
    @Test
    public void testUserCfItemsPerUserRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new UserCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new UserCfItemsPerUserRecommBatchOp()
            .setUserCol("user")
            .setReservedCols("user")
            .setK(1)
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	prediction_result
1	{"item":"[1]","score":"[0.4609327677584255]"}
2	{"item":"[1]","score":"[0.1843731071033702]"}
2	{"item":"[1]","score":"[0.1843731071033702]"}
4	{"item":"[2]","score":"[0.16388720631410686]"}
4	{"item":"[2]","score":"[0.16388720631410686]"}
4	{"item":"[2]","score":"[0.16388720631410686]"}

UserCf：打分推荐 (UserCfRateRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.UserCfRateRecommBatchOp

Python 类名：UserCfRateRecommBatchOp

功能介绍

UserCF 打分是使用UserCF模型，于预测user对item的评分。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])
```

```
data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = UserCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = UserCfRateRecommBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.UserCfRateRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.UserCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class UserCfRateRecommBatchOpTest {
    @Test
    public void testUserCfRateRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new UserCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new UserCfRateRecommBatchOp()
            .setUserCol("user")
```

```
            .setItemCol("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	item	rating	prediction_result
1	1	0.6000	0.6000
2	2	0.8000	0.3000
2	3	0.6000	0.4000
4	1	0.6000	0.6000
4	2	0.3000	0.8000
4	3	0.4000	0.6000

UserCf：相似users推荐 (UserCfSimilarUsersRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.UserCfSimilarUsersRecommBatchOp

Python 类名：UserCfSimilarUsersRecommBatchOp

功能介绍

用UserCF模型为user推荐相似的user list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
```

```
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = UserCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = UserCfSimilarUsersRecommBatchOp()\
    .setUserCol("user")\
    .setReservedCols(["user"])\
    .setK(1)\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.recommendation.UserCfSimilarUsersRecommBatchOp
;
import com.alibaba.alink.operator.batch.recommendation.UserCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class UserCfSimilarUsersRecommBatchOpTest {
    @Test
    public void testUserCfSimilarUsersRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new UserCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new UserCfSimilarUsersRecommBatchOp()
            .setUserCol("user")
            .setReservedCols("user")
            .setK(1)
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	prediction_result
1	{"user":"[4]","similarities":"[0.7682212795973759]"}
2	{"user":"[4]","similarities":"[0.6145770236779007]"}
2	{"user":"[4]","similarities":"[0.6145770236779007]"}
4	{"user":"[1]","similarities":"[0.7682212795973759]"}
4	{"user":"[1]","similarities":"[0.7682212795973759]"}
4	{"user":"[1]","similarities":"[0.7682212795973759]"}

UserCf训练 (UserCfTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.UserCfTrainBatchOp

Python 类名：UserCfTrainBatchOp

功能介绍

UserCF 是一种被广泛使用的协同过滤算法，用给定打分数据训练一个推荐模型， 用于预测user对item的评分、对user推荐itemlist，或者对item推荐userlist等。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
userCol	User列列名	User列列名	String	✓		
k	相似集合元素数目	相似集合元素数目	Integer			64
rateCol	打分列列名	打分列列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
similarityThreshold	相似阈值	只有大于该阈值的Object才会被计算	Double			1.0E-4
similarityType	距离度量方式	聚类使用的距离类型	String		"COSINE", "JACCARD", "PEARSON"	"COSINE"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = UserCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = UserCfRateRecommBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.UserCfRateRecommBatchOp;
import com.alibaba.alink.operator.batch.recommendation.UserCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class UserCfTrainBatchOpTest {
    @Test
    public void testUserCfTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new UserCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new UserCfRateRecommBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

user	item	rating	prediction_result
1	1	0.6000	0.6000
2	2	0.8000	0.3000
2	3	0.6000	0.4000
4	1	0.6000	0.6000
4	2	0.3000	0.8000
4	3	0.4000	0.6000

UserCf：UsersPerItem推荐 (UserCfUsersPerItemRecommBatchOp)

Java 类名：com.alibaba.alink.operator.batch.recommendation.UserCfUsersPerItemRecommBatchOp

Python 类名：UserCfUsersPerItemRecommBatchOp

功能介绍

用UserCF模型 为item推荐user list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean			false
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]	null
k	推荐TOP数量	推荐TOP数量	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = UserCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = UserCfUsersPerItemRecommBatchOp()\
    .setItemCol("item")\
    .setReservedCols(["item"])\
    .setK(1)\
    .setRecommCol("prediction_result");

predictor.linkFrom(model, data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.UserCfTrainBatchOp;
import
com.alibaba.alink.operator.batch.recommendation.UserCfUsersPerItemRecommBatchOp
;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class UserCfUsersPerItemRecommBatchOpTest {
```

```
    @Test
    public void testUserCfUsersPerItemRecommBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new UserCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        BatchOperator <?> predictor = new UserCfUsersPerItemRecommBatchOp()
            .setItemCol("item")
            .setReservedCols("item")
            .setK(1)
            .setRecommCol("prediction_result");
        predictor.linkFrom(model, data).print();
    }
}
```

运行结果

item	prediction_result
1	{"user":"[1]","score":"[0.23046638387921276]"}
2	{"user":"[4]","score":"[0.2458308094711603]"}
3	{"user":"[4]","score":"[0.1843731071033702]"}
1	{"user":"[1]","score":"[0.23046638387921276]"}
2	{"user":"[4]","score":"[0.2458308094711603]"}
3	{"user":"[4]","score":"[0.1843731071033702]"}

PyTorch预测 (PyTorchPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dl.PyTorchPredictBatchOp

Python 类名：PyTorchPredictBatchOp

功能介绍

加载TorchScript模型进行预测

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
inputShapes	输入tensor形状	输入tensor形状	String	✓		
inputTypes	输入tensor数据类型	输入tensor数据类型	String	✓		
modelPath	模型路径	模型路径	String	✓		
outputCols	输出结果列列名数组	输出结果列列名数组，必选	String[]	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
reservedCols	算法保留列名	算法保留列	String[]			null

<!--

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df_data = pd.DataFrame([
    [str(np.random.rand(1 * 3 * 224 * 224)).strip('[]')]
])

train_data = BatchOperator.fromDataframe(df_data, schemaStr='f string')

url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/model.pt";
```

```
pytorch = PyTorchPredictBatchOp() \
  .setModelPath(url)\
  .setInputShapes("1,3,224,224") \
  .setInputTypes("float") \
  .setSelectedCols(["f"])\
  .setOutputCols(["p"])

pytorch.linkFrom(train_data).print()
```

DeepFM分类预测 (DeepFMClassifierPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dl.ctr.DeepFMClassifierPredictBatchOp

Python 类名：DeepFMClassifierPredictBatchOp

功能介绍

该组件提供 DeepFM 分类算法的批式预测功能。

预测数据所使用的特征列需要和训练数据保持一致，包括列名和类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
inferBatchSize	推理数据批大小	推理数据批大小	Integer			256
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
# 以下代码需要补齐一些参数，并将结果导出或者打印。

LABEL_COL = "label"

CATE_COLS = ["cf01", "cf02", "cf03", "cf04", "cf05",
             "cf06", "cf07", "cf08", "cf09", "cf10",
             "cf11", "cf12", "cf13", "cf14", "cf15",
             "cf16", "cf17", "cf18", "cf19", "cf20",
             "cf21", "cf22", "cf23", "cf24", "cf25",
             "cf26"]

NUM_COLS = ["nf01", "nf02", "nf03", "nf04", "nf05",
```

```
            "nf06", "nf07", "nf08", "nf09", "nf10",
            "nf11", "nf12", "nf13"]

schemaStr = "label DOUBLE,nf01 DOUBLE,nf02 DOUBLE,nf03 DOUBLE,nf04 DOUBLE,nf05
DOUBLE,nf06 DOUBLE,nf07 " + "DOUBLE,nf08 DOUBLE,nf09 DOUBLE,nf10 DOUBLE,nf11
DOUBLE,nf12 DOUBLE,nf13 DOUBLE,cf01 BIGINT,cf02 BIGINT," + "cf03 BIGINT,cf04
BIGINT,cf05 BIGINT,cf06 BIGINT,cf07 BIGINT,cf08 BIGINT,cf09 BIGINT,cf10
BIGINT,cf11 " + "BIGINT,cf12 BIGINT,cf13 BIGINT,cf14 BIGINT,cf15 BIGINT,cf16
BIGINT,cf17 BIGINT,cf18 BIGINT,cf19 BIGINT," + "cf20 BIGINT,cf21 BIGINT,cf22
BIGINT,cf23 BIGINT,cf24 BIGINT,cf25 BIGINT,cf26 BIGINT"

source = CsvSourceBatchOp() \
    .setFilePath(
    "https://alink-test.oss-cn-beijing-internal.aliyuncs.com/jiqi-
temp/tf_ut/tf_script/criteo_random_10w_test_data_fe") \
    .setSchemaStr(schemaStr)

numEpochs = 2
batchSize = 256
dnnHiddenUnits = [400, 400, 400]

deepFMTrain = DeepFMClassifierTrainBatchOp() \
    .setFeatureCols([*NUM_COLS, *CATE_COLS]) \
    .setCategoricalCols(CATE_COLS) \
    .setLabelCol("label") \
    .setDropoutRate(0.5) \
    .setNumFactor(10) \
    .setBatchSize(batchSize) \
    .setLambda1(0.1) \
    .setNumEpochs(numEpochs) \
    .setIntraOpParallelism(1) \
    .setDnnHiddenUnits(json.dumps(dnnHiddenUnits)) \
    .setCheckpointFilePath(checkpointFilePath) \
    .setNumPSs(2) \
    .setNumWorkers(2) \
    .linkFrom(source)

deepFMPredict = DeepFMClassifierPredictBatchOp() \
    .setPredictionCol("pred") \
    .setPredictionDetailCol("pred_detail") \
    .linkFrom(deepFMTrain, source)
```

DeepFM分类训练 (DeepFMClassifierTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dl.ctr.DeepFMClassifierTrainBatchOp

Python 类名：DeepFMClassifierTrainBatchOp

功能介绍

该组件提供 DeepFM 分类算法的训练功能，目标分类必须是两个。

输入数据为原始特征，包括数值型特征 (numerical) 和类别型特征 (categorical)。

对于类别型特征，组件内实现了从字符串到整数的映射，不需要额外的操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
checkpointFilePath	保存 checkpoint 的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estimator` 的 `model_dir` 传入，需要为所有 worker 都能访问到的目录	String	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
batchSize	数据批大小	数据批大小	Integer		
dnnHiddenUnits	DNN 隐含单元数	DNN 层数以及各层的隐含单元数，例如 [128, 128]	String		
dropoutRate	dropout 层概率	dropout 层的概率	Double		

evalConfig	EasyRec eval 配置	EasyRec 配置中的 evalConfig 部分，JSON 或者 prototxt 格式	String			
exportConfig	EasyRec export 配置	EasyRec 配置中的 ExportConfig 部分，JSON 或者 prototxt 格式	String			
featureDefs	FeatureConfigs 配置	EasyRec 配置中的 FeatureConfigs 部分，prototxt 格式	String			
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			
lambdaDnn	DNN L2正则化系数	DNN L2正则化系数	Double			
lambdaEmbedding	Embedding 正则化系数	DNN Embedding 正则化系数	Double			
learningRate	学习率	学习率	Double			
modelConfig	EasyRec model 配置	EasyRec 配置中的 modelConfig 部分，JSON 或者 prototxt 格式	String			
numEpochs	epoch数	epoch数	Integer			
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer			

numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer			
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			
removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只会删除必要的文件	Boolean			
trainConfig	EasyRec train 配置	EasyRec 配置中的 trainConfig 部分，JSON 或者 prototxt 格式；设置后，会覆盖组件的学习率、intra 并发度参数；配置中的 num_steps 不会起作用	String			

参数说明

对于参数 numWorkers 和 numPSs，两者总和不能超过运行 Alink 作业总的 Worker 数，建议两者按 3:1 或者 4:1 分配。

在进行训练前，总的数据集会均分到 TensorFlow Worker 角色的机器上，并写到硬盘上。 如果数据量过大时，可能出现硬盘空间不够的报错，此时可以增加 Worker 角色的数量。

对于参数 checkpointFilePath，需要在分布式训练时设置，它的取值会传递给底层 TensorFlow Estimator 的 model_dir 参数。 这个参数支持 TensorFlow 和 TensorFlow-io 官方所支持的文件系统和格式，例如 OSS 的路径为：

```
除此之外，对于 OSS，Alink 还支持这种写法：

```oss://[bucket-name]/[path]?host=[endpoint]&access_key_id=
[access_id]&access_key_secret=[access_key]
```

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
# 以下代码需要补齐一些参数，并将结果导出或者打印。

LABEL_COL = "label"

CATE_COLS = ["cf01", "cf02", "cf03", "cf04", "cf05",
             "cf06", "cf07", "cf08", "cf09", "cf10",
             "cf11", "cf12", "cf13", "cf14", "cf15",
             "cf16", "cf17", "cf18", "cf19", "cf20",
             "cf21", "cf22", "cf23", "cf24", "cf25",
             "cf26"]

NUM_COLS = ["nf01", "nf02", "nf03", "nf04", "nf05",
            "nf06", "nf07", "nf08", "nf09", "nf10",
            "nf11", "nf12", "nf13"]

schemaStr = "label DOUBLE,nf01 DOUBLE,nf02 DOUBLE,nf03 DOUBLE,nf04 DOUBLE,nf05
DOUBLE,nf06 DOUBLE,nf07 " + "DOUBLE,nf08 DOUBLE,nf09 DOUBLE,nf10 DOUBLE,nf11
DOUBLE,nf12 DOUBLE,nf13 DOUBLE,cf01 BIGINT,cf02 BIGINT," + "cf03 BIGINT,cf04
BIGINT,cf05 BIGINT,cf06 BIGINT,cf07 BIGINT,cf08 BIGINT,cf09 BIGINT,cf10
BIGINT,cf11 " + "BIGINT,cf12 BIGINT,cf13 BIGINT,cf14 BIGINT,cf15 BIGINT,cf16
BIGINT,cf17 BIGINT,cf18 BIGINT,cf19 BIGINT," + "cf20 BIGINT,cf21 BIGINT,cf22
BIGINT,cf23 BIGINT,cf24 BIGINT,cf25 BIGINT,cf26 BIGINT"

source = CsvSourceBatchOp() \
    .setFilePath(
```
```

```
    "https://alink-test.oss-cn-beijing-internal.aliyuncs.com/jiqi-
temp/tf_ut/tf_script/criteo_random_10w_test_data_fe") \
    .setSchemaStr(schemaStr)

numEpochs = 2
batchSize = 256
dnnHiddenUnits = [400, 400, 400]

deepFMTrain = DeepFMClassifierTrainBatchOp() \
    .setFeatureCols([*NUM_COLS, *CATE_COLS]) \
    .setCategoricalCols(CATE_COLS) \
    .setLabelCol("label") \
    .setDropoutRate(0.5) \
    .setNumFactor(10) \
    .setBatchSize(batchSize) \
    .setLambda1(0.1) \
    .setNumEpochs(numEpochs) \
    .setIntraOpParallelism(1) \
    .setDnnHiddenUnits(json.dumps(dnnHiddenUnits)) \
    .setCheckpointFilePath(checkpointFilePath) \
    .setNumPSs(2) \
    .setNumWorkers(2) \
    .linkFrom(source)

deepFMPredict = DeepFMClassifierPredictBatchOp() \
    .setPredictionCol("pred") \
    .setPredictionDetailCol("pred_detail") \
    .linkFrom(deepFMTrain, source)
```

DeepFM回归预测 (DeepFMRegressorPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dl.ctr.DeepFMRegressorPredictBatchOp

Python 类名：DeepFMRegressorPredictBatchOp

功能介绍

该组件提供 DeepFM 回归算法的批式预测功能。

预测数据所使用的特征列需要和训练数据保持一致，包括列名和类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
inferBatchSize	推理数据批大小	推理数据批大小	Integer			256
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
# 以下代码需要补齐一些参数，并将结果导出或者打印。

LABEL_COL = "label"

CATE_COLS = ["cf01", "cf02", "cf03", "cf04", "cf05",
             "cf06", "cf07", "cf08", "cf09", "cf10",
             "cf11", "cf12", "cf13", "cf14", "cf15",
             "cf16", "cf17", "cf18", "cf19", "cf20",
             "cf21", "cf22", "cf23", "cf24", "cf25",
             "cf26"]

NUM_COLS = ["nf01", "nf02", "nf03", "nf04", "nf05",
            "nf06", "nf07", "nf08", "nf09", "nf10",
            "nf11", "nf12", "nf13"]
```

```
schemaStr = "label DOUBLE,nf01 DOUBLE,nf02 DOUBLE,nf03 DOUBLE,nf04 DOUBLE,nf05
DOUBLE,nf06 DOUBLE,nf07 " + "DOUBLE,nf08 DOUBLE,nf09 DOUBLE,nf10 DOUBLE,nf11
DOUBLE,nf12 DOUBLE,nf13 DOUBLE,cf01 BIGINT,cf02 BIGINT," + "cf03 BIGINT,cf04
BIGINT,cf05 BIGINT,cf06 BIGINT,cf07 BIGINT,cf08 BIGINT,cf09 BIGINT,cf10
BIGINT,cf11 " + "BIGINT,cf12 BIGINT,cf13 BIGINT,cf14 BIGINT,cf15 BIGINT,cf16
BIGINT,cf17 BIGINT,cf18 BIGINT,cf19 BIGINT," + "cf20 BIGINT,cf21 BIGINT,cf22
BIGINT,cf23 BIGINT,cf24 BIGINT,cf25 BIGINT,cf26 BIGINT"

source = CsvSourceBatchOp() \
    .setFilePath("https://alink-test.oss-cn-beijing-internal.aliyuncs.com/jiqi-
temp/tf_ut/tf_script/criteo_random_10w_test_data_fe") \
    .setSchemaStr(schemaStr)

numEpochs = 2
batchSize = 256
dnnHiddenUnits = [400, 400, 400]

deepFMTrain = DeepFMRegressorTrainBatchOp() \
    .setFeatureCols([*NUM_COLS, *CATE_COLS]) \
    .setCategoricalCols(CATE_COLS) \
    .setLabelCol("label") \
    .setDropoutRate(0.5) \
    .setNumFactor(10) \
    .setBatchSize(batchSize) \
    .setLambda1(0.1) \
    .setNumEpochs(numEpochs) \
    .setIntraOpParallelism(1) \
    .setDnnHiddenUnits(json.dumps(dnnHiddenUnits)) \
    .setCheckpointFilePath(checkpointFilePath) \
    .setNumPSs(2) \
    .setNumWorkers(2) \
    .linkFrom(source)

deepFMPredict = DeepFMRegressorPredictBatchOp() \
    .setPredictionCol("pred") \
    .linkFrom(deepFMTrain, source)
```

DeepFM回归训练 (DeepFMRegressorTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dl.ctr.DeepFMRegressorTrainBatchOp

Python 类名：DeepFMRegressorTrainBatchOp

功能介绍

该组件提供 DeepFM 回归算法的训练功能。

输入数据为原始特征，包括数值型特征 (numerical) 和类别型特征 (categorical)。

对于类别型特征，组件内实现了从字符串到整数的映射，不需要额外的操作。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
checkpointFilePath	保存 checkpoint 的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estimator` 的 `model_dir` 传入，需要为所有 worker 都能访问到的目录	String	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
batchSize	数据批大小	数据批大小	Integer		
dnnHiddenUnits	DNN 隐含单元数	DNN 层数以及各层的隐含单元数，例如 [128, 128]	String		
dropoutRate	dropout 层概率	dropout 层的概率	Double		

evalConfig	EasyRec eval 配置	EasyRec 配置中的 evalConfig 部分，JSON 或者 prototxt 格式	String			
exportConfig	EasyRec export 配置	EasyRec 配置中的 ExportConfig 部分，JSON 或者 prototxt 格式	String			
featureDefs	FeatureConfigs 配置	EasyRec 配置中的 FeatureConfigs 部分，prototxt 格式	String			
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			
lambdaDnn	DNN L2正则化系数	DNN L2正则化系数	Double			
lambdaEmbedding	Embedding 正则化系数	DNN Embedding 正则化系数	Double			
learningRate	学习率	学习率	Double			
modelConfig	EasyRec model 配置	EasyRec 配置中的 modelConfig 部分，JSON 或者 prototxt 格式	String			
numEpochs	epoch数	epoch数	Integer			
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer			

numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer			
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			
removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只会删除必要的文件	Boolean			
trainConfig	EasyRec train 配置	EasyRec 配置中的 trainConfig 部分，JSON 或者 prototxt 格式；设置后，会覆盖组件的学习率、intra 并发度参数；配置中的 num_steps 不会起作用	String			

参数说明

对于参数 numWorkers 和 numPSs，两者总和不能超过运行 Alink 作业总的 Worker 数，建议两者按 3:1 或者 4:1 分配。

在进行训练前，总的数据集会均分到 TensorFlow Worker 角色的机器上，并写到硬盘上。 如果数据量过大时，可能出现硬盘空间不够的报错，此时可以增加 Worker 角色的数量。

对于参数 checkpointFilePath，需要在分布式训练时设置，它的取值会传递给底层 TensorFlow Estimator 的 model_dir 参数。 这个参数支持 TensorFlow 和 TensorFlow-io 官方所支持的文件系统和格式，例如 OSS 的路径为：

```
除此之外，对于 OSS，Alink 还支持这种写法：

```oss://[bucket-name]/[path]?host=[endpoint]&access_key_id=
[access_id]&access_key_secret=[access_key]
```

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
# 以下代码需要补齐一些参数，并将结果导出或者打印。

LABEL_COL = "label"

CATE_COLS = ["cf01", "cf02", "cf03", "cf04", "cf05",
             "cf06", "cf07", "cf08", "cf09", "cf10",
             "cf11", "cf12", "cf13", "cf14", "cf15",
             "cf16", "cf17", "cf18", "cf19", "cf20",
             "cf21", "cf22", "cf23", "cf24", "cf25",
             "cf26"]

NUM_COLS = ["nf01", "nf02", "nf03", "nf04", "nf05",
            "nf06", "nf07", "nf08", "nf09", "nf10",
            "nf11", "nf12", "nf13"]

schemaStr = "label DOUBLE,nf01 DOUBLE,nf02 DOUBLE,nf03 DOUBLE,nf04 DOUBLE,nf05
DOUBLE,nf06 DOUBLE,nf07 " + "DOUBLE,nf08 DOUBLE,nf09 DOUBLE,nf10 DOUBLE,nf11
DOUBLE,nf12 DOUBLE,nf13 DOUBLE,cf01 BIGINT,cf02 BIGINT," + "cf03 BIGINT,cf04
BIGINT,cf05 BIGINT,cf06 BIGINT,cf07 BIGINT,cf08 BIGINT,cf09 BIGINT,cf10
BIGINT,cf11 " + "BIGINT,cf12 BIGINT,cf13 BIGINT,cf14 BIGINT,cf15 BIGINT,cf16
BIGINT,cf17 BIGINT,cf18 BIGINT,cf19 BIGINT," + "cf20 BIGINT,cf21 BIGINT,cf22
BIGINT,cf23 BIGINT,cf24 BIGINT,cf25 BIGINT,cf26 BIGINT"

source = CsvSourceBatchOp() \
    .setFilePath("https://alink-test.oss-cn-beijing-internal.aliyuncs.com/jiqi-
```
```

```
temp/tf_ut/tf_script/criteo_random_10w_test_data_fe") \
    .setSchemaStr(schemaStr)

numEpochs = 2
batchSize = 256
dnnHiddenUnits = [400, 400, 400]

deepFMTrain = DeepFMRegressorTrainBatchOp() \
    .setFeatureCols([*NUM_COLS, *CATE_COLS]) \
    .setCategoricalCols(CATE_COLS) \
    .setLabelCol("label") \
    .setDropoutRate(0.5) \
    .setNumFactor(10) \
    .setBatchSize(batchSize) \
    .setLambda1(0.1) \
    .setNumEpochs(numEpochs) \
    .setIntraOpParallelism(1) \
    .setDnnHiddenUnits(json.dumps(dnnHiddenUnits)) \
    .setCheckpointFilePath(checkpointFilePath) \
    .setNumPSs(2) \
    .setNumWorkers(2) \
    .linkFrom(source)

deepFMPredict = DeepFMRegressorPredictBatchOp() \
    .setPredictionCol("pred") \
    .linkFrom(deepFMTrain, source)
```

PyTorch脚本预测 (PyTorchScriptPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.dl.pytorch.PyTorchScriptPredictBatchOp

Python 类名：PyTorchScriptPredictBatchOp

功能介绍

该组件支持用户传入PyTorch脚本，使用传入的批数据进行处理、预测，并可以将预测好的的数据输出回 Alink 端。

用户可以提供自己编写的 PyTorch 脚本文件， 脚本的编写可以参考 Alink PyTorch 组件使用指南.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
batchSize	batch size	batch size	Integer			256
mainScriptFile	主脚本文件路径	主脚本文件路径	String			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			""
reservedCols	算法保留列名	算法保留列	String[]			null

scriptFiles	所有脚本文件路径	所有脚本文件路径	String[]			
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
userFile	用户依赖文件包路径，tar.gzip或者zip格式	用户依赖文件包路径 url	String			""

脚本路径说明

脚本路径可以使用http/https/hdfs/oss等路径，OSS路径需要为：oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx

输出数据说明

脚本中可以输出一些数据，这些数据将传回 Alink 端，形成 Flink Table 的形式，进行后续的处理。

Stepwise二分类筛选预测 (BinarySelectorPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.BinarySelectorPredictBatchOp

Python 类名：BinarySelectorPredictBatchOp

功能介绍

使用Stepwise逻辑回归方法，进行特征筛选。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCol	计算列对应的列名	计算列对应的列名, 默认值是null	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$3$0:1.0 1:7.0 2:9.0", "1.0 7.0 9.0", 1.0, 7.0, 9.0, 1.0],
    ["$3$0:1.0 1:3.0 2:3.0", "2.0 3.0 3.0", 2.0, 3.0, 3.0, 1.0],
    ["$3$0:1.0 1:2.0 2:4.0", "3.0 2.0 4.0", 3.0, 2.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "2.0 3.0 4.0", 2.0, 3.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 5.0 8.0", 1.0, 5.0, 8.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 6.0 3.0", 1.0, 6.0, 3.0, 0.0]
```

```
])

batchData = BatchOperator.fromDataframe(df, schemaStr='svec string, vec string,
f0 double, f1 double, f2 double, label double')

selector = BinarySelectorTrainBatchOp()\
                .setAlphaEntry(0.65)\
                .setAlphaStay(0.7)\
                .setSelectedCol("vec")\
                .setLabelCol("label")\
                .setForceSelectedCols([1])

batchData.link(selector)

predict = BinarySelectorPredictBatchOp()\
            .setPredictionCol("pred")\
            .setReservedCols(["label"])

predict.linkFrom(selector, batchData).print()
```

运行结果

```
   label     pred
0    1.0  7.0 9.0
1    1.0  3.0 3.0
2    0.0  2.0 4.0
3    0.0  3.0 4.0
4    0.0  5.0 8.0
5    0.0  6.0 3.0
```

Stepwise二分类筛选训练 (BinarySelectorTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.BinarySelectorTrainBatchOp

Python 类名：BinarySelectorTrainBatchOp

功能介绍

使用Stepwise逻辑回归方法，进行特征筛选。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
alphaEntry	筛选阈值	筛选阈值	Double			0.05
alphaStay	移除阈值	移除阈值	Double			0.05
forceSelectedCols	强制选择的列	强制选择的列	int[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0

l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
method	方法	方法	String		"ScoreTest", "MarginalContribution"	"ScoreTest"
optimMethod	优化方法	优化方法	String		"LBFGS", "NETWON"	"LBFGS"
selectedCol	计算列对应的列名	计算列对应的列名，默认值是null	String			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$3$0:1.0 1:7.0 2:9.0", "1.0 7.0 9.0", 1.0, 7.0, 9.0, 1.0],
    ["$3$0:1.0 1:3.0 2:3.0", "2.0 3.0 3.0", 2.0, 3.0, 3.0, 1.0],
    ["$3$0:1.0 1:2.0 2:4.0", "3.0 2.0 4.0", 3.0, 2.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "2.0 3.0 4.0", 2.0, 3.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 5.0 8.0", 1.0, 5.0, 8.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 6.0 3.0", 1.0, 6.0, 3.0, 0.0]
```

```
])

batchData = BatchOperator.fromDataframe(df, schemaStr='svec string, vec string,
f0 double, f1 double, f2 double, label double')

selector = BinarySelectorTrainBatchOp()\
                .setAlphaEntry(0.65)\
                .setAlphaStay(0.7)\
                .setSelectedCol("vec")\
                .setLabelCol("label")\
                .setForceSelectedCols([1])

batchData.link(selector)

predict = BinarySelectorPredictBatchOp()\
            .setPredictionCol("pred")\
            .setReservedCols(["label"])

predict.linkFrom(selector, batchData).print()
```

运行结果

```
   label pred
0    1.0    7
1    1.0    3
2    0.0    2
3    0.0    3
4    0.0    5
5    0.0    6
```

评分卡分箱训练 (BinningTrainForScorecardBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.BinningTrainForScorecardBatchOp

Python 类名：BinningTrainForScorecardBatchOp

功能介绍

使用等频，等宽或自动分箱的方法对数据进行分箱操作，输入为连续或离散的特征数据，输出为每个特征的分箱规则，在分箱组件中点键右键选择**我要分箱**，可以对分箱结果进行查看和编辑。

算法简介

信用评分卡模型在国外是一种成熟的预测方法，尤其在信用风险评估以及金融风险控制领域更是得到了比较广泛的使用，其原理是将模型变量WOE编码方式离散化之后运用logistic回归模型进行建模，其中本文介绍的分箱算法就是这里说的WOE编码过程。

分箱通常是指对连续变量进行区间划分，将连续变量划分成几个区间变量，主要目的是为了避免“过拟合”，使评分结果更具有稳健性和预测性。这里有个典型的例子就是：用决策树进行评分看起来效果不错，但往往都会因为”过拟合”，模型无法实际应用。

分箱支持等频、等宽两种方法，输入为连续或离散的特征数据，输出为每个特征的分箱规则，分箱结果支持合并、拆分等。

算法原理

分箱方法

等频分箱

等频分箱是对特征数据进行排序，按分位点的方式选取用户指定的N个分位点作为分箱边界，若相邻分位点相同则将两个分箱合并，因此分箱结果中有可能少于用户指定的分箱个数。

等宽分箱

等宽分箱是对特征数据按最大值和最小值等平均分成N份，在每一等份的边界作为分箱的边界。

离散变量分箱

字符串类型变量即为离散变量，离散变量的分箱不使用上述的三种分箱方法，对于离散变量的每一种取值会单独分成一个分箱，仅当某个取值小于用户指定的最小阈值时，该取值会被统一归入ELSE分箱中。

相关公式

WOE的计算公式如下:

$$ WOE_i=ln(\dfrac{py_i}{pn_i}) $$

其中，py_i是这个组中正样本个数占所有正样本个数的比例，pn_i是这个组中负样本个数占所有负样本个数的比例。

IV的计算公式如下：

$$ IV_i=(py_i-pn_i)*WOE_i $$

实际使用时，一般用IV评估变量的重要性，IV越大，变量对模型的影响越大，在单个变量内部，再利用WOE判断每个bin的区分度如何，WOE越大，区分度越大。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
binningMethod	连续特征分箱方法	连续特征分箱方法	String		"QUANTILE", "BUCKET"	"QUANTIL
discreteThresholds	离散个数阈值	离散个数阈值，低于该阈值的离散样本将不会单独成一个组别。	Integer			-21474836
discreteThresholdsArray	离散个数阈值	离散个数阈值，每一列对应数组中一个元素。	Integer[]			null

discreteThresholdsMap	离散分箱离散为ELSE的最小阈值, 形式如col0:3,col1:4	离散分箱离散为ELSE的最小阈值，形式如col0:3,col1:4。	String			null
fromUserDefined	是否读取用户自定义JSON	是否读取用户自定义JSON,true则为用户自定义分箱,false则按参数配置分箱。	Boolean			false
labelCol	标签列名	输入表中的标签列名	String			null
leftOpen	是否左开右闭	左开右闭为true，左闭右开为false	Boolean			true
numBuckets	quantile个数	quantile个数,对所有列有效。	Integer			2
numBucketsArray	quantile个数	quantile个数,每一列对应数组中一个元素。	Integer[]			null

numBucketsMap	用户定义的bucket个数，形式如col0:3,col1:4	用户定义的bucket个数，形式如col0:3,col1:4	String			null
positiveLabelValueString	正样本	正样本对应的字符串格式。	String			"1"
userDefinedBin	用户定义的bin的json	用户定义的bin的json	String			

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, 1.0, True, 0, "A", 1],
    [1, 2.1, False, 2, "B", 1],
    [2, 1.1, True, 3, "C", 1],
    [3, 2.2, True, 1, "E", 0],
    [4, 0.1, True, 2, "A", 0],
    [5, 1.5, False, -4, "D", 1],
    [6, 1.3, True, 1, "B", 0],
    [7, 0.2, True, -1, "A", 1],
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id int,f0 double, f1
boolean, f2 int, f3 string, label int')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id int,f0 double, f1
boolean, f2 int, f3 string, label int')

train = BinningTrainForScorecardBatchOp()\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setLabelCol("label")\
    .setPositiveLabelValueString("1")\
```

```
    .linkFrom(inOp1)

predict = BinningPredictBatchOp()\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setEncode("INDEX")\
    .setReservedCols(["id", "label"])\
    .linkFrom(train, inOp1)

predict.lazyPrint(10)
train.print()

predict = BinningPredictStreamOp(train)\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setEncode("INDEX")\
    .setReservedCols(["id", "label"])\
    .linkFrom(inOp2)

predict.print()
StreamOperator.execute()
```

Java代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.feature.BinningPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

/**
 * Unit test for Binning.
 */
public class BinningTrainBatchOpTest {

    @Test
    public void test() throws Exception {
        List<Row> list = Arrays.asList(
            Row.of(0, 1.0, true, 0, "A", 1),
            Row.of(1, 2.1, false, 2, "B", 1),
            Row.of(2, 1.1, true, 3, "C", 1),
            Row.of(3, 2.2, true, 1, "E", 0),
```

```
            Row.of(4, 0.1, true, 2, "A", 0),
            Row.of(5, 1.5, false, -4, "D", 1),
            Row.of(6, 1.3, true, 1, "B", 0),
            Row.of(7, 0.2, true, -1, "A", 1)
        );

        BatchOperator batchOperator = new MemSourceBatchOp(list, "id int, f0
double, f1 boolean, f2 int, f3 string, label int");
        StreamOperator streamOperator = new MemSourceStreamOp(list, "id int, f0
double, f1 boolean, f2 int, f3 string, label int");

        BatchOperator train = new BinningTrainBatchOp()
            .setSelectedCols("f0", "f1", "f2", "f3")
            .setLabelCol("label")
            .setPositiveLabelValueString("1")
            .linkFrom(batchOperator);

        BatchOperator predict = new BinningPredictBatchOp()
            .setSelectedCols("f0", "f1", "f2", "f3")
            .setReservedCols("id", "label")
            .setEncode("INDEX")
            .linkFrom(train, batchOperator);

        predict.lazyPrint(10);
        train.print();

        StreamOperator predictStream = new BinningPredictStreamOp(train)
            .setSelectedCols("f0", "f1", "f2", "f3")
            .setEncode("INDEX")
            .setReservedCols("id", "label")
            .linkFrom(streamOperator);

        predictStream.print();
        StreamOperator.execute();
    }
}
```

运行结果

模型结果

```
                                FeatureBordersJson
0  [{"binDivideType":"QUANTILE","featureName":"f0...
1  [{"binDivideType":"QUANTILE","featureName":"f2...
2  [{"binDivideType":"DISCRETE","featureName":"f1...
3  [{"binDivideType":"DISCRETE","featureName":"f3...
```

预测结果

id	f0	f1	f2	f3	label
0	0	0	0	0	1
1	1	1	1	1	1
2	0	0	1	2	1
3	1	0	0	4	0
4	0	0	1	0	0
5	1	1	0	3	1
6	0	0	0	1	0
7	0	0	0	0	1

带约束的Stepwise二分类筛选预测 (ConstrainedBinarySelectorPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.ConstrainedBinarySelectorPredictBatchOp

Python 类名：ConstrainedBinarySelectorPredictBatchOp

功能介绍

使用带约束的Stepwise二分类方法，进行特征筛选。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCol	计算列对应的列名	计算列对应的列名, 默认值是null	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$3$0:1.0 1:7.0 2:9.0", "1.0 7.0 9.0", 1.0, 7.0, 9.0, 1.0],
    ["$3$0:1.0 1:3.0 2:3.0", "2.0 3.0 3.0", 2.0, 3.0, 3.0, 1.0],
    ["$3$0:1.0 1:2.0 2:4.0", "3.0 2.0 4.0", 3.0, 2.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "2.0 3.0 4.0", 2.0, 3.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 5.0 8.0", 1.0, 5.0, 8.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 6.0 3.0", 1.0, 6.0, 3.0, 0.0]
```

```
])

batchData = BatchOperator.fromDataframe(df, schemaStr='svec string, vec string,
f0 double, f1 double, f2 double, label double')

selector = ConstrainedBinarySelectorTrainBatchOp()\
                .setAlphaEntry(0.65)\
                .setAlphaStay(0.7)\
                .setSelectedCol("vec")\
                .setLabelCol("label")\
                .setForceSelectedCols([1,2])

constraint = pd.DataFrame([
    ['{"featureConstraint":[],"constraintBetweenFeatures":
{"name":"constraintBetweenFeatures","UP":[],"LO":[],"=":[[1,1.814],
[2,0.4]],"%":[],"<":[],">":[[1,3]]},"countZero":null,"elseNullSave":null}']
])
constraintData = BatchOperator.fromDataframe(constraint, schemaStr='data
string')

selector.linkFrom(batchData, constraintData)

predict = ConstrainedBinarySelectorPredictBatchOp()\
            .setPredictionCol("pred")\
            .setReservedCols(["label"])

predict.linkFrom(selector, batchData).print()
```

运行结果

```
   label     pred
0    1.0  7.0 9.0
1    1.0  3.0 3.0
2    0.0  2.0 4.0
3    0.0  3.0 4.0
4    0.0  5.0 8.0
5    0.0  6.0 3.0
```

带约束的Stepwise二分类筛选训练 (ConstrainedBinarySelectorTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.ConstrainedBinarySelectorTrainBatchOp

Python 类名：ConstrainedBinarySelectorTrainBatchOp

功能介绍

使用带约束的Stepwise二分类方法，进行特征筛选。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
alphaEntry	筛选阈值	筛选阈值	Double			0.05
alphaStay	移除阈值	移除阈值	Double			0.05
forceSelectedCols	强制选择的列	强制选择的列	int[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0

l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
method	方法	方法	String		"ScoreTest", "MarginalContribution"	"ScoreTest"
optimMethod	优化方法	优化方法	String		"SQP", "BARRIER"	"SQP"
selectedCol	计算列对应的列名	计算列对应的列名，默认值是null	String			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$3$0:1.0 1:7.0 2:9.0", "1.0 7.0 9.0", 1.0, 7.0, 9.0, 1.0],
    ["$3$0:1.0 1:3.0 2:3.0", "2.0 3.0 3.0", 2.0, 3.0, 3.0, 1.0],
    ["$3$0:1.0 1:2.0 2:4.0", "3.0 2.0 4.0", 3.0, 2.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "2.0 3.0 4.0", 2.0, 3.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 5.0 8.0", 1.0, 5.0, 8.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 6.0 3.0", 1.0, 6.0, 3.0, 0.0]
```

```
])

batchData = BatchOperator.fromDataframe(df, schemaStr='svec string, vec string,
f0 double, f1 double, f2 double, label double')

selector = ConstrainedBinarySelectorTrainBatchOp()\
                .setAlphaEntry(0.65)\
                .setAlphaStay(0.7)\
                .setSelectedCol("vec")\
                .setLabelCol("label")\
                .setForceSelectedCols([1,2])

constraint = pd.DataFrame([
    ['{"featureConstraint":[],"constraintBetweenFeatures":
{"name":"constraintBetweenFeatures","UP":[],"LO":[],"=":[[1,1.814],
[2,0.4]],"%":[],"<":[],">":[[1,3]]},"countZero":null,"elseNullSave":null}']
])
constraintData = BatchOperator.fromDataframe(constraint, schemaStr='data
string')

selector.linkFrom(batchData, constraintData)

predict = ConstrainedBinarySelectorPredictBatchOp()\
            .setPredictionCol("pred")\
            .setReservedCols(["label"])

predict.linkFrom(selector, batchData).print()
```

运行结果

```
   label     pred
0    1.0  7.0 9.0
1    1.0  3.0 3.0
2    0.0  2.0 4.0
3    0.0  3.0 4.0
4    0.0  5.0 8.0
5    0.0  6.0 3.0
```

带约束的线性回归训练 (ConstrainedLinearRegTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.ConstrainedLinearRegTrainBatchOp

Python 类名：ConstrainedLinearRegTrainBatchOp

功能介绍

- 带约束的线性回归是一个回归算法
- 和线性回归组件一样，带约束的线性回归支持稀疏、稠密两种数据格式
- 和线性回归组件一样，带约束的线性回归支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
constOptimMethod	优化方法	求解优化问题时选择的优化方法	String		"SQP", "Barrier", "LBFGS", "Newton"	"SQP"
constraint	约束	约束	String			""
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为1.0e-6	Double		[0.0, +inf)	1.0E-6
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
maxIter	最大迭代步数	最大迭代步数，默认为 100	Integer		[1, +inf)	100
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean			true
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

约束说明

约束有larger than value, larger than feature, less than value, less than feature, equal to feature, scale to feature这六种形式。

约束由ConstraintBetweenFeatures类控制，写完ConstraintBetweenFeatures实例以后存放于FeatureConstraint中。

约束可以通过constraint参数传入，也可以在linkFrom中通过表传入。但推荐直接通过constraint参数传入。

约束以如下格式传入，下面表示约束的意义为：

第2列的上界为7；第1列下界为3；第1列和第6列相等；第3列是第4列的7倍；第4列小于等于第5列；第5列大于等于第6列。

{"featureConstraint":[],"constraintBetweenFeatures":{"name":"constraintBetweenFeatures","UP":[[2,7.0]],"LO":[[1,3.0]],"=":[1,6],"%":[3,4,7.0],"<":[4,5],">":[5,6]}}

如果想通过feature colName的方式来表示约束，则以如下形式：

以下表示的是f1列下界是0,1.814，f1列大于等于f0列。

{"featureConstraint":[],"constraintBetweenFeatures":{"name":"constraintBetweenFeatures","UP":[],"LO":[["f1",0,1.814]],"=":[],"%":[],"<":[["f0",0,"f1",0]],">":[]}}

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1.0 7.0 9.0", 16.8],
    ["1.0 3.0 3.0", 6.7],
    ["1.0 2.0 4.0", 6.9],
    ["1.0 3.0 4.0", 8.0]
])

data = BatchOperator.fromDataframe(df, schemaStr="vec string, label double")

bfc = ConstraintBetweenFeatures()
bfc.addEqual(1, 3.)

f = FeatureConstraint()
f.addConstraintBetweenFeature(bfc)
cons = f.toString()

batchOp = ConstrainedLinearRegTrainBatchOp()\
    .setWithIntercept(True)\
    .setVectorCol("vec")\
    .setConstOptimMethod("barrier")\
    .setConstraint(cons)\
```

```
    .setLabelCol("label")

model = batchOp.linkFrom(data)

predict = LinearRegPredictBatchOp()\
    .setPredictionCol("pred")

predict.linkFrom(model, data).print()
```

运行结果

vec	label	pred
1.0 7.0 9.0	16.8	17.15
1.0 3.0 3.0	6.7	8.45
1.0 2.0 4.0	6.9	4.90
1.0 3.0 4.0	8.0	7.90

带约束的逻辑回归训练 (ConstrainedLogisticRegressionTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.ConstrainedLogisticRegressionTrainBatchOp

Python 类名：ConstrainedLogisticRegressionTrainBatchOp

功能介绍

- 带约束的逻辑回归是一个二分类算法
- 带约束的逻辑回归组件支持稀疏、稠密两种数据格式
- 带约束的支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
positiveLabelValueString	正样本	正样本对应的字符串格式。	String	✓		
constOptimMethod	优化方法	求解优化问题时选择的优化方法	String		"SQP", "Barrier", "LBFGS", "Newton"	"SQP"
constraint	约束	约束	String			""
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1.0E-6

featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean			true
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

约束说明

约束有larger than value, larger than feature, less than value, less than feature, equal to feature, scale to feature这六种形式。

约束由ConstraintBetweenFeatures类控制，写完ConstraintBetweenFeatures实例以后存放于FeatureConstraint中。

约束可以通过constraint参数传入，也可以在linkFrom中通过表传入。但推荐直接通过constraint参数传入。

约束以如下格式传入，下面表示约束的意义为：

第2列的上界为7；第1列下界为3；第1列和第6列相等；第3列是第4列的7倍；第4列小于等于第5列；第5列大于等于第6列。

{"featureConstraint":[],"constraintBetweenFeatures":{"name":"constraintBetweenFeatures","UP":[[2,7.0]],"LO":[[1,3.0]],"=":[1,6],"%":[3,4,7.0],"<":[4,5],">":[5,6]}}

如果想通过feature colName的方式来表示约束，则以如下形式：

以下表示的是f1列下界是0,1.814，f1列大于等于f0列。

{"featureConstraint":[],"constraintBetweenFeatures":{"name":"constraintBetweenFeatures","UP":[],"LO":[["f1",0,1.814]],"=":[],"%":[],"<":[["f0",0,"f1",0]],">":[]}}

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1, 1, 1, 1, 2],
    [1, 1, 0, 1, 2],
    [1, 0, 1, 1, 2],
    [1, 0, 1, 1, 2],
    [0, 1, 1, 0, 0],
    [0, 1, 1, 0, 0],
    [0, 1, 1, 0, 0],
    [0, 1, 1, 0, 0]
```

```
])

data = BatchOperator.fromDataframe(df, schemaStr="f0 int, f1 int, f2 int, f3
int, label bigint")

bfc = ConstraintBetweenFeatures()
bfc.addEqual("f1", 0, 1.814)

f = FeatureConstraint()
f.addConstraintBetweenFeature(bfc)

features = ["f0","f1","f2","f3"]

lr = ConstrainedLogisticRegressionTrainBatchOp()\
    .setConstraint(f.toString())\
    .setLabelCol("label")\
    .setFeatureCols(features)\
    .setConstOptimMethod("sqp")\
    .setPositiveLabelValueString("2")

model = lr.linkFrom(data)

predict = LogisticRegressionPredictBatchOp()\
    .setPredictionCol("lrpred")\
    .setReservedCols(["label"])

predict.linkFrom(model, data).print()
```

运行结果

label	lrpred
2	2
2	2
2	2
2	2
0	0
0	0
0	0
0	0

带约束的Stepwise回归筛选预测 (ConstrainedRegSelectorPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.ConstrainedRegSelectorPredictBatchOp

Python 类名：ConstrainedRegSelectorPredictBatchOp

功能介绍

使用带约束的Stepwise线性回归方法，进行特征筛选。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCol	计算列对应的列名	计算列对应的列名, 默认值是null	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$3$0:1.0 1:7.0 2:9.0", "1.0 7.0 9.0", 1.0, 7.0, 9.0, 1.0],
    ["$3$0:1.0 1:3.0 2:3.0", "2.0 3.0 3.0", 2.0, 3.0, 3.0, 1.0],
    ["$3$0:1.0 1:2.0 2:4.0", "3.0 2.0 4.0", 3.0, 2.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "2.0 3.0 4.0", 2.0, 3.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 5.0 8.0", 1.0, 5.0, 8.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 6.0 3.0", 1.0, 6.0, 3.0, 0.0]
```

```
])

batchData = BatchOperator.fromDataframe(df, schemaStr='svec string, vec string,
f0 double, f1 double, f2 double, label double')

selector = ConstrainedRegSelectorTrainBatchOp()\
                .setAlphaEntry(0.65)\
                .setAlphaStay(0.7)\
                .setSelectedCol("vec")\
                .setLabelCol("label")\
                .setForceSelectedCols([1,2])

constraint = pd.DataFrame([
        ['{"featureConstraint":[],"constraintBetweenFeatures":
{"name":"constraintBetweenFeatures","UP":[],"LO":[],"=":[[1,1.814],
[2,0.4]],"%":[],"<":[],">":[[1,2]]},"countZero":null,"elseNullSave":null}']
])
constraintData = BatchOperator.fromDataframe(constraint, schemaStr='data
string')

selector.linkFrom(batchData, constraintData)

predict = ConstrainedRegSelectorPredictBatchOp()\
            .setPredictionCol("pred")\
            .setReservedCols(["label"])

predict.linkFrom(selector, batchData).print()
```

运行结果

```
   label     pred
0    1.0  7.0 9.0
1    1.0  3.0 3.0
2    0.0  2.0 4.0
3    0.0  3.0 4.0
4    0.0  5.0 8.0
5    0.0  6.0 3.0
```

带约束的Stepwise回归筛选训练 (ConstrainedRegSelectorTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.ConstrainedRegSelectorTrainBatchOp

Python 类名：ConstrainedRegSelectorTrainBatchOp

功能介绍

使用带约束的Stepwise线性回归方法，进行特征筛选。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
alphaEntry	筛选阈值	筛选阈值	Double			0.05
alphaStay	移除阈值	移除阈值	Double			0.05
forceSelectedCols	强制选择的列	强制选择的列	int[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0

l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
method	方法	方法	String		"FTest", "MarginalContribution"	"FTest"
optimMethod	优化方法	优化方法	String		"SQP", "BARRIER"	"SQP"
selectedCol	计算列对应的列名	计算列对应的列名, 默认值是null	String			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$3$0:1.0 1:7.0 2:9.0", "1.0 7.0 9.0", 1.0, 7.0, 9.0, 1.0],
    ["$3$0:1.0 1:3.0 2:3.0", "2.0 3.0 3.0", 2.0, 3.0, 3.0, 1.0],
    ["$3$0:1.0 1:2.0 2:4.0", "3.0 2.0 4.0", 3.0, 2.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "2.0 3.0 4.0", 2.0, 3.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 5.0 8.0", 1.0, 5.0, 8.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 6.0 3.0", 1.0, 6.0, 3.0, 0.0]
```

```
])

batchData = BatchOperator.fromDataframe(df, schemaStr='svec string, vec string,
f0 double, f1 double, f2 double, label double')

selector = ConstrainedRegSelectorTrainBatchOp()\
                .setAlphaEntry(0.65)\
                .setAlphaStay(0.7)\
                .setSelectedCol("vec")\
                .setLabelCol("label")\
                .setForceSelectedCols([1,2])

constraint = pd.DataFrame([
        ['{"featureConstraint":[],"constraintBetweenFeatures":
{"name":"constraintBetweenFeatures","UP":[],"LO":[],"=":[[1,1.814],
[2,0.4]],"%":[],"<":[],">":[[1,2]]},"countZero":null,"elseNullSave":null}']
])
constraintData = BatchOperator.fromDataframe(constraint, schemaStr='data
string')

selector.linkFrom(batchData, constraintData)

predict = ConstrainedRegSelectorPredictBatchOp()\
            .setPredictionCol("pred")\
            .setReservedCols(["label"])

predict.linkFrom(selector, batchData).print()
```

运行结果

```
   label     pred
0    1.0  7.0 9.0
1    1.0  3.0 3.0
2    0.0  2.0 4.0
3    0.0  3.0 4.0
4    0.0  5.0 8.0
5    0.0  6.0 3.0
```

Stepwise回归筛选预测 (RegressionSelectorPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.RegressionSelectorPredictBatchOp

Python 类名：RegressionSelectorPredictBatchOp

功能介绍

使用Stepwise线性回归方法，进行特征筛选。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCol	计算列对应的列名	计算列对应的列名, 默认值是null	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$3$0:1.0 1:7.0 2:9.0", "1.0 7.0 9.0", 1.0, 7.0, 9.0, 1.0],
    ["$3$0:1.0 1:3.0 2:3.0", "2.0 3.0 3.0", 2.0, 3.0, 3.0, 1.0],
    ["$3$0:1.0 1:2.0 2:4.0", "3.0 2.0 4.0", 3.0, 2.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "2.0 3.0 4.0", 2.0, 3.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 5.0 8.0", 1.0, 5.0, 8.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 6.0 3.0", 1.0, 6.0, 3.0, 0.0]
```

```
])

batchData = BatchOperator.fromDataframe(df, schemaStr='svec string, vec string,
f0 double, f1 double, f2 double, label double')

selector = RegressionSelectorTrainBatchOp()\
                .setAlphaEntry(0.65)\
                .setAlphaStay(0.7)\
                .setSelectedCol("vec")\
                .setLabelCol("label")\
                .setForceSelectedCols([1])

batchData.link(selector)

predict =  RegressionSelectorPredictBatchOp()\
            .setPredictionCol("pred")\
            .setReservedCols(["label"])

predict.linkFrom(selector, batchData).print()
```

运行结果

```
   label pred
0    1.0    7
1    1.0    3
2    0.0    2
3    0.0    3
4    0.0    5
5    0.0    6
```

Stepwise回归筛选预测 (RegressionSelectorTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.RegressionSelectorTrainBatchOp

Python 类名：RegressionSelectorTrainBatchOp

功能介绍

使用Stepwise线性回归方法，进行特征筛选。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
alphaEntry	筛选阈值	筛选阈值	Double			0.05
alphaStay	移除阈值	移除阈值	Double			0.05
forceSelectedCols	强制选择的列	强制选择的列	int[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0

l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
method	方法	方法	String		"FTest", "MarginalContribution"	"FTest"
optimMethod	优化方法	优化方法	String		"LBFGS", "NETWON"	"LBFGS"
selectedCol	计算列对应的列名	计算列对应的列名，默认值是null	String			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$3$0:1.0 1:7.0 2:9.0", "1.0 7.0 9.0", 1.0, 7.0, 9.0, 1.0],
    ["$3$0:1.0 1:3.0 2:3.0", "2.0 3.0 3.0", 2.0, 3.0, 3.0, 1.0],
    ["$3$0:1.0 1:2.0 2:4.0", "3.0 2.0 4.0", 3.0, 2.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "2.0 3.0 4.0", 2.0, 3.0, 4.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 5.0 8.0", 1.0, 5.0, 8.0, 0.0],
    ["$3$0:1.0 1:3.0 2:4.0", "1.0 6.0 3.0", 1.0, 6.0, 3.0, 0.0]
```

```
])

batchData = BatchOperator.fromDataframe(df, schemaStr='svec string, vec string,
f0 double, f1 double, f2 double, label double')

selector = RegressionSelectorTrainBatchOp()\
                .setAlphaEntry(0.65)\
                .setAlphaStay(0.7)\
                .setSelectedCol("vec")\
                .setLabelCol("label")\
                .setForceSelectedCols([1])

batchData.link(selector)

predict =  RegressionSelectorPredictBatchOp()\
            .setPredictionCol("pred")\
            .setReservedCols(["label"])

predict.linkFrom(selector, batchData).print()
```

运行结果

```
   label pred
0    1.0    7
1    1.0    3
2    0.0    2
3    0.0    3
4    0.0    5
5    0.0    6
```

评分卡预测 (ScorecardPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.ScorecardPredictBatchOp

Python 类名：ScorecardPredictBatchOp

功能介绍

评分卡是在信用风险评估领域常用的建模工具，本实现的原理是通过分箱输入将原始变量离散化后再使用线性模型（逻辑回归，线性回归等）进行模型训练，其中包括特征工程/分数转换功能等等，同时也支持训练过程中的给变量添加约束条件。

注：若未指定分箱输入，则评分卡训练过程完全等价于一般的逻辑回归/线性回归。

特征工程

评分卡区别于普通的线性模型的最大的地方在于，评分卡在使用线性模型进行训练之前会对数据进行一定的特征工程处理，本评分卡中提供了两种特征工程方法，都是需要先经过分箱将特征编码，

```
i. ASSEMBLERED_VECTOR，将所有变量根据分箱结果进行编码生成一个统一的向量。
ii. WOE，即将变量的原始值使用变量落入的分箱所对应的WOE值进行替换。
iii. NULL，不进行编码。
```

注：使用ASSEMBLERED_VECTOR时，每个原始变量的不同组之间可以设置相关的约束，具体参见后续章节。

分数转换

评分卡的信用评分等场景中，需要通过线性变换将预测得到的样本的odds转换成分数，通常通过如下的线性变换来进行：

$$ log(odds)=\sum_{i}w_ix_i=a*scaledScore+b $$

用户通过如下三个参数来指定这个线性变换关系：

```
scaledValue: 给出一个分数的基准点
odds: 在给定的分数基准点处的odds值
pdo: (Point Double Odds) 分数增长多分odds值加倍
```

如scaledValue=800, odds=50, pdo=25, 则表示指定了上述直线中的两个点：

```
log(50) = a * 800 + b * log(50)
log(100) = a * 825 + b * log(100)
```

解出a和b，对模型中的分数做线性变换即可得到转换后的变量分。

训练过程支持约束

评分卡训练过程支持对变量添加约束，如可指定某个bin所对应的分数为固定值，或两个bin的分数满足一定的比例，再或者bin之间的分数有大小的限制，如设置bin的分数按bin的woe值排序等等，约束的实现依赖于底层的带约束的优化算法，约束可以在分箱的UI中进行设置，设置完成后分箱会生一个json格式的约束条件，并自动传递给后面连接的训练组件。

目前支持如下几种json约束:

```
"<": 变量的权重按顺序满足升序的约束
">": 变量的权重按顺序满足降序的约束
"=": 变量的权重等于固定值
"%": 变量之间的权重符定一定的比例关系
"UP": 变量的权重约束上限
"LO": 变量的权重约束下限
```

json约束以字符串的形式存储在表中，表为单行单列（字符串类型）的表，存储如下的Json字符串：

```
{
    "<": [
        [
            4,
            3
        ]
    ],
    "%": [
        [
            0,
            1,
            2
        ],
        [
            2,
            0,
            1
        ]
    ],
    "=": [
        [
            1,
            2
        ]
    ],
    "LO": [
        [
            3,
            5
        ]
    ],
```

```
    "name": "fc"
}
```

内置约束

优化算法

在高级选项中可以选择训练过程中使用的优化算法，目前支持下面四种优化算法:

```
L-BFGS
Newton's Method
Barrier Method
SQP
```

其中L-BFGS是一阶的优化算法支持较大规模的特征数据级，牛顿法是经典的二阶算法，收敛速度快，准确度高，但由于要计算二阶的Hessian Matrix，因此不适用于较大特征规模的问题，这两种算法均为无约束的优化算法，当选择这两种优化算法时会自动忽略约束条件。

当训练过程中有约束条件时，可以选择Barrier Method和SQP，这两种算法都是二阶的优化算法，在没有约束条件的情况下完全等价于牛顿法，这两种算法的计算性能和准确性差别不大，我们默认建议选择SQP。

如果用户对优化算法不太了解，建议默认选择"自动选择"，会自动根据用户任务的数据规模和约束情况来选择最合适的优化算法。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionScoreCol	预测分数列	预测分数列	String	✓		
calculateScorePerFeature	输出变量分	输出变量分	Boolean			false
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, True, 0, "A", 1],
    [2.1, False, 2, "B", 1],
    [1.1, True, 3, "C", 1],
    [2.2, True, 1, "E", 0],
    [0.1, True, 2, "A", 0],
    [1.5, False, -4, "D", 1],
    [1.3, True, 1, "B", 0],
    [0.2, True, -1, "A", 1],
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='f0 double, f1 boolean, f2
int, f3 string, label int')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='f0 double, f1 boolean, f2
int, f3 string, label int')

binning = BinningTrainBatchOp()\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setLabelCol("label")\
    .setPositiveLabelValueString("1")\
    .linkFrom(inOp1)

scorecard = ScorecardTrainBatchOp()\
    .setPositiveLabelValueString("0")\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setLabelCol("label")\
    .linkFrom(inOp1, binning)

predict = ScorecardPredictBatchOp()\
    .setPredictionScoreCol("score")\
    .setPredictionDetailCol("detail")\
    .linkFrom(scorecard, inOp1)

predict.lazyPrint(10)
scorecard.print()

predict = ScorecardPredictStreamOp(scorecard)\
    .setPredictionDetailCol("detail")\
    .setPredictionScoreCol("score")\
    .linkFrom(inOp2)
```

```
predict.print()
StreamOperator.execute()
```

运行结果

```
    f0     f1  f2 f3  label  \
0  1.0   True   0  A      1
1  2.1  False   2  B      1
2  1.1   True   3  C      1
3  2.2   True   1  E      0
4  0.1   True   2  A      0
5  1.5  False  -4  D      1
6  1.3   True   1  B      0
7  0.2   True  -1  A      1

                                              detail      score
0  {"0":"0.9999968435780205","1":"3.1564219794555... -12.666068
1  {"0":"0.9999978934455291","1":"2.1065544708598... -13.070455
2  {"0":"0.9999972659779248","1":"2.7340220751792... -12.809734
3  {"0":"1.676491567126348E-6","1":"0.99999832350...  13.298806
4  {"0":"5.571428878359264E-6","1":"0.99999442857...  12.097853
5                            {"0":"1.0","1":"0.0"} -53.828587
6  {"0":"3.3860911581307107E-6","1":"0.9999966139...  12.595831
7  {"0":"0.9999968435780205","1":"3.1564219794555... -12.666068
```

评分卡训练 (ScorecardTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.finance.ScorecardTrainBatchOp

Python 类名：ScorecardTrainBatchOp

功能介绍

评分卡是在信用风险评估领域常用的建模工具，本实现的原理是通过分箱输入将原始变量离散化后再使用线性模型（逻辑回归，线性回归等）进行模型训练，其中包括特征工程/分数转换功能等等，同时也支持训练过程中的给变量添加约束条件。

注：若未指定分箱输入，则评分卡训练过程完全等价于一般的逻辑回归/线性回归。

特征工程

评分卡区别于普通的线性模型的最大的地方在于，评分卡在使用线性模型进行训练之前会对数据进行一定的特征工程处理，本评分卡中提供了两种特征工程方法，都是需要先经过分箱将特征编码，

```
i. ASSEMBLERED_VECTOR，将所有变量根据分箱结果进行编码生成一个统一的向量。
ii. WOE，即将变量的原始值使用变量落入的分箱所对应的WOE值进行替换。
iii. NULL，不进行编码。
```

注：使用ASSEMBLERED_VECTOR时，每个原始变量的不同组之间可以设置相关的约束，具体参见后续章节。

分数转换

评分卡的信用评分等场景中，需要通过线性变换将预测得到的样本的odds转换成分数，通常通过如下的线性变换来进行：

$$ log(odds)=\sum_{i}w_ix_i=a*scaledScore+b $$

用户通过如下三个参数来指定这个线性变换关系：

```
scaledValue: 给出一个分数的基准点
odds: 在给定的分数基准点处的odds值
pdo: (Point Double Odds) 分数增长多分odds值加倍
```

如scaledValue=800, odds=50, pdo=25, 则表示指定了上述直线中的两个点：

```
log(50) = a * 800 + b * log(50)
log(100) = a * 825 + b * log(100)
```

解出a和b，对模型中的分数做线性变换即可得到转换后的变量分。

训练过程支持约束

评分卡训练过程支持对变量添加约束，如可指定某个bin所对应的分数为固定值，或两个bin的分数满足一定的比例，再或者bin之间的分数有大小的限制，如设置bin的分数按bin的woe值排序等等，约束的实现依赖于底层的带约束的优化算法，约束可以在分箱的UI中进行设置，设置完成后分箱会生一个json格式的约束条件，并自动传递给后面连接的训练组件。

目前支持如下几种json约束:

```
"<": 变量的权重按顺序满足升序的约束
">": 变量的权重按顺序满足降序的约束
"=": 变量的权重等于固定值
"%": 变量之间的权重符定一定的比例关系
"UP": 变量的权重约束上限
"LO": 变量的权重约束下限
```

json约束以字符串的形式存储在表中，表为单行单列（字符串类型）的表，存储如下的Json字符串：

```
{
    "<": [
        [
            4,
            3
        ]
    ],
    "%": [
        [
            0,
            1,
            2
        ],
        [
            2,
            0,
            1
        ]
    ],
    "=": [
        [
            1,
            2
        ]
    ],
    "LO": [
        [
            3,
            5
        ]
    ],
```

```
    "name": "fc"
}
```

内置约束

优化算法

在高级选项中可以选择训练过程中使用的优化算法，目前支持下面四种优化算法:

```
L-BFGS
Newton's Method
Barrier Method
SQP
```

其中L-BFGS是一阶的优化算法支持较大规模的特征数据级，牛顿法是经典的二阶算法，收敛速度快，准确度高，但由于要计算二阶的Hessian Matrix，因此不适用于较大特征规模的问题，这两种算法均为无约束的优化算法，当选择这两种优化算法时会自动忽略约束条件。

当训练过程中有约束条件时，可以选择Barrier Method和SQP，这两种算法都是二阶的优化算法，在没有约束条件的情况下完全等价于牛顿法，这两种算法的计算性能和准确性差别不大，我们默认建议选择SQP。

如果用户对优化算法不太了解，建议默认选择"自动选择"，会自动根据用户任务的数据规模和约束情况来选择最合适的优化算法。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
labelCol	标签列名	输入表中的标签列名	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
alphaEntry	筛选阈值	筛选阈值	Double			0.
alphaStay	移除阈值	移除阈值	Double			0.

constOptimMethod	优化方法	求解优化问题时选择的优化方法	String		"SQP", "Barrier", "LBFGS", "Newton"	"S
defaultWoe	默认Woe，在woe为Nan或NULL时替换	默认Woe，在woe为Nan或NULL时替换	Double			N
encode	编码方法	编码方法	String		"WOE", "ASSEMBLED_VECTOR", "NULL"	"A
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为1.0e-6	Double		[0.0, +inf)	1.
forceSelectedCols	强制选择的列	强制选择的列	String[]			
l1	L1 正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.
linearModelType	优化方法	优化方法	String		"LR", "LinearReg"	"L
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	1(

odds	分数基准点处的odds值	分数基准点处的odds值	Double			nu
pdo	分数增长pdo，odds加倍	分数增长pdo，odds加倍	Double			nu
positiveLabelValueString	正样本	正样本对应的字符串格式。	String			"1
scaleInfo	是否将模型进行分数转换	是否将模型进行分数转换	Boolean			fa
scaledValue	分数基准点	分数基准点	Double			nu
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	nu
withSelector	是否逐步回归	是否逐步回归	Boolean			fa

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, True, 0, "A", 1],
    [2.1, False, 2, "B", 1],
```

```
    [1.1, True, 3, "C", 1],
    [2.2, True, 1, "E", 0],
    [0.1, True, 2, "A", 0],
    [1.5, False, -4, "D", 1],
    [1.3, True, 1, "B", 0],
    [0.2, True, -1, "A", 1],
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='f0 double, f1 boolean, f2
int, f3 string, label int')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='f0 double, f1 boolean, f2
int, f3 string, label int')

binning = BinningTrainBatchOp()\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setLabelCol("label")\
    .setPositiveLabelValueString("1")\
    .linkFrom(inOp1)

scorecard = ScorecardTrainBatchOp()\
    .setPositiveLabelValueString("0")\
    .setSelectedCols(["f0", "f1", "f2", "f3"])\
    .setLabelCol("label")\
    .linkFrom(inOp1, binning)

predict = ScorecardPredictBatchOp()\
    .setPredictionScoreCol("score")\
    .setPredictionDetailCol("detail")\
    .linkFrom(scorecard, inOp1)

predict.lazyPrint(10)
scorecard.print()

predict = ScorecardPredictStreamOp(scorecard)\
    .setPredictionDetailCol("detail")\
    .setPredictionScoreCol("score")\
    .linkFrom(inOp2)

predict.print()
StreamOperator.execute()
```

运行结果

```
    f0     f1  f2 f3  label  \
0  1.0   True   0  A      1
1  2.1  False   2  B      1
2  1.1   True   3  C      1
```

```
3  2.2   True   1  E      0
4  0.1   True   2  A      0
5  1.5  False  -4  D      1
6  1.3   True   1  B      0
7  0.2   True  -1  A      1

                                                 detail      score
0  {"0":"0.9999968435780205","1":"3.1564219794555... -12.666068
1  {"0":"0.9999978934455291","1":"2.1065544708598... -13.070455
2  {"0":"0.9999972659779248","1":"2.7340220751792... -12.809734
3  {"0":"1.676491567126348E-6","1":"0.99999832350...  13.298806
4  {"0":"5.571428878359264E-6","1":"0.99999442857...  12.097853
5                             {"0":"1.0","1":"0.0"} -53.828587
6  {"0":"3.3860911581307107E-6","1":"0.9999966139...  12.595831
7  {"0":"0.9999968435780205","1":"3.1564219794555... -12.666068
```

CommonNeighborsBatchOp (CommonNeighborsBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.CommonNeighborsBatchOp

Python 类名：CommonNeighborsBatchOp

功能介绍

共同邻居算法（Common Neighbors）是一种常用的基本图分析算法，可以计算两个节点所共有的邻居节点， 发现社交场合中的共同好友、以及在消费领域共同感兴趣的商品，进一步推测两个节点之间的潜在关系和相近程度。适用于电商、社交等多种领域。

算法的输出有6列，前两列分别为两个节点的值，后面四列分别为共同邻居列表、共同邻居数量、Jaccard分数和Adamic分数。 Jaccard距离为 CommonNeighbors(a, b) / (size(a) + size(b) - CommonNeighbors(a,b)) Adamic Adar距离，首先计算每个节点的权重为 1/log(当前节点的邻居数量), Adamic Adar距离为两个节点的共同邻居的权重之和。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓		
targetCol	边表中终点所在列	边表中终点所在列	String	✓		
cnCol	输出表共同邻居数量列	输出表中共同邻居数量列名	String			"cn"
isBipartiteGraph	是否二部图	是否二部图	Boolean			false
needTransformID	Not available!	Not available!	Boolean			true
neighborsList	输出共同邻居列名	输出表中共同邻居列名	String			"neighbors_list"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([["a1", "11L"],\
        ["a1", "12L"],\
        ["a1", "16L"],\
        ["a2", "11L"],\
        ["a2", "12L"],\
        ["a3", "12L"],\
        ["a3", "13L"]])

data = BatchOperator.fromDataframe(df, schemaStr="source string, target
string")
CommonNeighborsBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .setIsBipartiteGraph(False)\
    .linkFrom(data)\
    .print()
```

Java代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CommonNeighborsBatchOpTest {

    @Test
    public void testGraph() throws Exception {
        List<Row> rows = Arrays.asList(
            Row.of("a1", "11L"),
            Row.of("a1", "12L"),
            Row.of("a1", "16L"),
            Row.of("a2", "11L"),
            Row.of("a2", "12L"),
            Row.of("a3", "12L"),
            Row.of("a3", "13L")
```

```
        );

        BatchOperator inputdata = new MemSourceBatchOp(rows, "source
string,target string");

        new CommonNeighborsBatchOp()
            .setSourceCol("source")
            .setTargetCol("target")
            .setIsBipartiteGraph(false)
            .linkFrom(inputdata)
            .print();
    }
}
```

运行结果

source	target	neighbors_list	cn	jaccards_score	adamic_score
a2	a1	11L,12L	2	0.6667	2.3529
11L	12L	a2,a1	2	0.6667	2.3529
13L	12L	a3	1	0.3333	1.4427
16L	12L	a1	1	0.3333	0.9102
16L	11L	a1	1	0.5000	0.9102
a3	a2	12L	1	0.3333	0.9102
12L	11L	a1,a2	2	0.6667	2.3529
11L	16L	a1	1	0.5000	0.9102
12L	13L	a3	1	0.3333	1.4427
a2	a3	12L	1	0.3333	0.9102
a1	a3	12L	1	0.2500	0.9102
12L	16L	a1	1	0.3333	0.9102
a1	a2	11L,12L	2	0.6667	2.3529
a3	a1	12L	1	0.2500	0.9102

标签传播分类 (CommunityDetectionClassifyBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.CommunityDetectionClassifyBatchOp

Python 类名：CommunityDetectionClassifyBatchOp

功能介绍

该算法为半监督的分类算法，原理为用已标记节点的标签信息去预测未标记节点的标签信息。 在算法执行过程中，每个节点的标签按相似度传播给相邻节点，在节点传播的每一步， 每个节点根据相邻节点的标签来更新自己的标签，与该节点相似度越大，其相邻节点对其标注的影响权值越大， 相似节点的标签越趋于一致，其标签就越容易传播。在标签传播过程中，保持已标注数据的标签不变， 使其像一个源头把标签传向未标注数据。 最终，当迭代过程结束时，相似节点的概率分布也趋于相似， 可以划分到同一个类别中，从而完成标签传播过程。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓		
targetCol	边表中终点所在列	边表中终点所在列	String	✓		
vertexCol	输入点表中点所在列	输入点表中点所在列	String	✓		
vertexLabelCol	输入点表中标签所在列	输入点表中标签所在列	String	✓		
delta	delta	delta参数	Double			0.2

edgeWeightCol	边权重列	表示边权重的列	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
k	K值	每轮迭代中，设置1/k的node不更新它们的值。这样的设定可能使得社区发现的效果更好。	Integer			40
maxIter	最大迭代次数	最大迭代次数	Integer		[1, +inf)	50
outputCols	输出列名	输出列名	String[]			["vertex","label"]
toUndirected	输入数据是否为有向图	表明输入的是有向图还是无向图，如果是有向图，算法就会将其转为无向图来处理。	Boolean			true
vertexWeightCol	点的权重所在列	点的权重所在列，如果不输入就自动补为1。	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[3, 1],\
[3, 0],\
[0, 1],\
[0, 2],\
[2, 1],\
[2, 4],\
[5, 4],\
[7, 4],\
[5, 6],\
[5, 8],\
[5, 7],\
[7, 8],\
[6, 8],\
[12, 10],\
[12, 11],\
[12, 13],\
[12, 9],\
[10, 9],\
[8, 9],\
[13, 9],\
[10, 7],\
[10, 11],\
[11, 13]])

edges = BatchOperator.fromDataframe(df, schemaStr="source int, target int")

df2 = pd.DataFrame([[2, 0],\
[4, 1],\
[7, 1],\
[8, 1],\
[9, 2],\
[10, 2]])

verteices = BatchOperator.fromDataframe(df2, schemaStr="vertex int, label
bigint")

communityDetectionClassify = CommunityDetectionClassifyBatchOp()\
                .setSourceCol("source")\
```

```
                .setTargetCol("target")\
                .setVertexCol("vertex")\
                .setVertexLabelCol("label")\
                .setOutputCols(["vertex", "label"])
communityDetectionClassify.linkFrom(edges, verteices).print()
```

Java代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CommunityDetectionClassifyBatchOpTest {
    @Test
    public void test() throws Exception {
        List <Row> edgeRows = Arrays.asList(
            Row.of(3, 1),
            Row.of(3, 0),
            Row.of(0, 1),
            Row.of(0, 2),
            Row.of(2, 1),
            Row.of(2, 4),
            Row.of(5, 4),
            Row.of(7, 4),
            Row.of(5, 6),
            Row.of(5, 8),
            Row.of(5, 7),
            Row.of(7, 8),
            Row.of(6, 8),
            Row.of(12, 10),
            Row.of(12, 11),
            Row.of(12, 13),
            Row.of(12, 9),
            Row.of(10, 9),
            Row.of(8, 9),
            Row.of(13, 9),
            Row.of(10, 7),
            Row.of(10, 11),
            Row.of(11, 13));

        BatchOperator edges = new MemSourceBatchOp(edgeRows,  "source int,
```

```
target int");

        List <Row> vertexRows = Arrays.asList(
            Row.of(2, 0L),
            Row.of(4, 1L),
            Row.of(7, 1L),
            Row.of(8, 1L),
            Row.of(9, 2L),
            Row.of(10, 2L));

        BatchOperator verteices = new MemSourceBatchOp(vertexRows,"vertex int,
label bigint");

        new CommunityDetectionClassifyBatchOp()
                .setSourceCol("source")
                .setTargetCol("target")
                .setVertexCol("vertex")
                .setVertexLabelCol("label")
                .setOutputCols("vertex", "label")
                .linkFrom(edges, verteices).print();
    }
}
```

运行结果

vertex	label
0	0
10	2
2	0
12	2
6	1
9	2
1	0
13	2
11	2
3	0
4	1
5	1

8	1
7	1

标签传播聚类 (CommunityDetectionClusterBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.CommunityDetectionClusterBatchOp

Python 类名：CommunityDetectionClusterBatchOp

功能介绍

图聚类是根据图的拓扑结构，进行子图的划分，使得子图内部节点的链接较多，子图之间的连接较少。 标签传播聚类算法的基本思路是节点的标签依赖其邻居节点的标签信息，影响程度由节点相似度决定，并通过传播迭代更新达到稳定。

标签传播聚类是一个迭代的过程，每一步迭代中，标签沿着边将自己的标签传给相邻的节点，相邻节点接收到所有边传输过来的标签后，计算节点所属的标签。 在下一轮传播时，继续将更新后的标签传播给自己的相邻节点。直到所有节点的标签不再改变。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓		
targetCol	边表中终点所在列	边表中终点所在列	String	✓		
vertexCol	Not available!	Not available!	String	✓		
delta	delta	delta参数	Double		(0.0, +inf)	0.2
edgeWeightCol	边权重列	表示边权重的列	String			null
k	K值	每轮迭代中，设置1/k的node不更新它们的值。这样的设定可能使得社区发现的效果更好。	Integer		[1, +inf)	40
maxIter	最大迭代次数	最大迭代次数	Integer		[1, +inf)	50
outputCols	输出列名	输出列名	String[]			["vertex","label"]

setStable	Not available!	Not available!	Boolean			false
toUndirected	输入数据是否为有向图	表明输入的是有向图还是无向图，如果是有向图，算法就会将其转为无向图来处理。	Boolean			true
vertexWeightCol	点的权重所在列	点的权重所在列，如果不输入就自动补为1。	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[3, 1],\
[3, 0],\
[0, 1],\
[0, 2],\
[2, 1],\
[2, 4],\
[5, 4],\
[7, 4],\
[5, 6],\
[5, 8],\
[5, 7],\
[7, 8],\
[6, 8],\
[12, 10],\
[12, 11],\
[12, 13],\
[12, 9],\
[10, 9],\
[8, 9],\
[13, 9],\
[10, 7],\
[10, 11],\
[11, 13]])
```

```
edges = BatchOperator.fromDataframe(df, schemaStr="source int, target int")

communityDetectionClusterBatchOp = CommunityDetectionClusterBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .setOutputCols(["vertex", "label"])
communityDetectionClusterBatchOp.linkFrom(edges).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CommunityDetectionClusterBatchOpTest {
    @Test
    public void testIntWithVertex() throws Exception {
        List<Row> datas = Arrays.asList(
                Row.of(3, 1),
                Row.of(3, 0),
                Row.of(0, 1),
                Row.of(0, 2),
                Row.of(2, 1),
                Row.of(2, 4),
                Row.of(5, 4),
                Row.of(7, 4),
                Row.of(5, 6),
                Row.of(5, 8),
                Row.of(5, 7),
                Row.of(7, 8),
                Row.of(6, 8),
                Row.of(12, 10),
                Row.of(12, 11),
                Row.of(12, 13),
                Row.of(12, 9),
                Row.of(10, 9),
                Row.of(8, 9),
                Row.of(13, 9),
                Row.of(10, 7),
                Row.of(10, 11),
```

```
                Row.of(11, 13));

        BatchOperator edges = new MemSourceBatchOp(datas, "source int, target
int");

        new CommunityDetectionClusterBatchOp()
            .setSourceCol("source")
            .setTargetCol("target")
            .setOutputCols("vertex", "label")
            .linkFrom(edges).print();
    }
}
```

运行结果

vertex	label
0	0
10	1
2	0
12	1
6	0
9	1
1	0
13	1
11	1
3	0
4	0
5	0
8	0
7	0

最大联通分量 (ConnectedComponentsBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.ConnectedComponentsBatchOp

Python 类名：ConnectedComponentsBatchOp

功能介绍

在无向图G中，若从顶点A到顶点B有路径相连，则称A和B是连通的； 在图G种存在若干子图，其中每个子图中所有顶点之间都是连通的，但在不同子图间不存在顶点连通，那么称图G的这些子图为最大连通子图。

最大联通子图的应用场景有ID-Mapping，节点分类，社区发现，反作弊等。比如识别作弊或者有风险的账号，与作弊账号在同一个子图中的账号，就有潜在的风险。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓		
targetCol	边表中终点所在列	边表中终点所在列	String	✓		
groupIdCol	输出表中点所属group所在列	输出表中点所属group所在列	String			"groupId"
maxIter	最大迭代次数	最大迭代次数	Integer		[1, +inf)	50
outIdCol	输出表中点id所在列	输出表中点id所在列	String			"node"
setStable	Not available!	Not available!	Boolean			false
toUndirected	输入数据是否为有向图	表明输入的是有向图还是无向图，如果是有向图，算法就会将其转为无向图来处理。	Boolean			true
vertexCol	Not available!	Not available!	String			"vertex"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[1, 2],\
[2, 3],\
[3, 4],\
[4, 5],\
[6, 7],\
[7, 8],\
[8, 9],\
[9, 6]])

data = BatchOperator.fromDataframe(df, schemaStr="source int, target int")
ConnectedComponentsBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .linkFrom(data)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ConnectedComponentsBatchOpTest {
    @Test
    public void test() throws Exception {
        List<Row> edgeRows = Arrays.asList(
            Row.of(1, 2),
            Row.of(2, 3),
            Row.of(3, 4),
            Row.of(4, 5),
            Row.of(6, 7),
```

```
            Row.of(7, 8),
            Row.of(8, 9),
            Row.of(9, 6)
        );
        BatchOperator edgeData = new MemSourceBatchOp(edgeRows, "source
int,target int");

        BatchOperator res = new ConnectedComponentsBatchOp()
            .setSetStable(false)
            .setSourceCol("source")
            .setTargetCol("target")
            .linkFrom(edgeData).print();
    }
}
```

运行结果

node	groupId
1	0
2	0
5	0
9	4
4	0
3	0
8	4
6	4
7	4

DeepWalk (DeepWalkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.DeepWalkBatchOp

Python 类名：DeepWalkBatchOp

功能介绍

DeepWalk是2014年提出的一个新的方法，用来为网络中的结点学习隐式特征表达，即将网络中的每一个点表示成连续特征空间中的一个点向量。DeepWalk是无监督特征学习方法，利用随机游走(Random Walk)及语言模型(Language modeling)，学习出的隐式特征能够捕捉到网络的结构信息。后续论文也提出了一些扩展，如结合损失函数进行有监督学习、结合文本信息等等

DeepWalk: Online Learning of Social Representations

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	起始点列名	用来指定起始点列	String	✓		
targetCol	中止点点列名	用来指定中止点列	String	✓		
walkLength	游走的长度	随机游走完向量的长度	Integer	✓		
walkNum	路径数目	每一个起始点游走出多少条路径	Integer	✓		
alpha	学习率	学习率	Double			0.025
batchSize	batch大小	batch大小，按行计算	Integer		[1, +inf)	
isToUndigraph	是否转无向图	选为true时，会将当前图转成无向图，然后再游走	Boolean			false
minCount	最小词频	最小词频	Integer			5
negative	负采样大小	负采样大小	Integer			5

numIter	迭代次数	迭代次数，默认为1。	Integer			1
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"true"
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	100
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
window	窗口大小	窗口大小	Integer			5

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["Bob", "Lucy", 1.],
    ["Lucy", "Bob", 1.],
    ["Lucy", "Bella", 1.],
    ["Bella", "Lucy", 1.],
    ["Alice", "Lisa", 1.],
    ["Lisa", "Alice", 1.],
    ["Lisa", "Karry", 1.],
    ["Karry", "Lisa", 1.],
    ["Karry", "Bella", 1.],
    ["Bella", "Karry", 1.]
])
source = BatchOperator.fromDataframe(df, schemaStr="start string, end string,
value double")

deepWalkBatchOp = DeepWalkBatchOp() \
  .setSourceCol("start")            \
  .setTargetCol("end")              \
```

```
  .setWeightCol("value")            \
  .setWalkNum(2)                    \
  .setWalkLength(2)                 \
  .setMinCount(1)                   \
  .setVectorSize(4)
deepWalkBatchOp.linkFrom(source).print()
```

运行结果

node	vec
Karry	0.03438692167401314,-0.04779096320271492,0.012648836709558964,-0.09576538950204849
Lisa	0.11595723778009415,-0.08507091552019119,0.1099027618765831,0.013517010025680065
Bella	0.05783883109688759,0.08286115527153015,-0.06497485190629959,0.026532595977187157
Alice	0.05775630846619606,-0.099935382604599,-0.022451162338256836,-0.023144230246543884
Lucy	0.11699658632278442,0.05271214246749878,-0.12347490340471268,-0.08684996515512466
Bob	-0.07306862622499466,-0.11596906185150146,-0.04183155298233032,0.03973118215799332

边聚类系数 (EdgeClusterCoefficientBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.EdgeClusterCoefficientBatchOp

Python 类名：EdgeClusterCoefficientBatchOp

功能介绍

对于给定的图，对于图中的每条边，分别输出边的两个顶点的度，两个顶点的公共邻点数目，以及该边的聚类系数。 聚类系数的计算公式为 commonNeighbor / min(neighobr1, neighbor2)

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
sourceCol	边表中起点所在列	边表中起点所在列	String	✓	所选列类型为 [INTEGER, LONG, STRING]	
targetCol	边表中终点所在列	边表中终点所在列	String	✓	所选列类型为 [INTEGER, LONG, STRING]	
outputCols	输出列的列名	输出列的列名	String[]			["node1","node2","neighbor1","neigh

toUndirected	输入数据是否为有向图	表明输入的是有向图还是无向图，如果是有向图，算法就会将其转为无向图来处理。	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[1.0, 2.0],\
[1.0, 3.0],\
[3.0, 2.0],\
[5.0, 2.0],\
[3.0, 4.0],\
[4.0, 2.0],\
[5.0, 4.0],\
[5.0, 1.0],\
[5.0, 3.0],\
[5.0, 6.0],\
[5.0, 8.0],\
[7.0, 6.0],\
[7.0, 1.0],\
[7.0, 5.0],\
[8.0, 6.0],\
```

```
[8.0, 4.0]])

data = BatchOperator.fromDataframe(df, schemaStr="source double, target
double")
EdgeClusterCoefficientBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .linkFrom(data)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;

public class EdgeClusterCoefficientBatchOpTest extends AlinkTestBase {
    @Test
    public void test() throws Exception {
        Row[] rows = new Row[] {
            Row.of(1.0, 2.0),
            Row.of(1.0, 3.0),
            Row.of(3.0, 2.0),
            Row.of(5.0, 2.0),
            Row.of(3.0, 4.0),
            Row.of(4.0, 2.0),
            Row.of(5.0, 4.0),
            Row.of(5.0, 1.0),
            Row.of(5.0, 3.0),
            Row.of(5.0, 6.0),
            Row.of(5.0, 8.0),
            Row.of(7.0, 6.0),
            Row.of(7.0, 1.0),
            Row.of(7.0, 5.0),
            Row.of(8.0, 6.0),
            Row.of(8.0, 4.0)
        };
        MemSourceBatchOp dataSource = new MemSourceBatchOp(Arrays.asList(rows),
"source double,target double");
        BatchOperator res = new EdgeClusterCoefficientBatchOp()
            .setSourceCol("source")
            .setTargetCol("target")
```

```
            .linkFrom(dataSource);
        res.print();
    }
}
```

运行结果

ndoe1	node2	neighbor1	neighbor2	commonNeighbor	edgeClusterCoefficient
7.0000	1.0000	3	4	1	0.3333
3.0000	1.0000	4	4	2	0.5000
6.0000	7.0000	3	3	1	0.3333
7.0000	5.0000	3	7	2	0.6667
4.0000	5.0000	4	7	3	0.7500
3.0000	5.0000	4	7	3	0.7500
1.0000	5.0000	4	7	3	0.7500
6.0000	5.0000	3	7	2	0.6667
8.0000	5.0000	3	7	2	0.6667
2.0000	5.0000	4	7	3	0.7500
3.0000	4.0000	4	4	2	0.5000
4.0000	8.0000	4	3	1	0.3333
6.0000	8.0000	3	3	1	0.3333
4.0000	2.0000	4	4	2	0.5000
3.0000	2.0000	4	4	3	0.7500
1.0000	2.0000	4	4	2	0.5000

KCore算法 (KCoreBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.KCoreBatchOp

Python 类名：KCoreBatchOp

功能介绍

对于给定的图，反复去除图中度小于等于k的点，直到图中所有点的度均大于k或者达到最大迭代次数为止。

k-Core算法是一种用来在图中找出符合指定核心度的紧密关联的子图结构，在k-Core的结果子图中，每个顶点至少具有k的度数，且所有顶点都至少与该子图中的 k 个其他节点相连。k-Core通常用来对一个图进行子图划分，通过去除不重要的顶点，将符合逾期的子图暴露出来进行进一步分析。k-Core由于其线性的时间复杂度和符合直观认识的可解释性，在风控金融，社交网络和生物学上都具有较多的应用场景。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
targetCol	边表中终点所在列	边表中终点所在列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
k	k的数目	反复去除图中度小于等于k的点	Integer		[1, +inf)	3

outCols	输出列名	输出列名	String[]			["node1","node2"]
toUndirected	输入数据是否为有向图	表明输入的是有向图还是无向图，如果是有向图，算法就会将其转为无向图来处理。	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[1, 2],\
        [1, 3],\
        [1, 4],\
        [2, 3],\
        [2, 4],\
        [3, 4],\
        [3, 5],\
        [3, 6],\
        [5, 6]])

data = BatchOperator.fromDataframe(df, schemaStr="source int, target int")
KCoreBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .setK(2)\
    .linkFrom(data)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KCoreBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
        List <Row> intInputRows = Arrays.asList(
            Row.of(1, 2),
            Row.of(1, 3),
            Row.of(1, 4),
            Row.of(2, 3),
            Row.of(2, 4),
            Row.of(3, 4),
            Row.of(3, 5),
            Row.of(3, 6),
            Row.of(5, 6)
        );
        BatchOperator data = new MemSourceBatchOp(intInputRows, "source int,
target int");
        KCoreBatchOp kCoreBatchOp = new KCoreBatchOp()
            .setSourceCol("source")
            .setTargetCol("target")
            .setK(2)
            .linkFrom(data);
        kCoreBatchOp.print();
    }
}
```

运行结果

node1	node2
1	2
1	3
1	4
3	1

3	2
3	4
2	1
2	3
2	4
4	1
4	2
4	3

Line (LineBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.LineBatchOp

Python 类名：LineBatchOp

功能介绍

line算法(Large-scale Information Network Embedding)是一种graph embedding算法，可以将网络中的每一个点表示成连续特征空间中的一个点向量。 line算法有一阶相似度和二阶相似度两种描述方法，一阶相似度用于描述图中成对顶点之间的局部相似度，二阶相似度以两个顶点的邻域特征来描述顶点的相似度。 line算法的论文：LINE: Large-scale Information Network Embedding

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默
sourceCol	起始点列名	用来指定起始点列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
targetCol	中止点点列名	用来指定中止点列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
batchSize	batch大小	batch大小,按行计算	Integer		[1, +inf)	
isToUndigraph	是否转无向图	选为true时，会将当前图转成无向图，然后再游走	Boolean			false
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
minRhoRate	最小学习率的比例	最小学习率的比例	Double		[0.0, 1.0]	0.00
negative	负采样大小	负采样大小	Integer			5

order	阶数	选择一阶优化或是二阶优化	String		"FirstOrder", "SecondOrder"	"First
rho	学习率	学习率	Double		[0.0, +inf)	0.025
sampleRatioPerPartition	采样率	每轮迭代在每个partition上采样样本的比率	Double		[0.0, +inf)	1.0
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	100
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([\
["1L", "5L", 1.],\
["2L", "5L", 1.],\
["3L", "5L", 1.],\
["4L", "5L", 1.],\
["1L", "6L", 1.],\
["2L", "6L", 1.],\
["3L", "6L", 1.],\
["4L", "6L", 1.],\
["7L", "6L", 15.],\
```

```
["7L", "8L", 1.],\
["7L", "9L", 1.],\
["7L", "10L", 1.]])

data = BatchOperator.fromDataframe(df, schemaStr="source string, target string,
weight double")
line = LineBatchOp()\
    .setOrder("firstorder")\
    .setRho(.025)\
    .setVectorSize(5)\
    .setNegative(5)\
    .setIsToUndigraph(False)\
    .setMaxIter(20)\
    .setSampleRatioPerPartition(2.)\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .setWeightCol("weight")
line.linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.graph.LineBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LineBatchOpTest {
    @Test
    public void testLineBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1L", "5L", 1.),
            Row.of("2L", "5L", 1.),
            Row.of("3L", "5L", 1.),
            Row.of("4L", "5L", 1.),
            Row.of("1L", "6L", 1.),
            Row.of("2L", "6L", 1.),
            Row.of("3L", "6L", 1.),
            Row.of("4L", "6L", 1.),
            Row.of("7L", "6L", 15.),
            Row.of("7L", "8L", 1.),
            Row.of("7L", "9L", 1.));
        BatchOperator <?> data = new MemSourceBatchOp(df, "source string,
```

```
target string, weight double");
        BatchOperator <?> line = new LineBatchOp()
            .setOrder("firstorder")
            .setRho(.025)
            .setVectorSize(5)
            .setNegative(5)
            .setIsToUndigraph(false)
            .setMaxIter(20)
            .setSampleRatioPerPartition(2.)
            .setSourceCol("source")
            .setTargetCol("target")
            .setWeightCol("weight");
        line.linkFrom(data).print();
    }
}
```

运行结果

vertexId	vertexVector
10L	0.21389429778837246,0.1911353696863806,0.1316112606087454,-0.15504651922643958,0.9
1L	0.43122351608756165,0.29783837576159716,-0.6242134421932172,-0.5699927640850769,0.
2L	0.46132608248237156,0.35541855098613856,-0.5135636636216717,-0.6265741465013136,0.(
3L	0.22368818387133887,0.36608001332291756,-0.8030241335180479,-0.34202409002367046,0
4L	0.3570337202077478,0.7126398702420819,-0.4696611127705199,-0.23265871937843668,-0.2
5L	0.45792127623648854,0.47070594164899743,-0.56933846250774,-0.47612043020847256,0.1
6L	-0.5015833272182806,-0.4481268255333818,0.4612455253782732,0.43140895120924494,-0.3
7L	-0.3152196336701024,-0.4607786197664082,0.6574313951991989,0.48164878283999957,0.1
8L	-0.42030975318452385,0.05099491454249831,0.4269511935747453,-0.2071107702180848,0.7
9L	-0.22769540933018892,-0.5154933470780315,0.5261821838436239,0.5103449434077099,0.3

MetaPath To Vector (MetaPath2VecBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.MetaPath2VecBatchOp

Python 类名：MetaPath2VecBatchOp

功能介绍

沿着之前random walk的思路往前走，metapath2vec的方法提出了控制随机游走的模式，这样就可以在生成的序列上根据节点类型的不同来控制序列游走，这样也就可以对异质网络（Heterogeneous Networks）进行表征学习。在游走之前需要设定一个metapath，也就是游走时节点类型的模式

metapath2vec: Scalable Representation Learning for Heterogeneous Networks

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
metaPath	游走的模式	一般为用字符串表示，例如"ABDFA"	String	✓		
sourceCol	起始点列名	用来指定起始点列	String	✓		
targetCol	中止点点列名	用来指定中止点列	String	✓		
typeCol	节点类型列名	用来指定节点类型列	String	✓		
vertexCol	节点列名	用来指定节点列	String	✓	所选列类型为[STRING]	
walkLength	游走的长度	随机游走完向量的长度	Integer	✓		
walkNum	路径数目	每一个起始点游走出多少条路径	Integer	✓		
alpha	学习率	学习率	Double			0
batchSize	batch大小	batch大小，按行计算	Integer		[1, +inf)	

isToUndigraph	是否转无向图	选为true时，会将当前图转成无向图，然后再游走	Boolean			f
minCount	最小词频	最小词频	Integer			5
mode	metapath中word2vec的模式，分别为metapath2vec和metapath2vecpp	metapath的模式	String		"METAPATH2VEC", "METAPATH2VECPP"	"
negative	负采样大小	负采样大小	Integer			5
numIter	迭代次数	迭代次数，默认为1。	Integer			1
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	1
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	r
window	窗口大小	窗口大小	Integer			5

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["Bob", "Lucy", 1.],
    ["Lucy", "Bob", 1.],
    ["Lucy", "Bella", 1.],
```

```
    ["Bella", "Lucy", 1.],
    ["Alice", "Lisa", 1.],
    ["Lisa", "Alice", 1.],
    ["Lisa", "Karry", 1.],
    ["Karry", "Lisa", 1.],
    ["Karry", "Bella", 1.],
    ["Bella", "Karry", 1.]
])

source = BatchOperator.fromDataframe(df, schemaStr="start string, end string,
value double")

df2 = pd.DataFrame([
    ["Bob", "A"],
    ["Bella", "A"],
    ["Karry", "A"],
    ["Lucy", "B"],
    ["Alice", "B"],
    ["Lisa", "B"],
    ["Karry", "B"]
])

type = BatchOperator.fromDataframe(df2, schemaStr="node string, type string")

metapathBatchOp = MetaPath2VecBatchOp() \
    .setSourceCol("start") \
    .setTargetCol("end") \
    .setWeightCol("value") \
    .setVertexCol("node") \
    .setTypeCol("type") \
    .setMetaPath("ABA") \
    .setWalkNum(2) \
    .setWalkLength(2) \
    .setMinCount(1) \
    .setVectorSize(4)
metapathBatchOp.linkFrom(source, type).print()
```

运行结果

node	vec
Karry	-0.028718041256070137,0.02825581468641758,0.12125638127326965,0.1207452341914177
Bella	0.03437831997871399,-0.0477546751499176,0.012570690363645554,-0.0958133116364479
Bob	0.024427175521850586,0.07044785469770432,-0.04175269603729248,-0.06182029843330383
Lucy	0.05776885524392128,0.08288335055112839,-0.06490718573331833,0.026563744992017746

MetaPath游走 (MetaPathWalkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.MetaPathWalkBatchOp

Python 类名：MetaPathWalkBatchOp

功能介绍

MataPathWalk [1] 是描述随机游走的一种算法。在给定的图上，每次迭代过程中，点都会按照一定的metaPath转移到它的邻居上，转移到每个邻居的概率和连接这两个点的边的Type相关。通过这样的随机游走可以获得固定长度的随机游走序列，这可以类比自然语言中的句子。

[1] Dong et al. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. KDD2017.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
metaPath	游走的模式	一般为用字符串表示，例如"ABDFA"	String	✓		
sourceCol	起始点列名	用来指定起始点列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
targetCol	中止点点列名	用来指定中止点列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
typeCol	节点类型列名	用来指定节点类型列	String	✓	所选列类型为[STRING]	

vertexCol	节点列名	用来指定节点列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
walkLength	游走的长度	随机游走完向量的长度	Integer	✓		
walkNum	路径数目	每一个起始点游走出多少条路径	Integer	✓		
delimiter	分隔符	用来分割字符串	String			" "
isToUndigraph	是否转无向图	选为true时，会将当前图转成无向图，然后再游走	Boolean			false
samplingMethod	起始点列名	用来指定起始点列	String			"ALIAS"
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [1, 1, 1.0],
    [1, 2, 1.0],
    [2, 3, 1.0],
    [3, 4, 1.0],
    [4, 2, 1.0],
    [3, 1, 1.0],
    [2, 4, 1.0],
    [4, 1, 1.0]])

graph = BatchOperator.fromDataframe(df, schemaStr="start int, dest int, weight
double")

df2 = pd.DataFrame([
    [1,"A"],
    [2,"B"],
    [3,"A"],
    [4,"B"]])

node = BatchOperator.fromDataframe(df2, schemaStr="node int, type string")

MetaPathWalkBatchOp() \
            .setWalkNum(10) \
            .setWalkLength(20) \
            .setSourceCol("start") \
            .setTargetCol("dest") \
            .setIsToUndigraph(True) \
            .setMetaPath("ABA") \
            .setVertexCol("node") \
            .setWeightCol("weight") \
            .setTypeCol("type").linkFrom(graph, node).print();
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.graph.MetaPathWalkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MetaPathWalkBatchOpTest {
```

```
    @Test
    public void testMetaPathWalkBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, 1, 1.0),
            Row.of(1, 2, 1.0),
            Row.of(2, 3, 1.0),
            Row.of(3, 4, 1.0),
            Row.of(4, 2, 1.0),
            Row.of(3, 1, 1.0),
            Row.of(2, 4, 1.0)
        );
        BatchOperator <?> graph = new MemSourceBatchOp(df, "start int, dest
int, weight double");
        List <Row> df2 = Arrays.asList(
            Row.of(1, "A"),
            Row.of(2, "B"),
            Row.of(3, "A")
        );
        BatchOperator <?> node = new MemSourceBatchOp(df2, "node int, type
string");
        new MetaPathWalkBatchOp()
            .setWalkNum(10)
            .setWalkLength(20)
            .setSourceCol("start")
            .setTargetCol("dest")
            .setIsToUndigraph(true)
            .setMetaPath("ABA")
            .setVertexCol("node")
            .setWeightCol("weight")
            .setTypeCol("type").linkFrom(graph, node).print();
    }
}
```

运行结果

path
1 4 3 2 3 2 1 2 1 2 1 2 1 2 1 2 3 2 3 2
1 4 3 2 1 4 3 2 3 4 3 2 1 4 1 4 1 2 3 2
1 2 3 4 3 2 1 4 3 4 3 4 1 2 3 2 3 4 3 2
3 4 3 4 1 4 1 4 1 2 1 2 1 2 3 2 1 4 3 2
1 4 1 2 3 2 1 2 3 2 3 2 3 2 1 2 1 4 3 2
3 2 3 4 3 2 3 2 3 4 1 2 3 2 1 2 1 4 3 2

3 4 3 4 1 2 3 4 1 4 1 4 3 2 3 4 3 2 1 2
3 4 1 4 1 4 3 4 1 2 1 2 1 4 1 4 3 2 1 2
1 2 1 4 3 4 3 2 3 2 1 2 1 2 1 2 3 2 1 2
3 2 1 2 3 2 3 4 3 2 1 4 3 4 1 2 1 2 1 2
1 2 1 4 1 4 1 2 1 2 3 2 3 4 1 2 1 2 1 4
1 2 1 4 3 2 1 2 1 4 1 4 1 4 1 2 1 2 1 2
1 4 1 4 1 2 1 2 1 4 1 2 3 2 1 2 1 2 1 2
3 4 3 4 3 2 1 2 3 2 3 4 1 2 1 2 1 2 1 4
3 2 3 4 3 4 1 4 3 2 3 4 3 2 3 2 3 4 1 4
3 2 3 2 3 4 3 2 1 2 1 4 1 2 1 2 3 4 1 4
1 4 3 4 1 2 1 4 1 2 3 2 1 2 1 2 3 4 1 4
3 4 3 2 3 4 1 2 1 4 3 4 3 2 1 4 1 4 1 2
3 4 3 4 1 2 3 2 1 4 1 4 3 2 1 4 1 4 1 4
1 4 1 2 3 2 3 4 1 2 1 2 1 2 3 2 1 4 1 2

备注

- 上述表中的结点类型可以是任意类型。
- 给定metaPath 后算法按照 metaPath 去游走，直到路径长度达到walkLength。
- 给定的metaPath必须首尾两个结点的Type一样，正确的格式例子： “ABCA”，“VDEEV”，“ABA”等。
- 如果给定metaPath为“ABDSA”，walkLength为10，则游走得到的结果将是 “ABDSABDSAB”，达到10个结点是会截断。

模块度计算 (ModularityCalBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.ModularityCalBatchOp

Python 类名：ModularityCalBatchOp

功能介绍

模块度是一种评价图社群划分好坏的指标，来评估网络结构中划分出来社区的紧密程度，取值范围为-0.5到1之间。通常认为大于0.3的图是划分出较为明显社群的。

要计算一个网络的模块度，需要构造一个具有相同节点度分布的随机网络作为参照。通俗地来说，模块度的物理含义是：在社团内，实际的边数与随机情况下的边数的差距。如果差距比较大，说明社团内部密集程度显著高于随机情况，社团划分的质量较好。模块度取值范围在[-0.5,1]之间。如果节点组中的连边数量超过了随机分配时所得到的期望连边数量，模块度为正数。没有超过，则为负数。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓		
targetCol	边表中终点所在列	边表中终点所在列	String	✓		
vertexCol	点列	输入点表中点信息所在列	String	✓		

vertexCommunityCol	社群信息列	输入点表中点的社群信息所在列	String	✓	所选列类型为[INTEGER, LONG]	
edgeWeightCol	边权重列	表示边权重的列	String			null
outCol	输出列名	输出列名	String			"modularity"
toUndirected	输入数据是否为有向图	表明输入的是有向图还是无向图，如果是有向图，算法就会将其转为无向图来处理。	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[3, 1],\
        [3, 0],\
        [0, 1],\
        [0, 2],\
        [2, 1],\
        [2, 4],\
        [5, 4],\
        [7, 4],\
        [5, 6],\
        [5, 8],\
```

```
        [5, 7],\
        [7, 8],\
        [6, 8],\
        [12, 10],\
        [12, 11],\
        [12, 13],\
        [12, 9],\
        [10, 9],\
        [8, 9],\
        [13, 9],\
        [10, 7],\
        [10, 11],\
        [11, 13]])

edges = BatchOperator.fromDataframe(df, schemaStr="source int, target int")

df2 = pd.DataFrame([[2, 0],\
        [4, 1],\
        [7, 1],\
        [8, 1],\
        [9, 2],\
        [10, 2]])

vertices = BatchOperator.fromDataframe(df2, schemaStr="vertex int, label
bigint")

communityDetectionClassify = CommunityDetectionClassifyBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .setVertexCol("vertex")\
    .setVertexLabelCol("label")\
    .setOutputCols(["vertex", "label"])
community = communityDetectionClassify.linkFrom(edges, vertices)


modularityCal = ModularityCalBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .setOutCol("modularity")\
    .setVertexCol("vertex")\
    .setVertexCommunityCol("label")
modularityCal.linkFrom(edges, community).print()
```

Java 代码

```
import org.apache.flink.types.Row;
```

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ModularityCalBatchOpTest extends AlinkTestBase {
    @Test
    public void testModularityCal() throws Exception {

        List <Row> edgesList = Arrays.asList(
            Row.of(3, 1),
            Row.of(3, 0),
            Row.of(0, 1),
            Row.of(0, 2),
            Row.of(2, 1),
            Row.of(2, 4),
            Row.of(5, 4),
            Row.of(7, 4),
            Row.of(5, 6),
            Row.of(5, 8),
            Row.of(5, 7),
            Row.of(7, 8),
            Row.of(6, 8),
            Row.of(12, 10),
            Row.of(12, 11),
            Row.of(12, 13),
            Row.of(12, 9),
            Row.of(10, 9),
            Row.of(8, 9),
            Row.of(13, 9),
            Row.of(10, 7),
            Row.of(10, 11),
            Row.of(11, 13));
        BatchOperator edges = new MemSourceBatchOp(edgesList, "source int,
target int");
        List <Row> nodesList = Arrays.asList(Row.of(2, 0),
            Row.of(4, 1),
            Row.of(7, 1),
            Row.of(8, 1),
            Row.of(9, 2),
            Row.of(10, 2));
        BatchOperator nodes = new MemSourceBatchOp(nodesList, "vertex int,
label int");

        CommunityDetectionClassifyBatchOp communityDetectionClassify = new
```

```
CommunityDetectionClassifyBatchOp()
            .setSourceCol("source")
            .setTargetCol("target")
            .setVertexCol("vertex")
            .setVertexLabelCol("label")
            .setOutputCols("vertex", "label")
            .linkFrom(edges, nodes);

        ModularityCalBatchOp modularityCal = new ModularityCalBatchOp()
            .setSourceCol("source")
            .setTargetCol("target")
            .setOutCol("modularity")
            .setVertexCol("vertex")
            .setVertexCommunityCol("label");
        modularityCal.linkFrom(edges, communityDetectionClassify).print();
    }
}
```

运行结果

modularity
0.522684

MultiSourceShortestPathBatchOp (MultiSourceShortestPathBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.MultiSourceShortestPathBatchOp

Python 类名：MultiSourceShortestPathBatchOp

功能介绍

给定的边以及多个源点，求图中所有点到给定源点的最短路径。输出所有点的根节点（给定源点之一）、距离和最短路径的节点序列。

对于不能连接到所有源点的节点，输出的根节点为null，距离为-1，节点序列为空。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓		
sourcePointCol	源点的列	源点的列	String	✓		
targetCol	边表中终点所在列	边表中终点所在列	String	✓		
distanceCol	输出表中距离所在列	输出表中距离所在列	String			"distance"
edgeWeightCol	边权重列	表示边权重的列	String			null
maxIter	最大迭代次数	最大迭代次数	Integer		[1, +inf)	50
nodeListCol	输出表最短路径上节点所在列	输出表中最短路径上节点所在列	String			"node_list"

rootCol	输出表中根节点所在列	输出表中根节点所在列	String			"root_node"
toUndirected	输入数据是否为有向图	表明输入的是有向图还是无向图，如果是有向图，算法就会将其转为无向图来处理。	Boolean			true
vertexCol	输出表中点所在列	输出表中点所在列	String			"vertex"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[0, 1, 0.4],
                   [1, 2, 1.3],
                   [2, 3, 1.0],
                   [3, 4, 1.0],
                   [4, 5, 1.0],
                   [5, 6, 1.0],
                   [6, 7, 1.0],
                   [7, 8, 1.0],
                   [8, 9, 1.0],
                   [9, 6, 1.0],
                   [19, 16, 1.0]])
source_df = pd.DataFrame([[1, 1],
                          [5, 5]])
data = BatchOperator.fromDataframe(df, schemaStr="source int, target int,
weight double")
source_data = BatchOperator.fromDataframe(source_df, schemaStr="source int,
target int")

MultiSourceShortestPathBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .setEdgeWeightCol("weight")\
    .setSourcePointCol("source")\
```

```
    .linkFrom(data, source_data)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Assert;
import org.junit.Test;

import java.util.List;

public class MultiSourceShortestPathBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
        Row[] rows = new Row[]{
            Row.of(0, 1, 0.4),
            Row.of(1, 2, 1.3),
            Row.of(2, 3, 1.0),
            Row.of(3, 4, 1.0),
            Row.of(4, 5, 1.0),
            Row.of(5, 6, 1.0),
            Row.of(6, 7, 1.0),
            Row.of(7, 8, 1.0),
            Row.of(8, 9, 1.0),
            Row.of(9, 6, 1.0),
            Row.of(19, 16, 1.0),
        };
        Row[] sources = new Row[]{
            Row.of(1, 1),
            Row.of(5, 5),
        };
        BatchOperator inData = new MemSourceBatchOp(rows, new String[]
{"source", "target", "weight"});
        BatchOperator sourceBatchOp = new MemSourceBatchOp(sources, new
String[]{"source", "target"});
        MultiSourceShortestPathBatchOp op = new
MultiSourceShortestPathBatchOp()
            .setSourceCol("source")
            .setTargetCol("target")
            .setEdgeWeightCol("weight")
            .setSourcePointCol("source");
```

```
        BatchOperator res = op.linkFrom(inData, sourceBatchOp);

        res.lazyPrint(20);
    }

}
```

运行结果

vertex	root_node	node_list	distance
19	null		-1.0000
16	null		-1.0000
0	1	0,1	0.4000
1	1	1	0.0000
2	1	2,1	1.3000
3	5	3,4,5	2.0000
4	5	4,5	1.0000
5	5	5	0.0000
8	5	8,7,6,5	3.0000
7	5	7,6,5	2.0000
6	5	6,5	1.0000
9	5	9,6,5	2.0000

备注

源点个数少，且图中点非常多的情况下，运行速度会比较慢，这种情况建议使用单源最短路径算法。

Node To Vector (Node2VecBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.Node2VecBatchOp

Python 类名：Node2VecBatchOp

功能介绍

node2vec是一种用于网络中的特征学习有效的可扩展算法，该算法可以使用SGD有效地优化，能根据网络中的既定原则，为发现符合不同等值的表示提供了灵活性

node2vec: Scalable Feature Learning for Networks

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	起始点列名	用来指定起始点列	String	✓		
targetCol	中止点点列名	用来指定中止点列	String	✓		
walkLength	游走的长度	随机游走完向量的长度	Integer	✓		
walkNum	路径数目	每一个起始点游走出多少条路径	Integer	✓		
alpha	学习率	学习率	Double			0.025
batchSize	batch大小	batch大小, 按行计算	Integer		[1, +inf)	
isToUndigraph	是否转无向图	选为true时，会将当前图转成无向图，然后再游走	Boolean			false
minCount	最小词频	最小词频	Integer			5
negative	负采样大小	负采样大小	Integer			5
numIter	迭代次数	迭代次数，默认为1。	Integer			1

p	p	p越小越趋向于访问到已经访问的节点，反之则趋向于访问没有访问过的节点	Double			1.0
q	q	q>1时行为类似于bfs趋向于访问和访问过的节点相连的节点，q<1时行为类似于dfs	Double			1.0
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"true"
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	100
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
window	窗口大小	窗口大小	Integer			5
wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["Bob", "Lucy", 1.],
```

```
    ["Lucy", "Bob", 1.],
    ["Lucy", "Bella", 1.],
    ["Bella", "Lucy", 1.],
    ["Alice", "Lisa", 1.],
    ["Lisa", "Alice", 1.],
    ["Lisa", "Karry", 1.],
    ["Karry", "Lisa", 1.],
    ["Karry", "Bella", 1.],
    ["Bella", "Karry", 1.]
])
source = BatchOperator.fromDataframe(df, schemaStr="start string, end string,
value double")

node2vecBatchOp = Node2VecBatchOp() \
  .setSourceCol("start")            \
  .setTargetCol("end")              \
  .setWeightCol("value")            \
  .setWalkNum(2)                    \
  .setWalkLength(2)                 \
  .setMinCount(1)                   \
  .setVectorSize(4)
node2vecBatchOp.linkFrom(source).print()
```

运行结果

node	vec
Karry	0.02435881271958351,0.0703350380063057,-0.04173225536942482,-0.06183897703886032
Bella	-0.028720347210764885,0.02828666940331459,0.12123052030801773,0.12075022608041763
Alice	0.03435942903161049,-0.04773801192641258,0.0125938905403018,-0.09576953202486038
Lisa	-0.07306616753339767,-0.11595576256513596,-0.04181118682026863,0.03970039263367653
Bob	0.0577755942940712,0.08282522112131119,-0.06487344205379486,0.026600968092679977
Lucy	0.05773818120360374 5,-0.09987597167491913,-0.022486409172415733,-0.02312176302075386

Node2Vec游走 (Node2VecWalkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.Node2VecWalkBatchOp

Python 类名：Node2VecWalkBatchOp

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	起始点列名	用来指定起始点列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
targetCol	中止点点列名	用来指定中止点列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
walkLength	游走的长度	随机游走完向量的长度	Integer	✓		
walkNum	路径数目	每一个起始点游走出多少条路径	Integer	✓		
delimiter	分隔符	用来分割字符串	String			" "
isToUndigraph	是否转无向图	选为true时，会将当前图转成无向图，然后再游走	Boolean			false

p	算法参数P	控制随机游走序列的跳转概率	Double			1.0
q	算法参数Q	控制随机游走序列的跳转概率	Double			1.0
samplingMethod	起始点列名	用来指定起始点列	String			"ALIAS"
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1, 1, 1.0],
    [1, 2, 1.0],
    [2, 3, 1.0],
    [3, 4, 1.0],
    [4, 2, 1.0],
    [3, 1, 1.0],
    [2, 4, 1.0],
    [4, 1, 1.0]])

source = BatchOperator.fromDataframe(df, schemaStr="start int, dest int, weight
double")
```

```
n2vWalkBatchOp = Node2VecWalkBatchOp() \
                 .setWalkNum(4) \
                 .setWalkLength(50) \
                 .setDelimiter(",") \
                 .setSourceCol("start") \
                 .setTargetCol("dest") \
                 .setIsToUndigraph(True) \
                 .setWeightCol("weight")

n2vWalkBatchOp.linkFrom(source).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.graph.Node2VecWalkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class Node2VecWalkBatchOpTest {
    @Test
    public void testNode2VecWalkBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, 1, 1.0),
            Row.of(1, 2, 1.0),
            Row.of(2, 3, 1.0),
            Row.of(3, 4, 1.0),
            Row.of(4, 2, 1.0),
            Row.of(3, 1, 1.0),
            Row.of(2, 4, 1.0)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df, "start int, dest
int, weight double");
        BatchOperator <?> n2vWalkBatchOp = new Node2VecWalkBatchOp()
            .setWalkNum(4)
            .setWalkLength(50)
            .setDelimiter(",")
            .setSourceCol("start")
            .setTargetCol("dest")
            .setIsToUndigraph(true)
            .setWeightCol("weight");
        n2vWalkBatchOp.linkFrom(source).print();
```

```
    }
}
```

运行结果

path
3,2,1,1,4,2,3,4,2,1,3,1,1,1,3,1,3,4,1,2,4,3,2,1,1,3,2,4,3,4,1,4,2,1,2,1,4,3,1,2,1,3,4,2,4,3,2,3,4,1
2,3,2,4,2,1,2,3,2,3,2,4,3,1,2,3,4,2,4,2,3,2,3,2,4,2,1,3,1,4,1,1,4,2,1,2,4,1,3,1,1,3,4,2,4,2,3,4,2,4
4,2,1,4,1,1,1,2,3,4,2,3,2,3,2,4,1,2,3,2,1,2,4,3,1,1,2,4,3,4,2,4,1,2,4,3,1,4,2,4,2,1,3,4,2,1,2,4,3,4
4,3,1,1,1,3,4,3,4,1,2,4,2,3,2,1,1,1,2,3,4,1,2,4,3,4,3,1,4,2,3,2,4,1,1,1,3,1,3,2,4,2,4,3,1,1,1,3,2,1
4,3,1,1,3,2,3,1,4,1,2,1,3,4,3,4,2,1,2,4,2,3,4,2,4,2,4,2,3,1,4,3,2,4,1,2,3,2,1,1,3,1,1,4,1,4,1,4,1,2
1,3,4,1,2,3,1,3,4,2,1,4,3,2,1,3,1,4,1,4,3,1,4,2,1,2,3,4,3,4,2,3,4,3,4,1,1,1,1,2,4,1,2,4,1,2,4,2,3,1
3,4,1,4,1,2,1,3,2,4,2,1,2,3,1,4,1,2,4,2,4,3,4,3,2,3,2,1,2,1,1,1,4,2,3,4,1,1,4,2,3,4,3,1,4,3,4,1,4,3
2,3,1,3,4,1,1,1,4,3,4,1,2,3,2,1,1,3,4,3,2,3,1,3,4,3,2,1,4,3,1,1,2,4,2,1,3,1,3,1,2,3,1,4,3,2,1,2,1,1
1,2,1,3,1,4,2,3,2,1,3,4,2,4,3,4,2,4,2,1,4,3,2,3,4,1,4,2,1,4,1,3,4,2,1,4,1,4,3,1,3,2,4,1,1,4,1,1,2,3
1,2,4,2,1,2,4,3,4,3,4,3,1,2,1,2,3,2,3,4,2,4,3,2,3,2,3,1,1,1,4,2,1,4,1,2,1,2,1,1,2,4,2,1,4,2,3,1,1,4
3,4,1,1,1,2,4,2,4,1,1,1,2,3,2,4,2,1,2,3,4,1,1,3,4,1,2,4,3,2,1,1,4,1,2,1,2,4,2,4,1,3,4,2,1,3,4,3,2,1
3,2,3,4,3,1,2,4,2,4,1,2,3,4,3,1,2,4,3,2,3,2,3,2,3,4,3,2,4,3,4,2,4,3,1,1,4,1,4,1,3,1,2,3,4,1,4,3,4,2
1,4,2,4,1,4,1,2,1,1,2,3,1,4,3,4,1,3,1,3,4,2,4,3,2,3,2,3,4,1,3,2,3,4,2,3,4,1,2,4,2,3,4,3,2,3,1,4,2,3
2,4,3,1,1,3,1,2,4,2,4,3,2,1,2,4,3,2,3,1,1,3,2,1,3,1,4,1,3,1,1,1,1,2,1,3,1,1,3,4,2,1,2,1,1,2,4,3,1,2
4,2,1,4,3,2,1,1,3,1,4,1,3,2,4,2,4,2,4,3,4,1,3,4,2,3,2,1,3,1,1,1,4,2,1,2,4,1,4,2,1,4,1,2,4,2,1,4,1,2
2,4,1,4,3,2,1,3,1,3,1,4,1,2,4,1,3,4,2,1,1,3,1,3,1,1,1,1,1,2,4,1,4,3,1,2,1,2,4,1,3,4,1,2,3,1,4,2,4,3

随机游走 (RandomWalkBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.RandomWalkBatchOp

Python 类名：RandomWalkBatchOp

功能介绍

RandomWalk是deepwalk [1] 中描述随机游走的一种算法。 在给定的图上，每次迭代过程中，点都会转移到它的邻居上，转移到每个邻居的概率和连接这两个点的边的权重相关。 通过这样的随机游走可以获得固定长度的随机游走序列，这可以类比自然语言中的句子。

[1] Bryan Perozzi et al. DeepWalk: online learning of social representations. KDD 2014.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	起始点列名	用来指定起始点列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
targetCol	中止点点列名	用来指定中止点列	String	✓	所选列类型为[INTEGER, LONG, STRING]	
walkLength	游走的长度	随机游走完向量的长度	Integer	✓		
walkNum	路径数目	每一个起始点游走出多少条路径	Integer	✓		

delimiter	分隔符	用来分割字符串	String			" "
isToUndigraph	是否转无向图	选为true时，会将当前图转成无向图，然后再游走	Boolean			false
isWeightedSampling	是否为加权采样	该算法支持加权采样和随机采样两种采样方式	Boolean			true
samplingMethod	起始点列名	用来指定起始点列	String			"ALIAS"
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1, 1, 1.0],
    [1, 2, 1.0],
    [2, 3, 1.0],
    [3, 4, 1.0],
    [4, 2, 1.0],
```

```
    [3, 1, 1.0],
    [2, 4, 1.0],
    [4, 1, 1.0]])

graph = BatchOperator.fromDataframe(df, schemaStr="start int, dest int, weight
double")

RandomWalkBatchOp() \
            .setWalkNum(5) \
            .setWalkLength(20) \
            .setSourceCol("start") \
            .setTargetCol("dest") \
            .setIsToUndigraph(True) \
            .setWeightCol("weight").linkFrom(graph).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.graph.RandomWalkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RandomWalkBatchOpTest {
    @Test
    public void testRandomWalkBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, 1, 1.0),
            Row.of(1, 2, 1.0),
            Row.of(2, 3, 1.0),
            Row.of(3, 4, 1.0),
            Row.of(4, 2, 1.0),
            Row.of(3, 1, 1.0),
            Row.of(2, 4, 1.0)
        );
        BatchOperator <?> graph = new MemSourceBatchOp(df, "start int, dest
int, weight double");
        new RandomWalkBatchOp()
            .setWalkNum(5)
            .setWalkLength(20)
            .setSourceCol("start")
            .setTargetCol("dest")
            .setIsToUndigraph(true)
```

```
            .setWeightCol("weight").linkFrom(graph).print();
    }
}
```

运行结果

path
2 4 1 1 2 1 3 2 1 3 1 1 2 4 2 4 3 1 4 3
1 1 4 2 3 4 3 1 2 4 2 3 1 4 1 1 1 2 1 2
1 3 4 1 2 4 2 3 4 2 1 2 1 3 2 1 2 1 3 4
3 2 3 1 1 3 1 3 1 4 2 3 2 1 1 1 4 2 3 2
4 3 1 4 1 4 3 1 2 3 2 3 4 3 4 1 4 3 2 1
2 3 4 3 2 4 3 2 1 4 2 3 1 4 3 1 2 4 1 4
3 2 4 1 3 2 1 2 4 3 1 3 1 2 3 1 3 2 3 4
4 2 4 2 3 4 1 1 1 4 2 4 3 4 3 2 1 2 3 2
1 3 1 2 3 2 1 4 1 3 2 1 3 2 3 1 4 2 1 1
2 1 2 4 2 1 2 4 3 2 3 2 4 3 1 3 1 2 3 4
3 1 1 2 4 1 4 2 4 1 3 2 4 1 3 2 1 2 1 3
3 1 1 4 3 1 3 4 2 4 3 1 3 1 4 2 1 3 1 1
4 2 1 3 4 2 3 1 3 4 2 3 4 3 2 4 2 3 1 4
4 1 1 4 3 1 4 3 1 3 4 3 4 2 1 3 2 3 1 3
1 4 2 1 3 4 1 2 3 2 4 1 4 1 2 3 4 3 2 1
3 4 1 4 1 3 2 1 4 2 3 4 1 1 3 2 3 2 4 1
2 1 4 1 1 3 1 2 1 1 3 2 1 3 1 3 4 2 3 2
2 3 2 1 1 4 3 4 1 1 3 2 1 2 1 2 1 2 3 1
4 3 4 3 4 2 3 4 3 4 1 3 1 2 1 3 2 4 1 2
1 2 4 3 2 4 1 1 2 1 2 4 3 1 2 3 4 3 1 4

单源最短路径 (SingleSourceShortestPathBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.SingleSourceShortestPathBatchOp

Python 类名：SingleSourceShortestPathBatchOp

功能介绍

对于给定的图以及给定的单个源点，求给定点到图中所有点的最短路径。如果某点与源点不相连，输出的距离为长整型可取到的最大值。

单源最短路径的应用场景有网络设计、路径规划等。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓		
sourcePoint	源点的值	源点的值	String	✓		
targetCol	边表中终点所在列	边表中终点所在列	String	✓		
distanceCol	输出表中距离所在列	输出表中距离所在列	String			"distance"
edgeWeightCol	边权重列	表示边权重的列	String			null
maxIter	最大迭代次数	最大迭代次数	Integer		[1, +inf)	50

toUndirected	输入数据是否为有向图	表明输入的是有向图还是无向图，如果是有向图，算法就会将其转为无向图来处理。	Boolean			true
vertexCol	输出表中点所在列	输出表中点所在列	String			"vertex"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[1, 2],\
[2, 3],\
[3, 4],\
[4, 5],\
[5, 6],\
[6, 7],\
[7, 8],\
[8, 9],\
[9, 6]])

data = BatchOperator.fromDataframe(df, schemaStr="source double, target
double")
SingleSourceShortestPathBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .setSourcePoint("1")\
    .linkFrom(data)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
```

```
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

public class SingleSourceShortestPathBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
        Row[] rows = new Row[]{
            Row.of(1, 2, 1.0),
            Row.of(2, 3, 1.0),
            Row.of(3, 4, 1.0),
            Row.of(4, 5, 1.0),
            Row.of(5, 6, 1.0),
            Row.of(6, 7, 1.0),
            Row.of(7, 8, 1.0),
            Row.of(8, 9, 1.0),
            Row.of(9, 6, 1.0)
        };
        BatchOperator inData = new MemSourceBatchOp(rows, new String[]
{"source", "target", "weight"});
        SingleSourceShortestPathBatchOp op = new
SingleSourceShortestPathBatchOp()
            .setSourceCol("source")
            .setTargetCol("target")
            .setSourcePoint("1");

        BatchOperator res = op.linkFrom(inData);
        res.print();
    }
}
```

运行结果

vertex	distance
1	0.0000
2	1.0000
5	4.0000
9	6.0000
4	3.0000
3	2.0000
8	7.0000
6	5.0000

7	6.0000

树深度 (TreeDepthBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.TreeDepthBatchOp

Python 类名：TreeDepthBatchOp

功能介绍

对于给定的有向图，判断该有向图是否为一棵树或是否为森林。如果判定成功，则输出每个结点所在树的根结点以及该结点的深度。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓		
targetCol	边表中终点所在列	边表中终点所在列	String	✓		
edgeWeightCol	边权重列	表示边权重的列	String			null
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
outputOriginCol	输出表中源点所在列	输入表中源点所在列	String			"root"
outputVertexCol	输出表中点所在列	输入表中点所在列	String			"vertices"

toUndirected	输入数据是否为有向图	表明输入的是有向图还是无向图，如果是有向图，算法就会将其转为无向图来处理。	Boolean			true
treeDepthCol	输出表中树深度所在列	输出表中树深度所在列	String			"treeDepth"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[0, 1],\
[0, 2],\
[1, 3],\
[1, 4],\
[2, 5],\
[4, 6],\
[7, 8],\
[7, 9],\
[9, 10],\
[9, 11]])

data = BatchOperator.fromDataframe(df, schemaStr="source double, target
double")
TreeDepthBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .linkFrom(data)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;

public class TreeDepthBatchOpTest extends AlinkTestBase {

    @Test
    public void testTreeDepth() throws Exception {
        Row[] rows = new Row[] {
            Row.of(0.0, 1.0),
            Row.of(0.0, 2.0),
            Row.of(1.0, 3.0),
            Row.of(1.0, 4.0),
            Row.of(2.0, 5.0),
            Row.of(4.0, 6.0),
            Row.of(7.0, 8.0),
            Row.of(7.0, 9.0),
            Row.of(9.0, 10.0),
            Row.of(9.0, 11.0)
        };
        BatchOperator inData = new MemSourceBatchOp(Arrays.asList(rows),
"source double,target double");

        TreeDepthBatchOp op = new TreeDepthBatchOp()
            .setSourceCol("source")
            .setTargetCol("target")
            .linkFrom(inData);
        op.print();
    }
}
```

运行结果

vertices	root	treeDepth
0.0000	0.0000	0.0000
4.0000	0.0000	2.0000
6.0000	0.0000	3.0000
1.0000	0.0000	1.0000
2.0000	0.0000	1.0000
3.0000	0.0000	2.0000

5.0000	0.0000	2.0000
7.0000	7.0000	0.0000
8.0000	7.0000	1.0000
9.0000	7.0000	1.0000
11.0000	7.0000	2.0000
10.0000	7.0000	2.0000

计数三角形 (TriangleListBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.TriangleListBatchOp

Python 类名：TriangleListBatchOp

功能介绍

对于给定的图，计算图中边所能构成的三角形，并将其输出。

对网络图中进行三角形个数计数可以根据三角形数量反应网络中的稠密程度和质量。 可以用于社区发现，如微博中你关注的人也关注你，大家的关注关系中有很多三角形，说明社区很强很稳定，大家联系比较紧密；如果一个人只关注了很多人，却没有形成三角形，则说明社交群体很小很松散。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓		
targetCol	边表中终点所在列	边表中终点所在列	String	✓		
outputCols	输出列名	输出列的列名	String[]			["node1","node2","node3"]

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[1.0,2.0],\
[1.0,3.0],\
[1.0,4.0],\
[1.0,5.0],\
[1.0,6.0],\
[2.0,3.0],\
[4.0,3.0],\
```

```
[5.0,4.0],\
[5.0,6.0],\
[5.0,7.0],\
[6.0,7.0]])

data = BatchOperator.fromDataframe(df, schemaStr="source double, target
double")
TriangleListBatchOp()\
    .setSourceCol("source")\
    .setTargetCol("target")\
    .linkFrom(data)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;

public class TriangleListBatchOpTest extends AlinkTestBase {
    @Test
    public void testTriangleList() throws Exception {
        Row[] rows = new Row[] {
            Row.of(1.0, 2.0),
            Row.of(1.0, 3.0),
            Row.of(1.0, 4.0),
            Row.of(1.0, 5.0),
            Row.of(1.0, 6.0),
            Row.of(2.0, 3.0),
            Row.of(4.0, 3.0),
            Row.of(5.0, 4.0),
            Row.of(5.0, 6.0),
            Row.of(5.0, 7.0),
            Row.of(6.0, 7.0)
        };
        BatchOperator inData = new MemSourceBatchOp(Arrays.asList(rows),
"source double,target double");

        TriangleListBatchOp op = new TriangleListBatchOp()
            .setSourceCol("source").setOutputCols(new String[] {"node1",
"node2", "node3"})
            .setTargetCol("target").linkFrom(inData);
```

```
        op.print();
    }
}
```

运行结果

node1	node2	node3
6.0000	5.0000	1.0000
4.0000	5.0000	1.0000
7.0000	5.0000	6.0000
4.0000	1.0000	3.0000
2.0000	1.0000	3.0000

点聚类系数 (VertexClusterCoefficientBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.VertexClusterCoefficientBatchOp

Python 类名：VertexClusterCoefficientBatchOp

功能介绍

对于给定的图，计算图中的每个顶点，输出其邻居个数，其邻点间的边数，以及每个点的聚类系数。

参数说明

该组件有一个输入桩，表示图的边集。组件有一个输出桩，输出图中所有顶点，其邻点个数，其邻点间的边数，以及其对应的聚类系数。

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	边表中起点所在列	边表中起点所在列	String	✓		
targetCol	边表中终点所在列	边表中终点所在列	String	✓		
outputCols	输出列名	输出列所在列名	String[]			["vertexId","vertexDegree","edgeNum","coeffi

toUndirected	输入数据是否为有向图	表明输入的是有向图还是无向图，如果是有向图，算法就会将其转为无向图来处理。	Boolean			true

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([[1, 2],
                   [1, 3],
                   [1, 4],
                   [1, 5],
                   [1, 6],
                   [2, 3],
                   [4, 3],
                   [5, 4],
                   [5, 6],
                   [5, 7],
                   [6, 7]])

data = BatchOperator.fromDataframe(df, schemaStr="source double, target
double")
```

```
vertexClusterCoefficient = VertexClusterCoefficientBatchOp() \
    .setSourceCol("source") \
    .setTargetCol("target") \
    .setOutputCols(["vertexId", "vertexDegree", "edgeNum", "coefficient"])
vertexClusterCoefficient.linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

public class VertexClusterCoefficientBatchOpTest {
    @Test
    public void test() throws Exception {
        Row[] rows = new Row[] {
            Row.of(1, 2),
            Row.of(1, 3),
            Row.of(1, 4),
            Row.of(1, 5),
            Row.of(1, 6),
            Row.of(2, 3),
            Row.of(4, 3),
            Row.of(5, 4),
            Row.of(5, 6),
            Row.of(5, 7),
            Row.of(6, 7)
        };
        BatchOperator <?> inData = new MemSourceBatchOp(rows, "source int,
target int");
        VertexClusterCoefficientBatchOp op = new
VertexClusterCoefficientBatchOp()
            .setSourceCol("source")
            .setTargetCol("target")
            .setOutputCols(new String[] {"vertexId", "vertexDegree", "edgeNum",
"coefficient"})
            .linkFrom(inData);
        op.print();
    }
}
```

运行结果

vertexId	vertexDegree	edgeNum	coefficient

2	2	1	1.0000
1	5	4	0.4000
7	2	1	1.0000
5	4	3	0.5000
3	3	2	0.6667
4	3	2	0.6667
6	3	2	0.6667

点邻居搜索 (VertexNeighborSearchBatchOp)

Java 类名：com.alibaba.alink.operator.batch.graph.VertexNeighborSearchBatchOp

Python 类名：VertexNeighborSearchBatchOp

功能介绍

节点k度邻居子图查询算法，根据用户输入的图数据和一组节点，在图中查找它们的k度邻居，然后导出子图。在子图较小时，还提供图可视化功能。

参数说明

该组件至少接入一个输入桩，表示图的边集；可选接入第二个输入桩，表示图的点集。组件输出对应于输入桩，分别表示子图的边集和点集，其中点集在第二个输入桩有输入时才有输出。

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sources	其实节点集合	起始节点集合	String[]	✓		
depth	深度	寻找邻居的深度	Integer			1
edgeSourceCol	有向边起始节点列	表示有向边起始节点的列名	String			"source"
edgeTargetCol	有向边目标节点列	表示有向边目标节点的列名	String			"target"
isUndirected	是否为无向图	表示是否为无向图	Boolean			false
vertexIdCol	节点ID列	表示节点ID的列名	String			"id"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
```

```
    ["Alice", "Lisa", 1., "hello"],
        ["Lisa", "Alice", 1., "hello"],
        ["Lisa", "Karry", 1., "hello"],
        ["Karry", "Lisa", 1., "213"],
        ["Karry", "Bella", 1., "hello"],
        ["Bella", "Karry", 1., "h123ello"],
        ["Bella", "Lucy", 1., "hello"],
        ["Lucy", "Bella", 1., "hello"],
        ["Lucy", "Bob", 1., "123"],
        ["Bob", "Lucy", 1., "hello"],
        ["John", "Bob", 1., "hello"],
        ["Bob", "John", 1., "hello"],
        ["John", "Stella", 1., "123"],
        ["Stella", "John", 1., "hello"],
        ["Kate", "Stella", 1., "hello"],
        ["Stella", "Kate", 1., "hello"],
        ["Kate", "Jack", 1., "13"],
        ["Jack", "Kate", 1., "hello"],
        ["Jess", "Jack", 1., "13"],
        ["Jack", "Jess", 1., "hello"],
        ["Jess", "Jacob", 1., "hello"],
        ["Jacob", "Jess", 1., "123"]]
)
data = BatchOperator.fromDataframe(df, schemaStr="start string, end string,
value double, attr string")
op = VertexNeighborSearchBatchOp() \
    .setDepth(1) \
    .setSources(["John", "Lisa"]) \
    .setEdgeSourceCol("start") \
    .setEdgeTargetCol("end") \
    .setVertexIdCol("name") \
    .setIsUndirected(False)
op.linkFrom(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VertexNeighborSearchBatchOpTest {
    @Test
    public void testDemo() throws Exception {
```

```
        List <Row> edgesRows = Arrays.asList(
            Row.of("Alice", "Lisa", 1., "hello"),
            Row.of("Lisa", "Alice", 1., "hello"),
            Row.of("Lisa", "Karry", 1., "hello"),
            Row.of("Karry", "Lisa", 1., "213"),
            Row.of("Karry", "Bella", 1., "hello"),
            Row.of("Bella", "Karry", 1., "h123ello"),
            Row.of("Bella", "Lucy", 1., "hello"),
            Row.of("Lucy", "Bella", 1., "hello"),
            Row.of("Lucy", "Bob", 1., "123"),
            Row.of("Bob", "Lucy", 1., "hello"),
            Row.of("John", "Bob", 1., "hello"),
            Row.of("Bob", "John", 1., "hello"),
            Row.of("John", "Stella", 1., "123"),
            Row.of("Stella", "John", 1., "hello"),
            Row.of("Kate", "Stella", 1., "hello"),
            Row.of("Stella", "Kate", 1., "hello"),
            Row.of("Kate", "Jack", 1., "13"),
            Row.of("Jack", "Kate", 1., "hello"),
            Row.of("Jess", "Jack", 1., "13"),
            Row.of("Jack", "Jess", 1., "hello"),
            Row.of("Jess", "Jacob", 1., "hello"),
            Row.of("Jacob", "Jess", 1., "123")
        );

        MemSourceBatchOp edgesSource = new MemSourceBatchOp(edgesRows,
            "start string, end string, value double, attr string");
        VertexNeighborSearchBatchOp op = new VertexNeighborSearchBatchOp()
            .setDepth(1)
            .setSources("John", "Lisa")
            .setEdgeSourceCol("start")
            .setEdgeTargetCol("end")
            .setVertexIdCol("name")
            .setIsUndirected(false);
        op.linkFrom(edgesSource).print();
    }
}
```

运行结果

start	end	value	attr
Alice	Lisa	1.0000	hello
Lisa	Karry	1.0000	hello
John	Bob	1.0000	hello
John	Stella	1.0000	123

Lisa	Alice	1.0000	hello
Karry	Lisa	1.0000	213
Bob	John	1.0000	hello
Stella	John	1.0000	hello

二分类评估 (EvalBinaryClassBatchOp)

Java 类名：com.alibaba.alink.operator.batch.evaluation.EvalBinaryClassBatchOp

Python 类名：EvalBinaryClassBatchOp

功能介绍

对二分类算法的预测结果进行效果评估。

算法原理

在有监督二分类问题的评估中，每条样本都有一个真实的标签和一个由模型生成的预测。 这样每调样本点实际上可以划分为以下 4 个类别中的一类：

- True Positive（TP）： 标签为正类，预测为正类；
- True Negative（TN）： 标签为负类，预测为负类；
- False Positive（FP）： 标签为负类，预测为正类；
- False Negative（FN）： 标签为正类，预测为负类。

通常，用 TP, TN, FP, FN 分别表示属于各自类别的样本数。 基于这几个数量，可以定义大部分的二分类评估指标。

精确率

$Precision = \frac{TP}{TP + FP}$

召回率、敏感性

$Recall = \frac{TP}{TP + FN} = Sensitivity$

F-measure

$F1=\frac{2TP}{2TP+FP+FN}=\frac{2\cdot Precision \cdot Recall}{Precision+Recall}$

准确率

$Accuracy=\frac{TP + TN}{TP + TN + FP + FN}$

特异性

$Specificity=\frac{TN}{FP+TN}$

Kappa

$p_a =\frac{TP + TN}{TP + TN + FP + FN}$

$p_e = \frac{(TN + FP) *(TN + FN)* + *(FN + TP)* (FP + TP)}{(TP + TN + FP + FN) * (TP + TN + FP + FN)}$

$kappa = \frac{p_a - p_e}{1 - p_e}$

混淆矩阵

		Actual class	
		positive class	negative class
Predicted class	positive class	True Positive(TP)	False Positive(FP)
	negative class	False Negative(FN)	True Negative(TN)

二分类模型除了给出每条样本的预测标签之外，通常还会给出每条样本预测为正类的概率p，而预测标签是根据这个概率与阈值的关系确定的。 通常情况下，阈值会设为 0.5，概率大于 0.5 的预测为正类，小于 0.5 的预测为负类。 有一些评估指标会考虑这个阈值从 0 到 1 变化时，各个指标的变化情况，计算更加复杂的指标。

LogLoss

$LogLoss = -\frac{1}{n}\sum_{i}[y_i \log(y'_i) + (1-y_i) \log(1 - y'_i)]$

这里，$y_i\in [0,1]$表示样本i的真实标签（正类为 1，负类为0），y'_i表示样本i预测为正类的概率。

ROC（receiver operating characteristic）曲线

阈值从 0 到 1 变化时， 横坐标：$FPR = FP / (FP + TN)$ 和纵坐标：$TPR = TP / (TP + FN)$构成的曲线。

AUC (Area under curve)

ROC 曲线下的面积。

K-S 曲线

阈值从 0 到 1 变化时，横坐标阈值和纵坐标TPR和FPR构成的曲线。

KS 指标

K-S 曲线 中，两条曲线在纵轴方向上的最大差值。

Precision-Recall 曲线

阈值从 0 到 1 变化时，横坐标 Precision 和纵坐标 Recall 构成的曲线。

PRC 指标

Precision-Recall 曲线下的面积。

提升曲线（Lift Chart）

阈值从 0 到 1 变化时，横坐标$\frac{TP + FP}{N}$和纵坐标TP构成的曲线。

使用方式

该组件通常接二分类预测算法的输出端。

使用时，需要通过参数 `labelCol` 指定预测标签列，通过参数 `predictionDetailCol` 指定预测详细信息列（包含有预测概率）。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String	✓	所选列类型为[STRING]	
positiveLabelValueString	正样本	正样本对应的字符串格式。	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["prefix1", "{\"prefix1\": 0.9, \"prefix0\": 0.1}"],
    ["prefix1", "{\"prefix1\": 0.8, \"prefix0\": 0.2}"],
    ["prefix1", "{\"prefix1\": 0.7, \"prefix0\": 0.3}"],
    ["prefix0", "{\"prefix1\": 0.75, \"prefix0\": 0.25}"],
    ["prefix0", "{\"prefix1\": 0.6, \"prefix0\": 0.4}"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='label string, detailInput
```

```
string')

metrics =
EvalBinaryClassBatchOp().setLabelCol("label").setPredictionDetailCol("detailInp
ut").linkFrom(inOp).collectMetrics()
print("AUC:", metrics.getAuc())
print("KS:", metrics.getKs())
print("PRC:", metrics.getPrc())
print("Accuracy:", metrics.getAccuracy())
print("Macro Precision:", metrics.getMacroPrecision())
print("Micro Recall:", metrics.getMicroRecall())
print("Weighted Sensitivity:", metrics.getWeightedSensitivity())
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalBinaryClassBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.BinaryClassMetrics;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class EvalBinaryClassBatchOpTest {
    @Test
    public void testEvalBinaryClassBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("prefix1", "{\"prefix1\": 0.9, \"prefix0\": 0.1}"),
            Row.of("prefix1", "{\"prefix1\": 0.8, \"prefix0\": 0.2}"),
            Row.of("prefix1", "{\"prefix1\": 0.7, \"prefix0\": 0.3}"),
            Row.of("prefix0", "{\"prefix1\": 0.75, \"prefix0\": 0.25}"),
            Row.of("prefix0", "{\"prefix1\": 0.6, \"prefix0\": 0.4}")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "label string,
detailInput string");
        BinaryClassMetrics metrics = new
EvalBinaryClassBatchOp().setLabelCol("label").setPredictionDetailCol(
            "detailInput").linkFrom(inOp).collectMetrics();
        System.out.println("AUC:" + metrics.getAuc());
        System.out.println("KS:" + metrics.getKs());
        System.out.println("PRC:" + metrics.getPrc());
        System.out.println("Accuracy:" + metrics.getAccuracy());
        System.out.println("Macro Precision:" + metrics.getMacroPrecision());
        System.out.println("Micro Recall:" + metrics.getMicroRecall());
```

```
        System.out.println("Weighted Sensitivity:" +
metrics.getWeightedSensitivity());
    }
}
```

运行结果

```
AUC: 0.8333333333333334
KS: 0.6666666666666666
PRC: 0.9027777777777777
Accuracy: 0.6
Macro Precision: 0.8
Micro Recall: 0.6
Weighted Sensitivity: 0.6
```

聚类评估 (EvalClusterBatchOp)

Java 类名：com.alibaba.alink.operator.batch.evaluation.EvalClusterBatchOp

Python 类名：EvalClusterBatchOp

功能介绍

对聚类算法的预测结果进行效果评估。

算法原理

在下面的指标中，用 C_i 表示第 i 个簇， u_i 表示C_i 的中心，k 表示簇的总数。

Compactness(CP)

$\overline{CP*i}=*dfrac{1}{|C_i|}*sum*{x \in C_i}|x_i-u_i|$

$\overline{CP}=\dfrac{1}{k}\sum_{i=1}^{k}\overline{CP_k}$

CP越低意味着类内聚类距离越近。

Separation (SP)

$SP=\dfrac{2}{k^2-k}\sum*{i=1}^{k}*sum*{j=i+1}^{k}|u_i-u_j|$

SP越高意味类间聚类距离越远。

Davies-Bouldin Index (DB)

$DB=\dfrac{1}{k}\sum*{i=1}^{k}*max*{i \not=j}(\dfrac{\overline{CP_i}+\overline{CP_j}}{|u_i-u_j|})$

DB越小意味着类内距离越小，同时类间距离越大。

Calinski-Harabasz Index (Variance Ratio Criterion, VRC)

用 u 表示数据集整体的中心点，

$SSB=\sum_{i=1}^{k}n_i|u_i-u|^2,$

$SSW=\sum*{i=1}^{k}*sum*{x \in C_i}|x_i-u_i|,$

$\mathrm{VRC}=\dfrac{SSB}{SSW}*\dfrac{N-k}{k-1}$

VRC越大意味着聚类质量越好。

另有一部分聚类评价指标称作外部指标（external criterion）。 这些指标在评估时除了有每个样本点所属的簇之外，还假设每个样本点有类别标签。

在下面的指标中，N表示簇的总数，用 ω_k 表示簇k所包含的样本点集合，c_j 表示类别标签为 j 的样本点集合。

Purity

$Purity(\Omega, C)=\dfrac{1}{N}\sum_{k}\max_{j}|\omega_k \cap c_j|$

取值在 $[0,1]$ 区间内,越接近1表示同一个簇内相同类别的数据点越多，聚类结果越好。

Normalized Mutual Information (NMI), NMI在[0,1]区间内, 越接近1表示聚类结果越好

$H(\Omega)=-\sum_{k}\dfrac{\omega_k}{N}log\dfrac{\omega_k}{N},$

$H(C)=-\sum_{j}\dfrac{c_j}{N}log\dfrac{c_j}{N},$

$I(\Omega, C)=\sum_k\sum_j\dfrac{|\omega_k \cap c_j|}{N}\log \dfrac{N|\omega_k \cap c_j|}{|\omega_k||c_j|},$

$\mathrm{NMI}=\dfrac{2 * I(\Omega, C)}{H(\Omega) + H(C)}$

取值在 $[0,1]$ 区间内, 越接近1表示聚类结果越好。

Rand Index (RI)

对于任意一对样本点：

- 如果标签相同并且属于相同的簇，则认为是 TP (True Positive)；
- 如果标签不同并且属于不同的簇，则认为是 TN (True Negative)；
- 如果标签相同并且属于不同的簇，则认为是 FN (False Negative)；
- 如果标签不同并且属于相同的簇，则认为是 FP (False Positive)。

用 TP, TN, FN, FP 分别表示属于各自类别的样本点对的个数，$N(k,j)$ 表示簇 k 内类别为 j 的样本点个数，那么有：

$TP+FP=\sum_{j}\binom{c_j}{2},$

$TP+FN=\sum_{k}\binom{\omega_k}{2},$

$TP=\sum_{k}\sum_{j}\binom{N(k,j)}{2},$

$TP+TN+FP+FN= \binom{N}{2},$

$RI=\dfrac{TP+TN}{TP+TN+FP+FN}$

取值在 $[0,1]$ 区间内，越接近1表示聚类结果越好。

Adjusted Rand Index(ARI), ARI在[-1,1]区间内,越接近1表示聚类结果越好

$\mathrm{Index}=TP,$

$\mathrm{ExpectedIndex}=\dfrac{(TP+FP)(TP+FN)}{TP+TN+FP+FN},$

$\mathrm{MaxIndex}=\dfrac{TP+FP+TP+FN}{2},$

$ARI=\dfrac{\mathrm{Index} - \mathrm{ExpectedIndex}}{\mathrm{MaxIndex} - \mathrm{ExpectedIndex}}$ E 取值在 $[-1,1]$ 区间内，越接近1表示聚类结果越好。

使用方式

该组件通常接聚类算法的输出端。

使用时，需要通过 `predictionCol` 指定预测结果类。 通常还需要通过 `vectorCol` 指定样本点的坐标，这样才能计算评估指标。否则，只能输出样本点所属簇等基本信息。 另外，可以根据需要指定标签列 `labelCol`，这样可以计算外部指标。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
distanceType	距离度量方式	距离类型	String		"EUCLIDEAN", "COSINE", "CITYBLOCK"	"EUCLIDEAN"
labelCol	标签列名	输入表中的标签列名	String			null
vectorCol	向量列名	输入表中的向量列名	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    [0, "0 0 0"],
    [0, "0.1,0.1,0.1"],
    [0, "0.2,0.2,0.2"],
    [1, "9 9 9"],
    [1, "9.1 9.1 9.1"],
    [1, "9.2 9.2 9.2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')

metrics =
EvalClusterBatchOp().setVectorCol("vec").setPredictionCol("id").linkFrom(inOp).
collectMetrics()

print("Total Samples Number:", metrics.getCount())
print("Cluster Number:", metrics.getK())
print("Cluster Array:", metrics.getClusterArray())
print("Cluster Count Array:", metrics.getCountArray())
print("CP:", metrics.getCp())
print("DB:", metrics.getDb())
print("SP:", metrics.getSp())
print("SSB:", metrics.getSsb())
print("SSW:", metrics.getSsw())
print("CH:", metrics.getVrc())
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalClusterBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.ClusterMetrics;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class EvalClusterBatchOpTest {
    @Test
    public void testEvalClusterBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(0, "0.1,0.1,0.1"),
            Row.of(0, "0.2,0.2,0.2"),
            Row.of(1, "9 9 9"),
```

```
            Row.of(1, "9.1 9.1 9.1"),
            Row.of(1, "9.2 9.2 9.2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, vec
string");
        ClusterMetrics metrics = new
EvalClusterBatchOp().setVectorCol("vec").setPredictionCol("id").linkFrom(inOp)
            .collectMetrics();
        System.out.println("Total Samples Number:" + metrics.getCount());
        System.out.println("Cluster Number:" + metrics.getK());
        System.out.println("Cluster Array:" +
Arrays.toString(metrics.getClusterArray()));
        System.out.println("Cluster Count Array:" +
Arrays.toString(metrics.getCountArray()));
        System.out.println("CP:" + metrics.getCp());
        System.out.println("DB:" + metrics.getDb());
        System.out.println("SP:" + metrics.getSp());
        System.out.println("SSB:" + metrics.getSsb());
        System.out.println("SSW:" + metrics.getSsw());
        System.out.println("CH:" + metrics.getVrc());
    }
}
```

运行结果

```
Total Samples Number: 6
Cluster Number: 2
Cluster Array: ['0', '1']
Cluster Count Array: [3.0, 3.0]
CP: 0.11547005383792497
DB: 0.014814814814814791
SP: 15.588457268119896
SSB: 364.5
SSW: 0.1199999999999996
CH: 12150.000000000042
```

多分类评估 (EvalMultiClassBatchOp)

Java 类名：com.alibaba.alink.operator.batch.evaluation.EvalMultiClassBatchOp

Python 类名：EvalMultiClassBatchOp

功能介绍

对多分类算法的预测结果进行效果评估。

算法原理

在多分类问题的评估中，每条样本都有一个真实的标签和一个由模型生成的预测。 但与二分类问题不同，多分类算法中，总的类别数是大于2的，因此不能直接称作正类和负类。

在计算评估指标时，可以将某个类别选定为正类，将其他值都看作负类，这样可以计算每个类别（per-class）的指标。 进一步地，将每个类别各自的指标进行平均，可以得到模型总体的指标。 这里的"平均"有三种做法：

- Macro 平均：直接对各个类别的同一个指标求数值平均值，作为总体指标；
- 加权平均：以样本中各个类别所占的比例为权重，对各个类别的同一个指标求加权平均值，作为总体指标；
- Micro 平均：将各个类别看作正类时的 TP, TN, FN 相加，得到总的 TP, TN, FN 值，然后计算指标。在 Micro 平均时，micro-F1, micro-precision, micro-recall 都等于 accuracy。

所支持的每类别指标与平均指标见下：

精确率

$Precision = \frac{TP}{TP + FP}$

召回率、敏感性

$Recall = \frac{TP}{TP + FN} = Sensitivity$

F-measure

$F1=\frac{2TP}{2TP+FP+FN}=\frac{2\cdot Precision \cdot Recall}{Precision+Recall}$

准确率

$Accuracy=\frac{TP + TN}{TP + TN + FP + FN}$

特异性

$Specificity=\frac{TN}{FP+TN}$

Kappa

$p_a =\frac{TP + TN}{TP + TN + FP + FN}$

$p_e = \frac{(TN + FP) *(TN + FN)* + *(FN + TP)* (FP + TP)}{(TP + TN + FP + FN) * (TP + TN + FP + FN)}$

$kappa = \frac{p_a - p_e}{1 - p_e}$

混淆矩阵

		Actual class	
		positive class	negative class
Predicted class	positive class	True Positive(TP)	False Positive(FP)
	negative class	False Negative(FN)	True Negative(TN)

二分类模型除了给出每条样本i的预测标签之外，通常还会给出每条样本预测为为各个类别j的概率$p_{i,j}$。通常情况下，每条样本最大概率对应的类别为该样本的预测标签。

LogLoss

$LogLoss=- \frac{1}{n}\sum*{i} \sum*{j=1}^M y*{i,j}log(p*{i,j})$

使用方式

该组件通常接多分类预测算法的输出端。

使用时，需要通过参数 `labelCol` 指定预测标签列，通过参数 `predictionCol` 和 `predictionDetailCol` 指定预测结果列和预测详细信息列（包含有预测概率）。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		所选列类型为[STRING]	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["prefix1", "{\"prefix1\": 0.9, \"prefix0\": 0.1}"],
    ["prefix1", "{\"prefix1\": 0.8, \"prefix0\": 0.2}"],
    ["prefix1", "{\"prefix1\": 0.7, \"prefix0\": 0.3}"],
    ["prefix0", "{\"prefix1\": 0.75, \"prefix0\": 0.25}"],
    ["prefix0", "{\"prefix1\": 0.6, \"prefix0\": 0.4}"]])

inOp = BatchOperator.fromDataframe(df, schemaStr='label string, detailInput
string')

metrics =
EvalMultiClassBatchOp().setLabelCol("label").setPredictionDetailCol("detailInpu
t").linkFrom(inOp).collectMetrics()
print("Prefix0 accuracy:", metrics.getAccuracy("prefix0"))
print("Prefix1 recall:", metrics.getRecall("prefix1"))
print("Macro Precision:", metrics.getMacroPrecision())
print("Micro Recall:", metrics.getMicroRecall())
print("Weighted Sensitivity:", metrics.getWeightedSensitivity())
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalMultiClassBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.MultiClassMetrics;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class EvalMultiClassBatchOpTest {
    @Test
    public void testEvalMultiClassBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("prefix1", "{\"prefix1\": 0.9, \"prefix0\": 0.1}"),
```

```
            Row.of("prefix1", "{\"prefix1\": 0.8, \"prefix0\": 0.2}"),
            Row.of("prefix1", "{\"prefix1\": 0.7, \"prefix0\": 0.3}"),
            Row.of("prefix0", "{\"prefix1\": 0.75, \"prefix0\": 0.25}")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "label string,
detailInput string");
        MultiClassMetrics metrics = new
EvalMultiClassBatchOp().setLabelCol("label").setPredictionDetailCol(
            "detailInput").linkFrom(inOp).collectMetrics();
        System.out.println("Prefix0 accuracy:" +
metrics.getAccuracy("prefix0"));
        System.out.println("Prefix1 recall:" + metrics.getRecall("prefix1"));
        System.out.println("Macro Precision:" + metrics.getMacroPrecision());
        System.out.println("Micro Recall:" + metrics.getMicroRecall());
        System.out.println("Weighted Sensitivity:" +
metrics.getWeightedSensitivity());
    }
}
```

运行结果

```
Prefix0 accuracy: 0.6
Prefix1 recall: 1.0
Macro Precision: 0.8
Micro Recall: 0.6
Weighted Sensitivity: 0.6
```

多标签分类评估 (EvalMultiLabelBatchOp)

Java 类名：com.alibaba.alink.operator.batch.evaluation.EvalMultiLabelBatchOp

Python 类名：EvalMultiLabelBatchOp

功能介绍

对多标签分类算法的预测结果进行效果评估。

算法原理

在多标签分类问题中，每个样本点 i 所属标签集合记为 L_i，模型预测给出的预测集合记为 P_i；样本点总数记为 N。

Precision

$\frac{1}{N} \sum_{i=0}^{N-1} \frac{\left|P_i \cap L_i\right|}{\left|P_i\right|}$

Recall

$\frac{1}{N} \sum_{i=0}^{N-1} \frac{\left|L_i \cap P_i\right|}{\left|L_i\right|}$

Accuracy

$\frac{1}{N} \sum_{i=0}^{N - 1} \frac{\left|L_i \cap P_i \right|}{\left|L_i\right| + \left|P_i\right| - \left|L_i \cap P_i \right|}$

Hamming Loss, HL

$\frac{1}{N \cdot \left|L\right|} \sum_{i=0}^{N - 1} \left|L_i\right| + \left|P_i\right| - 2\left|L_i \cap P_i\right|$

Subset Accuracy, SA

$\frac{1}{N}\sum_{i=0}^{N-1}I[L_i =P_i]$

这里 $I[\cdot]$ 是指示函数，内部条件满足时值为1，其他时候为0。

F1 Measure

$\frac{1}{N} \sum_{i=0}^{N-1} 2 \frac{\left|P_i \cap L_i\right|}{\left|P_i\right| \cdot \left|L_i\right|}$

Micro Precision

$\frac{TP}{TP + FP}=\frac{\sum*{i=0}^{N-1} \left|P_i \cap L_i\right|}{\sum*{i=0}^{N-1} \left|P*i \cap L_i\right| + \sum*{i=0}^{N-1} \left|P_i - L_i\right|}$

Micro Recall

$\frac{TP}{TP + FN}=\frac{\sum*{i=0}^{N-1} \left|P_i \cap L_i\right|}{\sum*{i=0}^{N-1} \left|P*i \cap L_i\right| + \sum*{i=0}^{N-1} \left|L_i - P_i\right|}$

Micro F1

$2 \cdot \frac{TP}{2 \cdot TP + FP + FN}=2 \cdot \frac{\sum*{i=0}^{N-1} \left|P_i \cap L_i\right|}{2 \cdot \sum*{i=0}^{N-1} \left|P*i \cap L_i\right| + \sum*{i=0}^{N-1} \left|L*i - P_i\right| + \sum*{i=0}^{N-1} \left|P_i - L_i\right|}$

使用方式

该组件通常接多标签分类预测算法的输出端。

使用时，需要通过参数 `labelCol` 指定预测标签列，参数 `predictionCol` 和 `predictionCol` 指定预测结果列。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
labelRankingInfo	Object列列名	Object列列名	String			"object"
predictionRankingInfo	Object列列名	Object列列名	String			"object"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["{\"object\":\"[0.0, 1.0]\"}", "{\"object\":\"[0.0, 2.0]\"}"],
    ["{\"object\":\"[0.0, 2.0]\"}", "{\"object\":\"[0.0, 1.0]\"}"],
    ["{\"object\":\"[]\"}", "{\"object\":\"[0.0]\"}"],
    ["{\"object\":\"[2.0]\"}", "{\"object\":\"[2.0]\"}"],
    ["{\"object\":\"[2.0, 0.0]\"}", "{\"object\":\"[2.0, 0.0]\"}"],
    ["{\"object\":\"[0.0, 1.0, 2.0]\"}", "{\"object\":\"[0.0, 1.0]\"}"],
```

```
    ["{\"object\":\"[1.0]\"}", "{\"object\":\"[1.0, 2.0]\"}"]
])

source = BatchOperator.fromDataframe(df, "pred string, label string")

evalMultiLabelBatchOp: EvalMultiLabelBatchOp =
EvalMultiLabelBatchOp().setLabelCol("label").setPredictionCol("pred").linkFrom(
source)
metrics = evalMultiLabelBatchOp.collectMetrics()
print(metrics)
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalMultiLabelBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.MultiLabelMetrics;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class EvalMultiLabelBatchOpTest {
    @Test
    public void testEvalMultiLabelBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("{\"object\":\"[0.0, 1.0]\"}", "{\"object\":\"[0.0,
2.0]\"}"),
            Row.of("{\"object\":\"[0.0, 2.0]\"}", "{\"object\":\"[0.0,
1.0]\"}"),
            Row.of("{\"object\":\"[]\"}", "{\"object\":\"[0.0]\"}"),
            Row.of("{\"object\":\"[2.0]\"}", "{\"object\":\"[2.0]\"}"),
            Row.of("{\"object\":\"[2.0, 0.0]\"}", "{\"object\":\"[2.0,
0.0]\"}"),
            Row.of("{\"object\":\"[0.0, 1.0, 2.0]\"}", "{\"object\":\"[0.0,
1.0]\"}"),
            Row.of("{\"object\":\"[1.0]\"}", "{\"object\":\"[1.0, 2.0]\"}")
        );
        BatchOperator <?> source = new MemSourceBatchOp(df, "pred string, label
string");
        EvalMultiLabelBatchOp evalMultiLabelBatchOp =
            new EvalMultiLabelBatchOp().setLabelCol("label").setPredictionCol(
            "pred").linkFrom(source);
        MultiLabelMetrics metrics = evalMultiLabelBatchOp.collectMetrics();
        System.out.println(metrics.toString());
```

```
    }
}
```

运行结果

```
-------------------------------- Metrics: --------------------------------
microPrecision:0.7273
microF1:0.6957
subsetAccuracy:0.2857
precision:0.6667
recall:0.6429
accuracy:0.5476
f1:0.6381
microRecall:0.6667
hammingLoss:0.3333
```

排序评估 (EvalRankingBatchOp)

Java 类名：com.alibaba.alink.operator.batch.evaluation.EvalRankingBatchOp

Python 类名：EvalRankingBatchOp

功能介绍

对推荐排序算法的预测结果进行效果评估。

算法原理

在排序问题中，可以忽略顺序，将问题看作多标签分类问题。这样多标签分类评估中的指标都能使用。

在考虑顺序时，假设有M个用户，每个用户的真实标签集合为 $D*i* = *left{d_0, d_1, ..., d*{N-1}\right}$，模型推荐的集合为 $R*i* = *left[r_0, r_1, ..., r*{Q-1}\right]$，按相关程序递减排序。 定义$rel_D(r)$在满足$r\in D$时为1，否则为0。 那么，还可以支持以下评估指标。

Hit Rate

$HitRate=\frac{1}{M}\sum*{i=0}^{M-1} \sum*{j=0}^{\left|D\right| - 1}rel_{{d_0}}(R_i(j))$

Average Reciprocal Hit Rank

$ARHR=\frac{1}{M}\sum*{i=0}^{M-1} \sum*{j=0}^{\left|D\right| - 1}\frac{1}{j+1}rel_{{d_0}}(R_i(j))$

Precision at k

$p(k)=\frac{1}{M} \sum*{i=0}^{M-1} {\frac{1}{k} \sum*{j=0}^{\min(\left|D\right|, k) - 1} rel_{D_i}(R_i(j))}$

这里 k 是一个参数。

Recall at k

$recall(k)=\frac{1}{M} \sum*{i=0}^{M-1} {\frac{1}{N} \sum*{j=0}^{\min(\left|D\right|, k) - 1} rel_{D_i}(R_i(j))}$

这里 k 是一个参数。

MAP (Mean Average Precision)

$MAP=\frac{1}{M} \sum*{i=0}^{M-1} {\frac{1}{\left|D_i\right|} \sum*{j=0}^{Q-1} \frac{rel_{D_i}(R_i(j))}{j + 1}}$

NDCG at k (Normalized Discounted Cumulative Gain)

$NDCG(k)=\frac{1}{M} \sum*{i=0}^{M-1} {\frac{1}{IDCG(D_i, k)}\sum*{j=0}^{n-1} \frac{rel_{D_i}(R_i(j))}{\ln(j+2)}},$

其中，$n = \min\left(\max\left(|R*i*|,|*D_i*|\right),k\right)$， $*IDCG(D, k)* = *sum*{j=0}^{\min(\left|D\right|, k) - 1} \frac{1}{\ln(j+2)}$。

这里 k 是一个参数。

使用方式

该组件通常接推荐排序预测算法的输出端。

使用时，需要通过参数 `labelCol` 指定预测标签列，参数 `predictionCol` 和 `predictionCol` 指定预测结果列。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
labelRankingInfo	Object列列名	Object列列名	String			"object"
predictionRankingInfo	Object列列名	Object列列名	String			"object"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["{\"object\":\"[1, 6, 2, 7, 8, 3, 9, 10, 4, 5]\"}", "{\"object\":\"[1, 2, 3, 4, 5]\"}"],
    ["{\"object\":\"[4, 1, 5, 6, 2, 7, 3, 8, 9, 10]\"}", "{\"object\":\"[1, 2, 3]\"}"],
    ["{\"object\":\"[1, 2, 3, 4, 5]\"}", "{\"object\":\"[]\"}"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='pred string, label string')

metrics = EvalRankingBatchOp().setPredictionCol('pred').setLabelCol('label').linkFrom(inOp).collectMetrics()
print(metrics)
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalRankingBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.RankingMetrics;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class EvalRankingBatchOpTest {
    @Test
    public void testEvalRankingBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("{\"object\":\"[1, 6, 2, 7, 8, 3, 9, 10, 4, 5]\"}", "
{\"object\":\"[1, 2, 3, 4, 5]\"}"),
            Row.of("{\"object\":\"[4, 1, 5, 6, 2, 7, 3, 8, 9, 10]\"}", "
{\"object\":\"[1, 2, 3]\"}"),
            Row.of("{\"object\":\"[1, 2, 3, 4, 5]\"}", "{\"object\":\"[]\"}")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "pred string, label
string");
        RankingMetrics metrics = new
EvalRankingBatchOp().setPredictionCol("pred").setLabelCol("label").linkFrom(inO
p)
            .collectMetrics();
        System.out.println(metrics.toString());
    }
}
```

运行结果

```
-------------------------------- Metrics: --------------------------------
microPrecision:0.32
averageReciprocalHitRank:0.5
precision:0.2667
accuracy:0.2667
f1:0.3761
hitRate:0.6667
microRecall:1
microF1:0.4848
subsetAccuracy:0
recall:0.6667
```

```
map:0.355
hammingLoss:0.5667
```

回归评估 (EvalRegressionBatchOp)

Java 类名：com.alibaba.alink.operator.batch.evaluation.EvalRegressionBatchOp

Python 类名：EvalRegressionBatchOp

功能介绍

对回归算法的预测结果进行效果评估，支持下列评估指标。

算法原理

对于每条样本i，y_i表示样本的真实标签，f_i表示回归模型预测的值。样本总数记为N。 支持计算下面的评估指标：

SST 总平方和(Sum of Squared for Total)

$SST=\sum_{i=1}^{N}(y_i-\bar{y})^2$

SSE 误差平方和(Sum of Squares for Error)

$SSE=\sum_{i=1}^{N}(y_i-f_i)^2$

SSR 回归平方和(Sum of Squares for Regression)

$SSR=\sum_{i=1}^{N}(f_i-\bar{y})^2$

R^2 判定系数(Coefficient of Determination)

$R^2=1-\dfrac{SSE}{SST}$

R 多重相关系数(Multiple Correlation Coeffient)

$R=\sqrt{R^2}$

MSE 均方误差(Mean Squared Error)

$MSE=\dfrac{1}{N}\sum_{i=1}^{N}(f_i-y_i)^2$

RMSE 均方根误差(Root Mean Squared Error)

$RMSE=\sqrt{MSE}$

SAE/SAD 绝对误差(Sum of Absolute Error/Difference)

$SAE=\sum_{i=1}^{N}|f_i-y_i|$

MAE/MAD 平均绝对误差(Mean Absolute Error/Difference)

$MAE=\dfrac{1}{N}\sum_{i=1}^{N}|f_i-y_i|$

MAPE 平均绝对百分误差(Mean Absolute Percentage Error)

$MAPE=\dfrac{100}{N}\sum_{i=1}^{N}|\dfrac{f_i-y_i}{y_i}|$

explained variance 解释方差

$\mathrm{ExplainedVariance}=\dfrac{SSR}{N}$

使用方式

该组件通常接回归预测算法的输出端。

使用时，需要通过参数 `labelCol` 指定预测标签列，通过参数 `predictionCol` 指定预测结果列。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, 0],
    [8, 8],
    [1, 2],
    [9, 10],
    [3, 1],
    [10, 7]
])
```

```
inOp = BatchOperator.fromDataframe(df, schemaStr='pred int, label int')

metrics =
EvalRegressionBatchOp().setPredictionCol("pred").setLabelCol("label").linkFrom(
inOp).collectMetrics()

print("Total Samples Number:", metrics.getCount())
print("SSE:", metrics.getSse())
print("SAE:", metrics.getSae())
print("RMSE:", metrics.getRmse())
print("R2:", metrics.getR2())
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalRegressionBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.RegressionMetrics;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class EvalRegressionBatchOpTest {
    @Test
    public void testEvalRegressionBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, 0),
            Row.of(8, 8),
            Row.of(1, 2),
            Row.of(9, 10),
            Row.of(3, 1),
            Row.of(10, 7)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "pred int, label
int");
        RegressionMetrics metrics = new
EvalRegressionBatchOp().setPredictionCol("pred").setLabelCol("label").linkFrom(
            inOp).collectMetrics();
        System.out.println("Total Samples Number:" + metrics.getCount());
        System.out.println("SSE:" + metrics.getSse());
        System.out.println("SAE:" + metrics.getSae());
        System.out.println("RMSE:" + metrics.getRmse());
        System.out.println("R2:" + metrics.getR2());
```

```
    }
}
```

运行结果

```
Total Samples Number: 6.0
SSE: 15.0
SAE: 7.0
RMSE: 1.5811388300841898
R2: 0.8282442748091603
```

时间序列评估 (EvalTimeSeriesBatchOp)

Java 类名：com.alibaba.alink.operator.batch.evaluation.EvalTimeSeriesBatchOp

Python 类名：EvalTimeSeriesBatchOp

功能介绍

对时间序列结果进行评估。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
labelCol	标签列名	输入表中的标签列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
            [1, datetime.datetime.fromtimestamp(1), 10.0, 10.5],
            [1, datetime.datetime.fromtimestamp(2), 11.0, 10.5],
            [1, datetime.datetime.fromtimestamp(3), 12.0, 11.5],
            [1, datetime.datetime.fromtimestamp(4), 13.0, 12.5],
            [1, datetime.datetime.fromtimestamp(5), 14.0, 13.5],
            [1, datetime.datetime.fromtimestamp(6), 15.0, 14.5],
            [1, datetime.datetime.fromtimestamp(7), 16.0, 14.5],
            [1, datetime.datetime.fromtimestamp(8), 17.0, 14.5],
```

```
            [1, datetime.datetime.fromtimestamp(9), 18.0, 14.5],
            [1, datetime.datetime.fromtimestamp(10), 19.0, 16.5]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val double,
pred double', op_type='batch')

cmex = source.link(
    EvalTimeSeriesBatchOp()\
        .setLabelCol("val")\
        .setPredictionCol("pred")
).collectMetrics()

print(cmex.getMse())
print(cmex.getMae())
print(cmex.getRmse())
print(cmex.getSse())
print(cmex.getSst())
print(cmex.getSsr())
print(cmex.getSae())
print(cmex.getMape())
print(cmex.getSmape())
print(cmex.getND())
print(cmex.getCount())
print(cmex.getYMean())
print(cmex.getPredictionMean())
```

Java 代码

```
package com.alibaba.alink.operator.batch.evaluation;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;


public class EvalTimeSeriesBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
```

```
        List <Row> mTableData = Arrays.asList(
            Row.of(1, new Timestamp(1), 10.0, 10.5),
            Row.of(1, new Timestamp(2), 11.0, 10.5),
            Row.of(1, new Timestamp(3), 12.0, 11.5),
            Row.of(1, new Timestamp(4), 13.0, 12.5),
            Row.of(1, new Timestamp(5), 14.0, 13.5),
            Row.of(1, new Timestamp(6), 15.0, 14.5),
            Row.of(1, new Timestamp(7), 16.0, 14.5),
            Row.of(1, new Timestamp(8), 17.0, 14.5),
            Row.of(1, new Timestamp(9), 18.0, 14.5),
            Row.of(1, new Timestamp(10), 19.0, 16.5)
        );

        MemSourceBatchOp source = new MemSourceBatchOp(mTableData, new String[]
{"id", "ts", "val", "pred"});

        source.link(
            new EvalTimeSeriesBatchOp()
                .setLabelCol("val")
                .setPredictionCol("pred")
        ).lazyPrintMetrics();

        BatchOperator.execute();
    }

}
```

运行结果

{"predictionMean":"13.3","SSE":"28.5","count":"10.0","SMAPE":"8.606434351991227","MAPE":"8.11462584972
Variance":"5.0","MSE":"2.85"}

大规模DeepWalk (HugeDeepWalkTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.huge.HugeDeepWalkTrainBatchOp

Python 类名：HugeDeepWalkTrainBatchOp

功能介绍

DeepWalk是2014年提出的一个新的方法，用来为网络中的结点学习隐式特征表达，即将网络中的每一个点表示成连续特征空间中的一个点向量。DeepWalk是无监督特征学习方法，利用随机游走(Random Walk)及语言模型(Language modeling)，学习出的隐式特征能够捕捉到网络的结构信息。后续论文也提出了一些扩展，如结合损失函数进行有监督学习、结合文本信息等等

DeepWalk: Online Learning of Social Representations

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	起始点列名	用来指定起始点列	String	✓		
targetCol	中止点点列名	用来指定中止点列	String	✓		
walkLength	游走的长度	随机游走完向量的长度	Integer	✓		
walkNum	路径数目	每一个起始点游走出多少条路径	Integer	✓		
alpha	学习率	学习率	Double			0.025
batchSize	batch大小	batch大小，按行计算	Integer		[1, +inf)	
isToUndigraph	是否转无向图	选为true时，会将当前图转成无向图，然后再游走	Boolean			false
minCount	最小词频	最小词频	Integer			5
negative	负采样大小	负采样大小	Integer			5

numCheckpoint	checkPoint数目	checkPoint数目	Integer			1
numIter	迭代次数	迭代次数，默认为1。	Integer			1
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"true"
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	100
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
window	窗口大小	窗口大小	Integer			5

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ["Bob", "Lucy", 1.],
    ["Lucy", "Bob", 1.],
    ["Lucy", "Bella", 1.],
    ["Bella", "Lucy", 1.],
    ["Alice", "Lisa", 1.],
    ["Lisa", "Alice", 1.],
    ["Lisa", "Karry", 1.],
    ["Karry", "Lisa", 1.],
    ["Karry", "Bella", 1.],
    ["Bella", "Karry", 1.]
])
source =  BatchOperator.fromDataframe(df_data, schemaStr='start string, end
string, value double')
```

```
deepWalkBatchOp = HugeDeepWalkTrainBatchOp() \
  .setSourceCol("start")            \
  .setTargetCol("end")              \
  .setWeightCol("value")            \
  .setWalkNum(2)                    \
  .setWalkLength(2)                 \
  .setMinCount(1)                   \
  .setVectorSize(4)
deepWalkBatchOp.linkFrom(source).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.huge.HugeDeepWalkTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HugeDeepWalkTrainBatchOpTest {
    @Test
    public void testHugeDeepWalkTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("Bob", "Lucy", 1.),
            Row.of("Lucy", "Bob", 1.),
            Row.of("Lucy", "Bella", 1.),
            Row.of("Bella", "Lucy", 1.),
            Row.of("Alice", "Lisa", 1.),
            Row.of("Lisa", "Alice", 1.),
            Row.of("Lisa", "Karry", 1.),
            Row.of("Karry", "Lisa", 1.),
            Row.of("Karry", "Bella", 1.),
            Row.of("Bella", "Karry", 1.)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df_data, "start string,
end string, value double");
        BatchOperator <?> deepWalkBatchOp = new HugeDeepWalkTrainBatchOp()
            .setSourceCol("start")
            .setTargetCol("end")
            .setWeightCol("value")
            .setWalkNum(2)
            .setWalkLength(2)
            .setMinCount(1)
            .setVectorSize(4);
```

```
        deepWalkBatchOp.linkFrom(source).print();
    }
}
```

运行结果

node	vec
Karry	0.03438692167401314,-0.04779096320271492,0.012648836709558964,-0.09576538950204849
Lisa	0.11595723778009415,-0.08507091552019119,0.1099027618765831,0.013517010025680065
Bella	0.05783883109688759,0.08286115527153015,-0.06497485190629959,0.026532595977187157
Alice	0.05775630846619606,-0.099935382604599,-0.022451162338256836,-0.023144230246543884
Lucy	0.11699658632278442,0.05271214246749878,-0.12347490340471268,-0.08684996515512466
Bob	-0.07306862622499466,-0.11596906185150146,-0.04183155298233032,0.03973118215799332

Huge FM 预测 (HugeFmPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.huge.HugeFmPredictBatchOp

Python 类名：HugeFmPredictBatchOp

功能介绍

Factorization Machine(FM) 预测。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
  [0, "0:5.1,1:3.5,2:1.4,3:0.2"],
  [1, "0:4.9,1:3.0,2:1.4,3:0.2"],
  [0, "0:4.7,1:3.2,2:1.3,3:0.2"],
  [0, "0:4.6,1:3.1,2:1.5,3:0.2"],
  [1, "0:5.0,1:3.6,2:1.4,3:0.2"],
  [0, "0:5.4,1:3.9,2:1.7,3:0.4"],
  [0, "0:4.6,1:3.4,2:1.4,3:0.3"],
  [1, "0:5.0,1:3.4,2:1.5,3:0.2"],
  [0, "0:4.4,1:2.9,2:1.4,3:0.2"],
```

```
  [0, "0:4.9,1:3.1,2:1.5,3:0.1"],
])

data =  BatchOperator.fromDataframe(df_data, schemaStr='label bigint, features
string')
trainer = HugeFmTrainBatchOp().setVectorCol("features").setLabelCol("label")

model = trainer.linkFrom(data)

predictor = HugeFmPredictBatchOp().setPredictionCol("prediction_result")\
  .setPredictionDetailCol("prediction_detail").setReservedCols(["label"])
output = predictor.linkFrom(model, data)
output.print()
```

运行结果

```
   label  prediction_result            prediction_detail
0      0                0.0  {"0":0.716016,"1":0.283984}
1      1                0.0  {"0":0.701164,"1":0.298836}
2      0                0.0  {"0":0.698423,"1":0.301577}
3      0                0.0  {"0":0.696045,"1":0.303955}
4      1                0.0  {"0":0.715208,"1":0.284792}
5      0                0.0  {"0":0.732697,"1":0.267303}
6      0                0.0  {"0":0.699416,"1":0.300584}
7      1                0.0  {"0":0.712598,"1":0.287402}
8      0                0.0  {"0":0.685639,"1":0.314361}
9      0                0.0  {"0":0.705375,"1":0.294625}
```

Huge FM 训练 (HugeFmTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.huge.HugeFmTrainBatchOp

Python 类名：HugeFmTrainBatchOp

功能介绍

Factorization Machine(FM) 模型训练。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]
blockSize	数据分块大小	数据分块大小。表示每次迭代每个worker处理的数据条数。默认系统自动决定。	Integer		
dim	维数	维数。逗号分隔的三个整数，分别表示0次项、1次项、2次项的长度。	String		
initStdev	初始化参数的标准差	初始化参数的标准差	Double		
lambda	正则化系数	正则化系数。逗号分隔的三个浮点数，分别表示0次项、1次项、2次项的正则化系数。	String		
learnRate	学习率	学习率	Double		
numCheckpoints	checkpoint数目	checkpoint数目	Integer		

numEpochs	epoch数	epoch数	Integer		
task	模型类型	模型类型。回归模型或二分类模型。"regression" or "binary_classification"	String		"REGRESSION", "BINARY_CLASSIFICA

关于输入数据：

- 输入特征按kv格式存储，如"1:1.0,3:1.0,5:0.5"
- Label列要求是数值型，如果是训练二分类模型，label值必须是0或1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
  [0, "0:5.1,1:3.5,2:1.4,3:0.2"],
  [1, "0:4.9,1:3.0,2:1.4,3:0.2"],
  [0, "0:4.7,1:3.2,2:1.3,3:0.2"],
  [0, "0:4.6,1:3.1,2:1.5,3:0.2"],
  [1, "0:5.0,1:3.6,2:1.4,3:0.2"],
  [0, "0:5.4,1:3.9,2:1.7,3:0.4"],
  [0, "0:4.6,1:3.4,2:1.4,3:0.3"],
  [1, "0:5.0,1:3.4,2:1.5,3:0.2"],
  [0, "0:4.4,1:2.9,2:1.4,3:0.2"],
  [0, "0:4.9,1:3.1,2:1.5,3:0.1"],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='label bigint, features
string')
trainer = HugeFmTrainBatchOp().setVectorCol("features").setLabelCol("label")

model = trainer.linkFrom(data)
model.print()
```

运行结果

```
   model_id  feature_id                                      feature_weights
0         0         NaN  {"vectorColName":"\"features\"","labelColName"...
1         1          -1                                   [-0.024405986]
```

```
2          1          0  [0.030789409,-0.03626042,0.0875104,0.04609344,...
3          1          1  [0.013547433,-0.030760048,-0.02232726,-0.02868...
4          1          2  [-0.0060028853,0.059482295,0.0034232587,-0.039...
5          1          3  [-0.060445864,-0.017359639,0.020320095,0.04753...
```

大规模带标签的Word2Vec (HugeLabeledWord2VecTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.huge.HugeLabeledWord2VecTrainBatchOp

Python 类名：HugeLabeledWord2VecTrainBatchOp

功能介绍

Word2Vec是Google在2013年开源的一个将词表转为向量的算法，其利用神经网络，可以通过训练，将词映射到K维度空间向量，甚至对于表示词的向量进行操作还能和语义相对应，由于其简单和高效引起了很多人的关注。

Google Word2Vec的工具包相关链接：https://code.google.com/p/word2vec/

支持metapath2vec++训练：metapath2vec: Scalable Representation Learning forHeterogeneous Networks

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	计算列对应的列名	计算列对应的列名	String	✓	所选列类型为 [STRING]	
typeCol	节点类型列名	用来指定节点类型列	String	✓		
vertexCol	节点列名	用来指定节点列	String	✓	所选列类型为 [STRING]	
alpha	学习率	学习率	Double			0.025
batchSize	batch大小	batch大小, 按行计算	Integer		[1, +inf)	
minCount	最小词频	最小词频	Integer			5
negative	负采样大小	负采样大小	Integer			5
numCheckpoint	checkPoint数目	checkPoint 数目	Integer			1
numIter	迭代次数	迭代次数，默认为1。	Integer			1
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"true"

vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	100
window	窗口大小	窗口大小	Integer			5
wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

tokens = pd.DataFrame([
    ["Bob Lucy Bella"]
])

nodeType = pd.DataFrame([
    ["Bob", "A"],
    ["Bella", "A"],
    ["Karry", "A"],
    ["Lucy", "B"],
    ["Alice", "B"],
    ["Lisa", "B"]
])

source = BatchOperator.fromDataframe(tokens, schemaStr='tokens string')
typed = BatchOperator.fromDataframe(nodeType, schemaStr='node string, type
string')

labeledWord2vecBatchOp = HugeLabeledWord2VecTrainBatchOp() \
  .setSelectedCol("tokens")         \
  .setVertexCol("node")             \
  .setTypeCol("type")               \
  .setMinCount(1)                   \
  .setVectorSize(4)
labeledWord2vecBatchOp.linkFrom(source, typed).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.huge.HugeLabeledWord2VecTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HugeLabeledWord2VecTrainBatchOpTest {
    @Test
    public void testHugeLabeledWord2VecTrainBatchOp() throws Exception {
        List <Row> tokens = Arrays.asList(
            Row.of("Bob Lucy Bella")
        );
        List <Row> nodeType = Arrays.asList(
            Row.of("Bob", "A"),
            Row.of("Bella", "A"),
            Row.of("Karry", "A"),
            Row.of("Lucy", "B"),
            Row.of("Alice", "B"),
            Row.of("Lisa", "B")
        );
        BatchOperator <?> source = new MemSourceBatchOp(tokens, "tokens
string");
        BatchOperator <?> typed = new MemSourceBatchOp(nodeType, "node string,
type string");
        BatchOperator <?> labeledWord2vecBatchOp = new
HugeLabeledWord2VecTrainBatchOp()
            .setSelectedCol("tokens")
            .setVertexCol("node")
            .setTypeCol("type")
            .setMinCount(1)
            .setVectorSize(4);
        labeledWord2vecBatchOp.linkFrom(source, typed).print();
    }
}
```

运行结果

word	vec
Lucy	0.03437602147459984,-0.04761518910527229,0.012536839582026005,-0.09563367068767548
Bob	0.057709891349077225,0.08290477842092514,-0.06487766653299332,0.026675613597035408
Bella	0.02439533919095993,0.07039660215377808,-0.04170553758740425,-0.061801809817552567

大规模MethPath2Vec (HugeMetaPath2VecTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.huge.HugeMetaPath2VecTrainBatchOp

Python 类名：HugeMetaPath2VecTrainBatchOp

功能介绍

沿着之前random walk的思路往前走，metapath2vec的方法提出了控制随机游走的模式，这样就可以在生成的序列上根据节点类型的不同来控制序列游走，这样也就可以对异质网络（Heterogeneous Networks）进行表征学习。在游走之前需要设定一个metapath，也就是游走时节点类型的模式

metapath2vec: Scalable Representation Learning for Heterogeneous Networks

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
metaPath	游走的模式	一般为用字符串表示，例如"ABDFA"	String	✓		
sourceCol	起始点列名	用来指定起始点列	String	✓		
targetCol	中止点点列名	用来指定中止点列	String	✓		
typeCol	节点类型列名	用来指定节点类型列	String	✓		
vertexCol	节点列名	用来指定节点列	String	✓	所选列类型为[STRING]	
walkLength	游走的长度	随机游走完向量的长度	Integer	✓		
walkNum	路径数目	每一个起始点游走出多少条路径	Integer	✓		
alpha	学习率	学习率	Double			0

batchSize	batch大小	batch大小，按行计算	Integer		[1, +inf)	
isToUndigraph	是否转无向图	选为true时，会将当前图转成无向图，然后再游走	Boolean			f
minCount	最小词频	最小词频	Integer			5
mode	metapath中word2vec的模式，分别为metapath2vec和metapath2vecpp	metapath的模式	String		"METAPATH2VEC", "METAPATH2VECPP"	"
negative	负采样大小	负采样大小	Integer			5
numCheckpoint	checkPoint 数目	checkPoint数目	Integer			1
numIter	迭代次数	迭代次数，默认为1。	Integer			1
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	1
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	r
window	窗口大小	窗口大小	Integer			5

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df_data = pd.DataFrame([
    ["Bob", "Lucy", 1.],
    ["Lucy", "Bob", 1.],
    ["Lucy", "Bella", 1.],
    ["Bella", "Lucy", 1.],
    ["Alice", "Lisa", 1.],
    ["Lisa", "Alice", 1.],
    ["Lisa", "Karry", 1.],
    ["Karry", "Lisa", 1.],
    ["Karry", "Bella", 1.],
    ["Bella", "Karry", 1.]
])
source =  BatchOperator.fromDataframe(df_data, schemaStr='start string, end
string, value double')
nodeType = pd.DataFrame([
    ["Bob", "A"],
    ["Bella", "A"],
    ["Karry", "A"],
    ["Lucy", "B"],
    ["Alice", "B"],
    ["Lisa", "B"],
    ["Karry", "B"]
])
type = BatchOperator.fromDataframe(nodeType, schemaStr='node string, type
string')
metapathBatchOp = HugeMetaPath2VecTrainBatchOp() \
  .setSourceCol("start")            \
  .setTargetCol("end")              \
  .setWeightCol("value")            \
  .setVertexCol("node")             \
  .setTypeCol("type")               \
  .setMetaPath("ABA")               \
  .setWalkNum(2)                    \
  .setWalkLength(2)                 \
  .setMinCount(1)                   \
  .setVectorSize(4)
metapathBatchOp.linkFrom(source, type).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.huge.HugeMetaPath2VecTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;
```

```
import java.util.Arrays;
import java.util.List;

public class HugeMetaPath2VecTrainBatchOpTest {
    @Test
    public void testHugeMetaPath2VecTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("Bob", "Lucy", 1.),
            Row.of("Lucy", "Bob", 1.),
            Row.of("Lucy", "Bella", 1.),
            Row.of("Bella", "Lucy", 1.),
            Row.of("Alice", "Lisa", 1.),
            Row.of("Lisa", "Alice", 1.),
            Row.of("Lisa", "Karry", 1.),
            Row.of("Karry", "Lisa", 1.),
            Row.of("Karry", "Bella", 1.),
            Row.of("Bella", "Karry", 1.)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df_data, "start string,
end string, value double");
        List <Row> nodeType = Arrays.asList(
            Row.of("Bob", "A"),
            Row.of("Bella", "A"),
            Row.of("Karry", "A"),
            Row.of("Lucy", "B"),
            Row.of("Alice", "B"),
            Row.of("Lisa", "B"),
            Row.of("Karry", "B")
        );
        BatchOperator <?> type = new MemSourceBatchOp(nodeType, "node string,
type string");
        BatchOperator <?> metapathBatchOp = new HugeMetaPath2VecTrainBatchOp()
            .setSourceCol("start")
            .setTargetCol("end")
            .setWeightCol("value")
            .setVertexCol("node")
            .setTypeCol("type")
            .setMetaPath("ABA")
            .setWalkNum(2)
            .setWalkLength(2)
            .setMinCount(1)
            .setVectorSize(4);
        metapathBatchOp.linkFrom(source, type).print();
    }
}
```

运行结果

node	vec
Karry	-0.028718041256070137,0.02825581468641758,0.12125638127326965,0.1207452341914177
Bella	0.03437831997871399,-0.0477546751499176,0.012570690363645554,-0.0958133116364479
Bob	0.024427175521850586,0.07044785469770432,-0.04175269603729248,-0.06182029843330383
Lucy	0.05776885524392128,0.08288335055112839,-0.06490718573331833,0.026563744992017746

大规模Node2Vec (HugeNode2VecTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.huge.HugeNode2VecTrainBatchOp

Python 类名：HugeNode2VecTrainBatchOp

功能介绍

node2vec是一种用于网络中的特征学习有效的可扩展算法，该算法可以使用SGD有效地优化，能根据网络中的既定原则，为发现符合不同等值的表示提供了灵活性

node2vec: Scalable Feature Learning for Networks

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sourceCol	起始点列名	用来指定起始点列	String	✓		
targetCol	中止点点列名	用来指定中止点列	String	✓		
walkLength	游走的长度	随机游走完向量的长度	Integer	✓		
walkNum	路径数目	每一个起始点游走出多少条路径	Integer	✓		
alpha	学习率	学习率	Double			0.025
batchSize	batch大小	batch大小, 按行计算	Integer		[1, +inf)	
isToUndigraph	是否转无向图	选为true时，会将当前图转成无向图，然后再游走	Boolean			false
minCount	最小词频	最小词频	Integer			5
negative	负采样大小	负采样大小	Integer			5
numCheckpoint	checkPoint数目	checkPoint 数目	Integer			1

numIter	迭代次数	迭代次数，默认为1。	Integer			1
p	p	p越小越趋向于访问到已经访问的节点，反之则趋向于访问没有访问过的节点	Double			1.0
q	q	q>1时行为类似于bfs趋向于访问和访问过的节点相连的节点，q<1时行为类似于dfs	Double			1.0
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"true"
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	100
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
window	窗口大小	窗口大小	Integer			5
wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df_data = pd.DataFrame([
    ["Bob", "Lucy", 1.],
    ["Lucy", "Bob", 1.],
    ["Lucy", "Bella", 1.],
    ["Bella", "Lucy", 1.],
    ["Alice", "Lisa", 1.],
    ["Lisa", "Alice", 1.],
    ["Lisa", "Karry", 1.],
    ["Karry", "Lisa", 1.],
    ["Karry", "Bella", 1.],
    ["Bella", "Karry", 1.]
])
source =  BatchOperator.fromDataframe(df_data, schemaStr='start string, end
string, value double')

node2vecBatchOp = HugeNode2VecTrainBatchOp() \
  .setSourceCol("start")            \
  .setTargetCol("end")              \
  .setWeightCol("value")            \
  .setWalkNum(2)                    \
  .setWalkLength(2)                 \
  .setMinCount(1)                   \
  .setVectorSize(4)
node2vecBatchOp.linkFrom(source).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.huge.HugeNode2VecTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HugeNode2VecTrainBatchOpTest {
    @Test
    public void testHugeNode2VecTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("Bob", "Lucy", 1.),
            Row.of("Lucy", "Bob", 1.),
            Row.of("Lucy", "Bella", 1.),
            Row.of("Bella", "Lucy", 1.),
            Row.of("Alice", "Lisa", 1.),
            Row.of("Lisa", "Alice", 1.),
```

```
            Row.of("Lisa", "Karry", 1.),
            Row.of("Karry", "Lisa", 1.),
            Row.of("Karry", "Bella", 1.),
            Row.of("Bella", "Karry", 1.)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df_data, "start string,
end string, value double");
        BatchOperator <?> node2vecBatchOp = new HugeNode2VecTrainBatchOp()
            .setSourceCol("start")
            .setTargetCol("end")
            .setWeightCol("value")
            .setWalkNum(2)
            .setWalkLength(2)
            .setMinCount(1)
            .setVectorSize(4);
        node2vecBatchOp.linkFrom(source).print();
    }
}
```

运行结果

node	vec
Karry	0.02435881271958351,0.0703350380063057,-0.04173225536942482,-0.06183897703886032
Bella	-0.028720347210764885,0.02828666940331459,0.12123052030801773,0.12075022608041763
Alice	0.03435942903161049,-0.04773801192641258,0.0125938905403018,-0.09576953202486038
Lisa	-0.07306616753339767,-0.11595576256513596,-0.04181118682026863,0.03970039263367653
Bob	0.0577755942940712,0.08282522112131119,-0.06487344205379486,0.026600968092679977
Lucy	0.057738181203603745,-0.09987597167491913,-0.022486409172415733,-0.02312176302075386

大规模Word2Vec (HugeWord2VecTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.huge.HugeWord2VecTrainBatchOp

Python 类名：HugeWord2VecTrainBatchOp

功能介绍

Word2Vec是Google在2013年开源的一个将词表转为向量的算法，其利用神经网络，可以通过训练，将词映射到K维度空间向量，甚至对于表示词的向量进行操作还能和语义相对应，由于其简单和高效引起了很多人的关注。

Word2Vec的工具包相关链接：https://code.google.com/p/word2vec/

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	计算列对应的列名	计算列对应的列名	String	✓	所选列类型为 [STRING]	
alpha	学习率	学习率	Double			0.025
batchSize	batch大小	batch大小, 按行计算	Integer		[1, +inf)	
minCount	最小词频	最小词频	Integer			5
negative	负采样大小	负采样大小	Integer			5
numCheckpoint	checkPoint数目	checkPoint 数目	Integer			1
numIter	迭代次数	迭代次数，默认为1。	Integer			1
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String			"true"
vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)	100
window	窗口大小	窗口大小	Integer			5
wordDelimiter	单词分隔符	单词之间的分隔符	String			" "

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

tokens = pd.DataFrame([
    ["A B C"]
])

source = BatchOperator.fromDataframe(tokens, schemaStr='tokens string')

word2vecBatchOp = HugeWord2VecTrainBatchOp() \
  .setSelectedCol("tokens")         \
  .setMinCount(1)                   \
  .setVectorSize(4)
word2vecBatchOp.linkFrom(source).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.huge.HugeWord2VecTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HugeWord2VecTrainBatchOpTest {
    @Test
    public void testHugeWord2VecTrainBatchOp() throws Exception {
        List <Row> tokens = Arrays.asList(
            Row.of("A B C")
        );
        BatchOperator <?> source = new MemSourceBatchOp(tokens, "tokens
string");
        BatchOperator <?> word2vecBatchOp = new HugeWord2VecTrainBatchOp()
            .setSelectedCol("tokens")
            .setMinCount(1)
            .setVectorSize(4);
        word2vecBatchOp.linkFrom(source).print();
```

```
    }
}
```

运行结果

word	vec
A	0.024366257712244987,0.07037621736526489,-0.04168345779180527,-0.06180821731686592
B	0.05771925672888756,0.08288027346134186,-0.06486544758081436,0.026565641164779663
C	0.034414440393447876,-0.047638311982154846,0.012538374401628971,-0.09579437971115112

备注

如果不输入vecTable的情况下是随机初始化，可能会造成两次结果相同的item的embedding结果绝对值差别比较大，请注意

BoxPlot异常检测 (BoxPlotOutlierBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.BoxPlotOutlierBatchOp

Python 类名：BoxPlotOutlierBatchOp

功能介绍

- BoxPlot算法又叫做箱线图算法, 是一种常用的异常检测算法.
- 它由五个数值点组成：最小值(min)，下四分位数(Q1)，中位数(median)，上四分位数(Q3)，最大值(max)，大于Q3 + K *IQR和小于Q1 - K* IQR的点定义为异常值(Outlier)。
- k通常取值为1.5或者3.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
direction	方向	检测异常的方向	String		"POSITIVE", "NEGATIVE", "BOTH"	"BOTH"
featureCol	特征列名	特征列名，默认选最左边的列	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
groupCols	分组列名数组	分组列名，多列，可选，默认不选	String[]			null

maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer			
maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double			
maxSampleNumPerGroup	每组最大样本数目	每组最大样本数目	Integer			
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import pandas as pd
df = pd.DataFrame([
                [0.73, 0],
                [0.24, 0],
                [0.63, 0],
                [0.55, 0],
                [0.73, 0],
                [0.41, 0]
        ])

dataOp = BatchOperator.fromDataframe(df, schemaStr='val double, label int')

outlierOp = BoxPlotOutlierBatchOp()\
            .setFeatureCol("val")\
            .setOutlierThreshold(3.0)\
            .setPredictionCol("pred")\
            .setPredictionDetailCol("pred_detail")

evalOp = EvalOutlierBatchOp()\
            .setLabelCol("label")\
            .setPredictionDetailCol("pred_detail")\
            .setOutlierValueStrings(["1"]);

metrics = dataOp\
            .link(outlierOp)\
            .link(evalOp)\
```

```
            .collectMetrics()

print(metrics)
```

Java 代码

```
package com.alibaba.alink.operator.batch.outlier;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalOutlierBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.OutlierMetrics;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Assert;
import org.junit.Test;

public class BoxPlotOutlierBatchOpTest extends AlinkTestBase {
    @Test
    public void test() throws Exception {
        BatchOperator <?> data = new MemSourceBatchOp(
            new Object[][] {
                {0.73, 0},
                {0.24, 0},
                {0.63, 0},
                {0.55, 0},
                {0.73, 0},
                {0.41, 0},
            },
            new String[]{"val", "label"});

        BatchOperator <?> outlier = new BoxPlotOutlierBatchOp()
            .setFeatureCol("val")
            .setOutlierThreshold(3.0)
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail");

        EvalOutlierBatchOp eval = new EvalOutlierBatchOp()
            .setLabelCol("label")
            .setPredictionDetailCol("pred_detail")
            .setOutlierValueStrings("1");

        OutlierMetrics metrics = data
            .link(outlier)
            .link(eval)
            .collectMetrics();

        Assert.assertEquals(1.0, metrics.getAccuracy(), 10e-6);
```

```
    }
}
```

运行结果

```
---------------------------------- Metrics: ----------------------------------
Outlier values: [1]             Normal values: [0]
Auc:NaN Accuracy:1      Precision:1     Recall:0        F1:0
|Pred\Real|Outlier|Normal|
|---------|-------|------|
|  Outlier|      0|     0|
|   Normal|      0|     6|
```

DBSCAN异常检测 (DbscanOutlierBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.DbscanOutlierBatchOp

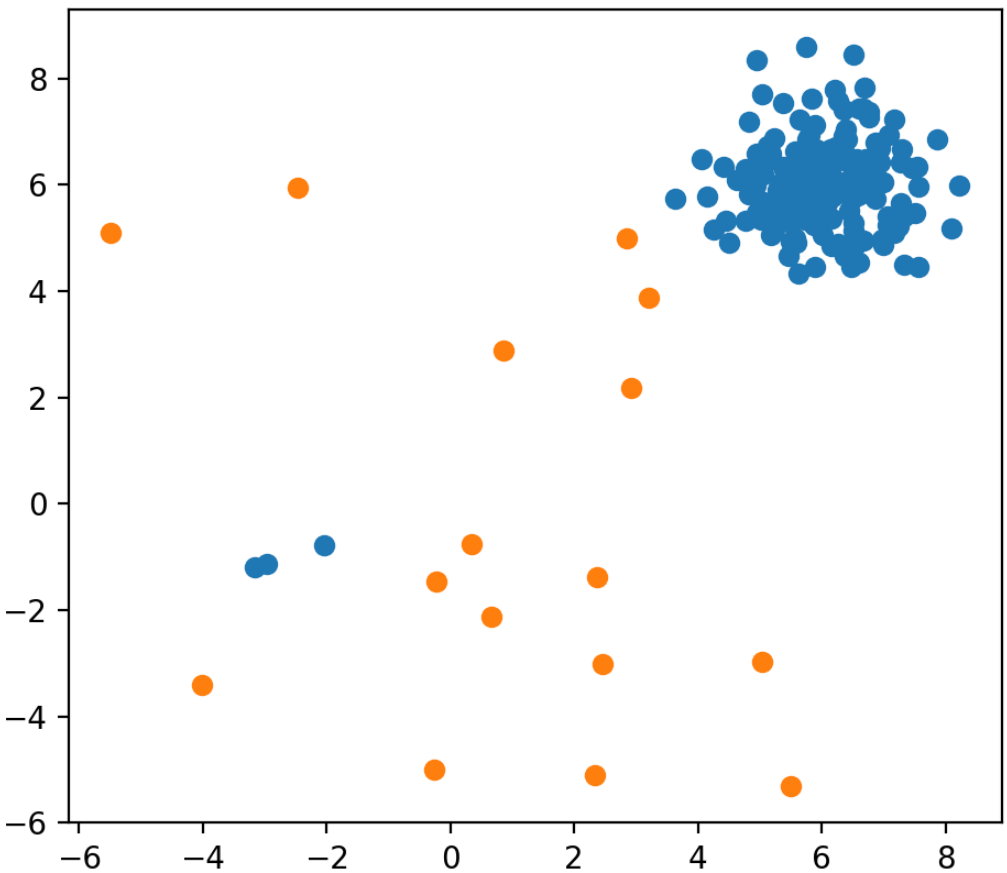
Python 类名：DbscanOutlierBatchOp

功能介绍

DBSCAN，Density-Based Spatial Clustering of Applications with Noise，是一个比较有代表性的基于密度的聚类算法。与划分和层次聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在噪声的空间数据库中发现任意形状的聚类。 基于DBSCAN聚类的异常检测算法将规模过小的簇视为异常。

使用方法

使用DBSCAN算法进行异常检测需要设置聚类半径，距离计算方法和簇最小规模。



当聚类半径设为0.1，用欧氏距离度量，簇最小规模为3时，如图是DBSCAN算法的异常检测效果，

距离度量方式

参数名称	参数描述	说明
EUCLIDEAN	$d(x-c)=(x-c)(x-c)^{'}$	欧式距离
COSINE	$d(x-c)=0.5-0.5*\frac{xc^{'}}{\sqrt{xx^{'}}\sqrt{cc^{'}}}$	夹角余弦距离
CITYBLOCK		街区距离

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认
predictionCol	预测结果列名	预测结果列名	String	✓		
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE", "INNERPRODUCT", "CITYBLOCK", "JACCARD", "PEARSON"	"EUCLI
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
groupCols	分组列名数组	分组列名，多列，可选，默认不选	String[]			null

maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer			
maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double			
maxSampleNumPerGroup	每组最大样本数目	每组最大样本数目	Integer			
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
tensorCol	tensor列	tensor列	String		所选列类型为[BOOL_TENSOR, BYTE_TENSOR, DOUBLE_TENSOR, FLOAT_TENSOR, INT_TENSOR, LONG_TENSOR, STRING, STRING_TENSOR, TENSOR, UBYTE_TENSOR]	null

vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

JAVA 代码

ESD异常检测 (EsdOutlierBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.EsdOutlierBatchOp

Python 类名：EsdOutlierBatchOp

功能介绍

ESD算法是一种常用的异常检测算法.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
alpha	置信度	置信度	Double			0.05
direction	方向	检测异常的方向	String		"POSITIVE", "NEGATIVE", "BOTH"	"BOTH"
featureCol	特征列名	特征列名，默认选最左边的列	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
groupCols	分组列名数组	分组列名，多列，可选，默认不选	String[]			null

maxIter	最大迭代步数	最大迭代步数	Integer			
maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer			
maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double			
maxSampleNumPerGroup	每组最大样本数目	每组最大样本数目	Integer			
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import pandas as pd
df = pd.DataFrame([
                [0.73, 0],
                [0.24, 0],
                [0.63, 0],
                [0.55, 0],
                [0.73, 0],
                [0.41, 0]
        ])

dataOp = BatchOperator.fromDataframe(df, schemaStr='val double, label int')

outlierOp = EsdOutlierBatchOp()\
            .setFeatureCol("val")\
            .setOutlierThreshold(3.0)\
            .setPredictionCol("pred")\
            .setPredictionDetailCol("pred_detail")

evalOp = EvalOutlierBatchOp()\
            .setLabelCol("label")\
            .setPredictionDetailCol("pred_detail")\
            .setOutlierValueStrings(["1"]);

metrics = dataOp\
            .link(outlierOp)\
            .link(evalOp)\
```

```
            .collectMetrics()

print(metrics)
```

Java 代码

```
package com.alibaba.alink.operator.batch.outlier;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalOutlierBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.OutlierMetrics;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Assert;
import org.junit.Test;

public class EsdOutlierBatchOpTest extends AlinkTestBase {
    @Test
    public void test() throws Exception {
        BatchOperator <?> data = new MemSourceBatchOp(
            new Object[][] {
                {0.73, 0},
                {0.24, 0},
                {0.63, 0},
                {0.55, 0},
                {0.73, 0},
                {0.41, 0},
            },
            new String[]{"val", "label"});

        BatchOperator <?> outlier = new EsdOutlierBatchOp()
            .setFeatureCol("val")
            .setOutlierThreshold(3.0)
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail");

        EvalOutlierBatchOp eval = new EvalOutlierBatchOp()
            .setLabelCol("label")
            .setPredictionDetailCol("pred_detail")
            .setOutlierValueStrings("1");

        OutlierMetrics metrics = data
            .link(outlier)
            .link(eval)
            .collectMetrics();

        Assert.assertEquals(1.0, metrics.getAccuracy(), 10e-6);
```

```
    }
}
```

运行结果

无

HBOS序列异常检测 (HbosOutlier4GroupedDataBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.HbosOutlier4GroupedDataBatchOp

Python 类名：HbosOutlier4GroupedDataBatchOp

功能介绍

Histogram-based Outlier Score 使用直方图统计结果，描述异常值，算法较为简单，上手方便。

文献或出处

1. HBOS

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
inputMTableCol	输入列名	输入序列的列名	String	✓		
outputMTableCol	输出列名	输出序列的列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
k	K	直方图 bin 的数量	Integer		[1, +inf)	10

maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer			
maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double			
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
tensorCol	tensor列	tensor列	String		所选列类型为[BOOL_TENSOR, BYTE_TENSOR, DOUBLE_TENSOR, FLOAT_TENSOR, INT_TENSOR, LONG_TENSOR, STRING, STRING_TENSOR, TENSOR, UBYTE_TENSOR]	null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
import pandas as pd
df = pd.DataFrame([
    [1, 1, 10.0],
    [1, 2, 11.0],
    [1, 3, 12.0],
    [1, 4, 13.0],
    [1, 5, 14.0],
    [1, 6, 15.0],
    [1, 7, 16.0],
    [1, 8, 17.0],
    [1, 9, 18.0],
    [1, 10, 19.0]
])

dataOp = BatchOperator.fromDataframe(
    df, schemaStr='group_id int, id int, val double')

outlierOp = dataOp.link(
    GroupByBatchOp()
    .setGroupByPredicate("group_id")
    .setSelectClause("mtable_agg(id, val) as data")
).link(
    HbosOutlier4GroupedDataBatchOp()
    .setInputMTableCol("data")
    .setOutputMTableCol("pred")
    .setFeatureCols(["val"])
    .setPredictionCol("detect_pred")
)

outlierOp.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
```

```
import com.alibaba.alink.operator.batch.outlier.HbosOutlier4GroupedDataBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class HbosOutlier4GroupedDataBatchOpTest {
    @Test
    public void test() throws Exception {
        List <Row> mTableData = Arrays.asList(
            Row.of(1, 1, 10.0),
            Row.of(1, 2, 11.0),
            Row.of(1, 3, 12.0),
            Row.of(1, 4, 13.0),
            Row.of(1, 5, 14.0),
            Row.of(1, 6, 15.0),
            Row.of(1, 7, 16.0),
            Row.of(1, 8, 17.0),
            Row.of(1, 9, 18.0),
            Row.of(1, 10, 19.0)
        );

        MemSourceBatchOp dataOp = new MemSourceBatchOp(mTableData, new String[]
{"group_id", "id", "val"});

        BatchOperator <?> outlierOp = dataOp.link(
            new GroupByBatchOp()
                .setGroupByPredicate("group_id")
                .setSelectClause("group_id, mtable_agg(id, val) as data")
        ).link(
            new HbosOutlier4GroupedDataBatchOp()
                .setInputMTableCol("data")
                .setOutputMTableCol("pred")
                .setFeatureCols("val")
                .setPredictionCol("detect_pred")
        );

        outlierOp.print();
    }
}
```

运行结果

group_id	data	pred
1	MTable(10,2)(id,val)	MTable(10,3)(id,val,detect_pred)

	1	10.0000	1	10.0000	false
	2	11.0000	2	11.0000	false
	3	12.0000	3	12.0000	false
	4	13.0000	4	13.0000	false
	5	14.0000	5	14.0000	false

HBOS异常检测 (HbosOutlierBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.HbosOutlierBatchOp

Python 类名：HbosOutlierBatchOp

功能介绍

Histogram-based Outlier Score 使用直方图统计结果，描述异常值，算法较为简单，上手方便。

文献或出处

1. HBOS

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
groupCols	分组列名数组	分组列名，多列，可选，默认不选	String[]			null
k	K	直方图 bin 的数量	Integer		[1, +inf)	10

maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer			
maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double			
maxSampleNumPerGroup	每组最大样本数目	每组最大样本数目	Integer			
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
tensorCol	tensor列	tensor列	String		所选列类型为[BOOL_TENSOR, BYTE_TENSOR, DOUBLE_TENSOR, FLOAT_TENSOR, INT_TENSOR, LONG_TENSOR, STRING, STRING_TENSOR, TENSOR, UBYTE_TENSOR]	null

vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
import pandas as pd
df = pd.DataFrame([
    [0.73, 0],
    [0.24, 0],
    [0.63, 0],
    [0.55, 0],
    [0.73, 0],
    [0.41, 0]
])

dataOp = BatchOperator.fromDataframe(df, schemaStr='val double, label int')

outlierOp = HbosOutlierBatchOp()\
    .setFeatureCols(["val"])\
    .setOutlierThreshold(3.0)\
    .setPredictionCol("pred")\
    .setPredictionDetailCol("pred_detail")

evalOp = EvalOutlierBatchOp()\
    .setLabelCol("label")\
    .setPredictionDetailCol("pred_detail")\
    .setOutlierValueStrings(["1"])

metrics = dataOp\
    .link(outlierOp)\
    .link(evalOp)\
    .collectMetrics()
```

```
print(metrics)
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalOutlierBatchOp;
import com.alibaba.alink.operator.batch.outlier.HbosOutlierBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.OutlierMetrics;
import org.junit.Assert;
import org.junit.Test;

public class HbosOutlierBatchOpTest {

    @Test
    public void test() throws Exception {
        BatchOperator <?> data = new MemSourceBatchOp(
            new Object[][] {
                {0.73, 0},
                {0.24, 0},
                {0.63, 0},
                {0.55, 0},
                {0.73, 0},
                {0.41, 0},
            },
            new String[] {"val", "label"});

        BatchOperator <?> outlier = new HbosOutlierBatchOp()
            .setFeatureCols("val")
            .setOutlierThreshold(3.0)
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail");

        EvalOutlierBatchOp eval = new EvalOutlierBatchOp()
            .setLabelCol("label")
            .setPredictionDetailCol("pred_detail")
            .setOutlierValueStrings("1");

        OutlierMetrics metrics = data
            .link(outlier)
            .link(eval)
            .collectMetrics();

        Assert.assertEquals(1.0, metrics.getAccuracy(), 10e-6);

    }
```

```
}
```

运行结果

---------- Metrics: ---------- Outlier values: [1] Normal values: [0] Auc:NaN Accuracy:1 Precision:1 Recall:0 F1:0 |Pred\Real|Outlier|Normal| |---------|-------|------| | Outlier| 0| 0| | Normal| 0| 6|

IForest模型异常检测预测 (IForestModelOutlierPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.IForestModelOutlierPredictBatchOp

Python 类名：IForestModelOutlierPredictBatchOp

功能介绍

iForest 可以识别数据中异常点，在异常检测领域有比较好的效果。算法使用 sub-sampling 方法，降低了算法的计算复杂度。

文献或出处

1. Isolation Forest

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
import pandas as pd
df = pd.DataFrame([
[0.73, 0],
[0.24, 0],
[0.63, 0],
[0.55, 0],
[0.73, 0],
[0.41, 0]
])

dataOp = BatchOperator.fromDataframe(df, schemaStr='val double, label int')

trainOp = IForestModelOutlierTrainBatchOp()\
.setFeatureCols(["val"])

predOp = IForestModelOutlierPredictBatchOp()\
.setOutlierThreshold(3.0)\
.setPredictionCol("pred")\
.setPredictionDetailCol("pred_detail")

predOp.linkFrom(trainOp.linkFrom(dataOp), dataOp)

evalOp = EvalOutlierBatchOp()\
.setLabelCol("label")\
.setPredictionDetailCol("pred_detail")\
.setOutlierValueStrings(["1"]);

metrics = predOp\
.link(evalOp)\
.collectMetrics()

print(metrics)
```

Java 代码

```
package com.alibaba.alink.operator.batch.outlier;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalOutlierBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.OutlierMetrics;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Assert;
import org.junit.Test;

public class IForestModelOutlierTrainBatchOpTest extends AlinkTestBase {
```

```
    @Test
    public void test() {
        BatchOperator <?> data = new MemSourceBatchOp(
            new Object[][] {
                {0.73, 0},
                {0.24, 0},
                {0.63, 0},
                {0.55, 0},
                {0.73, 0},
                {0.41, 0},
            },
            new String[]{"val", "label"});

        IForestModelOutlierTrainBatchOp trainOp = new
IForestModelOutlierTrainBatchOp()
            .setFeatureCols("val");

        IForestModelOutlierPredictBatchOp predOp = new
IForestModelOutlierPredictBatchOp()
            .setOutlierThreshold(3.0)
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail");

        predOp.linkFrom(trainOp.linkFrom(data), data);

        EvalOutlierBatchOp eval = new EvalOutlierBatchOp()
            .setLabelCol("label")
            .setPredictionDetailCol("pred_detail")
            .setOutlierValueStrings("1");

        OutlierMetrics metrics = predOp
            .link(eval)
            .collectMetrics();

        Assert.assertEquals(1.0, metrics.getAccuracy(), 10e-6);
    }
}
```

运行结果

---------- Metrics: ---------- Outlier values: [1] Normal values: [0] Auc:NaN Accuracy:1 Precision:1 Recall:0 F1:0 |Pred\Real|Outlier|Normal| |---------|-------|------| | Outlier| 0| 0| | Normal| 0| 6|

IForest模型异常检测训练 (IForestModelOutlierTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.IForestModelOutlierTrainBatchOp

Python 类名：IForestModelOutlierTrainBatchOp

功能介绍

iForest 可以识别数据中异常点，在异常检测领域有比较好的效果。算法使用 sub-sampling 方法，降低了算法的计算复杂度。

文献或出处

1. Isolation Forest

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
numTrees	模型中树的棵数	模型中树的棵数	Integer			100
subsamplingSize	每棵树的样本采样行数	每棵树的样本采样行数，默认 256，最小 2，最大 100000 .	Integer		[1, 100000]	256

tensorCol	tensor列	tensor列	String		所选列类型为[BOOL_TENSOR, BYTE_TENSOR, DOUBLE_TENSOR, FLOAT_TENSOR, INT_TENSOR, LONG_TENSOR, STRING, STRING_TENSOR, TENSOR, UBYTE_TENSOR]	null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

代码示例

Python 代码

```
import pandas as pd
df = pd.DataFrame([
[0.73, 0],
[0.24, 0],
[0.63, 0],
[0.55, 0],
[0.73, 0],
[0.41, 0]
])

dataOp = BatchOperator.fromDataframe(df, schemaStr='val double, label int')

trainOp = IForestModelOutlierTrainBatchOp()\
.setFeatureCols(["val"])

predOp = IForestModelOutlierPredictBatchOp()\
.setOutlierThreshold(3.0)\
.setPredictionCol("pred")\
.setPredictionDetailCol("pred_detail")

predOp.linkFrom(trainOp.linkFrom(dataOp), dataOp)

evalOp = EvalOutlierBatchOp()\
.setLabelCol("label")\
.setPredictionDetailCol("pred_detail")\
.setOutlierValueStrings(["1"]);
```

```
metrics = predOp\
.link(evalOp)\
.collectMetrics()

print(metrics)
```

Java 代码

```
package com.alibaba.alink.operator.batch.outlier;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalOutlierBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.OutlierMetrics;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Assert;
import org.junit.Test;

public class IForestModelOutlierTrainBatchOpTest extends AlinkTestBase {

    @Test
    public void test() {
        BatchOperator <?> data = new MemSourceBatchOp(
            new Object[][] {
                {0.73, 0},
                {0.24, 0},
                {0.63, 0},
                {0.55, 0},
                {0.73, 0},
                {0.41, 0},
            },
            new String[]{"val", "label"});

        IForestModelOutlierTrainBatchOp trainOp = new
IForestModelOutlierTrainBatchOp()
            .setFeatureCols("val");

        IForestModelOutlierPredictBatchOp predOp = new
IForestModelOutlierPredictBatchOp()
            .setOutlierThreshold(3.0)
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail");

        predOp.linkFrom(trainOp.linkFrom(data), data);

        EvalOutlierBatchOp eval = new EvalOutlierBatchOp()
            .setLabelCol("label")
```

```
            .setPredictionDetailCol("pred_detail")
            .setOutlierValueStrings("1");

        OutlierMetrics metrics = predOp
            .link(eval)
            .collectMetrics();

        Assert.assertEquals(1.0, metrics.getAccuracy(), 10e-6);
    }
}
```

运行结果

---------- Metrics: ---------- Outlier values: [1] Normal values: [0] Auc:NaN Accuracy:1 Precision:1 Recall:0 F1:0 |Pred\Real|Outlier|Normal| |---------|-------|------| | Outlier| 0| 0| | Normal| 0| 6|

IForest序列异常检测 (IForestOutlier4GroupedDataBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.IForestOutlier4GroupedDataBatchOp

Python 类名：IForestOutlier4GroupedDataBatchOp

功能介绍

iForest 可以识别数据中异常点，在异常检测领域有比较好的效果。算法使用 sub-sampling 方法，降低了算法的计算复杂度。

文献或出处

1. Isolation Forest

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
inputMTableCol	输入列名	输入序列的列名	String	✓		
outputMTableCol	输出列名	输出序列的列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer			

maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double			
numTrees	模型中树的棵数	模型中树的棵数	Integer			100
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
subsamplingSize	每棵树的样本采样行数	每棵树的样本采样行数，默认 256，最小 2，最大 100000.	Integer		[1, 100000]	256
tensorCol	tensor列	tensor列	String		所选列类型为[BOOL_TENSOR, BYTE_TENSOR, DOUBLE_TENSOR, FLOAT_TENSOR, INT_TENSOR, LONG_TENSOR, STRING, STRING_TENSOR, TENSOR, UBYTE_TENSOR]	null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
import pandas as pd
df = pd.DataFrame([
    [1, 1, 10.0],
    [1, 2, 11.0],
    [1, 3, 12.0],
    [1, 4, 13.0],
    [1, 5, 14.0],
    [1, 6, 15.0],
    [1, 7, 16.0],
    [1, 8, 17.0],
    [1, 9, 18.0],
    [1, 10, 19.0]
])

dataOp = BatchOperator.fromDataframe(
    df, schemaStr='group_id int, id int, val double')

outlierOp = dataOp.link(
    GroupByBatchOp()
    .setGroupByPredicate("group_id")
    .setSelectClause("mtable_agg(id, val) as data")
).link(
    IForestOutlier4GroupedDataBatchOp()
    .setInputMTableCol("data")
    .setOutputMTableCol("pred")
    .setFeatureCols(["val"])
    .setPredictionCol("detect_pred")
)

outlierOp.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
```

```
import
com.alibaba.alink.operator.batch.outlier.IForestOutlier4GroupedDataBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class IForestOutlier4GroupedDataBatchOpTest {
    @Test
    public void test() throws Exception {
        List <Row> mTableData = Arrays.asList(
            Row.of(1, 1, 10.0),
            Row.of(1, 2, 11.0),
            Row.of(1, 3, 12.0),
            Row.of(1, 4, 13.0),
            Row.of(1, 5, 14.0),
            Row.of(1, 6, 15.0),
            Row.of(1, 7, 16.0),
            Row.of(1, 8, 17.0),
            Row.of(1, 9, 18.0),
            Row.of(1, 10, 19.0)
        );

        MemSourceBatchOp dataOp = new MemSourceBatchOp(mTableData, new String[]
{"group_id", "id", "val"});

        BatchOperator <?> outlierOp = dataOp.link(
            new GroupByBatchOp()
                .setGroupByPredicate("group_id")
                .setSelectClause("group_id, mtable_agg(id, val) as data")
        ).link(
            new IForestOutlier4GroupedDataBatchOp()
                .setInputMTableCol("data")
                .setOutputMTableCol("pred")
                .setFeatureCols("val")
                .setPredictionCol("detect_pred")
        );

        outlierOp.print();
    }
}
```

运行结果

group_id	data	pred

1	MTable(10,2)(id,val)	MTable(10,3)(id,val,detect_pred)			
	1	10.0000	1	10.0000	false
	2	11.0000	2	11.0000	false
	3	12.0000	3	12.0000	false
	4	13.0000	4	13.0000	false
	5	14.0000	5	14.0000	false

IForest异常检测 (IForestOutlierBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.IForestOutlierBatchOp

Python 类名：IForestOutlierBatchOp

功能介绍

iForest 可以识别数据中异常点，在异常检测领域有比较好的效果。算法使用 sub-sampling 方法，降低了算法的计算复杂度。

文献或出处

1. Isolation Forest

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
groupCols	分组列名数组	分组列名，多列，可选，默认不选	String[]			null
maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer			

maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double			
maxSampleNumPerGroup	每组最大样本数目	每组最大样本数目	Integer			
numTrees	模型中树的棵数	模型中树的棵数	Integer			100
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
subsamplingSize	每棵树的样本采样行数	每棵树的样本采样行数，默认 256，最小 2，最大 100000.	Integer		[1, 100000]	256
tensorCol	tensor 列	tensor 列	String		所选列类型为 [BOOL_TENSOR, BYTE_TENSOR, DOUBLE_TENSOR, FLOAT_TENSOR, INT_TENSOR, LONG_TENSOR, STRING, STRING_TENSOR, TENSOR, UBYTE_TENSOR]	null

vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
import pandas as pd
df = pd.DataFrame([
[0.73, 0],
[0.24, 0],
[0.63, 0],
[0.55, 0],
[0.73, 0],
[0.41, 0]
])

dataOp = BatchOperator.fromDataframe(df, schemaStr='val double, label int')

outlierOp = IForestOutlierBatchOp()\
.setFeatureCols(["val"])\
.setOutlierThreshold(3.0)\
.setPredictionCol("pred")\
.setPredictionDetailCol("pred_detail")

evalOp = EvalOutlierBatchOp()\
.setLabelCol("label")\
.setPredictionDetailCol("pred_detail")\
.setOutlierValueStrings(["1"]);

metrics = dataOp\
.link(outlierOp)\
.link(evalOp)\
.collectMetrics()

print(metrics)
```

Java 代码

```
package com.alibaba.alink.operator.batch.outlier;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalOutlierBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.OutlierMetrics;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Assert;
import org.junit.Test;

public class IForestOutlierBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {

        BatchOperator <?> data = new MemSourceBatchOp(
            new Object[][] {
                {0.73, 0},
                {0.24, 0},
                {0.63, 0},
                {0.55, 0},
                {0.73, 0},
                {0.41, 0},
            },
            new String[]{"val", "label"});

        BatchOperator <?> outlier = new IForestOutlierBatchOp()
            .setFeatureCols("val")
            .setOutlierThreshold(3.0)
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail");

        EvalOutlierBatchOp eval = new EvalOutlierBatchOp()
            .setLabelCol("label")
            .setPredictionDetailCol("pred_detail")
            .setOutlierValueStrings("1");

        OutlierMetrics metrics = data
            .link(outlier)
            .link(eval)
            .collectMetrics();

        Assert.assertEquals(1.0, metrics.getAccuracy(), 10e-6);

    }
}
```

运行结果

---------- Metrics: ---------- Outlier values: [1] Normal values: [0] Auc:NaN Accuracy:1 Precision:1 Recall:0 F1:0 |Pred\Real|Outlier|Normal| |---------|-------|------| | Outlier| 0| 0| | Normal| 0| 6|

KSigma异常检测 (KSigmaOutlierBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.KSigmaOutlierBatchOp

Python 类名：KSigmaOutlierBatchOp

功能介绍

KSigma算法是一种常用的异常检测算法,如果整体数据服从正态分布，则如果一个点偏离均值K倍标准差，则该点被视为异常点.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
direction	方向	检测异常的方向	String		"POSITIVE", "NEGATIVE", "BOTH"	"BOTH"
featureCol	特征列名	特征列名，默认选最左边的列	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
groupCols	分组列名数组	分组列名，多列，可选，默认不选	String[]			null

maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer			
maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double			
maxSampleNumPerGroup	每组最大样本数目	每组最大样本数目	Integer			
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import pandas as pd
df = pd.DataFrame([
                [0.73, 0],
                [0.24, 0],
                [0.63, 0],
                [0.55, 0],
                [0.73, 0],
                [0.41, 0]
        ])

dataOp = BatchOperator.fromDataframe(df, schemaStr='val double, label int')

outlierOp = KSigmaOutlierBatchOp()\
            .setFeatureCol("val")\
            .setOutlierThreshold(3.0)\
            .setPredictionCol("pred")\
            .setPredictionDetailCol("pred_detail")

evalOp = EvalOutlierBatchOp()\
            .setLabelCol("label")\
            .setPredictionDetailCol("pred_detail")\
            .setOutlierValueStrings(["1"]);

metrics = dataOp\
            .link(outlierOp)\
            .link(evalOp)\
```

```
            .collectMetrics()

print(metrics)
```

Java 代码

```
package com.alibaba.alink.operator.batch.outlier;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalOutlierBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.OutlierMetrics;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Assert;
import org.junit.Test;

public class KSigmaOutlierBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
        BatchOperator <?> data = new MemSourceBatchOp(
            new Object[][] {
                {0.73, 0},
                {0.24, 0},
                {0.63, 0},
                {0.55, 0},
                {0.73, 0},
                {0.41, 0},
            },
            new String[]{"val", "label"});

        BatchOperator <?> outlier = new KSigmaOutlierBatchOp()
            .setFeatureCol("val")
            .setOutlierThreshold(3.0)
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail");

        EvalOutlierBatchOp eval = new EvalOutlierBatchOp()
            .setLabelCol("label")
            .setPredictionDetailCol("pred_detail")
            .setOutlierValueStrings("1");

        OutlierMetrics metrics = data
            .link(outlier)
            .link(eval)
            .collectMetrics();
```

```
        Assert.assertEquals(1.0, metrics.getAccuracy(), 10e-6);

    }
}
```

运行结果

```
-------------------------------- Metrics: --------------------------------
Outlier values: [1]             Normal values: [0]
Auc:NaN Accuracy:1      Precision:1     Recall:0        F1:0
|Pred\Real|Outlier|Normal|
|---------|-------|------|
|  Outlier|      0|     0|
|   Normal|      0|     6|
```

局部核密度估计异常检测 (KdeOutlierBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.KdeOutlierBatchOp

Python 类名：KdeOutlierBatchOp

功能介绍

KDE（Kernel Density Estimation核密度估计）是一种通过数据样本集，得到总体的概率分布的非参数估计方法。KDE异常检测算法将概率密度小的点视为异常点。

算法原理

该组件以每个点的数据、带宽作为参数，根据设置的核函数（高斯核或线性核）估计样本中每个数据点及其附近的概率密度函数。

- 带宽(bandwidth)：带宽设的越小，误差越小，但方差越大，KDE整体曲线就越陡峭，反之，就越平坦。不同的带宽对拟合结果的影响可能很大。
- 核函数(kernel)：用来对每个数据点得到光滑的、积分为1的概率密度估计。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范
bandwidth	KDE带宽	核密度函数带宽参数	Double	✓	[0.0, +inf)
predictionCol	预测结果列名	预测结果列名	String	✓	
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN "COSINE", "INNERPRO "CITYBLOCK "JACCARD", "PEARSON"
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为 [BIGDECIMA BIGINTEGE BYTE, DOUB FLOAT, INTE LONG, SHO

groupCols	分组列名数组	分组列名，多列，可选，默认不选	String[]		
kernelType	核密度函数类型	核密度函数类型，可取为"GAUSSIAN"，"LINEAR"	String		"GAUSSIAN" "LINEAR"
maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer		
maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double		
maxSampleNumPerGroup	每组最大样本数目	每组最大样本数目	Integer		
numNeighbors	相邻点个数	计算KDE时使用的相邻点个数(默认使用全部点)	Integer		
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
tensorCol	tensor列	tensor列	String		所选列类型为 [BOOL_TEN BYTE_TENS DOUBLE_TE FLOAT_TEN INT_TENSO LONG_TENS STRING, STRING_TE TENSOR, UBYTE_TEN

vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VE SPARSE_VE STRING, VE
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		

代码示例

Python 代码

```
import pandas as pd
df = pd.DataFrame([
                [-1.1],
                [0.2],
                [101.1],
                [0.3]
        ])

dataOp = BatchOperator.fromDataframe(df, schemaStr='val double')

outlierOp = KdeOutlierBatchOp()\
            .setFeatureCols(["val"])\
            .setBandwidth(4.0)\
            .setOutlierThreshold(15.0)\
            .setPredictionCol("pred")\
            .setPredictionDetailCol("pred_detail")

dataOp.link(outlierOp).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.outlier.KdeOutlierBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

public class KdeOutlierBatchOpTest {

    @Test
    public void testBatchOp() throws Exception {
        BatchOperator <?> data = new MemSourceBatchOp(
```

```
            new Object[][] {
                {-1.1},
                {0.2},
                {101.1},
                {0.3}}
            , new String[] {"val"}
        );

        KdeOutlierBatchOp kdeOutlierBatchOp = new KdeOutlierBatchOp()
            .setBandwidth(4.)
            .setOutlierThreshold(15.)
            .setFeatureCols("val")
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail")
            .linkFrom(data)
            .print();
    }
}
```

运行结果

val	pred	pred_detail
-1.1	false	{"outlier_score":"13.881629239608612","KDE":"0.0720376537032619","is_outlier":"false"}
0.2	false	{"outlier_score":"13.603363005188747","KDE":"0.07351123392197714","is_outlier":"false"
101.1	true	{"outlier_score":"40.106052394096","KDE":"0.02493389252508955","is_outlier":"true"}
0.3	false	{"outlier_score":"13.640235707107605","KDE":"0.07331251610842206","is_outlier":"false

局部异常因子异常检测 (LofOutlierBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.LofOutlierBatchOp

Python 类名：LofOutlierBatchOp

功能介绍

根据数据样本的局部异常因子值（Local Outlier Factor, LOF）判断样本是否异常。

算法原理

LOF 是根据样本点间距离关系计算得到的数值，用$d(\cdot, \cdot)$表示两个样本点的距离。

LOF 的计算过程包含以下几个步骤：

1. 对于样本点 p，找到其最近的 k 个样本点（不包含自身），称作 p 的 k-最近邻，记为$N*p$；其中的距离最大值记为样本点 p 的 k-距离： $\textrm{k-distance}(p)=\max*{o\in N_p}d(p, o)$；
2. 对于样本点 p 的 k-最近邻，计算每个样本点的到达距离（reach-distance）： $\textrm{reach}(p, o)=\max(\textrm{k-distance}(p),d(p, o)) $；
3. 定义样本点 p 的局部可达性密度（local reachability density, lrd）为： $\textrm{lrd}*p = 1/(\sum*{o\in N_p} \textrm{reach}(p, o) /|N_p|)$；
4. 样本点的 p 的局部异常因子 LOF 可以通过 lrd 来计算：$\textrm{lof}*p = \sum*{o\in N_p}\frac{\textrm{lrd}_o}{\textrm{lrd}_p} / |N_p|$.

需要注意的是，当有大于 k 个样本点具有完全一样的坐标（特征）时，会导致某些点的 lrd 值计算出现除 0 的情况。此时应该在计算中增加个极小的数值来避免出现这种情况。

在判定采样点是否为异常值。原论文建议取 1.5 为阈值， LOF 值大于 1.5 的可以认为是异常点。当然也可以采用其他阈值或者按一定比例进行判定。

使用方式

在使用组件时，k-最近邻的 k 值通过参数 numNeighbors 指定，采样点的坐标（特征）可以通过参数 featureCols 或者参数 vectorCol 指定。 通过参数 distanceType 可以指定采样点间的距离计算方式，默认为欧式距离。

在数据量大时，LOF 算法计算速度会比较慢。此时可以通过参数 maxSampleNumPerGroup，将数据分隔为若干组分别进行计算和判断。

在判定采样点是否为异常点时，可以通过设置参数 outlierThreshold 根据阈值判断，也可以根据数异常点数量（参数 maxOutlierNumPerGroup）、比例（参数 maxOutlierRatio）等来判断。

文献索引

LOF: Identifying Density-Based Local Outliers

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认
predictionCol	预测结果列名	预测结果列名	String	✓		
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE", "INNERPRODUCT", "CITYBLOCK", "JACCARD", "PEARSON"	"EUCLI
featureCols	特征列名数组	特征列名数组，默认全选	String[]		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	null
groupCols	分组列名数组	分组列名，多列，可选，默认不选	String[]			null
maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer			
maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double			
maxSampleNumPerGroup	每组最大样本数目	每组最大样本数目	Integer			

numNeighbors	相邻点个数	构造近邻图使用的相邻点个数	Integer		[1, +inf)	5
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
tensorCol	tensor列	tensor列	String		所选列类型为[BOOL_TENSOR, BYTE_TENSOR, DOUBLE_TENSOR, FLOAT_TENSOR, INT_TENSOR, LONG_TENSOR, STRING, STRING_TENSOR, TENSOR, UBYTE_TENSOR]	null
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
import pandas as pd
df = pd.DataFrame([
[0.73, 0],
[0.24, 0],
[0.63, 0],
[0.55, 0],
[0.73, 0],
[0.41, 0]
])

dataOp = BatchOperator.fromDataframe(df, schemaStr='val double, label int')

outlierOp = LofOutlierBatchOp()\
    .setFeatureCols(["val"])\
    .setOutlierThreshold(3.0)\
    .setPredictionCol("pred")\
    .setPredictionDetailCol("pred_detail")

evalOp = EvalOutlierBatchOp()\
    .setLabelCol("label")\
    .setPredictionDetailCol("pred_detail")\
    .setOutlierValueStrings(["1"])

metrics = dataOp\
    .link(outlierOp)\
    .link(evalOp)\
    .collectMetrics()

print(metrics)
```

Java 代码

```
package examples;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.evaluation.EvalOutlierBatchOp;
import com.alibaba.alink.operator.batch.outlier.LofOutlierBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.common.evaluation.OutlierMetrics;
import org.junit.Test;

public class LofOutlierBatchOpTest {
    @Test
```

```
    public void testLofOutlierBatchOp() throws Exception {
        BatchOperator <?> data = new MemSourceBatchOp(
            new Object[][] {
                {0.73, 0},
                {0.24, 0},
                {0.63, 0},
                {0.55, 0},
                {0.73, 0},
                {0.41, 0},
            },
            new String[] {"val", "label"});

        BatchOperator <?> outlier = new LofOutlierBatchOp()
            .setFeatureCols("val")
            .setOutlierThreshold(3.0)
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail");

        EvalOutlierBatchOp eval = new EvalOutlierBatchOp()
            .setLabelCol("label")
            .setPredictionDetailCol("pred_detail")
            .setOutlierValueStrings("1");

        OutlierMetrics metrics = data
            .link(outlier)
            .link(eval)
            .collectMetrics();

        System.out.println(metrics);
    }
}
```

运行结果

```
-------------------------------- Metrics: --------------------------------
Outlier values: [1]        Normal values: [0]
Auc:NaN    Accuracy:1    Precision:1    Recall:0    F1:0
|Pred\Real|Outlier|Normal|
|---------|-------|------|
|  Outlier|      0|     0|
|   Normal|      0|     6|
```

One-Class SVM异常检测 (OcsvmOutlierBatchOp)

Java 类名：com.alibaba.alink.operator.batch.outlier.OcsvmOutlierBatchOp

Python 类名：OcsvmOutlierBatchOp

功能介绍

与传统SVM不同的是，one-class SVM是一种非监督的学习算法，经常被用来做异常点检测。在该算法的训练集中只有一类positive（或者negative）的数据，而没有（或存在极少量）另外一类，通常称其为异常点。该算法需要学习（learn）的就是边界（boundary），而不是最大间隔（maximum margin），通过边界对异常点进行预测。

参数说明

名称	中文名称	描述	类型	是否必须?
predictionCol	预测结果列名	预测结果列名	String	✓
coef0	Kernel函数的相关参数coef0	Kernel函数的相关参数，只有在POLY和SIGMOID时起作用。	Double	
degree	多项式阶数	多项式的阶数，默认2	Integer	
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double	
featureCols	特征列名数组	特征列名数组，默认全选	String[]	
gamma	Kernel函数的相关参数gamma	Kernel函数的相关参数，只在 RBF, POLY 和 SIGMOID 时起作用. 如果不设置默认取1/d，d为特征维度.	Double	

groupCols	分组列名数组	分组列名，多列，可选，默认不选	String[]	
kernelType	核函数类型	核函数类型，可取为"RBF"，"POLY"，"SIGMOID"，"LINEAR"	String	
maxOutlierNumPerGroup	每组最大异常点数目	每组最大异常点数目	Integer	
maxOutlierRatio	最大异常点比例	算法检测异常点的最大比例	Double	
maxSampleNumPerGroup	每组最大样本数目	每组最大样本数目	Integer	
nu	异常点比例上界参数nu	该参数取值范围是(0,1)，该值与支持向量的数目正向相关。	Double	
outlierThreshold	异常评分阈值	只有评分大于该阈值才会被认为是异常点	Double	
predictionDetailCol	预测详细信息列名	预测详细信息列名	String	
tensorCol	tensor列	tensor列	String	
vectorCol	向量列名	向量列对应的列名，默认值是null	String	
numThreads	组件多线程线程个数	组件多线程线程个数	Integer	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
[0.730967787376657,0.24053641567148587,0.6374174253501083,0.5504370051176339],
[0.7308781907032909,0.41008081149220166,0.20771484130971707,0.3327170559595112]
,
[0.7311469360199058,0.9014476240300544,0.49682259343089075,0.9858769332362016],
[0.731057369148862,0.07099203475193139,0.06712000939049956,0.768156984078079],
[0.7306094602878371,0.9187140138555101,0.9186071189908658,0.6795571637816596],
[0.730519863614471,0.08825840967622589,0.4889045498516358,0.461837214623537],
[0.7307886238322471,0.5796252073129174,0.7780122870716483,0.1149970919002733],
[0.7306990420600421,0.7491696031336331,0.34830970303125697,0.8972771427421047]]
)

# load data
data = BatchOperator.fromDataframe(df, schemaStr="x1 double, x2 double, x3
double, x4 double")

OcsvmOutlierBatchOp() \
            .setFeatureCols(["x1", "x2", "x3", "x4"]) \
            .setGamma(0.5) \
            .setNu(0.1) \
            .setKernelType("RBF") \
            .setPredictionCol("pred").linkFrom(data).print();
```

Java 代码

```
package com.alibaba.alink.operator.batch.outlier;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

public class OcsvmBatchOpTest {
    @Test
    public void testOutlier() throws Exception {

        BatchOperator <?> data = new MemSourceBatchOp(
            new Object[][] {
```

```
                {0.730967787376657, 0.24053641567148587, 0.6374174253501083,
0.5504370051176339},
                {0.7308781907032909, 0.41008081149220166, 0.20771484130971707,
0.3327170559595112},
                {0.7311469360199058, 0.9014476240300544, 0.49682259343089075,
0.9858769332362016},
                {0.731057369148862, 0.07099203475193139, 0.06712000939049956,
0.768156984078079},
                {0.7306094602878371, 0.9187140138555101, 0.9186071189908658,
0.6795571637816596},
                {0.730519863614471, 0.08825840967622589, 0.4889045498516358,
0.461837214623537},
                {0.7307886238322471, 0.5796252073129174, 0.7780122870716483,
0.11499709190022733},
                {0.7306990420600421, 0.7491696031336331, 0.34830970303125697,
0.8972771427421047}
            },
            new String[] {"x1", "x2", "x3", "x4"});

        new OcsvmOutlierBatchOp()
            .setFeatureCols("x1", "x2", "x3", "x4")
            .setGamma(0.5)
            .setNu(0.2)
            .setKernelType("RBF")
            .setPredictionCol("pred").linkFrom(data).print();
    }
}
```

运行结果

x1	x2	x3	x4	pred
0.7310	0.2405	0.6374	0.5504	false
0.7309	0.4101	0.2077	0.3327	false
0.7311	0.9014	0.4968	0.9859	false
0.7311	0.0710	0.0671	0.7682	false
0.7306	0.9187	0.9186	0.6796	true
0.7305	0.0883	0.4889	0.4618	false
0.7308	0.5796	0.7780	0.1150	false
0.7307	0.7492	0.3483	0.8973	false

分组语义向量距离 (CrossGroupSemanticVectorDistanceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.CrossGroupSemanticVectorDistanceBatchOp

Python 类名：CrossGroupSemanticVectorDistanceBatchOp

功能介绍

在对应分组内计算查找向量的最近邻。 需要指定两个输入，输入中包含分组列，数据ID列和向量。 如果是稠密格式的向量，需要保证两个向量长度相等。支持 EUCLIDEAN、COSINE、INNERPRODUCT（内积）、CITYBLOCK（曼哈顿距离）、JACCARD、PEARSON 六种距离计算方法。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
idCol	Id列名	用来区分不同的样本	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE", "INNERPRODUCT", "CITYBLOCK", "JACCARD", "PEARSON"	"EUCLIDEAN"
groupCol	分组单列名	分组单列名，可选	String			null

topN	前N的数据	挑选最近的N个数据	Integer		[1, +inf)	5

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
                [1,1,"16.3, 1.1, 1.1"],
                [1,2,"16.8, 1.4, 1.5"],
                [1,3,"19.2, 1.7, 1.8"],
                [1,4,"10.0, 1.7, 1.7"],
                [2,5,"19.5, 1.8, 1.9"],
                [2,6,"20.9, 1.8, 1.8"],
                [2,7,"21.1, 1.9, 1.8"],
                [2,8,"20.9, 2.0, 2.1"],
                [1,9,"20.3, 2.3, 2.4"],
                [1,10,"22.0, 2.4, 2.5"]
        ])

inOp = BatchOperator.fromDataframe(df, schemaStr='class int, id int, vec
string')

op =
CrossGroupSemanticVectorDistanceBatchOp().setGroupCol("class").setVectorCol("ve
c").setIdCol("id").setDistanceType("euclidean").setTopN(3)
op.linkFrom(inOp, inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;
```

```
import java.util.Arrays;
import java.util.List;

public class CrossGroupSemanticVectorDistanceBatchOpTest {
    @Test
    public void testCrossGroup() throws Exception {
        List <Row> list = Arrays.asList(
            Row.of(1,1,"16.3, 1.1, 1.1"),
            Row.of(1,2,"16.8, 1.4, 1.5"),
            Row.of(1,3,"19.2, 1.7, 1.8"),
            Row.of(1,4,"10.0, 1.7, 1.7"),
            Row.of(2,5,"19.5, 1.8, 1.9"),
            Row.of(2,6,"20.9, 1.8, 1.8"),
            Row.of(2,7,"21.1, 1.9, 1.8"),
            Row.of(2,8,"20.9, 2.0, 2.1"),
            Row.of(1,9,"20.3, 2.3, 2.4"),
            Row.of(1,10,"22.0, 2.4, 2.5")
        );
        BatchOperator inOp = new MemSourceBatchOp(list, "class int, id int, vec
string");

        new CrossGroupSemanticVectorDistanceBatchOp()
            .setGroupCol("class")
            .setVectorCol("vec")
            .setIdCol("id")
            .setDistanceType("euclidean")
            .setTopN(3)
            .linkFrom(inOp, inOp).print();
    }
}
```

运行结果

origin_group_id	original_id	near_group_id	near_id	distance	rank
1	9	1	10	1.7059	3
1	9	1	3	1.3892	2
1	9	1	9	0.0000	1
1	3	1	2	2.4372	3
1	3	1	9	1.3892	2
1	3	1	3	0.0000	1
1	2	1	3	2.4372	3
1	2	1	1	0.7071	2

1	2	1	2	0.0000	1
2	6	2	8	0.3606	3
2	6	2	7	0.2236	2
2	6	2	6	0.0000	1
2	7	2	8	0.3742	3
2	7	2	6	0.2236	2
2	7	2	7	0.0000	1
1	4	1	2	6.8096	3
1	4	1	1	6.3569	2
1	4	1	4	0.0000	1
2	8	2	7	0.3742	3
2	8	2	6	0.3606	2
2	8	2	8	0.0000	1
1	10	1	3	2.9698	3
1	10	1	9	1.7059	2
1	10	1	10	0.0000	1
1	1	1	3	3.0430	3
1	1	1	2	0.7071	2
1	1	1	1	0.0000	1
2	5	2	8	1.4283	3
2	5	2	6	1.4036	2
2	5	2	5	0.0000	1

huge语义向量距离 (HugeSemanticVectorDistanceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.HugeSemanticVectorDistanceBatchOp

Python 类名：HugeSemanticVectorDistanceBatchOp

功能介绍

基于算法语义向量结果（如word2vec生成的词向量），计算给定的词（或者句子）的扩展词（或者句子）， 即计算其中某一向量距离最近的向量集合。其中一个用法是：基于word2vec生成的词向量结果，根据输入的词返回最为相似的词列表。

支持 EUCLIDEAN、COSINE、INNERPRODUCT（内积）、CITYBLOCK（曼哈顿距离）、JACCARD、PEARSON 六种距离计算方法。

算法说明

该算法是一个O(N^2)复杂度的算法，计算量非常大，下面给出计算规模与计算时间的估计：如果左右样本集合都是100万样本， 100维特征，使用100个worker计算该规模的样本语义向量距离计算时间估计约为20个小时左右。

left_num：左边数据集（字典集合）的样本数 right_num：右边数据集（待查询的集合）的样本数 feature_dim：样本的特征为度

计算时间与上面三个量成正比： time = O(left_num *right_num* feature_dim)

如果业务数据量特别大，或者特征维度特别大，建议结合业需求，采用下面的方式减少计算时间：

- 建议1：降低特征维度
- 建议2：降低样本数（左集合，或右集合的样本数目）
- 建议3：如果样本数和特征维度不能降低，建议将右边的集合拆成多份，分几次来求top K紧邻，这个效果和合并起来是一样的，只是拆分任务，让每一次运行的时间减少，防止机器或环境的不稳定引起不必要的计算浪费。
- 建议4：当字典集合 和 待查询集合 的数据量存在很大差异，量级的差异，可以通过参数 isRotateDict 来减少通信量。如果字典集合小，isRotateDict 设置为true，反之 设置为false，默认为false，目前该参数设置只在D2模式下可用。

与正常版本的区别

本算法为为Huge版本语义向量距离，与 SemanticVectorDistanceBatchOp 的主要区别是：

- Huge版可支持模型比较大的情形，当模型数据大于2G的时候，建议使用。（HugeSemanticVectorDistanceBatchOp）
- 正常的版本通信量相对于Huge版更小，因而稳定性和效率在模型不大的情况下会更优，当模型小于2G的时候建议使用。（SemanticVectorDistanceBatchOp）

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
idCol	Id列名	用来区分不同的样本	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE", "INNERPRODUCT", "CITYBLOCK", "JACCARD", "PEARSON"	"EUCLIDEAN"
groupIdCol	分组Id列名	支持做分组的语义向量距离	String			null
topN	前N的数据	挑选最近的N个数据	Integer		[1, +inf)	5

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ["id_1", "2.0,3.0"],
```

```
        ["id_2", "2.1,3.1"],
        ["id_3", "200.1,300.1"],
        ["id_4", "200.2,300.2"],
        ["id_5", "200.3,300.3"],
        ["id_6", "200.4,300.4"],
        ["id_7", "200.5,300.5"],
        ["id_8", "200.6,300.6"],
        ["id_9", "2.1,3.1"],
        ["id_10", "2.1,3.1"],
        ["id_11", "2.1,3.1"],
        ["id_12", "2.1,3.1"],
        ["id_16", "300.,3.2"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id string, vec string')
op =
HugeSemanticVectorDistanceBatchOp().setVectorCol("vec").setIdCol("id").setDista
nceType("euclidean").setTopN(3).linkFrom(inOp1, inOp1)
op.print()
```

Java代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

/**
 * Test for HugeSemanticVectorDistanceBatchOp
 */
public class HugeSemanticVectorDistanceBatchOpTest extends AlinkTestBase {

    @Test
    public void testHugeSemanticVectorDistanceBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("id_1", "2.0,3.0"),
            Row.of("id_2", "2.1,3.1"),
            Row.of("id_3", "200.1,300.1"),
            Row.of("id_4", "200.2,300.2"),
            Row.of("id_5", "200.3,300.3"),
            Row.of("id_6", "200.4,300.4"),
            Row.of("id_7", "200.5,300.5"),
            Row.of("id_8", "200.6,300.6"),
```

```
            Row.of("id_9", "2.1,3.1"),
            Row.of("id_10", "2.1,3.1"),
            Row.of("id_11", "2.1,3.1"),
            Row.of("id_12", "2.1,3.1"),
            Row.of("id_16", "300.,3.2"));

        MemSourceBatchOp data = new MemSourceBatchOp(df, "id string,vec
string");
        HugeSemanticVectorDistanceBatchOp op = new
HugeSemanticVectorDistanceBatchOp()
            .setVectorCol("vec")
            .setIdCol("id")
            .setDistanceType("euclidean")
            .setTopN(3)
            .linkFrom(data, data);
        op.print();
    }
}
```

运行结果

original_id	near_id	distance	rank
id_1	id_9	0.1414	3
id_1	id_11	0.1414	2
id_1	id_1	0.0000	1
id_3	id_5	0.2828	3
id_3	id_4	0.1414	2
id_3	id_3	0.0000	1
id_5	id_6	0.1414	3
id_5	id_4	0.1414	2
id_5	id_5	0.0000	1
id_7	id_6	0.1414	3
id_6	id_6	0.0000	1
id_8	id_6	0.2828	3
id_8	id_7	0.1414	2
id_8	id_8	0.0000	1
id_10	id_10	0.0000	3

id_10	id_12	0.0000	2
id_10	id_2	0.0000	1
id_12	id_10	0.0000	3
id_12	id_12	0.0000	2
id_12	id_2	0.0000	1

- 以上未显示全部结果

语义向量距离 (SemanticVectorDistanceBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.SemanticVectorDistanceBatchOp

Python 类名：SemanticVectorDistanceBatchOp

功能介绍

基于算法语义向量结果（如word2vec生成的词向量），计算给定的词（或者句子）的扩展词（或者句子）， 即计算其中某一向量距离最近的向量集合。其中一个用法是：基于word2vec生成的词向量结果，根据输入的词返回最为相似的词列表。 支持 EUCLIDEAN、COSINE、INNERPRODUCT（内积）、CITYBLOCK（曼哈顿距离）、JACCARD、PEARSON 六种距离计算方法。

补充信息

该算法是一个O(N^2)复杂度的算法，计算量非常大，下面给出计算规模与计算时间的估计：如果左右样本集合都是100万样本， 100维特征，使用100个worker计算该规模的样本语义向量距离计算时间估计约为20个小时左右。

left_num：左边数据集（字典集合）的样本数 right_num：右边数据集（待查询的集合）的样本数 feature_dim：样本的特征为度

计算时间与上面三个量成正比： time = O(left_num *right_num* feature_dim)

如果业务数据量特别大，或者特征维度特别大，建议结合业需求，采用下面的方式减少计算时间：

- 建议1：降低特征维度
- 建议2：降低样本数（左集合，或右集合的样本数目）
- 建议3：如果样本数和特征维度不能降低，建议将右边的集合拆成多份，分几次来求top K紧邻，这个效果和合并起来是一样的，只是拆分任务，让每一次运行的时间减少，防止机器或环境的不稳定引起不必要的计算浪费。
- 建议4：当字典集合 和 待查询集合 的数据量存在很大差异，量级的差异，可以通过参数 isRotateDict 来减少通信量。如果字典集合小，isRotateDict 设置为true，反之 设置为false，默认为false，目前该参数设置只在D2模式下可用。

该算法目前存在两个版本，一个为语义向量距离，一个为语义向量距离-Huge版，这两个版本之间的区别是：

Huge版可支持模型比较大的情形，当模型数据大于2G的时候，建议使用。（HugeSemanticVectorDistanceBatchOp）

正常的版本通信量相对于Huge版更小，因而稳定性和效率在模型不大的情况下会更优，当模型小于2G的时候建议使用。（SemanticVectorDistanceBatchOp）

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
idCol	Id列名	用来区分不同的样本	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE", "INNERPRODUCT", "CITYBLOCK", "JACCARD", "PEARSON"	"EUCLIDEAN"
topN	前N的数据	挑选最近的N个数据	Integer		[1, +inf)	5

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        ["id_1", "2.0,3.0"],
        ["id_2", "2.1,3.1"],
        ["id_3", "200.1,300.1"],
        ["id_4", "200.2,300.2"],
        ["id_5", "200.3,300.3"],
        ["id_6", "200.4,300.4"],
        ["id_7", "200.5,300.5"],
        ["id_8", "200.6,300.6"],
```

```
        ["id_9", "2.1,3.1"],
        ["id_10", "2.1,3.1"],
        ["id_11", "2.1,3.1"],
        ["id_12", "2.1,3.1"],
        ["id_16", "300.,3.2"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id string, vec string')
op =
SemanticVectorDistanceBatchOp().setVectorCol("vec").setIdCol("id").setDistanceT
ype("euclidean").setTopN(3).linkFrom(inOp1, inOp1)
op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

/**
 * Test for SemanticVectorDistance
 */
public class SemanticVectorDistanceBatchOpTest extends AlinkTestBase {
    @Test
    public void test() throws Exception {
        List <Row> rows = Arrays.asList(
            Row.of("id_1", "2.0,3.0"),
            Row.of("id_2", "2.1,3.1"),
            Row.of("id_3", "200.1,300.1"),
            Row.of("id_4", "200.2,300.2"),
            Row.of("id_5", "200.3,300.3"),
            Row.of("id_6", "200.4,300.4"),
            Row.of("id_7", "200.5,300.5"),
            Row.of("id_8", "200.6,300.6"),
            Row.of("id_9", "2.1,3.1"),
            Row.of("id_10", "2.1,3.1"),
            Row.of("id_11", "2.1,3.1"),
            Row.of("id_12", "2.1,3.1"),
            Row.of("id_16", "300.,3.2")
        );
        BatchOperator data = new MemSourceBatchOp(rows, "id string,vec
```

```
string");
        SemanticVectorDistanceBatchOp op = new SemanticVectorDistanceBatchOp()
            .setVectorCol("vec")
            .setIdCol("id")
            .setDistanceType("euclidean")
            .setTopN(3)
            .linkFrom(data, data);
        op.print();
    }
}
```

运行结果

original_id	near_id	distance	rank
id_1	id_2	0.1414	3
id_1	id_9	0.1414	2
id_1	id_1	0.0000	1
id_3	id_5	0.2828	3
id_3	id_4	0.1414	2
id_3	id_3	0.0000	1
id_5	id_6	0.1414	3
id_5	id_4	0.1414	2
id_5	id_5	0.0000	1
id_7	id_6	0.1414	3
id_6	id_6	0.0000	1
id_8	id_6	0.2828	3
id_8	id_7	0.1414	2
id_8	id_8	0.0000	1
id_10	id_2	0.0000	3
id_10	id_9	0.0000	2
id_10	id_10	0.0000	1
id_12	id_2	0.0000	3
id_12	id_9	0.0000	2
id_12	id_10	0.0000	1

注意：计算结果没有在表格中全部列出。

simhash (SimHashBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.SimHashBatchOp

Python 类名：SimHashBatchOp

功能介绍

SimHash是快速计算两个集合有多相似所用的技术
 具体细节可以参考：
https://en.wikipedia.org/wiki/SimHash

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
base	输出的进制表示	输出的进制表示	Integer			10
bitNum	Hamming码长度	Hamming码长度	Integer			64
outputCol	输出列名	输出列名	String			"hash_value"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, 'That is an English Book!'],
    [1, 'Do you like math?'],
    [2, 'Have a good day!']
```

```
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id long, text string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id long, text string')

op = SimHashBatchOp().setSelectedCol("text")
print(BatchOperator.collectToDataframe(op.linkFrom(inOp1)))

op = SimHashStreamOp().setSelectedCol("text")
op.linkFrom(inOp2).print()
StreamOperator.execute()
```

运行结果

```
   id                  text
0   0  11542653382991809927
1   1   6817740987335049141
2   2   4007439417379517431
```

simrank (SimrankBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.SimrankBatchOp

Python 类名：SimrankBatchOp

功能介绍

SimRank 是一种基于图的拓扑结构信息来衡量任意两个对象间相似程度的模型。 SimRank的核心思想为：如果两个对象和被相似的对象所引用，那么这两个对象也相似。 SimRank++ 是SimRank的拓展，它在模型中引入了边的权重。 本算法组件计算二部图中两部分顶点的 Simrank++ 相似度。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
itemCol	Item列列名	Item列列名	String	✓	
userCol	User列列名	User列列名	String	✓	
decayFactor	衰减因子	衰减因子	Double		
modelType	simrank模型类型	simrank模型类型，"simrank"或"simrankplusplus"	String		"SIMRANK", "SIMRANKPLUSPLUS
numWalks	随机游走次数	随机游走次数	Integer		
topK	topK	表示取前topK个相似的item进行输出	Integer		
walkLength	随机游走长度	随机游走长度	Integer		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
  ["pc", "Hp.com"],
  ["camera", "Hp.com"],
  ["Digital camera", "Hp.com"],
  ["camera", "Bestbuy.com"],
  ["Digital camera", "Bestbuy.com"],
  ["tv", "Bestbuy.com"],
  ["flower", "Teleflora.com"],
  ["flower", "Orchids.com"],
])



batch_data = BatchOperator.fromDataframe(df, schemaStr="product string, website
string")


simrank = SimrankBatchOp().setUserCol("website") \
  .setItemCol("product").setModelType("simrank").setNumWalks(1000)

output = simrank.linkFrom(batch_data)
output.print()
```

脚本运行结果

product	similar	score
Digital camera	pc: camera :tv	0.63016176:0.61392087:0.60078394
camera	pc:Digital camera:tv	0.62772226:0.61392087:0.60322344
pc	Digital camera: camera :tv	0.63016176:0.62772226:0.43094572
tv	camera:Digital camera:pc	0.60322344:0.60078394:0.43094572

字符串近似最近邻预测 (StringApproxNearestNeighborPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.StringApproxNearestNeighborPredictBatchOp

Python 类名：StringApproxNearestNeighborPredictBatchOp

功能介绍

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

SimhashHamming（SimHash_Hamming_Distance)相似度=1-距离/64.0，应选择metric的参数为SIMHASH_HAMMING_SIM。

MinHash应选择metric的参数为MINHASH_SIM。

Jaccard应选择metric的参数为JACCARD_SIM。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
radius	radius值	radius值	Double			null
reservedCols	算法保留列名	算法保留列	String[]			null
topN	TopN的值	TopN的值	Integer		[1, +inf)	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "abcde", "aabce"],
    [1, "aacedw", "aabbed"],
    [2, "cdefa", "bbcefa"],
    [3, "bdefh", "ddeac"],
    [4, "acedm", "aeefbc"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')

train =
StringApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1"
).setMetric("SIMHASH_HAMMING_SIM").linkFrom(inOp)
predict =
StringApproxNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3).
linkFrom(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.StringApproxNearestNeighborPredictB
atchOp;
import
com.alibaba.alink.operator.batch.similarity.StringApproxNearestNeighborTrainBat
chOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StringApproxNearestNeighborPredictBatchOpTest {
    @Test
    public void testStringApproxNearestNeighborPredictBatchOp() throws
Exception {
        List <Row> df = Arrays.asList(
```

```
            Row.of(0, "abcde", "aabce"),
            Row.of(1, "aacedw", "aabbed"),
            Row.of(2, "cdefa", "bbcefa"),
            Row.of(3, "bdefh", "ddeac"),
            Row.of(4, "acedm", "aeefbc")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        BatchOperator <?> train = new
StringApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1"
)
            .setMetric("SIMHASH_HAMMING_SIM").linkFrom(inOp);
        BatchOperator <?> predict = new
StringApproxNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3)
            .linkFrom(train, inOp);
        predict.print();
    }
}
```

运行结果

id	text1	text2
0	abcde	{"ID":"[0,1,2]","METRIC":"[0.953125,0.921875,0...
1	aacedw	{"ID":"[0,1,4]","METRIC":"[0.9375,0.90625,0.85...
2	cdefa	{"ID":"[0,1,4]","METRIC":"[0.890625,0.859375,0...
3	bdefh	{"ID":"[4,2,1]","METRIC":"[0.9375,0.90625,0.89...
4	acedm	{"ID":"[1,0,4]","METRIC":"[0.921875,0.921875,0...

字符串近似最近邻训练 (StringApproxNearestNeighborTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.StringApproxNearestNeighborTrainBatchOp

Python 类名：StringApproxNearestNeighborTrainBatchOp

功能介绍

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

SimhashHamming（SimHash_Hamming_Distance)相似度=1-距离/64.0，应选择metric的参数为SIMHASH_HAMMING_SIM。

MinHash应选择metric的参数为MINHASH_SIM。

Jaccard应选择metric的参数为JACCARD_SIM。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
idCol	id列名	id列名	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		

metric	距离类型	用于计算的距离类型	String		"SIMHASH_HAMMING_SIM", "SIMHASH_HAMMING", "MINHASH_JACCARD_SIM", "JACCARD_SIM"	"SIMHASH_HAM
numBucket	分桶个数	分桶个数	Integer			10
numHashTables	哈希表个数	哈希表的数目	Integer			10
seed	采样种子	采样种子	Long			0

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "abcde", "aabce"],
    [1, "aacedw", "aabbed"],
    [2, "cdefa", "bbcefa"],
    [3, "bdefh", "ddeac"],
    [4, "acedm", "aeefbc"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')
```

```
train =
StringApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1"
).setMetric("SIMHASH_HAMMING_SIM").linkFrom(inOp)
predict =
StringApproxNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3).
linkFrom(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.StringApproxNearestNeighborPredictB
atchOp;
import
com.alibaba.alink.operator.batch.similarity.StringApproxNearestNeighborTrainBat
chOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StringApproxNearestNeighborTrainBatchOpTest {
    @Test
    public void testStringApproxNearestNeighborTrainBatchOp() throws Exception
{
        List <Row> df = Arrays.asList(
            Row.of(0, "abcde", "aabce"),
            Row.of(1, "aacedw", "aabbed"),
            Row.of(2, "cdefa", "bbcefa"),
            Row.of(3, "bdefh", "ddeac"),
            Row.of(4, "acedm", "aeefbc")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        BatchOperator <?> train = new
StringApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1"
)
            .setMetric("SIMHASH_HAMMING_SIM").linkFrom(inOp);
        BatchOperator <?> predict = new
StringApproxNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3)
            .linkFrom(train, inOp);
        predict.print();
```

```
    }
}
```

运行结果

id	text1	text2
0	abcde	{"ID":"[0,1,2]","METRIC":"[0.953125,0.921875,0...
1	aacedw	{"ID":"[0,1,4]","METRIC":"[0.9375,0.90625,0.85...
2	cdefa	{"ID":"[0,1,4]","METRIC":"[0.890625,0.859375,0...
3	bdefh	{"ID":"[4,2,1]","METRIC":"[0.9375,0.90625,0.89...
4	acedm	{"ID":"[1,0,4]","METRIC":"[0.921875,0.921875,0...

字符串最近邻预测 (StringNearestNeighborPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.StringNearestNeighborPredictBatchOp

Python 类名：StringNearestNeighborPredictBatchOp

功能介绍

本算法支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine四种相似度精确计算方式，通过选择metric参数可计算不同的相似度。

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

Levenshtein（Levenshtein Distance), 相似度=(1-距离)/length，length为两个字符长度的最大值，应选metric的参数为LEVENSHTEIN_SIM。

LCS（Longest Common SubString), 相似度=(1-距离)/length，length为两个字符长度的最大值，应选择metric的参数为LCS_SIM。

SSK（String Subsequence Kernel）支持相似度计算，应选择metric的参数为SSK。

Cosine（Cosine）支持相似度计算，应选择metric的参数为COSINE。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
radius	radius值	radius值	Double			null
reservedCols	算法保留列名	算法保留列	String[]			null
topN	TopN的值	TopN的值	Integer		[1, +inf)	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "abcde", "aabce"],
    [1, "aacedw", "aabbed"],
    [2, "cdefa", "bbcefa"],
    [3, "bdefh", "ddeac"],
    [4, "acedm", "aeefbc"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')

train =
StringNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1").setM
etric("LEVENSHTEIN_SIM").linkFrom(inOp)
predict =
StringNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3).linkFr
om(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.StringNearestNeighborPredictBatchOp
;
import
com.alibaba.alink.operator.batch.similarity.StringNearestNeighborTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StringNearestNeighborPredictBatchOpTest {
    @Test
```

```
    public void testStringNearestNeighborPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "abcde", "aabce"),
            Row.of(1, "aacedw", "aabbed"),
            Row.of(2, "cdefa", "bbcefa"),
            Row.of(3, "bdefh", "ddeac"),
            Row.of(4, "acedm", "aeefbc")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        BatchOperator <?> train = new
StringNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1")
            .setMetric("LEVENSHTEIN_SIM").linkFrom(inOp);
        BatchOperator <?> predict = new
StringNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3)
            .linkFrom(train, inOp);
        predict.print();
    }
}
```

运行结果

id	text1	text2
0	abcde	{"ID":"[0,1,4]","METRIC":"[0.6,0.5,0.19999999999999996]"}
1	aacedw	{"ID":"[1,0,4]","METRIC":"[0.5,0.33333333333333337,0.33333333333333337]"}
2	cdefa	{"ID":"[2,3,1]","METRIC":"[0.5,0.5,0.33333333333333337]"}
3	bdefh	{"ID":"[2,3,4]","METRIC":"[0.4,0.4,0.19999999999999996]"}
4	acedm	{"ID":"[4,3,2]","METRIC":"[0.33333333333333337,0.33333333333333337,0.33333333333333337]"}

字符串最近邻训练 (StringNearestNeighborTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.StringNearestNeighborTrainBatchOp

Python 类名：StringNearestNeighborTrainBatchOp

功能介绍

本算法支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine四种相似度精确计算方式，通过选择metric参数可计算不同的相似度。

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

Levenshtein（Levenshtein Distance), 相似度=(1-距离)/length，length为两个字符长度的最大值，应选metric的参数为LEVENSHTEIN_SIM。

LCS（Longest Common SubString), 相似度=(1-距离)/length，length为两个字符长度的最大值，应选择metric的参数为LCS_SIM。

SSK（String Subsequence Kernel）支持相似度计算，应选择metric的参数为SSK。

Cosine（Cosine）支持相似度计算，应选择metric的参数为COSINE。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
idCol	id列名	id列名	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	

lambda	匹配字符权重	匹配字符权重，SSK中使用	Double			0.5
metric	距离类型	用于计算的距离类型	String		"LEVENSHTEIN_SIM", "LEVENSHTEIN", "LCS_SIM", "LCS", "SSK", "COSINE"	"LEVENSHTEIN_SIM"
windowSize	窗口大小	窗口大小	Integer		[1, +inf)	2

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "abcde", "aabce"],
    [1, "aacedw", "aabbed"],
    [2, "cdefa", "bbcefa"],
    [3, "bdefh", "ddeac"],
    [4, "acedm", "aeefbc"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')

train =
StringNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1").setM
etric("LEVENSHTEIN_SIM").linkFrom(inOp)
predict =
StringNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3).linkFr
om(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.StringNearestNeighborPredictBatchOp
;
import
com.alibaba.alink.operator.batch.similarity.StringNearestNeighborTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StringNearestNeighborTrainBatchOpTest {
    @Test
    public void testStringNearestNeighborTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "abcde", "aabce"),
            Row.of(1, "aacedw", "aabbed"),
            Row.of(2, "cdefa", "bbcefa"),
            Row.of(3, "bdefh", "ddeac"),
            Row.of(4, "acedm", "aeefbc")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        BatchOperator <?> train = new
StringNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1")
            .setMetric("LEVENSHTEIN_SIM").linkFrom(inOp);
        BatchOperator <?> predict = new
StringNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3)
            .linkFrom(train, inOp);
        predict.print();
    }
}
```

运行结果

id	text1	text2
0	abcde	{"ID":"[0,1,4]","METRIC":"[0.6,0.5,0.19999999999999996]"}
1	aacedw	{"ID":"[1,0,4]","METRIC":"[0.5,0.33333333333333337,0.33333333333333337]"}
2	cdefa	{"ID":"[2,3,1]","METRIC":"[0.5,0.5,0.33333333333333337]"}

3	bdefh	{"ID":"[2,3,4]","METRIC":"[0.4,0.4,0.19999999999999996]"}
4	acedm	{"ID":"[4,3,2]","METRIC":" [0.33333333333333337,0.33333333333333337,0.33333333333333337]"}

字符串两两相似度计算 (StringSimilarityPairwiseBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.StringSimilarityPairwiseBatchOp

Python 类名：StringSimilarityPairwiseBatchOp

功能介绍

字符相似度是计算两两文章或者句子之间的相似度： 支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine，SimHashHamming，MinHash和Jaccard七种相似度计算方式，通过选择metric参数可计算不同的相似度。

Levenshtein（Levenshtein Distance）支持距离和相似度两种方式，相似度=(1-距离)/length，length为两个字符长度的最大值，距离应选择metric的参数为LEVENSHTEIN，相似度应选metric的参数为LEVENSHTEIN_SIM。

LCS（Longest Common SubString）支持距离和相似度两种参数，相似度=(1-距离)/length，length为两个字符长度的最大值，距离应选择metric的参数为LCS，相似度应选择metric的参数为LCS_SIM。

SSK（String Subsequence Kernel）支持相似度计算，应选择metric的参数为SSK。

Cosine（Cosine）支持相似度计算，应选择metric的参数为COSINE。

SimhashHamming（SimHash_Hamming_Distance），支持距离和相似度两种方式，相似度=1-距离/64.0，距离应选择metric的参数为SIMHASH_HAMMING，相似度应选择metric的参数为SIMHASH_HAMMING_SIM。

MinHash 支持相似度计算，应选择metric的参数为MINHASH_SIM。

Jaccard 支持相似度计算，应选择metric的参数为JACCARD_SIM。

Alink上字符相似度算法包括Batch组件和Stream组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		

selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为 [STRING]	
lambda	匹配字符权重	匹配字符权重，SSK中使用	Double			0.5
metric	度量类型	计算距离时，可以取不同的度量	String		"LEVENSHTEIN", "LEVENSHTEIN_SIM", "LCS", "LCS_SIM", "SSK", "COSINE", "SIMHASH_HAMMING", "SIMHASH_HAMMING_SIM", "JACCARD_SIM"	"LEVENSHTEI
numBucket	分桶个数	分桶个数	Integer			10
numHashTables	哈希表个数	哈希表的数目	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
seed	采样种子	采样种子	Long			0
windowSize	窗口大小	窗口大小	Integer		[1, +inf)	2

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "abcde", "aabce"],
    [1, "aacedw", "aabbed"],
    [2, "cdefa", "bbcefa"],
    [3, "bdefh", "ddeac"],
    [4, "acedm", "aeefbc"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id long, text1 string,
text2 string')

batchOp = StringSimilarityPairwiseBatchOp().setSelectedCols(["text1",
"text2"]).setMetric("LEVENSHTEIN").setOutputCol("LEVENSHTEIN")
batchOp.linkFrom(inOp1).print()

streamOp = StringSimilarityPairwiseStreamOp().setSelectedCols(["text1",
"text2"]).setMetric("COSINE").setOutputCol("COSINE")
streamOp.linkFrom(inOp2).print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;
```

```
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.StringSimilarityPairwiseBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.similarity.StringSimilarityPairwiseStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StringSimilarityPairwiseBatchOpTest {
    @Test
    public void testStringSimilarityPairwiseBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "abcde", "aabce"),
            Row.of(1, "aacedw", "aabbed"),
            Row.of(2, "cdefa", "bbcefa"),
            Row.of(3, "bdefh", "ddeac"),
            Row.of(4, "acedm", "aeefbc")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text1
string, text2 string");
        BatchOperator <?> batchOp = new
StringSimilarityPairwiseBatchOp().setSelectedCols("text1", "text2").setMetric(
            "LEVENSHTEIN").setOutputCol("LEVENSHTEIN");
        batchOp.linkFrom(inOp1).print();
        StreamOperator <?> streamOp = new
StringSimilarityPairwiseStreamOp().setSelectedCols("text1", "text2")
            .setMetric("COSINE").setOutputCol("COSINE");
        streamOp.linkFrom(inOp2).print();
        StreamOperator.execute();
    }
}
```

运行结果

id	text1	text2	LEVENSHTEIN
0	abcde	aabce	2.0000
1	aacedw	aabbed	3.0000
2	cdefa	bbcefa	3.0000

3	bdefh	ddeac	3.0000
4	acedm	aeefbc	4.0000

id	text1	text2	COSINE
1	aacedw	aabbed	0.4000
4	acedm	aeefbc	0.0000
0	abcde	aabce	0.5000
2	cdefa	bbcefa	0.4472
3	bdefh	ddeac	0.2500

文本近似最近邻预测 (TextApproxNearestNeighborPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.TextApproxNearestNeighborPredictBatchOp

Python 类名：TextApproxNearestNeighborPredictBatchOp

功能介绍

文本相似度是在字符串相似度的基础上，基于词，计算两两文章或者句子之间的相似度，文章或者句子需要以空格分割的文本，计算方式和字符串相似度类似：支持SimHashHamming，MinHash和Jaccard三种近似相似度计算方式，通过选择metric参数可计算不同的相似度。

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

SimhashHamming（SimHash_Hamming_Distance)相似度=1-距离/64.0，应选择metric的参数为SIMHASH_HAMMING_SIM。

MinHash应选择metric的参数为MINHASH_SIM。

Jaccard应选择metric的参数为JACCARD_SIM。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
radius	radius值	radius值	Double			null
reservedCols	算法保留列名	算法保留列	String[]			null
topN	TopN的值	TopN的值	Integer		[1, +inf)	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "a b c d e", "a a b c e"],
    [1, "a a c e d w", "a a b b e d"],
    [2, "c d e f a", "b b c e f a"],
    [3, "b d e f h", "d d e a c"],
    [4, "a c e d m", "a e e f b c"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')

train =
TextApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1").
setMetric("SIMHASH_HAMMING_SIM").linkFrom(inOp)
predict =
TextApproxNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3).li
nkFrom(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.TextApproxNearestNeighborPredictBat
chOp;
import
com.alibaba.alink.operator.batch.similarity.TextApproxNearestNeighborTrainBatch
Op;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TextApproxNearestNeighborPredictBatchOpTest {
    @Test
    public void testTextApproxNearestNeighborPredictBatchOp() throws Exception
```

```
{
        List <Row> df = Arrays.asList(
            Row.of(0, "a b c d e", "a a b c e"),
            Row.of(1, "a a c e d w", "a a b b e d"),
            Row.of(2, "c d e f a", "b b c e f a"),
            Row.of(3, "b d e f h", "d d e a c"),
            Row.of(4, "a c e d m", "a e e f b c")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        BatchOperator <?> train = new
TextApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1")
            .setMetric("SIMHASH_HAMMING_SIM").linkFrom(inOp);
        BatchOperator <?> predict = new
TextApproxNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3)
            .linkFrom(train, inOp);
        predict.print();
    }
}
```

运行结果

id	text1	text2
0	a b c d e	{"ID":"[0,1,2]","METRIC":"[0.953125,0.921875,0.90625]"}
1	a a c e d w	{"ID":"[0,1,4]","METRIC":"[0.9375,0.90625,0.859375]"}
2	c d e f a	{"ID":"[0,1,4]","METRIC":"[0.890625,0.859375,0.8125]"}
3	b d e f h	{"ID":"[4,2,1]","METRIC":"[0.9375,0.90625,0.890625]"}
4	a c e d m	{"ID":"[1,0,4]","METRIC":"[0.921875,0.921875,0.90625]"}

文本近似最近邻训练 (TextApproxNearestNeighborTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.TextApproxNearestNeighborTrainBatchOp

Python 类名：TextApproxNearestNeighborTrainBatchOp

功能介绍

文本相似度是在字符串相似度的基础上，基于词，计算两两文章或者句子之间的相似度，文章或者句子需要以空格分割的文本，计算方式和字符串相似度类似：支持SimHashHamming，MinHash和Jaccard三种近似相似度计算方式，通过选择metric参数可计算不同的相似度。

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

SimhashHamming（SimHash_Hamming_Distance)相似度=1-距离/64.0，应选择metric的参数为SIMHASH_HAMMING_SIM。

MinHash应选择metric的参数为MINHASH_SIM。

Jaccard应选择metric的参数为JACCARD_SIM。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
idCol	id列名	id列名	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		

metric	距离类型	用于计算的距离类型	String		"SIMHASH_HAMMING_SIM", "SIMHASH_HAMMING", "MINHASH_JACCARD_SIM", "JACCARD_SIM"	"SIMHASH_HAM
numBucket	分桶个数	分桶个数	Integer			10
numHashTables	哈希表个数	哈希表的数目	Integer			10
seed	采样种子	采样种子	Long			0

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "a b c d e", "a a b c e"],
    [1, "a a c e d w", "a a b b e d"],
    [2, "c d e f a", "b b c e f a"],
    [3, "b d e f h", "d d e a c"],
    [4, "a c e d m", "a e e f b c"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')
```

```
train =
TextApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1").
setMetric("SIMHASH_HAMMING_SIM").linkFrom(inOp)
predict =
TextApproxNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3).li
nkFrom(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.TextApproxNearestNeighborPredictBat
chOp;
import
com.alibaba.alink.operator.batch.similarity.TextApproxNearestNeighborTrainBatch
Op;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TextApproxNearestNeighborTrainBatchOpTest {
    @Test
    public void testTextApproxNearestNeighborTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "a b c d e", "a a b c e"),
            Row.of(1, "a a c e d w", "a a b b e d"),
            Row.of(2, "c d e f a", "b b c e f a"),
            Row.of(3, "b d e f h", "d d e a c"),
            Row.of(4, "a c e d m", "a e e f b c")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        BatchOperator <?> train = new
TextApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1")
            .setMetric("SIMHASH_HAMMING_SIM").linkFrom(inOp);
        BatchOperator <?> predict = new
TextApproxNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3)
            .linkFrom(train, inOp);
        predict.print();
    }
}
```

运行结果

id	text1	text2
0	a b c d e	{"ID":"[0,1,2]","METRIC":"[0.953125,0.921875,0.90625]"}
1	a a c e d w	{"ID":"[0,1,4]","METRIC":"[0.9375,0.90625,0.859375]"}
2	c d e f a	{"ID":"[0,1,4]","METRIC":"[0.890625,0.859375,0.8125]"}
3	b d e f h	{"ID":"[4,2,1]","METRIC":"[0.9375,0.90625,0.890625]"}
4	a c e d m	{"ID":"[1,0,4]","METRIC":"[0.921875,0.921875,0.90625]"}

文本最近邻预测 (TextNearestNeighborPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.TextNearestNeighborPredictBatchOp

Python 类名：TextNearestNeighborPredictBatchOp

功能介绍

文本相似度是在字符串相似度的基础上，基于词，计算两两文章或者句子之间的相似度，文章或者句子需要以空格分割的文本，计算方式和字符串相似度类似：支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine三种精确相似度计算方式，通过选择metric参数可计算不同的相似度。

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

Levenshtein（Levenshtein Distance）相似度=(1-距离)/length，length为两个字符长度的最大值离，应选metric的参数为LEVENSHTEIN_SIM。

LCS（Longest Common SubString）相似度=(1-距离)/length，length为两个字符长度的最大值，应选择metric的参数为LCS_SIM。

SSK（String Subsequence Kernel）支持相似度计算，应选择metric的参数为SSK。

Cosine（Cosine）支持相似度计算，应选择metric的参数为COSINE。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
radius	radius值	radius值	Double			null
reservedCols	算法保留列名	算法保留列	String[]			null
topN	TopN的值	TopN的值	Integer		[1, +inf)	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "a b c d e", "a a b c e"],
    [1, "a a c e d w", "a a b b e d"],
    [2, "c d e f a", "b b c e f a"],
    [3, "b d e f h", "d d e a c"],
    [4, "a c e d m", "a e e f b c"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')

train =
TextNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1").setMet
ric("LEVENSHTEIN_SIM").linkFrom(inOp)
predict =
TextNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3).linkFrom
(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.TextNearestNeighborPredictBatchOp;
import
com.alibaba.alink.operator.batch.similarity.TextNearestNeighborTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TextNearestNeighborPredictBatchOpTest {
    @Test
    public void testTextNearestNeighborPredictBatchOp() throws Exception {
```

```
        List <Row> df = Arrays.asList(
            Row.of(0, "a b c d e", "a a b c e"),
            Row.of(1, "a a c e d w", "a a b b e d"),
            Row.of(2, "c d e f a", "b b c e f a"),
            Row.of(3, "b d e f h", "d d e a c"),
            Row.of(4, "a c e d m", "a e e f b c")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        BatchOperator <?> train = new
TextNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1")
            .setMetric("LEVENSHTEIN_SIM").linkFrom(inOp);
        BatchOperator <?> predict =
            new
TextNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3).linkFrom
(
            train, inOp);
        predict.print();
    }
}
```

运行结果

id	text1	text2
0	a b c d e	{"ID":"[0,1,4]","METRIC":"[0.6,0.5,0.19999999999999996]"}
1	a a c e d w	{"ID":"[1,0,4]","METRIC":"[0.5,0.3333333333333337,0.3333333333333337]"}
2	c d e f a	{"ID":"[3,2,4]","METRIC":"[0.5,0.5,0.3333333333333337]"}
3	b d e f h	{"ID":"[3,2,4]","METRIC":"[0.4,0.4,0.19999999999999996]"}
4	a c e d m	{"ID":"[3,2,4]","METRIC":"[0.3333333333333337,0.3333333333333337,0.3333333333333337]"}

文本最近邻训练 (TextNearestNeighborTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.TextNearestNeighborTrainBatchOp

Python 类名：TextNearestNeighborTrainBatchOp

功能介绍

文本相似度是在字符串相似度的基础上，基于词，计算两两文章或者句子之间的相似度，文章或者句子需要以空格分割的文本，计算方式和字符串相似度类似：支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine三种精确相似度计算方式，通过选择metric参数可计算不同的相似度。

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

Levenshtein（Levenshtein Distance）相似度=(1-距离)/length，length为两个字符长度的最大值离，应选metric的参数为LEVENSHTEIN_SIM。

LCS（Longest Common SubString）相似度=(1-距离)/length，length为两个字符长度的最大值，应选择metric的参数为LCS_SIM。

SSK（String Subsequence Kernel）支持相似度计算，应选择metric的参数为SSK。

Cosine（Cosine）支持相似度计算，应选择metric的参数为COSINE。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
idCol	id列名	id列名	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[STRING]	

lambda	匹配字符权重	匹配字符权重，SSK中使用	Double			0.5
metric	距离类型	用于计算的距离类型	String		"LEVENSHTEIN_SIM", "LEVENSHTEIN", "LCS_SIM", "LCS", "SSK", "COSINE"	"LEVENSHTEIN_SIM"
windowSize	窗口大小	窗口大小	Integer		[1, +inf)	2

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "a b c d e", "a a b c e"],
    [1, "a a c e d w", "a a b b e d"],
    [2, "c d e f a", "b b c e f a"],
    [3, "b d e f h", "d d e a c"],
    [4, "a c e d m", "a e e f b c"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')

train =
TextNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1").setMet
ric("LEVENSHTEIN_SIM").linkFrom(inOp)
predict =
TextNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3).linkFrom
(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.TextNearestNeighborPredictBatchOp;
import
com.alibaba.alink.operator.batch.similarity.TextNearestNeighborTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TextNearestNeighborTrainBatchOpTest {
    @Test
    public void testTextNearestNeighborTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "a b c d e", "a a b c e"),
            Row.of(1, "a a c e d w", "a a b b e d"),
            Row.of(2, "c d e f a", "b b c e f a"),
            Row.of(3, "b d e f h", "d d e a c"),
            Row.of(4, "a c e d m", "a e e f b c")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        BatchOperator <?> train = new
TextNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("text1")
            .setMetric("LEVENSHTEIN_SIM").linkFrom(inOp);
        BatchOperator <?> predict =
            new
TextNearestNeighborPredictBatchOp().setSelectedCol("text2").setTopN(3).linkFrom
(
            train, inOp);
        predict.print();
    }
}
```

运行结果

id	text1	text2
0	a b c d e	{"ID":"[0,1,4]","METRIC":"[0.6,0.5,0.19999999999999996]"}
1	a a c e d w	{"ID":"[1,0,4]","METRIC":"[0.5,0.33333333333333337,0.33333333333333337]"}

2	c d e f a	{"ID":"[3,2,4]","METRIC":"[0.5,0.5,0.33333333333333337]"}
3	b d e f h	{"ID":"[3,2,4]","METRIC":"[0.4,0.4,0.19999999999999996]"}
4	a c e d m	{"ID":"[3,2,4]","METRIC":"[0.33333333333333337,0.33333333333333337,0.33333333333333337]"}

文本两两相似度计算 (TextSimilarityPairwiseBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.TextSimilarityPairwiseBatchOp

Python 类名：TextSimilarityPairwiseBatchOp

功能介绍

文章相似度是在字符串相似度的基础上，基于词，计算两两文章或者句子之间的相似度，文章或者句子需要以空格分割的文本，计算方式和字符串相似度类似： 支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine，SimHashHamming，MinHash和Jaccard七种相似度计算方式，通过选择metric参数可计算不同的相似度。

Levenshtein（Levenshtein Distance）支持距离和相似度两种方式，相似度=(1-距离)/length，length为两个字符长度的最大值，距离应选择metric的参数为LEVENSHTEIN，相似度应选metric的参数为LEVENSHTEIN_SIM。

LCS（Longest Common SubString）支持距离和相似度两种参数，相似度=(1-距离)/length，length为两个字符长度的最大值，距离应选择metric的参数为LCS，相似度应选择metric的参数为LCS_SIM。

SSK（String Subsequence Kernel）支持相似度计算，应选择metric的参数为SSK。

Cosine（Cosine）支持相似度计算，应选择metric的参数为COSINE。

SimhashHamming（SimHash_Hamming_Distance)，支持距离和相似度两种方式，相似度=1-距离/64.0，距离应选择metric的参数为SIMHASH_HAMMING，相似度应选择metric的参数为SIMHASH_HAMMING_SIM。

MinHash 支持相似度计算，应选择metric的参数为MINHASH_SIM。

Jaccard 支持相似度计算，应选择metric的参数为JACCARD_SIM。

Alink上文本相似度算法包括Batch组件和Stream组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		

selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为 [STRING]	
lambda	匹配字符权重	匹配字符权重，SSK中使用	Double			0.5
metric	度量类型	计算距离时，可以取不同的度量	String		"LEVENSHTEIN", "LEVENSHTEIN_SIM", "LCS", "LCS_SIM", "SSK", "COSINE", "SIMHASH_HAMMING", "SIMHASH_HAMMING_SIM", "JACCARD_SIM"	"LEVENSHTEI
numBucket	分桶个数	分桶个数	Integer			10
numHashTables	哈希表个数	哈希表的数目	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
seed	采样种子	采样种子	Long			0
windowSize	窗口大小	窗口大小	Integer		[1, +inf)	2

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "a b c d e", "a a b c e"],
    [1, "a a c e d w", "a a b b e d"],
    [2, "c d e f a", "b b c e f a"],
    [3, "b d e f h", "d d e a c"],
    [4, "a c e d m", "a e e f b c"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id long, text1 string,
text2 string')

batchOp = TextSimilarityPairwiseBatchOp().setSelectedCols(["text1",
"text2"]).setMetric("LEVENSHTEIN").setOutputCol("output")
batchOp.linkFrom(inOp1).print()

streamOp = TextSimilarityPairwiseStreamOp().setSelectedCols(["text1",
"text2"]).setMetric("COSINE").setOutputCol("output")
streamOp.linkFrom(inOp2).print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;
```

```
import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.TextSimilarityPairwiseBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import
com.alibaba.alink.operator.stream.similarity.TextSimilarityPairwiseStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TextSimilarityPairwiseBatchOpTest {
    @Test
    public void testTextSimilarityPairwiseBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "a b c d e", "a a b c e"),
            Row.of(1, "a a c e d w", "a a b b e d"),
            Row.of(2, "c d e f a", "b b c e f a"),
            Row.of(3, "b d e f h", "d d e a c"),
            Row.of(4, "a c e d m", "a e e f b c")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text1
string, text2 string");
        BatchOperator <?> batchOp = new
TextSimilarityPairwiseBatchOp().setSelectedCols("text1", "text2").setMetric(
            "LEVENSHTEIN").setOutputCol("output");
        batchOp.linkFrom(inOp1).print();
        StreamOperator <?> streamOp = new
TextSimilarityPairwiseStreamOp().setSelectedCols("text1", "text2").setMetric(
            "COSINE").setOutputCol("output");
        streamOp.linkFrom(inOp2).print();
        StreamOperator.execute();
    }
}
```

运行结果

id	text1	text2	output
0	a b c d e	a a b c e	2.0
1	a a c e d w	a a b b e d	3.0
2	c d e f a	b b c e f a	3.0

3	b d e f h	d d e a c	3.0
4	a c e d m	a e e f b c	4.0

向量近似最近邻预测 (VectorApproxNearestNeighborPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.VectorApproxNearestNeighborPredictBatchOp

Python 类名：VectorApproxNearestNeighborPredictBatchOp

功能介绍

该功能由训练和预测组成，该组件为预测功能

该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
radius	radius值	radius值	Double			null
reservedCols	算法保留列名	算法保留列	String[]			null
topN	TopN的值	TopN的值	Integer		[1, +inf)	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "1 1 1"],
    [2, "2 2 2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
train =
VectorApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("vec").
linkFrom(inOp)
predict =
VectorApproxNearestNeighborPredictBatchOp().setSelectedCol("vec").setTopN(3).li
nkFrom(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.VectorApproxNearestNeighborPredictB
atchOp;
import
com.alibaba.alink.operator.batch.similarity.VectorApproxNearestNeighborTrainBat
chOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorApproxNearestNeighborPredictBatchOpTest {
    @Test
    public void testVectorApproxNearestNeighborPredictBatchOp() throws
Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "1 1 1"),
            Row.of(2, "2 2 2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, vec
string");
        BatchOperator <?> train = new
VectorApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("vec")
```

```
            .linkFrom(inOp);
        BatchOperator <?> predict = new
VectorApproxNearestNeighborPredictBatchOp().setSelectedCol("vec").setTopN(3)
            .linkFrom(train, inOp);
        predict.print();
    }
}
```

运行结果

id	vec
0	{"ID":"[0,1,2]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}
1	{"ID":"[1,2,0]","METRIC":"[0.0,1.7320508075688772,1.7320508075688772]"}
2	{"ID":"[2,1,0]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}

向量近似最近邻训练 (VectorApproxNearestNeighborTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.VectorApproxNearestNeighborTrainBatchOp

Python 类名：VectorApproxNearestNeighborTrainBatchOp

功能介绍

该功能由训练和预测组成，训练时指定距离计算方式，生成最近邻模型

可选择的距离计算方式包含EUCLIDEAN和JACCARD两种，同时支持KDTREE和LSH两种近似方法。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
idCol	id列名	id列名	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
metric	距离度量方式	距离类型	String		"EUCLIDEAN", "JACCARD"	"EUCLIDEA
numHashTables	哈希表的数目	哈希表的数目	Integer			1

numProjectionsPerTable	每个哈希表中的哈希函数个数	每个哈希表中的哈希函数个数	Integer			1
projectionWidth	桶的宽度	桶的宽度	Double			1.0
seed	采样种子	采样种子	Long			0
solver	近似方法	近似方法，包括KDTREE和LSH	String		"KDTREE", "LSH"	"KDTREE"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "1 1 1"],
    [2, "2 2 2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
train =
```

```
VectorApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("vec").
linkFrom(inOp)
predict =
VectorApproxNearestNeighborPredictBatchOp().setSelectedCol("vec").setTopN(3).li
nkFrom(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.VectorApproxNearestNeighborPredictB
atchOp;
import
com.alibaba.alink.operator.batch.similarity.VectorApproxNearestNeighborTrainBat
chOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorApproxNearestNeighborTrainBatchOpTest {
    @Test
    public void testVectorApproxNearestNeighborTrainBatchOp() throws Exception
{
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "1 1 1"),
            Row.of(2, "2 2 2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, vec
string");
        BatchOperator <?> train = new
VectorApproxNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("vec")
            .linkFrom(inOp);
        BatchOperator <?> predict = new
VectorApproxNearestNeighborPredictBatchOp().setSelectedCol("vec").setTopN(3)
            .linkFrom(train, inOp);
        predict.print();
    }
}
```

运行结果

id	vec
0	{"ID":"[0,1,2]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}
1	{"ID":"[1,2,0]","METRIC":"[0.0,1.7320508075688772,1.7320508075688772]"}
2	{"ID":"[2,1,0]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}

向量最近邻预测 (VectorNearestNeighborPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.VectorNearestNeighborPredictBatchOp

Python 类名：VectorNearestNeighborPredictBatchOp

功能介绍

该组件为向量最近邻预测功能，接收 VectorNearestNeighborTrainBatchOp 训练的模型

该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出前N个最近邻，如果填写了radius，则输出radius范围内的邻居。如果两个同时设置，则输出radius范围内前N个最近邻。

如果不设置 OutputCol，输出列会替换输入的向量列。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
radius	radius值	radius值	Double			null
reservedCols	算法保留列名	算法保留列	String[]			null
topN	TopN的值	TopN的值	Integer		[1, +inf)	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "1 1 1"],
    [2, "2 2 2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
train =
VectorNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("vec").linkFr
om(inOp)
predict =
VectorNearestNeighborPredictBatchOp().setSelectedCol("vec").setTopN(3).linkFrom
(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.VectorNearestNeighborPredictBatchOp
;
import
com.alibaba.alink.operator.batch.similarity.VectorNearestNeighborTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorNearestNeighborPredictBatchOpTest {
    @Test
    public void testVectorNearestNeighborPredictBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "1 1 1"),
```

```
            Row.of(2, "2 2 2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, vec
string");
        BatchOperator <?> train =
            new
VectorNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("vec").linkFr
om(
            inOp);
        BatchOperator <?> predict =
            new
VectorNearestNeighborPredictBatchOp().setSelectedCol("vec").setTopN(3).linkFrom
(
            train, inOp);
        predict.print();
    }
}
```

运行结果

id	vec
0	{"ID":"[0,1,2]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}
1	{"ID":"[1,2,0]","METRIC":"[0.0,1.7320508075688772,1.7320508075688772]"}
2	{"ID":"[2,1,0]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}

向量最近邻训练 (VectorNearestNeighborTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.VectorNearestNeighborTrainBatchOp

Python 类名：VectorNearestNeighborTrainBatchOp

功能介绍

该组件为向量最近邻的训练过程，在计算时与 VectorNearestNeighborPredictBatchOp 配合使用。

支持的距离计算方式包含EUCLIDEAN，COSINE，INNERPRODUCT（内积），CITYBLOCK（曼哈顿距离），JACCARD，PEARSON

默认距离EUCLIDEAN

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
idCol	id列名	id列名	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
metric	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE", "INNERPRODUCT", "CITYBLOCK", "JACCARD", "PEARSON"	"EUCLIDEAN"

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "1 1 1"],
    [2, "2 2 2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
train =
VectorNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("vec").linkFr
om(inOp)
predict =
VectorNearestNeighborPredictBatchOp().setSelectedCol("vec").setTopN(3).linkFrom
(train, inOp)
predict.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import
com.alibaba.alink.operator.batch.similarity.VectorNearestNeighborPredictBatchOp
;
import
com.alibaba.alink.operator.batch.similarity.VectorNearestNeighborTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorNearestNeighborTrainBatchOpTest {
    @Test
    public void testVectorNearestNeighborTrainBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "1 1 1"),
            Row.of(2, "2 2 2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, vec
string");
```

```
        BatchOperator <?> train =
            new
VectorNearestNeighborTrainBatchOp().setIdCol("id").setSelectedCol("vec").linkFr
om(
            inOp);
        BatchOperator <?> predict =
            new
VectorNearestNeighborPredictBatchOp().setSelectedCol("vec").setTopN(3).linkFrom
(
            train, inOp);
        predict.print();
    }
}
```

运行结果

id	vec
0	{"ID":"[0,1,2]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}
1	{"ID":"[1,2,0]","METRIC":"[0.0,1.7320508075688772,1.7320508075688772]"}
2	{"ID":"[2,1,0]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}

向量对相似度计算 (VectorSimilarityPairwiseBatchOp)

Java 类名：com.alibaba.alink.operator.batch.similarity.VectorSimilarityPairwiseBatchOp

Python 类名：VectorSimilarityPairwiseBatchOp

功能介绍

数据中包含至少两个向量，使用 VectorSimilarityPairwiseBatchOp 计算两个向量的相似度。

如果是稠密格式的向量，需要保证两个向量长度相等。支持 EUCLIDEAN、COSINE、INNERPRODUCT（内积）、CITYBLOCK（曼哈顿距离）、JACCARD、PEARSON 六种距离计算方法。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	所选列类型为[DENSE_VECTOR, SPARSE_VECTOR, STRING, VECTOR]	
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE", "INNERPRODUCT", "CITYBLOCK", "JACCARD", "PEARSON"	"EUCLIDEAN"
reservedCols	算法保留列名	算法保留列	String[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0","1 1 1"],
    [1, "1 1 1","2 2 2"],
    [2, "2 2 2","0 0 0"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec1 string, vec2
string')


op = VectorSimilarityPairwiseBatchOp()\
    .setSelectedCols(["vec1","vec2"])\
    .setOutputCol("res")\
    .setDistanceType("EUCLIDEAN")
op.linkFrom(inOp).print()
```

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorSimilarityPairwiseBatchOpTest {
```

```
    @Test
    public void test() throws Exception {
        List <Row> rows = Arrays.asList(
            Row.of(0, "0 0 0","1 1 1"),
            Row.of(1, "1 1 1","2 2 2"),
            Row.of(2, "2 2 2","0 0 0")
        );

        BatchOperator data = new MemSourceBatchOp(rows, "id int, vec1 string,
vec2 string");
        VectorSimilarityPairwiseBatchOp op = new
VectorSimilarityPairwiseBatchOp()
            .setSelectedCols("vec1", "vec2")
            .setDistanceType("EUCLIDEAN")
            .setOutputCol("res")
            .linkFrom(data);
        op.print();
    }
}
```

运行结果

id	vec1	vec2	res
0	0 0 0	1 1 1	1.7321
1	1 1 1	2 2 2	1.7321
2	2 2 2	0 0 0	3.4641

数据预览 (PreviewDataBatchOp)

Java 类名：com.alibaba.alink.operator.batch.utils.PreviewDataBatchOp

Python 类名：PreviewDataBatchOp

功能介绍

数据预览用于对数据源或者组件产生的表进行预览，可以用于batch组件和stream组件。

其中，batch数据预览在当前运行实验结束后、下次运行之前均能查看数据；而stream组件在实验运行中可以即时预览数据，在试验运行结束后，可以通过时间轴查看历史数据。

参数说明

函数名称：

com.alibaba.alink.batchoperator.utils.PreviewDataBatchOp,

com.alibaba.alink.streamoperator.utils.PreviewDataStreamOp

输入桩个数：1，数据集

输出桩个数：1，用于预览的数据子集

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
maxCount	最多预览的数据项数，默认为 100	最多预览的数据项数目	Integer			100

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
source = RandomTableSourceBatchOp().setNumRows(1000).setNumCols(50)
preview = PreviewDataBatchOp().setVizName("Preview")
preview.linkFrom(source)
BatchOperator.execute()
```

运行结果

脚本运行后从 Alink Web 的“可视化”页面进行查看。

批式数据打印 (PrintBatchOp)

Java 类名：com.alibaba.alink.operator.batch.utils.PrintBatchOp

Python 类名：PrintBatchOp

功能介绍

打印表中数据。

使用方式

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
       ["0,0,0"],
       ["0.1,0.1,0.1"],
       ["0.2,0.2,0.2"],
       ["9,9,9"],
       ["9.1,9.1,9.1"],
       ["9.2,9.2,9.2"]
])

# batch source
inOp = BatchOperator.fromDataframe(df, schemaStr='y string')

inOp.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class PrintBatchOpTest {
    @Test
    public void testPrintBatchOp() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("0,0,0"),
            Row.of("0.1,0.1,0.1"),
            Row.of("0.2,0.2,0.2"),
            Row.of("9,9,9"),
            Row.of("9.1,9.1,9.1"),
            Row.of("9.2,9.2,9.2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "y string");
        inOp.print();
    }
}
```

运行结果

y| |---| 0,0,0| 0.1,0.1,0.1| 0.2,0.2,0.2| 9,9,9| 9.1,9.1,9.1| 9.2,9.2,9.2|

UDF (UDFBatchOp)

Java 类名：com.alibaba.alink.operator.batch.utils.UDFBatchOp

Python 类名：UDFBatchOp

UDF/UDTF 定义

PyAlink 提供了基于 Python 的 UDF/UDTF 支持，方便进行灵活的数据处理。 PyAlink 所定义的 UDF/UDTF 即可以用于 PyAlink 提供的 UDF/UDTF 组件，也可以用于所提供的 `sqlQuery` 函数。

我们提供了 `udf` 和 `udtf` 函数来帮助构造 UDF/UDTF。 两个函数使用时都需要提供一个函数体、输入类型和返回类型。

- 函数体对于 UDF 而言，是直接用 `return` 返回值的 Python 函数，或者 lambda 函数； 对于 UDTF 而言，是用 `yield` 来多次返回值的 Python 函数。
- 输入类型均为 `DataType` 类型的 Python list。
- 输出类型，UDF 为单个 `DataType` 类型，UDTF为 `DataType` 类型的 Python list。

`DataType` 类型可以直接用 `DataTypes.DOUBLE()` 等类似的函数得到。

以下是定义 UDF/UDTF 的代码示例：

```
# 4种 UDF 定义

# ScalarFunction
class PlusOne(ScalarFunction):
    def eval(self, x, y):
        return x + y + 10
f_udf1 = udf(PlusOne(), input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_type=DataTypes.DOUBLE())

# function + decorator
@udf(input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_type=DataTypes.DOUBLE())
def f_udf2(x, y):
    return x + y + 20

# function
def f_udf3(x, y):
    return x + y + 30
f_udf3 = udf(f_udf3, input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_type=DataTypes.DOUBLE())

# lambda function
f_udf4 = udf(lambda x, y: x + y + 40
               , input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_type=DataTypes.DOUBLE())
```

```
udfs = [
    f_udf1,
    f_udf2,
    f_udf3,
    f_udf4
]

# 4种 UDTF 定义

# TableFunction
class SplitOp(TableFunction):
    def eval(self, *args):
        for index, arg in enumerate(args):
            yield index, arg
f_udtf1 = udtf(SplitOp(), [DataTypes.DOUBLE(), DataTypes.DOUBLE()],
[DataTypes.INT(), DataTypes.DOUBLE()])

# function + decorator
@udtf(input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()], result_types=
[DataTypes.INT(), DataTypes.DOUBLE()])
def f_udtf2(*args):
    for index, arg in enumerate(args):
        yield index, arg

# function
def f_udtf3(*args):
    for index, arg in enumerate(args):
        yield index, arg
f_udtf3 = udtf(f_udtf3, input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_types=[DataTypes.INT(), DataTypes.DOUBLE()])

# lambda function
f_udtf4 = udtf(lambda *args: [ (yield index, arg) for index, arg in
enumerate(args) ]
               , input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_types=[DataTypes.INT(), DataTypes.DOUBLE()])

udtfs = [
    f_udtf1,
    f_udtf2,
    f_udtf3,
    f_udtf4
]
```

UDF/UDTF 组件使用

在流和批两种场景中，分别提供了 UDF/UDTF 对应的 Operator：

- `UDFBatchOp`
- `UDFStreamOp`
- `UDTFBatchOp`
- `UDTFStreamOp`

它们的参数包括：

- `setFunc`：设置 UDF 或 UDTF，由前文的 `udf` 或 `udtf` 函数产生；
- setSelectedCols：选择参与计算的列；
- setOutputCol/setOutputCols：设置结果列名，其中 UDF 允许1列，UDTF 允许多列；
- setReservedCols：设置保留列。

```
source = CsvSourceBatchOp() \
    .setSchemaStr(
    "sepal_length double, sepal_width double, petal_length double, petal_width
double, category string") \
    .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv")

for index, f in enumerate(udfs):
    udfBatchOp = UDFBatchOp() \
        .setFunc(f) \
        .setSelectedCols(["sepal_length", "sepal_width"]) \
        .setOutputCol("sepal_length") \
        .linkFrom(source)
    df = udfBatchOp.collectToDataframe()
    print(df)

stream_source = CsvSourceStreamOp() \
    .setSchemaStr(
    "sepal_length double, sepal_width double, petal_length double, petal_width
double, category string") \
    .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv")

for index, f in enumerate(udtfs):
    udtfStreamOp = UDTFStreamOp() \
        .setFunc(f) \
        .setSelectedCols(["sepal_length", "sepal_width"]) \
        .setOutputCols(["index", "sepal_length"]) \
        .linkFrom(stream_source)
    udtfStreamOp.print()
    StreamOperator.execute()
```

除了使用组件形式以外，Operator 下还提供了 `udf` 和 `udtf` 方法，参数与上文中的 Operator 一致：

```
udf(self, func, selectedCols, outputCol, resultType, reservedCols=None)
udtf(self, func, selectedCols, outputCols, resultTypes, reservedCols=None)
```

SQL 中使用 UDF/UDTF

PyAlink 提供了更多对于 SQL 的支持。 `BatchOperator` 和 `StreamOperator` 提供了 `registerTableName` 和 `registerFunction` 的方法，用于将 Operator 对应的 Table 和 UDF/UDTF 注册。 `BatchOperator` 和 `StreamOperator` 还提供了 `sqlQuery` 静态函数来支持 SQL 功能。

SQL的使用可以参考下面的代码：

```
source.registerTableName("A")
for index, f in enumerate(udfs):
    name = "plus" + str(index)
    print(name, f)
    BatchOperator.registerFunction(name, f)
    BatchOperator.sqlQuery("select " + name + "(sepal_width, petal_width) as t
from A where sepal_width > 4").print()

stream_source.registerTableName("A")
for index, f in enumerate(udtfs):
    name = "split" + str(index)
    print(name, f)
    StreamOperator.registerFunction(name, f)
    StreamOperator.sqlQuery("select sepal_width, index, v from A, LATERAL
TABLE(" + name + "(sepal_width, petal_length)) as T(index, v) where sepal_width
> 4").print()
    StreamOperator.execute()
```

UDTF (UDTFBatchOp)

Java 类名：com.alibaba.alink.operator.batch.utils.UDTFBatchOp

Python 类名：UDTFBatchOp

UDF/UDTF 定义

PyAlink 提供了基于 Python 的 UDF/UDTF 支持，方便进行灵活的数据处理。 PyAlink 所定义的 UDF/UDTF 即可以用于 PyAlink 提供的 UDF/UDTF 组件，也可以用于所提供的 `sqlQuery` 函数。

我们提供了 `udf` 和 `udtf` 函数来帮助构造 UDF/UDTF。 两个函数使用时都需要提供一个函数体、输入类型和返回类型。

- 函数体对于 UDF 而言，是直接用 `return` 返回值的 Python 函数，或者 lambda 函数； 对于 UDTF 而言，是用 `yield` 来多次返回值的 Python 函数。
- 输入类型均为 `DataType` 类型的 Python list。
- 输出类型，UDF 为单个 `DataType` 类型，UDTF为 `DataType` 类型的 Python list。

`DataType` 类型可以直接用 `DataTypes.DOUBLE()` 等类似的函数得到。

以下是定义 UDF/UDTF 的代码示例：

```
# 4种 UDF 定义

# ScalarFunction
class PlusOne(ScalarFunction):
    def eval(self, x, y):
        return x + y + 10
f_udf1 = udf(PlusOne(), input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_type=DataTypes.DOUBLE())

# function + decorator
@udf(input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_type=DataTypes.DOUBLE())
def f_udf2(x, y):
    return x + y + 20

# function
def f_udf3(x, y):
    return x + y + 30
f_udf3 = udf(f_udf3, input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_type=DataTypes.DOUBLE())

# lambda function
f_udf4 = udf(lambda x, y: x + y + 40
               , input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_type=DataTypes.DOUBLE())
```

```
udfs = [
    f_udf1,
    f_udf2,
    f_udf3,
    f_udf4
]

# 4种 UDTF 定义

# TableFunction
class SplitOp(TableFunction):
    def eval(self, *args):
        for index, arg in enumerate(args):
            yield index, arg
f_udtf1 = udtf(SplitOp(), [DataTypes.DOUBLE(), DataTypes.DOUBLE()],
[DataTypes.INT(), DataTypes.DOUBLE()])

# function + decorator
@udtf(input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()], result_types=
[DataTypes.INT(), DataTypes.DOUBLE()])
def f_udtf2(*args):
    for index, arg in enumerate(args):
        yield index, arg

# function
def f_udtf3(*args):
    for index, arg in enumerate(args):
        yield index, arg
f_udtf3 = udtf(f_udtf3, input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_types=[DataTypes.INT(), DataTypes.DOUBLE()])

# lambda function
f_udtf4 = udtf(lambda *args: [ (yield index, arg) for index, arg in
enumerate(args) ]
               , input_types=[DataTypes.DOUBLE(), DataTypes.DOUBLE()],
result_types=[DataTypes.INT(), DataTypes.DOUBLE()])

udtfs = [
    f_udtf1,
    f_udtf2,
    f_udtf3,
    f_udtf4
]
```

UDF/UDTF 组件使用

在流和批两种场景中，分别提供了 UDF/UDTF 对应的 Operator：

- `UDFBatchOp`
- `UDFStreamOp`
- `UDTFBatchOp`
- `UDTFStreamOp`

它们的参数包括：

- `setFunc`：设置 UDF 或 UDTF，由前文的 `udf` 或 `udtf` 函数产生；
- setSelectedCols：选择参与计算的列；
- setOutputCol/setOutputCols：设置结果列名，其中 UDF 允许1列，UDTF 允许多列；
- setReservedCols：设置保留列。

```
source = CsvSourceBatchOp() \
    .setSchemaStr(
    "sepal_length double, sepal_width double, petal_length double, petal_width
double, category string") \
    .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv")

for index, f in enumerate(udfs):
    udfBatchOp = UDFBatchOp() \
        .setFunc(f) \
        .setSelectedCols(["sepal_length", "sepal_width"]) \
        .setOutputCol("sepal_length") \
        .linkFrom(source)
    df = udfBatchOp.collectToDataframe()
    print(df)

stream_source = CsvSourceStreamOp() \
    .setSchemaStr(
    "sepal_length double, sepal_width double, petal_length double, petal_width
double, category string") \
    .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv")

for index, f in enumerate(udtfs):
    udtfStreamOp = UDTFStreamOp() \
        .setFunc(f) \
        .setSelectedCols(["sepal_length", "sepal_width"]) \
        .setOutputCols(["index", "sepal_length"]) \
        .linkFrom(stream_source)
    udtfStreamOp.print()
    StreamOperator.execute()
```

除了使用组件形式以外，Operator 下还提供了 `udf` 和 `udtf` 方法，参数与上文中的 Operator 一致：

```
udf(self, func, selectedCols, outputCol, resultType, reservedCols=None)
udtf(self, func, selectedCols, outputCols, resultTypes, reservedCols=None)
```

SQL 中使用 UDF/UDTF

PyAlink 提供了更多对于 SQL 的支持。 `BatchOperator` 和 `StreamOperator` 提供了 `registerTableName` 和 `registerFunction` 的方法，用于将 Operator 对应的 Table 和 UDF/UDTF 注册。 `BatchOperator` 和 `StreamOperator` 还提供了 `sqlQuery` 静态函数来支持 SQL 功能。

SQL的使用可以参考下面的代码：

```
source.registerTableName("A")
for index, f in enumerate(udfs):
    name = "plus" + str(index)
    print(name, f)
    BatchOperator.registerFunction(name, f)
    BatchOperator.sqlQuery("select " + name + "(sepal_width, petal_width) as t
from A where sepal_width > 4").print()

stream_source.registerTableName("A")
for index, f in enumerate(udtfs):
    name = "split" + str(index)
    print(name, f)
    StreamOperator.registerFunction(name, f)
    StreamOperator.sqlQuery("select sepal_width, index, v from A, LATERAL
TABLE(" + name + "(sepal_width, petal_length)) as T(index, v) where sepal_width
> 4").print()
    StreamOperator.execute()
```

MFCC特征提取 (ExtractMfccFeatureBatchOp)

Java 类名：com.alibaba.alink.operator.batch.audio.ExtractMfccFeatureBatchOp

Python 类名：ExtractMfccFeatureBatchOp

功能介绍

- 从数据中提取 MFCC 特征。
- 支持Alink Vector、一维或两维Alink FloatTensor格式的数据

使用方式

用于声学特征提取，通常与ReadAudioToTensor组件一起使用，连接在其后

文献索引

[1] Davis S, Mermelstein P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences[J]. IEEE transactions on acoustics, speech, and signal processing, 1980, 28(4): 357-366.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sampleRate	采样率	采样率	Integer	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		
hopTime	相邻窗口时间间隔	相邻窗口时间间隔	Double			0.032
numMfcc	mfcc参数	mfcc参数	Integer			128
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
windowTime	一个窗口的时间	一个窗口的时间	Double			0.128
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

```
dataDir = "https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/audio";

df = pd.DataFrame([
    ["246.wav"],
    ["247.wav"]
])

allFiles = BatchOperator.fromDataframe(df, schemaStr='wav_file_path string')
SAMPLE_RATE = 16000

readOp = ReadAudioToTensorBatchOp().setRootFilePath(dataDir) \
.setSampleRate(SAMPLE_RATE) \
.setRelativeFilePathCol("wav_file_path") \
.setOutputCol("tensor") \
.linkFrom(allFiles)

mfccOp = ExtractMfccFeatureBatchOp() \
.setSampleRate(SAMPLE_RATE) \
.setSelectedCol("tensor") \
.linkFrom(readOp)

mfccOp.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

public class ExtractMfccFeatureBatchOpTest extends AlinkTestBase {
    @Test
    public void testExtractMfccFeatureBatchOp() throws Exception {
        String dataDir = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/audio";
        String[] allFiles = {"246.wav", "247.wav"};
        int sampleRate = 16000;
        String tensorName = "tensor";
        String mfccName = "mfcc";
        String wavFile = "wav_file_path";
        BatchOperator source = new MemSourceBatchOp(allFiles, wavFile)
                .link(new ReadAudioToTensorBatchOp()
                        .setRootFilePath(dataDir)
                        .setSampleRate(sampleRate)
```

```
                        .setRelativeFilePathCol(wavFile)
                        .setDuration(2)
                        .setOutputCol(tensorName)
                )
                .link(new ExtractMfccFeatureBatchOp()
                        .setSelectedCol(tensorName)
                        .setSampleRate(sampleRate)
                        .setWindowTime(0.128)
                        .setHopTime(0.032)
                        .setNumMfcc(26)
                        .setOutputCol(mfccName))
                .select(new String[]{wavFile, mfccName})
                .print();
    }
}
```

运行结果

wav_file_path	mfcc
246.wav	FLOAT#59,26,1#48.78127 -32.02646 12.432438 ...
247.wav	FLOAT#59,26,1#-50.62911 -13.844937 24.176699 ...

音频转张量 (ReadAudioToTensorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.audio.ReadAudioToTensorBatchOp

Python 类名：ReadAudioToTensorBatchOp

功能介绍

读取音频文件，并转换为 Alink FloatTensor 格式。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
relativeFilePathCol	文件路径列	文件路径列	String	✓	所选列类型为[STRING]	
rootFilePath	文件路径	文件路径	String	✓		
sampleRate	采样率	采样率	Integer	✓		
duration	采样持续时间	采样持续时间	Double			
offset	采样开始时刻	采样开始时刻	Double			0.0
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

```
dataDir = "https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/audio";

df = pd.DataFrame([
```

```
    ["246.wav"],
    ["247.wav"]
])

allFiles = BatchOperator.fromDataframe(df, schemaStr='wav_file_path string')
SAMPLE_RATE = 16000

readOp = ReadAudioToTensorBatchOp().setRootFilePath(dataDir) \
.setSampleRate(SAMPLE_RATE) \
.setRelativeFilePathCol("wav_file_path") \
.setOutputCol("tensor") \
.linkFrom(allFiles)

readOp.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

public class ReadAudioToTensorBatchOpTest extends AlinkTestBase {

    @Test
    public void testReadAudioToTensorOp() throws Exception {
        String dataDir = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/audio";
        String[] allFiles = {"246.wav", "247.wav"};
        int sampleRate = 16000;
        BatchOperator source = new MemSourceBatchOp(allFiles, "wav_file_path")
                .link(new ReadAudioToTensorBatchOp()
                        .setRootFilePath(DATA_DIR)
                        .setSampleRate(sampleRate)
                        .setRelativeFilePathCol("wav_file_path")
                        .setDuration(2)
                        .setOutputCol("tensor")
                );
        source.print();
    }

}
```

运行结果

wav_file_path	tensor

246.wav	FLOAT#32000,1#-7.324219E-4 -0.0010986328 -9.460449E-4 ...
247.wav	FLOAT#32000,1#-0.0057678223 -0.0051574707 -0.0036315918 ...

图片转张量 (ReadImageToTensorBatchOp)

Java 类名：com.alibaba.alink.operator.batch.image.ReadImageToTensorBatchOp

Python 类名：ReadImageToTensorBatchOp

功能介绍

将图片列转换为张量。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
relativeFilePathCol	文件路径列	文件路径列	String	✓	所选列类型为[STRING]	
rootFilePath	文件路径	文件路径	String	✓		
imageHeight	图片高度	图片高度	Integer			
imageWidth	图片宽度	图片宽度	Integer			
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
df_data = pd.DataFrame([
    'sphx_glr_plot_scripted_tensor_transforms_001.png'
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr = 'path string')

ReadImageToTensorBatchOp()\
    .setRootFilePath("https://pytorch.org/vision/stable/_images/")\
    .setRelativeFilePathCol("path")\
    .setOutputCol("tensor")\
```

```
    .linkFrom(batch_data)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Collections;
import java.util.List;

public class ReadImageToTensorBatchOpTest {

    @Test
    public void testReadImageToTensorBatchOp() throws Exception {

        List <Row> data = Collections.singletonList(
            Row.of("sphx_glr_plot_scripted_tensor_transforms_001.png")
        );

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "path
string");

        new ReadImageToTensorBatchOp()
            .setRootFilePath("https://pytorch.org/vision/stable/_images/")
            .setRelativeFilePathCol("path")
            .setOutputCol("tensor")
            .linkFrom(memSourceBatchOp)
            .print();
    }

}
```

运行结果

| path | tensor | |----------+----------| | sphx_glr_plot_scripted_tensor_transforms_001.png | FLOAT#250,520,4#1.0 1.0 1.0... |

张量转图片 (WriteTensorToImageBatchOp)

Java 类名：com.alibaba.alink.operator.batch.image.WriteTensorToImageBatchOp

Python 类名：WriteTensorToImageBatchOp

功能介绍

将张量列转换为图片，并写入根目录对应的相对路径列中，然后原样输出结果。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
relativeFilePathCol	文件路径列	文件路径列	String	✓	所选列类型为 [STRING]	
rootFilePath	文件路径	文件路径	String	✓		
tensorCol	tensor列	tensor列	String	✓	所选列类型为 [BOOL_TENSOR, BYTE_TENSOR, DOUBLE_TENSOR, FLOAT_TENSOR, INT_TENSOR, LONG_TENSOR, STRING, STRING_TENSOR, TENSOR, UBYTE_TENSOR]	
imageType	图片类型	图片类型	String		"PNG", "JPEG"	"PNG"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
df_data = pd.DataFrame([
    'sphx_glr_plot_scripted_tensor_transforms_001.png'
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr = 'path string')

readImageToTensorBatchOp = ReadImageToTensorBatchOp()\
    .setRootFilePath("https://pytorch.org/vision/stable/_images/")\
    .setRelativeFilePathCol("path")\
    .setOutputCol("tensor")

writeTensorToImageBatchOp = WriteTensorToImageBatchOp()\
            .setRootFilePath("/tmp/write_tensor_to_image")\
            .setTensorCol("tensor")\
            .setImageType("png")\
            .setRelativeFilePathCol("path")

batch_data.link(readImageToTensorBatchOp).link(writeTensorToImageBatchOp).print
()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.params.image.HasImageType.ImageType;
import org.junit.Test;

import java.util.Collections;
import java.util.List;

public class WriteTensorToImageBatchOpTest {

    @Test
    public void testWriteTensorToImageBatchOp() throws Exception {

        List <Row> data = Collections.singletonList(
            Row.of("sphx_glr_plot_scripted_tensor_transforms_001.png")
        );

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "path
string");

        ReadImageToTensorBatchOp readImageToTensorBatchOp = new
ReadImageToTensorBatchOp()
            .setRootFilePath("https://pytorch.org/vision/stable/_images/")
            .setRelativeFilePathCol("path")
```

```
            .setOutputCol("tensor");

        WriteTensorToImageBatchOp writeTensorToImageBatchOp = new
WriteTensorToImageBatchOp()
            .setRootFilePath("/tmp/write_tensor_to_image")
            .setTensorCol("tensor")
            .setImageType(ImageType.PNG)
            .setRelativeFilePathCol("path");


memSourceBatchOp.link(readImageToTensorBatchOp).link(writeTensorToImageBatchOp)
.print();
    }

}
```

运行结果

可以在 /tmp/write_tensor_to_image/sphx_glr_plot_scripted_tensor_transforms_001.png 中找到 https://pytorch.org/vision/stable/_images/sphx_glr_plot_scripted_tensor_transforms_001.png

同时组件的输出结果为：

| path | tensor | |----------+----------| | sphx_glr_plot_scripted_tensor_transforms_001.png | FLOAT#250,520,4#255.0 255.0... |

ONNX模型预测 (OnnxModelPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.onnx.OnnxModelPredictBatchOp

Python 类名：OnnxModelPredictBatchOp

功能介绍

加载 ONNX 模型进行预测。

使用方式

模型路径 `modelPath` 需要是 ONNX 模型。

参与模型预测的数据通过参数 `selectedCols` 设置，需要注意以下几点：

- ONNX 模型使用 input name 来标识模型输入桩的，因此需要设置 `inputNames`，与 `selectedCols` 一一对应，表明某列对应某输入桩。`inputNames` 不填写时，默认与列名一致。
- 仅支持输入桩为 `Tensor` 类型，不支持 `Sequences` 和 `Maps` 类型。
- 所选择的列的类型需要是 `float, double, int, long, byte, string` 类型及其对应的 Alink `Tensor` 类型。

模型输出信息通过参数 `outputSchemaStr` 指定，包括输出列名以及名称，需要注意以下几点：

- ONNX 模型使用 output name 来标识模型输出桩的，因此需要设置 `outputNames`，与 `outputSchemaStr` 一一对应，表明某列对应某输入桩。`outputNames` 不填写时，默认与列名一致。
- 仅支持输出桩为 `Tensor` 类型，不支持 `Sequences` 和 `Maps` 类型。
- `outputSchemaStr` 填写的输出类型需要是对应的输出桩类型，例如 输出桩类型 为 Float 类型的 Tensor 时，对应的 Alink 类型可以是 `TENSOR` 或者 `FLOAT_TENSOR`，当输出仅包含一个元素时，还可以是 `FLOAT`。

组件使用的是 ONNX 1.11.0 版本，当有 GPU 时，自动使用 GPU 进行推理，否则使用 CPU 进行推理。

在 Windows 下运行时，如果遇到 `UnsatisfiedLinkError`，请下载 Visual C++ 2019 Redistributable Packages 并重启，然后重新运行。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelPath	模型的URL路径	模型的URL路径	String	✓		
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如 "f0 string, f1 bigint, f2 double"	String	✓		

inputNames	ONNX 模型输入名	ONNX 模型输入名，用逗号分隔，需要与输入列一一对应，默认与选择列相同	String[]			null
outputNames	ONNX 模型输出名	ONNX 模型输出名，用逗号分隔，并且与输出 Schema 一一对应，默认与输出 Schema 中的列名相同	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

模型路径说明

模型路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
test = AkSourceBatchOp()\
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

test = VectorToTensorBatchOp()\
    .setTensorDataType("float")\
    .setTensorShape([1, 1, 28, 28])\
    .setSelectedCol("vec")\
    .setOutputCol("tensor")\
    .setReservedCols(["label"])\
    .linkFrom(test)

predictor = OnnxModelPredictBatchOp() \
    .setModelPath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/cnn_mnist_pytorch.onnx") \
```

```
    .setSelectedCols(["tensor"]) \
    .setInputNames(["0"]) \
    .setOutputNames(["21"]) \
    .setOutputSchemaStr("probabilities FLOAT_TENSOR")

test = predictor.linkFrom(test).select("label, probabilities")
test.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.VectorToTensorBatchOp;
import com.alibaba.alink.operator.batch.source.AkSourceBatchOp;
import org.junit.Test;

public class OnnxModelPredictBatchOpTest {
    @Test
    public void testOnnxModelPredictBatchOp() throws Exception {
        BatchOperator.setParallelism(1);
        BatchOperator <?> test = new AkSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

        test = new VectorToTensorBatchOp()
            .setTensorDataType("float")
            .setTensorShape(1, 1, 28, 28)
            .setSelectedCol("vec")
            .setOutputCol("tensor")
            .setReservedCols("label")
            .linkFrom(test);

        BatchOperator <?> predictor = new OnnxModelPredictBatchOp()
            .setModelPath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/cnn_mnist_pytorch.onnx")
            .setSelectedCols("tensor")
            .setInputNames("0")
            .setOutputNames("21")
            .setOutputSchemaStr("probabilities FLOAT_TENSOR");

        test = predictor.linkFrom(test).select("label, probabilities");
        test.print();
    }
}
```

PyTorch模型预测 (TorchModelPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.pytorch.TorchModelPredictBatchOp

Python 类名：TorchModelPredictBatchOp

功能介绍

加载 TorchScript 模型进行预测。

使用方式

模型路径 `modelPath` 需要是一个通过 `torch.jit` 导出的模型文件路径。

参与模型预测的数据通过参数 `selectedCols` 设置，需要注意以下几点：

- TorchScript 模型调用 `forward` 方法时是通过位置来传入参数的，所以 `selectedCols` 中各列的顺序是有意义的。
- 所选择的列的类型需要是 Alink `Tensor` 类型或者 4 种基本数据类型（`Long, Double, Boolean, String` 及其兼容类型），不接受其他类型。

模型输出信息通过参数 `outputSchemaStr` 指定，包括输出列名以及名称，需要注意以下几点：

- 输出列的数量需要与模型输出结果匹配。
- 输出类型可以是 Alink `Tensor` 类型或者 Alink 支持的类型，如果从模型预测输出的结果转换到指定类型失败那么将报错；暂不支持列表或字典类型。

组件使用的是 PyTorch 1.8.1 CPU 版本，如果需要使用 GPU 功能，可以自行替换插件文件。

在 Windows 下运行时，如果遇到 `UnsatisfiedLinkError`，请下载 Visual C++ 2015 Redistributable Packages 并重启，然后重新运行。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelPath	模型的URL路径	模型的URL路径	String	✓		
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如 "f0 string, f1 bigint, f2 double"	String	✓		

selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4
reservedCols	算法保留列名	算法保留列	String[]			null

模型路径说明

模型路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
test = AkSourceBatchOp()\
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

test = VectorToTensorBatchOp()\
    .setTensorDataType("float")\
    .setTensorShape([1, 1, 28, 28])\
    .setSelectedCol("vec")\
    .setOutputCol("tensor")\
    .setReservedCols(["label"])\
    .linkFrom(test)

predictor = TorchModelPredictBatchOp()\
    .setModelPath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/mnist_model_pytorch.pt")\
    .setSelectedCols(["tensor"])\
    .setOutputSchemaStr("probabilities FLOAT_TENSOR")

test = predictor.linkFrom(test).select("label, probabilities")
test.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.VectorToTensorBatchOp;
import com.alibaba.alink.operator.batch.pytorch.TorchModelPredictBatchOp;
import com.alibaba.alink.operator.batch.source.AkSourceBatchOp;
import org.junit.Test;

public class TorchModelPredictBatchOpTest {

    @Test
    public void testTorchModelPredictBatchOp() throws Exception {
        BatchOperator.setParallelism(1);
        BatchOperator <?> test = new AkSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

        test = new VectorToTensorBatchOp()
            .setTensorDataType("float")
            .setTensorShape(1, 1, 28, 28)
            .setSelectedCol("vec")
            .setOutputCol("tensor")
            .setReservedCols("label")
            .linkFrom(test);

        BatchOperator <?> predictor = new TorchModelPredictBatchOp()
            .setModelPath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/mnist_model_pytorch.pt")
            .setSelectedCols("tensor")
            .setOutputSchemaStr("probabilities FLOAT_TENSOR");

        test = predictor.linkFrom(test).select("label, probabilities");
        test.print();
    }
}
```

TF2表模型训练 (TF2TableModelTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.tensorflow.TF2TableModelTrainBatchOp

Python 类名：TF2TableModelTrainBatchOp

功能介绍

该组件支持用户传入 TensorFlow2 脚本，进行模型训练。

用户需要提供自己编写的 TensorFlow2 脚本文件。 关于脚本的编写，请先阅读下面的介绍。

脚本中必须要将模型保存为 SavedModel 格式，并导出到指定的目录下 (`TrainTaskConfig#saved_model_dir`)。

这个组件的输出可以接入 `TFTableModelPredictBatchOp/StreamOp` 进行预测。

TensorFlow 自定义脚本类组件的概况

Alink TensorFlow 自定义脚本类的组件，基于的是从 Alink 进程拉起 Python 进程、执行 Python 代码的能力。 通过这个能力，Alink 可以将数据传递给 Python 进程，在 Python 进程中执行自定义代码，然后将处理的结果返回给 Alink。

下表列出了自定义脚本类组件包含的具体组件及其区别：

	TF 版本	传入数据	传入参数类型	输出数据
TensorFlowBatchOp	1.15.2	批数据	BatchTaskConfig	自定义
TensorFlow2BatchOp	2.3.1	批数据	BatchTaskConfig	自定义
TensorFlowStreamOp	1.15.2	流数据	StreamTaskConfig	自定义
TensorFlow2StreamOp	2.3.1	流数据	StreamTaskConfig	自定义
TFTableModelTrainBatchOp	1.15.2	批数据	TrainTaskConfig	要求将训练模型保存到指定目录，无其他输出
TF2TableModelTrainBatchOp	2.3.1	批数据	TrainTaskConfig	要求将训练模型保存到指定目录，无其他输出

代码编写

用户可以提供多个 Python 文件， 其中一个为主文件，作为自定义脚本的入口。

参数传递

在主文件中，必须包含一个名为 `main` 的函数，接受一个参数，参数的类型根据使用的组件不同而不同，具体见上表。

结合这三种 Config 的 源码 ，对这三种 Config 的字段进行说明：

- 三者共有的字段：
  - `tf_context`：`TFContext` 类型，可以调用 `flink_stream_dataset()` 获取 一个 `TFRecordDataset` ，但这个数据集只能扫描一次；
  - `num_workers`： 总的 worker 数；
  - `cluster`： `TF_CONFIG` 中的 `cluster` 字段；
  - `task_type`： `TF_CONFIG` 中的 `task.type` 字段，取值有 'chief'、'worker' 或者 'ps'
  - `task_index`： `TF_CONFIG` 中的 `task.index` 字段；
  - `work_dir`： 工作目录；
  - `user_params`： 用户自定义参数，字典类型，对应为组件 `setUserParams` 的值。
- `BatchTaskConfig` 有的字段：
  - `dataset_file`： 将 `tf_context.flink_stream_dataset()` 得到的数据集写到本地文件中，从而可以读取多次；
  - `dataset_length`： 数据条数；
  - `output_writer`： 一个用于将数据写回 Alink 的工具，见下面说明。
- `StreamTaskConfig` 有的字段：
  - `dataset_fn`： 调用后返回的一个 `DataSet` ；
  - `output_writer`： 一个用于将数据写回 Alink 的工具，见下面说明。
- `TrainTaskConfig` 有的字段：
  - `dataset_file`： 将 `tf_context.flink_stream_dataset()` 得到的数据集写到本地文件中，从而可以读取多次；
  - `dataset_length`： 数据条数；
  - `saved_model_dir`： 训练完成后，必须将模型以 SavedModel 的格式导出到这个目录下。

数据输入

首先需要说明一下 TensorFlow 进程与数据集之间的关系。 当 Alink 作业本身的并发度大于 1 时，会有多个 Worker 同时执行任务，数据会根据任务的配置分布在各个 Worker 上。 在进入 TF 组件对应的任务时，各个 Worker 会启动一个 TF 进程，此时各个 Worker 会将其拥有的数据传递给 TF 进程。

这里**每个 TF 进程只能访问到它所在 Worker 的数据，而访问不了其他 Worker 的数据**。 这一点与某些 TensorFlow 分布式训练的写法不同：在一些 TensorFlow 分布式训练的写法中，数据集中存储在某些共享文件系统（例如 HDFS）上，整体作为模型训练数据，各个 TF 进程通过 shard 的形式读取部分数据。

从 Alink 传到 TensorFlow 进程的数据集为 `TFRecordDataset` 格式，每条数据是序列化后的 `tf.train.Example` 实例，可以通过 `tf.parse_single_example` 来进行解析。 其中， `parse_single_example` 的 `features` 参数与原本数据集的列名和类型对应，例如 `tf.int64` 、 `tf.float32` 、 `tf.string` 。

数据输出

通过 `output_writer` 可以从 TensorFlow 进程往 Alink 写回数据。 写出的数据需要是一个序列化后的 `tf.train.Example` 实例。 `Example` 实例所含的 `features` 需要与组件参数 `OutputSchemaStr` 中的列名和类型对应。

分布式训练

在代码中可以获取环境变量 `TF_CONFIG`，从而可以写分布式训练的代码，包括 Estimator + PS 与 AllReduce 的模式。

akdl 库

在 Alink 提供的 akdl 库 中，提供了一些便捷调用的函数，方便书写代码。具体例子可以参考 `alink_dl_predictors/predictor-tf/src/test/resources/tf_dnn_batch.py` 。

但需要注意的是： akdl 库内的写法采用的是 TF1 或者 TF2 中 TF1 兼容模式的写法，因此可能不能满足您的需要，例如 TF2 动态图运行模式等等。（即使仅引入 akdl 包中的头文件，也可能导致运行不了一些纯 TF2 写法的代码。）这个时候就需要您另外书写代码了。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
mainScriptFile	主脚本文件路径	主脚本文件路径，需要是参数 userFiles 中的一项，并且包含 main 函数	String	✓		
userFiles	所有自定义脚本文件的路径	所有自定义脚本文件的路径	String	✓		
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer			null
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer			null
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			""

selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
userParams	自定义参数	用户自定义参数，JSON 字典格式的字符串	String			"{}"

脚本路径说明

脚本路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
import json

source = RandomTableSourceBatchOp() \
    .setNumRows(100) \
    .setNumCols(10)

colNames = source.getColNames()
source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as
label")
label = "label"

userParams = {
    'featureCols': json.dumps(colNames),
    'labelCol': label,
    'batch_size': 16,
    'num_epochs': 1
}

tf2TableModelTrainBatchOp = TF2TableModelTrainBatchOp() \
    .setUserFiles(["https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/tf_dnn_train.py"]) \
    .setMainScriptFile("https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/tf_dnn_train.py") \
```

```
    .setUserParams(json.dumps(userParams)) \
    .linkFrom(source)
tf2TableModelTrainBatchOp.print()
```

Java 代码

```
import com.alibaba.alink.common.utils.JsonConverter;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.RandomTableSourceBatchOp;
import com.alibaba.alink.operator.batch.tensorflow.TFTableModelTrainBatchOp;
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

public class TFTableModelTrainBatchOpTest {

    @Test
    public void testTFTableModelTrainBatchOp() throws Exception {
        BatchOperator<?> source = new RandomTableSourceBatchOp()
            .setNumRows(100L)
            .setNumCols(10);

        String[] colNames = source.getColNames();
        source = source.select("*, case when RAND() > 0.5 then 1. else 0. end
as label");
        String label = "label";

        Map <String, Object> userParams = new HashMap <>();
        userParams.put("featureCols", JsonConverter.toJson(colNames));
        userParams.put("labelCol", label);
        userParams.put("batch_size", 16);
        userParams.put("num_epochs", 1);

        TF2TableModelTrainBatchOp tfTableModelTrainBatchOp = new
TF2TableModelTrainBatchOp()
            .setUserFiles(new String[] {"https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/tf_dnn_train.py"})
            .setMainScriptFile("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/tf_dnn_train.py")
            .setUserParams(JsonConverter.toJson(userParams))
            .linkFrom(source);
        tfTableModelTrainBatchOp.print();
    }
}
```

TF SavedModel模型预测 (TFSavedModelPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.tensorflow.TFSavedModelPredictBatchOp

Python 类名：TFSavedModelPredictBatchOp

功能介绍

该组件支持直接使用 SavedModel 进行预测。

模型路径需要时一个压缩文件，解压后能得到一个目录，目录内包含 SavedModel 的文件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelPath	模型的URL路径	模型的URL路径	String	✓		
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
graphDefTag	graph标签	graph标签	String			"serve"
inputSignatureDefs	输入SignatureDef	SavedModel模型的输入SignatureDef名，用逗号分隔，需要与输入列一一对应，默认与选择列相同	String[]			null

intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4
outputSignatureDefs	TF 输出 SignatureDef 名	模型的输出 SignatureDef 名，多个输出时用逗号分隔，并且与输出 Schema 一一对应，默认与输出 Schema 中的列名相同	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
signatureDefKey	signature标签	signature标签	String			"serving_default"

模型路径说明

模型路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
test = AkSourceBatchOp()\
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

test = VectorToTensorBatchOp()\
    .setTensorDataType("float")\
    .setTensorShape([1, 28, 28, 1])\
    .setSelectedCol("vec")\
```

```
    .setOutputCol("tensor")\
    .setReservedCols(["label"])\
    .linkFrom(test)

predictor = TFSavedModelPredictBatchOp()\
    .setModelPath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/mnist_model_tf.zip")\
    .setSelectedCols(["tensor"])\
    .setInputSignatureDefs(["input_1"])\
    .setOutputSignatureDefs(["output_1"])\
    .setOutputSchemaStr("probabilities FLOAT_TENSOR")

test = predictor.linkFrom(test).select("label, probabilities")
test.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.VectorToTensorBatchOp;
import com.alibaba.alink.operator.batch.source.AkSourceBatchOp;
import com.alibaba.alink.operator.batch.tensorflow.TFSavedModelPredictBatchOp;
import org.junit.Test;

public class TFSavedModelPredictBatchOpTest {

    @Test
    public void testTFSavedModelPredictBatchOp() throws Exception {
        BatchOperator.setParallelism(1);
        BatchOperator <?> test = new AkSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

        test = new VectorToTensorBatchOp()
            .setTensorDataType("float")
            .setTensorShape(1, 28, 28, 1)
            .setSelectedCol("vec")
            .setOutputCol("tensor")
            .setReservedCols("label")
            .linkFrom(test);

        BatchOperator <?> predictor = new TFSavedModelPredictBatchOp()
            .setModelPath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/mnist_model_tf.zip")
            .setSelectedCols("tensor")
            .setInputSignatureDefs(new String[] {"input_1"})
            .setOutputSignatureDefs(new String[] {"output_1"})
            .setOutputSchemaStr("probabilities FLOAT_TENSOR");
```

```
        test = predictor.linkFrom(test).select("label, probabilities");
        test.print();
    }
}
```

TF表模型预测 (TFTableModelPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.tensorflow.TFTableModelPredictBatchOp

Python 类名：TFTableModelPredictBatchOp

功能介绍

使用 `TFTableModelTrainBatchOp` 或者 `TF2TableModelTrainBatchOp` 训练产生的模型进行预测。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
graphDefTag	graph标签	graph标签	String			"serve"
inputSignatureDefs	输入SignatureDef	SavedModel模型的输入SignatureDef名，用逗号分隔，需要与输入列一一对应，默认与选择列相同	String[]			null
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4

outputSignatureDefs	TF 输出SignatureDef名	模型的输出SignatureDef名，多个输出时用逗号分隔，并且与输出Schema一一对应，默认与输出Schema中的列名相同	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
signatureDefKey	signature标签	signature标签	String			"serving_default"

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
import json

source = RandomTableSourceBatchOp() \
    .setNumRows(100) \
    .setNumCols(10)

colNames = source.getColNames()
source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as
label")
label = "label"

userParams = {
    'featureCols': json.dumps(colNames),
    'labelCol': label,
    'batch_size': 16,
    'num_epochs': 1
}

tfTableModelTrainBatchOp = TFTableModelTrainBatchOp() \
    .setUserFiles(["https://alink-release.oss-cn-beijing.aliyuncs.com/data-
```

```
files/tf_dnn_train.py"]) \
    .setMainScriptFile("https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/tf_dnn_train.py") \
    .setUserParams(json.dumps(userParams)) \
    .linkFrom(source)

tfTableModelPredictBatchOp = TFTableModelPredictBatchOp() \
    .setOutputSchemaStr("logits double") \
    .setOutputSignatureDefs(["logits"]) \
    .setSignatureDefKey("predict") \
    .setSelectedCols(colNames) \
    .linkFrom(tfTableModelTrainBatchOp, source)
tfTableModelPredictBatchOp.print()
```

Java 代码

```
import com.alibaba.alink.common.utils.JsonConverter;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.RandomTableSourceBatchOp;
import com.alibaba.alink.operator.batch.tensorflow.TFTableModelPredictBatchOp;
import com.alibaba.alink.operator.batch.tensorflow.TFTableModelTrainBatchOp;
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

public class TFTableModelPredictBatchOpTest {
    @Test
    public void testTFTableModelPredictBatchOp() throws Exception {
        BatchOperator<?> source = new RandomTableSourceBatchOp()
            .setNumRows(100L)
            .setNumCols(10);

        String[] colNames = source.getColNames();
        source = source.select("*, case when RAND() > 0.5 then 1. else 0. end
as label");
        String label = "label";

        Map <String, Object> userParams = new HashMap <>();
        userParams.put("featureCols", JsonConverter.toJson(colNames));
        userParams.put("labelCol", label);
        userParams.put("batch_size", 16);
        userParams.put("num_epochs", 1);

        TFTableModelTrainBatchOp tfTableModelTrainBatchOp = new
TFTableModelTrainBatchOp()
            .setUserFiles(new String[] {"https://alink-release.oss-cn-
```

```
beijing.aliyuncs.com/data-files/tf_dnn_train.py"})
            .setMainScriptFile("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/tf_dnn_train.py")
            .setUserParams(JsonConverter.toJson(userParams))
            .linkFrom(source);

        TFTableModelPredictBatchOp tfTableModelPredictBatchOp = new
TFTableModelPredictBatchOp()
            .setOutputSchemaStr("logits double")
            .setOutputSignatureDefs(new String[]{"logits"})
            .setSignatureDefKey("predict")
            .setSelectedCols(colNames)
            .linkFrom(tfTableModelTrainBatchOp, source);
        tfTableModelPredictBatchOp.print();
    }
}
```

运行结果

col0	col1	col2	col3	col4	col5	col6	col7	col8	c
0.7310	0.2405	0.6374	0.5504	0.5975	0.3332	0.3852	0.9848	0.8792	0.9
0.2750	0.1289	0.1466	0.0232	0.5467	0.9645	0.1045	0.6251	0.4108	0.1
0.9907	0.4872	0.7462	0.7332	0.8173	0.8389	0.5267	0.8993	0.1339	0.0
0.9786	0.7224	0.7150	0.1432	0.4630	0.0045	0.0715	0.3484	0.3388	0.8
0.9715	0.8657	0.6126	0.1790	0.2176	0.8545	0.0097	0.6923	0.7713	0.1
...	...	...	...	...	...	...	...	...	...

TF表模型训练 (TFTableModelTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.tensorflow.TFTableModelTrainBatchOp

Python 类名：TFTableModelTrainBatchOp

功能介绍

该组件支持用户传入 TensorFlow 脚本，进行模型训练。

用户需要提供自己编写的 TensorFlow 脚本文件。 关于脚本的编写，请先阅读下面的介绍。

脚本中必须要将模型保存为 SavedModel 格式，并导出到指定的目录下 (`TrainTaskConfig#saved_model_dir`)。

这个组件的输出可以接入 `TFTableModelPredictBatchOp/StreamOp` 进行预测。

TensorFlow 自定义脚本类组件的概况

Alink TensorFlow 自定义脚本类的组件，基于的是从 Alink 进程拉起 Python 进程、执行 Python 代码的能力。 通过这个能力，Alink 可以将数据传递给 Python 进程，在 Python 进程中执行自定义代码，然后将处理的结果返回给 Alink。

下表列出了自定义脚本类组件包含的具体组件及其区别：

	TF 版本	传入数据	传入参数类型	输出数据
TensorFlowBatchOp	1.15.2	批数据	BatchTaskConfig	自定义
TensorFlow2BatchOp	2.3.1	批数据	BatchTaskConfig	自定义
TensorFlowStreamOp	1.15.2	流数据	StreamTaskConfig	自定义
TensorFlow2StreamOp	2.3.1	流数据	StreamTaskConfig	自定义
TFTableModelTrainBatchOp	1.15.2	批数据	TrainTaskConfig	要求将训练模型保存到指定目录，无其他输出
TF2TableModelTrainBatchOp	2.3.1	批数据	TrainTaskConfig	要求将训练模型保存到指定目录，无其他输出

代码编写

用户可以提供多个 Python 文件， 其中一个为主文件，作为自定义脚本的入口。

参数传递

在主文件中，必须包含一个名为 `main` 的函数，接受一个参数，参数的类型根据使用的组件不同而不同，具体见上表。

结合这三种 Config 的 源码 ，对这三种 Config 的字段进行说明：

- 三者共有的字段：
  - `tf_context`：`TFContext` 类型，可以调用 `flink_stream_dataset()` 获取 一个 `TFRecordDataset` ，但这个数据集只能扫描一次；
  - `num_workers`： 总的 worker 数；
  - `cluster`： `TF_CONFIG` 中的 `cluster` 字段；
  - `task_type`： `TF_CONFIG` 中的 `task.type` 字段，取值有 'chief'、'worker' 或者 'ps'
  - `task_index`： `TF_CONFIG` 中的 `task.index` 字段；
  - `work_dir`： 工作目录；
  - `user_params`： 用户自定义参数，字典类型，对应为组件 `setUserParams` 的值。
- `BatchTaskConfig` 有的字段：
  - `dataset_file`： 将 `tf_context.flink_stream_dataset()` 得到的数据集写到本地文件中，从而可以读取多次；
  - `dataset_length`： 数据条数；
  - `output_writer`： 一个用于将数据写回 Alink 的工具，见下面说明。
- `StreamTaskConfig` 有的字段：
  - `dataset_fn`： 调用后返回的一个 `DataSet` ；
  - `output_writer`： 一个用于将数据写回 Alink 的工具，见下面说明。
- `TrainTaskConfig` 有的字段：
  - `dataset_file`： 将 `tf_context.flink_stream_dataset()` 得到的数据集写到本地文件中，从而可以读取多次；
  - `dataset_length`： 数据条数；
  - `saved_model_dir`： 训练完成后，必须将模型以 SavedModel 的格式导出到这个目录下。

数据输入

首先需要说明一下 TensorFlow 进程与数据集之间的关系。 当 Alink 作业本身的并发度大于 1 时，会有多个 Worker 同时执行任务，数据会根据任务的配置分布在各个 Worker 上。 在进入 TF 组件对应的任务时，各个 Worker 会启动一个 TF 进程，此时各个 Worker 会将其拥有的数据传递给 TF 进程。

这里**每个 TF 进程只能访问到它所在 Worker 的数据，而访问不了其他 Worker 的数据**。 这一点与某些 TensorFlow 分布式训练的写法不同：在一些 TensorFlow 分布式训练的写法中，数据集中存储在某些共享文件系统（例如 HDFS）上，整体作为模型训练数据，各个 TF 进程通过 shard 的形式读取部分数据。

从 Alink 传到 TensorFlow 进程的数据集为 `TFRecordDataset` 格式，每条数据是序列化后的 `tf.train.Example` 实例，可以通过 `tf.parse_single_example` 来进行解析。 其中， `parse_single_example` 的 `features` 参数与原本数据集的列名和类型对应，例如 `tf.int64` 、 `tf.float32` 、 `tf.string` 。

数据输出

通过 `output_writer` 可以从 TensorFlow 进程往 Alink 写回数据。 写出的数据需要是一个序列化后的 `tf.train.Example` 实例。 `Example` 实例所含的 `features` 需要与组件参数 `OutputSchemaStr` 中的列名和类型对应。

分布式训练

在代码中可以获取环境变量 `TF_CONFIG`，从而可以写分布式训练的代码，包括 Estimator + PS 与 AllReduce 的模式。

akdl 库

在 Alink 提供的 akdl 库 中，提供了一些便捷调用的函数，方便书写代码。具体例子可以参考 `alink_dl_predictors/predictor-tf/src/test/resources/tf_dnn_batch.py` 。

但需要注意的是： akdl 库内的写法采用的是 TF1 或者 TF2 中 TF1 兼容模式的写法，因此可能不能满足您的需要，例如 TF2 动态图运行模式等等。（即使仅引入 akdl 包中的头文件，也可能导致运行不了一些纯 TF2 写法的代码。）这个时候就需要您另外书写代码了。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
mainScriptFile	主脚本文件路径	主脚本文件路径，需要是参数 userFiles 中的一项，并且包含 main 函数	String	✓		
userFiles	所有自定义脚本文件的路径	所有自定义脚本文件的路径	String	✓		
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer			null
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer			null
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			""

selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
userParams	自定义参数	用户自定义参数，JSON 字典格式的字符串	String			"{}"

脚本路径说明

脚本路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
import json

source = RandomTableSourceBatchOp() \
    .setNumRows(100) \
    .setNumCols(10)

colNames = source.getColNames()
source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as
label")
label = "label"

userParams = {
    'featureCols': json.dumps(colNames),
    'labelCol': label,
    'batch_size': 16,
    'num_epochs': 1
}

tfTableModelTrainBatchOp = TFTableModelTrainBatchOp() \
    .setUserFiles(["https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/tf_dnn_train.py"]) \
    .setMainScriptFile("https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/tf_dnn_train.py") \
```

```
    .setUserParams(json.dumps(userParams)) \
    .linkFrom(source)
tfTableModelTrainBatchOp.print()
```

Java 代码

```
import com.alibaba.alink.common.utils.JsonConverter;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.RandomTableSourceBatchOp;
import com.alibaba.alink.operator.batch.tensorflow.TFTableModelTrainBatchOp;
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

public class TFTableModelTrainBatchOpTest {

    @Test
    public void testTFTableModelTrainBatchOp() throws Exception {
        BatchOperator<?> source = new RandomTableSourceBatchOp()
            .setNumRows(100L)
            .setNumCols(10);

        String[] colNames = source.getColNames();
        source = source.select("*, case when RAND() > 0.5 then 1. else 0. end
as label");
        String label = "label";

        Map <String, Object> userParams = new HashMap <>();
        userParams.put("featureCols", JsonConverter.toJson(colNames));
        userParams.put("labelCol", label);
        userParams.put("batch_size", 16);
        userParams.put("num_epochs", 1);

        TFTableModelTrainBatchOp tfTableModelTrainBatchOp = new
TFTableModelTrainBatchOp()
            .setUserFiles(new String[] {"https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/tf_dnn_train.py"})
            .setMainScriptFile("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/tf_dnn_train.py")
            .setUserParams(JsonConverter.toJson(userParams))
            .linkFrom(source);
        tfTableModelTrainBatchOp.print();
    }
}
```

TensorFlow2自定义脚本 (TensorFlow2BatchOp)

Java 类名：com.alibaba.alink.operator.batch.tensorflow.TensorFlow2BatchOp

Python 类名：TensorFlow2BatchOp

功能介绍

该组件支持用户传入 TensorFlow2 脚本，使用传入的批数据进行任意处理，并可以将数据输出回 Alink 端。

用户需要提供自己编写的 TensorFlow2 脚本文件。 关于脚本的编写，请先阅读下面的介绍。

TensorFlow 自定义脚本类组件的概况

Alink TensorFlow 自定义脚本类的组件，基于的是从 Alink 进程拉起 Python 进程、执行 Python 代码的能力。 通过这个能力，Alink 可以将数据传递给 Python 进程，在 Python 进程中执行自定义代码，然后将处理的结果返回给 Alink。

下表列出了自定义脚本类组件包含的具体组件及其区别：

	TF 版本	传入数据	传入参数类型	输出数据
TensorFlowBatchOp	1.15.2	批数据	BatchTaskConfig	自定义
TensorFlow2BatchOp	2.3.1	批数据	BatchTaskConfig	自定义
TensorFlowStreamOp	1.15.2	流数据	StreamTaskConfig	自定义
TensorFlow2StreamOp	2.3.1	流数据	StreamTaskConfig	自定义
TFTableModelTrainBatchOp	1.15.2	批数据	TrainTaskConfig	要求将训练模型保存到指定目录，无其他输出
TF2TableModelTrainBatchOp	2.3.1	批数据	TrainTaskConfig	要求将训练模型保存到指定目录，无其他输出

代码编写

用户可以提供多个 Python 文件， 其中一个为主文件，作为自定义脚本的入口。

参数传递

在主文件中，必须包含一个名为 `main` 的函数，接受一个参数，参数的类型根据使用的组件不同而不同，具体见上表。

结合这三种 Config 的 源码 ，对这三种 Config 的字段进行说明：

- 三者共有的字段：
  - `tf_context`：`TFContext` 类型，可以调用 `flink_stream_dataset()` 获取 一个 `TFRecordDataset` ，但这个数据集只能扫描一次；
  - `num_workers`： 总的 worker 数；
  - `cluster`： `TF_CONFIG` 中的 `cluster` 字段；
  - `task_type`： `TF_CONFIG` 中的 `task.type` 字段，取值有 'chief'、'worker' 或者 'ps'
  - `task_index`： `TF_CONFIG` 中的 `task.index` 字段；
  - `work_dir`： 工作目录；
  - `user_params`： 用户自定义参数，字典类型，对应为组件 `setUserParams` 的值。
- `BatchTaskConfig` 有的字段：
  - `dataset_file`： 将 `tf_context.flink_stream_dataset()` 得到的数据集写到本地文件中，从而可以读取多次；
  - `dataset_length`： 数据条数；
  - `output_writer`： 一个用于将数据写回 Alink 的工具，见下面说明。
- `StreamTaskConfig` 有的字段：
  - `dataset_fn`： 调用后返回的一个 `DataSet` ；
  - `output_writer`： 一个用于将数据写回 Alink 的工具，见下面说明。
- `TrainTaskConfig` 有的字段：
  - `dataset_file`： 将 `tf_context.flink_stream_dataset()` 得到的数据集写到本地文件中，从而可以读取多次；
  - `dataset_length`： 数据条数；
  - `saved_model_dir`： 训练完成后，必须将模型以 SavedModel 的格式导出到这个目录下。

数据输入

首先需要说明一下 TensorFlow 进程与数据集之间的关系。 当 Alink 作业本身的并发度大于 1 时，会有多个 Worker 同时执行任务，数据会根据任务的配置分布在各个 Worker 上。 在进入 TF 组件对应的任务时，各个 Worker 会启动一个 TF 进程，此时各个 Worker 会将其拥有的数据传递给 TF 进程。

这里**每个 TF 进程只能访问到它所在 Worker 的数据，而访问不了其他 Worker 的数据**。 这一点与某些 TensorFlow 分布式训练的写法不同：在一些 TensorFlow 分布式训练的写法中，数据集中存储在某些共享文件系统（例如 HDFS）上，整体作为模型训练数据，各个 TF 进程通过 shard 的形式读取部分数据。

从 Alink 传到 TensorFlow 进程的数据集为 `TFRecordDataset` 格式，每条数据是序列化后的 `tf.train.Example` 实例，可以通过 `tf.parse_single_example` 来进行解析。 其中， `parse_single_example` 的 `features` 参数与原本数据集的列名和类型对应，例如 `tf.int64` 、 `tf.float32` 、 `tf.string` 。

数据输出

通过 `output_writer` 可以从 TensorFlow 进程往 Alink 写回数据。 写出的数据需要是一个序列化后的 `tf.train.Example` 实例。 `Example` 实例所含的 `features` 需要与组件参数 `OutputSchemaStr` 中的列名和类型对应。

分布式训练

在代码中可以获取环境变量 `TF_CONFIG` ，从而可以写分布式训练的代码，包括 Estimator + PS 与 AllReduce 的模式。

akdl 库

在 Alink 提供的 akdl 库 中，提供了一些便捷调用的函数，方便书写代码。具体例子可以参考 `alink_dl_predictors/predictor-tf/src/test/resources/tf_dnn_batch.py` 。

但需要注意的是： akdl 库内的写法采用的是 TF1 或者 TF2 中 TF1 兼容模式的写法，因此可能不能满足您的需要，例如 TF2 动态图运行模式等等。（即使仅引入 akdl 包中的头文件，也可能导致运行不了一些纯 TF2 写法的代码。）这个时候就需要您另外书写代码了。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
mainScriptFile	主脚本文件路径	主脚本文件路径，需要是参数 userFiles 中的一项，并且包含 main 函数	String	✓		
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如 "f0 string, f1 bigint, f2 double"	String	✓		
userFiles	所有自定义脚本文件的路径	所有自定义脚本文件的路径	String	✓		
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer			null
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer			null

pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			""
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
userParams	自定义参数	用户自定义参数，JSON 字典格式的字符串	String			"{}"

脚本路径说明

脚本路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

输出数据说明

脚本中可以输出并传回 Alink 端，形成 Flink Table 的形式，进行后续的处理。

即使没有数据需要输出，参数中的输出数据 Schema 也需要填写个非空的形式，例如 `dummy string`。

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
import json

source = RandomTableSourceBatchOp() \
    .setNumRows(100) \
    .setNumCols(10)

colNames = source.getColNames()
source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as
label")
label = "label"
```

```
userParams = {
    'featureCols': json.dumps(colNames),
    'labelCol': label,
    'batch_size': 16,
    'num_epochs': 1
}
tensorFlow2BatchOp = TensorFlow2BatchOp() \
    .setUserFiles(["https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/tf_dnn_batch.py"]) \
    .setMainScriptFile("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/tf_dnn_batch.py") \
    .setUserParams(json.dumps(userParams)) \
    .setOutputSchemaStr("model_id long, model_info string") \
    .linkFrom(source)
tensorFlow2BatchOp.print()
```

Java 代码

```
import com.alibaba.alink.common.utils.JsonConverter;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.RandomTableSourceBatchOp;
import com.alibaba.alink.operator.batch.tensorflow.TensorFlow2BatchOp;
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

public class TensorFlow2BatchOpTest {

    @Test
    public void testTensorFlow2BatchOp() throws Exception {
        BatchOperator <?> source = new RandomTableSourceBatchOp()
            .setNumRows(100L)
            .setNumCols(10);

        String[] colNames = source.getColNames();
        source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as label");
        String label = "label";
        Map <String, Object> userParams = new HashMap <>();
        userParams.put("featureCols", JsonConverter.toJson(colNames));
        userParams.put("labelCol", label);
        userParams.put("batch_size", 16);
        userParams.put("num_epochs", 1);

        TensorFlow2BatchOp tensorFlow2BatchOp = new TensorFlow2BatchOp()
            .setUserFiles(new String[] {"https://alink-release.oss-cn-
```

```
beijing.aliyuncs.com/data-files/tf_dnn_batch.py"})
            .setMainScriptFile("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/tf_dnn_batch.py")
            .setUserParams(JsonConverter.toJson(userParams))
            .setOutputSchemaStr("model_id long, model_info string")
            .linkFrom(source);
        tensorFlow2BatchOp.print();
    }
}
```

TensorFlow自定义脚本 (TensorFlowBatchOp)

Java 类名：com.alibaba.alink.operator.batch.tensorflow.TensorFlowBatchOp

Python 类名：TensorFlowBatchOp

功能介绍

该组件支持用户传入 TensorFlow 脚本，使用传入的批数据进行任意处理，并可以将数据写回 Alink 端。

用户需要提供自己编写的 TensorFlow 脚本文件。 关于脚本的编写，请先阅读下面的介绍。

TensorFlow 自定义脚本类组件的概况

Alink TensorFlow 自定义脚本类的组件，基于的是从 Alink 进程拉起 Python 进程、执行 Python 代码的能力。 通过这个能力，Alink 可以将数据传递给 Python 进程，在 Python 进程中执行自定义代码，然后将处理的结果返回给 Alink。

下表列出了自定义脚本类组件包含的具体组件及其区别：

	TF 版本	传入数据	传入参数类型	输出数据
TensorFlowBatchOp	1.15.2	批数据	BatchTaskConfig	自定义
TensorFlow2BatchOp	2.3.1	批数据	BatchTaskConfig	自定义
TensorFlowStreamOp	1.15.2	流数据	StreamTaskConfig	自定义
TensorFlow2StreamOp	2.3.1	流数据	StreamTaskConfig	自定义
TFTableModelTrainBatchOp	1.15.2	批数据	TrainTaskConfig	要求将训练模型保存到指定目录，无其他输出
TF2TableModelTrainBatchOp	2.3.1	批数据	TrainTaskConfig	要求将训练模型保存到指定目录，无其他输出

代码编写

用户可以提供多个 Python 文件， 其中一个为主文件，作为自定义脚本的入口。

参数传递

在主文件中，必须包含一个名为 `main` 的函数，接受一个参数，参数的类型根据使用的组件不同而不同，具体见上表。

结合这三种 Config 的 源码 ，对这三种 Config 的字段进行说明：

- 三者共有的字段：
  - `tf_context`：`TFContext` 类型，可以调用 `flink_stream_dataset()` 获取 一个 `TFRecordDataset` ，但这个数据集只能扫描一次；
  - `num_workers`： 总的 worker 数；
  - `cluster`： `TF_CONFIG` 中的 `cluster` 字段；
  - `task_type`： `TF_CONFIG` 中的 `task.type` 字段，取值有 'chief'、'worker' 或者 'ps'
  - `task_index`： `TF_CONFIG` 中的 `task.index` 字段；
  - `work_dir`： 工作目录；
  - `user_params`： 用户自定义参数，字典类型，对应为组件 `setUserParams` 的值。
- `BatchTaskConfig` 有的字段：
  - `dataset_file`： 将 `tf_context.flink_stream_dataset()` 得到的数据集写到本地文件中，从而可以读取多次；
  - `dataset_length`： 数据条数；
  - `output_writer`： 一个用于将数据写回 Alink 的工具，见下面说明。
- `StreamTaskConfig` 有的字段：
  - `dataset_fn`： 调用后返回的一个 `DataSet` ；
  - `output_writer`： 一个用于将数据写回 Alink 的工具，见下面说明。
- `TrainTaskConfig` 有的字段：
  - `dataset_file`： 将 `tf_context.flink_stream_dataset()` 得到的数据集写到本地文件中，从而可以读取多次；
  - `dataset_length`： 数据条数；
  - `saved_model_dir`： 训练完成后，必须将模型以 SavedModel 的格式导出到这个目录下。

数据输入

首先需要说明一下 TensorFlow 进程与数据集之间的关系。 当 Alink 作业本身的并发度大于 1 时，会有多个 Worker 同时执行任务，数据会根据任务的配置分布在各个 Worker 上。 在进入 TF 组件对应的任务时，各个 Worker 会启动一个 TF 进程，此时各个 Worker 会将其拥有的数据传递给 TF 进程。

这里**每个 TF 进程只能访问到它所在 Worker 的数据，而访问不了其他 Worker 的数据**。 这一点与某些 TensorFlow 分布式训练的写法不同：在一些 TensorFlow 分布式训练的写法中，数据集中存储在某些共享文件系统（例如 HDFS）上，整体作为模型训练数据，各个 TF 进程通过 shard 的形式读取部分数据。

从 Alink 传到 TensorFlow 进程的数据集为 `TFRecordDataset` 格式，每条数据是序列化后的 `tf.train.Example` 实例，可以通过 `tf.parse_single_example` 来进行解析。 其中， `parse_single_example` 的 `features` 参数与原本数据集的列名和类型对应，例如 `tf.int64` 、 `tf.float32` 、 `tf.string` 。

数据输出

通过 `output_writer` 可以从 TensorFlow 进程往 Alink 写回数据。 写出的数据需要是一个序列化后的 `tf.train.Example` 实例。 `Example` 实例所含的 `features` 需要与组件参数 `OutputSchemaStr` 中的列名和类型对应。

分布式训练

在代码中可以获取环境变量 `TF_CONFIG` ，从而可以写分布式训练的代码，包括 Estimator + PS 与 AllReduce 的模式。

akdl 库

在 Alink 提供的 akdl 库 中，提供了一些便捷调用的函数，方便书写代码。具体例子可以参考 `alink_dl_predictors/predictor-tf/src/test/resources/tf_dnn_batch.py` 。

但需要注意的是： akdl 库内的写法采用的是 TF1 或者 TF2 中 TF1 兼容模式的写法，因此可能不能满足您的需要，例如 TF2 动态图运行模式等等。（即使仅引入 akdl 包中的头文件，也可能导致运行不了一些纯 TF2 写法的代码。）这个时候就需要您另外书写代码了。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
mainScriptFile	主脚本文件路径	主脚本文件路径，需要是参数 userFiles 中的一项，并且包含 main 函数	String	✓		
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如 "f0 string, f1 bigint, f2 double"	String	✓		
userFiles	所有自定义脚本文件的路径	所有自定义脚本文件的路径	String	✓		
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer			null
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer			null

pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			""
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
userParams	自定义参数	用户自定义参数，JSON 字典格式的字符串	String			"{}"

脚本路径说明

脚本路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

输出数据说明

脚本中可以输出并传回 Alink 端，形成 Flink Table 的形式，进行后续的处理。

即使没有数据需要输出，参数中的输出数据 Schema 也需要填写个非空的形式，例如 `dummy string`。

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
import json

source = RandomTableSourceBatchOp() \
    .setNumRows(100) \
    .setNumCols(10)

colNames = source.getColNames()
source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as
label");
label = "label"
```

```
userParams = {
    'featureCols': json.dumps(colNames),
    'labelCol': label,
    'batch_size': 16,
    'num_epochs': 1
}

tensorFlowBatchOp = TensorFlowBatchOp() \
    .setUserFiles(["https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/tf_dnn_batch.py"]) \
    .setMainScriptFile("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/tf_dnn_batch.py") \
    .setUserParams(json.dumps(userParams)) \
    .setOutputSchemaStr("model_id long, model_info string") \
    .linkFrom(source)
tensorFlowBatchOp.print()
```

Java 代码

```
import com.alibaba.alink.common.utils.JsonConverter;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.RandomTableSourceBatchOp;
import com.alibaba.alink.operator.batch.tensorflow.TensorFlowBatchOp;
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

public class TensorFlowBatchOpTest {

    @Test
    public void testTensorFlowBatchOp() throws Exception {
        BatchOperator <?> source = new RandomTableSourceBatchOp()
            .setNumRows(100L)
            .setNumCols(10);

        String[] colNames = source.getColNames();
        source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as label");
        String label = "label";

        Map <String, Object> userParams = new HashMap <>();
        userParams.put("featureCols", JsonConverter.toJson(colNames));
        userParams.put("labelCol", label);
        userParams.put("batch_size", 16);
        userParams.put("num_epochs", 1);
```

```
        TensorFlowBatchOp tensorFlowBatchOp = new TensorFlowBatchOp()
            .setUserFiles(new String[] {"https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/tf_dnn_batch.py"})
            .setMainScriptFile("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/tf_dnn_batch.py")
            .setUserParams(JsonConverter.toJson(userParams))
            .setOutputSchemaStr("model_id long, model_info string")
            .linkFrom(source);
        tensorFlowBatchOp.print();
    }
}
```

Arima (ArimaBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.ArimaBatchOp

Python 类名：ArimaBatchOp

功能介绍

给定分组，对每一组的数据进行Arima时间序列预测，给出下一时间段的结果。

算法原理

Arima全称为自回归积分滑动平均模型(Autoregressive Integrated Moving Average Model,简记ARIMA)，是由博克思(Box)和詹金斯(Jenkins)于70年代初提出一著名时间序列预测方法，所以又称为box-jenkins模型、博克思-詹金斯法.

Arima 详细介绍请见链接 https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
order	模型(p, d, q)	模型(p, d, q)	int[]	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
valueCol	value列，类型为MTable	value列，类型为MTable	String	✓	所选列类型为[M_TABLE]	
estMethod	估计方法	估计方法	String		"Mom", "Hr", "Css", "CssMle"	"CssMle"
predictNum	预测条数	预测条数	Integer			1
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			

reservedCols	算法保留列名	算法保留列	String[]			null
seasonalOrder	季节模型(p, d, q)	季节模型(p, d, q)	int[]			null
seasonalPeriod	季节周期	季节周期	Integer		[1, +inf)	1
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
            [1,  datetime.datetime.fromtimestamp(1), 10.0],
            [1,  datetime.datetime.fromtimestamp(2), 11.0],
            [1,  datetime.datetime.fromtimestamp(3), 12.0],
            [1,  datetime.datetime.fromtimestamp(4), 13.0],
            [1,  datetime.datetime.fromtimestamp(5), 14.0],
            [1,  datetime.datetime.fromtimestamp(6), 15.0],
            [1,  datetime.datetime.fromtimestamp(7), 16.0],
            [1,  datetime.datetime.fromtimestamp(8), 17.0],
            [1,  datetime.datetime.fromtimestamp(9), 18.0],
            [1,  datetime.datetime.fromtimestamp(10), 19.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val
double', op_type='batch')

source.link(
        GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("id, mtable_agg(ts, val) as data")
        ).link(ArimaBatchOp()
            .setValueCol("data")
```

```
            .setOrder([1, 2, 1])
            .setPredictNum(12)
            .setPredictionCol("predict")
        ).print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.timeseries;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

public class ArimaBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
        List <Row> mTableData = Arrays.asList(
            Row.of(1, new Timestamp(1), 10.0),
            Row.of(1, new Timestamp(2), 11.0),
            Row.of(1, new Timestamp(3), 12.0),
            Row.of(1, new Timestamp(4), 13.0),
            Row.of(1, new Timestamp(5), 14.0),
            Row.of(1, new Timestamp(6), 15.0),
            Row.of(1, new Timestamp(7), 16.0),
            Row.of(1, new Timestamp(8), 17.0),
            Row.of(1, new Timestamp(9), 18.0),
            Row.of(1, new Timestamp(10), 19.0)
        );

        MemSourceBatchOp source = new MemSourceBatchOp(mTableData, new String[]
{"id", "ts", "val"});

        source.link(
            new GroupByBatchOp()
                .setGroupByPredicate("id")
                .setSelectClause("mtable_agg(ts, val) as data")
        ).link(new ArimaBatchOp()
            .setValueCol("data")
            .setOrder(new int[] {1, 2, 1})
```

```
            .setPredictNum(12)
            .setPredictionCol("predict")
        ).print();
    }
}
```

运行结果

id	data	
1	{"data":{"ts":["1970-01-01 08:00:00.001","1970-01-01 08:00:00.002","1970-01-01 08:00:00.003","1970-01-01 08:00:00.004","1970-01-01 08:00:00.005","1970-01-01 08:00:00.006","1970-01-01 08:00:00.007","1970-01-01 08:00:00.008","1970-01-01 08:00:00.009","1970-01-01 08:00:00.01"],"val":[10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0]},"schema":"ts TIMESTAMP,val DOUBLE"}	{"data":{"ts":["1970-01-01 08:00:0 01-01 08:00:00.013","1970-01-01 08:00:00.015","1970-01-01 08:00 08:00:00.017","1970-01-01 08:00 08:00:00.019","1970-01-01 08:00 08:00:00.021","1970-01-01 08:00 [20.0,21.0,22.0,23.0,24.0,25.0,26 TIMESTAMP,val DOUBLE"}

t

AutoArima (AutoArimaBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.AutoArimaBatchOp

Python 类名：AutoArimaBatchOp

功能介绍

给定分组，对每一组的数据进行AutoArima时间序列预测，给出下一时间段的结果。

算法原理

Arima全称为自回归积分滑动平均模型(Autoregressive Integrated Moving Average Model,简记ARIMA)，是由博克思(Box)和詹金斯(Jenkins)于70年代初提出一著名时间序列预测方法，所以又称为box-jenkins模型、博克思-詹金斯法.

Arima 详细介绍请见链接 https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average

AutoArima是只需要指定MaxOrder, 不需要指定p/d/q, 对每个分组分别计算出最优的参数，给出预测结果。

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
valueCol	value列，类型为MTable	value列，类型为MTable	String	✓	所选列类型为[M_TABLE]	
d	d	d	Integer			-1
estMethod	估计方法	估计方法	String		"Mom", "Hr", "Css", "CssMle"	"CssMle"
icType	评价指标	评价指标	String		"AIC", "BIC", "HQIC"	"AIC"
maxOrder	模型(p, q)上限	模型(p, q)上限	Integer			10

maxSeasonalOrder	季节模型(p, q)上限	季节模型(p, q)上限	Integer			1
predictNum	预测条数	预测条数	Integer			1
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
seasonalPeriod	季节周期	季节周期	Integer		[1, +inf)	1
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
            [1,  datetime.datetime.fromtimestamp(1), 10.0],
            [1,  datetime.datetime.fromtimestamp(2), 11.0],
            [1,  datetime.datetime.fromtimestamp(3), 12.0],
            [1,  datetime.datetime.fromtimestamp(4), 13.0],
            [1,  datetime.datetime.fromtimestamp(5), 14.0],
            [1,  datetime.datetime.fromtimestamp(6), 15.0],
            [1,  datetime.datetime.fromtimestamp(7), 16.0],
            [1,  datetime.datetime.fromtimestamp(8), 17.0],
            [1,  datetime.datetime.fromtimestamp(9), 18.0],
            [1,  datetime.datetime.fromtimestamp(10), 19.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val
double', op_type='batch')

source.link(
        GroupByBatchOp()
```

```
            .setGroupByPredicate("id")
            .setSelectClause("id, mtable_agg(ts, val) as data")
        ).link(AutoArimaBatchOp()
            .setValueCol("data")
            .setPredictionCol("pred")
            .setPredictNum(12)
        ).print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.timeseries;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

public class AutoArimaBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
        List <Row> mTableData = Arrays.asList(
            Row.of(1, new Timestamp(1), 10.0),
            Row.of(1, new Timestamp(2), 11.0),
            Row.of(1, new Timestamp(3), 12.0),
            Row.of(1, new Timestamp(4), 13.0),
            Row.of(1, new Timestamp(5), 14.0),
            Row.of(1, new Timestamp(6), 15.0),
            Row.of(1, new Timestamp(7), 16.0),
            Row.of(1, new Timestamp(8), 17.0),
            Row.of(1, new Timestamp(9), 18.0),
            Row.of(1, new Timestamp(10), 19.0)
        );

        MemSourceBatchOp source = new MemSourceBatchOp(mTableData, new String[]
{"id", "ts", "val"});

        source.link(
            new GroupByBatchOp()
                .setGroupByPredicate("id")
                .setSelectClause("mtable_agg(ts, val) as data")
```

```
        ).link(new AutoArimaBatchOp()
            .setValueCol("data")
            .setPredictionCol("pred")
            .setPredictNum(12)
        ).print();
    }
}
```

运行结果

id	data	
1	{"data":{"ts":["1970-01-01 08:00:00.001","1970-01-01 08:00:00.002","1970-01-01 08:00:00.003","1970-01-01 08:00:00.004","1970-01-01 08:00:00.005","1970-01-01 08:00:00.006","1970-01-01 08:00:00.007","1970-01-01 08:00:00.008","1970-01-01 08:00:00.009","1970-01-01 08:00:00.01"],"val":[10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0]},"schema":"ts TIMESTAMP,val DOUBLE"}	{"data":{"ts":["1970-01-01 08:00:0 08:00:00.02","1970-01-01 08:00:0 [20.000043772632726,21.000149 TIMESTAMP,val DOUBLE"}

AutoGarch (AutoGarchBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.AutoGarchBatchOp

Python 类名：AutoGarchBatchOp

功能介绍

给定分组，对每一组的数据使用AutoGarch进行时间序列预测。

算法原理

garch(Generalized AutoRegressive Conditional Heteroskedasticity) 又称广义自回归条件异方差模型,

garch 详细介绍请见链接 https://en.wikipedia.org/wiki/Autoregressive_conditional_heteroskedasticity#GARCH

garch是只需要指定MaxOrder, 不需要指定p/d/q, 对每个分组分别计算出最优的参数，给出预测结果。

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
valueCol	value列，类型为MTable	value列，类型为MTable	String	✓	所选列类型为[M_TABLE]	
icType	评价指标	评价指标	String		"AIC", "BIC", "HQIC"	"AIC"
ifGARCH11	是否用garch11	是否用garch11	Boolean			true
maxOrder	模型(p, q)上限	模型(p, q)上限	Integer			10
minusMean	是否减去均值	是否减去均值	Boolean			true
predictNum	预测条数	预测条数	Integer			1

predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
            [1,  datetime.datetime.fromtimestamp(1), 10.0],
            [1,  datetime.datetime.fromtimestamp(2), 11.0],
            [1,  datetime.datetime.fromtimestamp(3), 12.0],
            [1,  datetime.datetime.fromtimestamp(4), 13.0],
            [1,  datetime.datetime.fromtimestamp(5), 14.0],
            [1,  datetime.datetime.fromtimestamp(6), 15.0],
            [1,  datetime.datetime.fromtimestamp(7), 16.0],
            [1,  datetime.datetime.fromtimestamp(8), 17.0],
            [1,  datetime.datetime.fromtimestamp(9), 18.0],
            [1,  datetime.datetime.fromtimestamp(10), 19.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val
double', op_type='batch')

source.link(
        GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("id, mtable_agg(ts, val) as data")
        ).link(
        AutoGarchBatchOp()
                .setValueCol("data")
                .setIcType("AIC")
```

```
                .setPredictNum(10)
                .setMaxOrder(4)
                .setIfGARCH11(False)
                .setMinusMean(False)
                .setPredictionCol("pred")
        ).print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.timeseries;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

public class AutoGarchBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
        List <Row> mTableData = Arrays.asList(
            Row.of(1, new Timestamp(1), 10.0),
            Row.of(1, new Timestamp(2), 11.0),
            Row.of(1, new Timestamp(3), 12.0),
            Row.of(1, new Timestamp(4), 13.0),
            Row.of(1, new Timestamp(5), 14.0),
            Row.of(1, new Timestamp(6), 15.0),
            Row.of(1, new Timestamp(7), 16.0),
            Row.of(1, new Timestamp(8), 17.0),
            Row.of(1, new Timestamp(9), 18.0),
            Row.of(1, new Timestamp(10), 19.0)
        );

        MemSourceBatchOp source = new MemSourceBatchOp(mTableData, new String[]
{"id", "ts", "val"});

        source.link(
            new GroupByBatchOp()
                .setGroupByPredicate("id")
                .setSelectClause("mtable_agg(ts, val) as data")
        ).link(
```

```
            new AutoGarchBatchOp()
                .setValueCol("data")
                .setIcType("AIC")
                .setPredictNum(10)
                .setMaxOrder(4)
                .setIfGARCH11(false)
                .setMinusMean(false)
                .setPredictionCol("pred")
        ).print();

    }

}
```

运行结果

id	data	pred
1	{"data":{"ts":["1970-01-01 08:00:00.001","1970-01-01 08:00:00.002","1970-01-01 08:00:00.003","1970-01-01 08:00:00.004","1970-01-01 08:00:00.005","1970-01-01 08:00:00.006","1970-01-01 08:00:00.007","1970-01-01 08:00:00.008","1970-01-01 08:00:00.009","1970-01-01 08:00:00.01"],"val":[10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0]},"schema":"ts TIMESTAMP,val DOUBLE"}	null

DeepAR预测 (DeepARPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.DeepARPredictBatchOp

Python 类名：DeepARPredictBatchOp

功能介绍

使用 DeepAR 进行时间序列训练和预测。

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
valueCol	value列，类型为MTable	value列，类型为MTable	String	✓	所选列类型为[M_TABLE]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictNum	预测条数	预测条数	Integer			1
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
    [0,  datetime.datetime.fromisoformat('2021-11-01 00:00:00'), 100.0],
    [0,  datetime.datetime.fromisoformat('2021-11-02 00:00:00'), 100.0],
    [0,  datetime.datetime.fromisoformat('2021-11-03 00:00:00'), 100.0],
    [0,  datetime.datetime.fromisoformat('2021-11-04 00:00:00'), 100.0],
    [0,  datetime.datetime.fromisoformat('2021-11-05 00:00:00'), 100.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, series
double', op_type='batch')

deepARTrainBatchOp = DeepARTrainBatchOp()\
    .setTimeCol("ts")\
    .setSelectedCol("series")\
    .setNumEpochs(10)\
    .setWindow(2)\
    .setStride(1)

groupByBatchOp = GroupByBatchOp()\
    .setGroupByPredicate("id")\
    .setSelectClause("mtable_agg(ts, series) as mtable_agg_series")

deepARPredictBatchOp = DeepARPredictBatchOp()\
            .setPredictNum(2)\
            .setPredictionCol("pred")\
            .setValueCol("mtable_agg_series")

deepARPredictBatchOp\
    .linkFrom(
        deepARTrainBatchOp.linkFrom(source),
        groupByBatchOp.linkFrom(source)
    )\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
```

```
import com.alibaba.alink.operator.batch.timeseries.DeepARPredictBatchOp;
import com.alibaba.alink.operator.batch.timeseries.DeepARTrainBatchOp;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

public class DeepARTrainBatchOpTest {

    @Test
    public void testDeepARTrainBatchOp() throws Exception {
        BatchOperator.setParallelism(1);

        List <Row> data = Arrays.asList(
            Row.of(0, Timestamp.valueOf("2021-11-01 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-02 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-03 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-04 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-05 00:00:00"), 100.0)
        );

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "id int,
ts timestamp, series double");

        DeepARTrainBatchOp deepARTrainBatchOp = new DeepARTrainBatchOp()
            .setTimeCol("ts")
            .setSelectedCol("series")
            .setNumEpochs(10)
            .setWindow(2)
            .setStride(1);

        GroupByBatchOp groupByBatchOp = new GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("mtable_agg(ts, series) as mtable_agg_series");

        DeepARPredictBatchOp deepARPredictBatchOp = new DeepARPredictBatchOp()
            .setPredictNum(2)
            .setPredictionCol("pred")
            .setValueCol("mtable_agg_series");

        deepARPredictBatchOp
            .linkFrom(
                deepARTrainBatchOp.linkFrom(memSourceBatchOp),
                groupByBatchOp.linkFrom(memSourceBatchOp)
            )
            .print();
```

```
    }
}
```

运行结果

| id | mtable_agg_series | pred | |----+----------+----------| | 0 | {"data":{"ts":["2021-11-01 00:00:00.0","2021-11-02 00:00:00.0","2021-11-03 00:00:00.0","2021-11-04 00:00:00.0","2021-11-05 00:00:00.0"],"series":[100.0,100.0,100.0,100.0,100.0]},"schema":"ts TIMESTAMP,series DOUBLE"} | {"data":{"ts":["2021-11-06 00:00:00.0","2021-11-07 00:00:00.0"],"series":[31.424224853515625,39.10265350341797]},"schema":"ts TIMESTAMP,series DOUBLE"} |

DeepAR训练 (DeepARTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.DeepARTrainBatchOp

Python 类名：DeepARTrainBatchOp

功能介绍

使用 DeepAR 进行时间序列训练和预测。

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

参数说明

名称	中文名称	描述	类型	是否必须?	取值范
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为TensorFlow中 `Estimator` 的 `model_dir` 传入，需要为所有worker 都能访问到的目录	String	✓	
timeCol	时间戳列(TimeStamp)	时间戳列(TimeStamp)	String	✓	所选列类型为[TIMESTAMI
batchSize	数据批大小	数据批大小	Integer		
intraOpParallelism	Op 间并发度	Op 间并发度	Integer		
learningRate	学习率	学习率	Double		
numEpochs	epoch数	epoch数	Integer		

numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer		
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer		
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String		

removeCheckpointBeforeTraining	是否在训练前移除checkpoint相关文件	是否在训练前移除checkpoint相关文件用于重新训练，只会删除必要的文件	Boolean		
selectedCol	计算列对应的列名	计算列对应的列名, 默认值是null	String		
stride	horizon大小	horizon大小	Integer		[1, +inf)
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VE SPARSE_VE STRING, VE
window	窗口大小	窗口大小	Integer		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
    [0,  datetime.datetime.fromisoformat('2021-11-01 00:00:00'), 100.0],
    [0,  datetime.datetime.fromisoformat('2021-11-02 00:00:00'), 100.0],
    [0,  datetime.datetime.fromisoformat('2021-11-03 00:00:00'), 100.0],
    [0,  datetime.datetime.fromisoformat('2021-11-04 00:00:00'), 100.0],
    [0,  datetime.datetime.fromisoformat('2021-11-05 00:00:00'), 100.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, series
double', op_type='batch')

deepARTrainBatchOp = DeepARTrainBatchOp()\
```

```
    .setTimeCol("ts")\
    .setSelectedCol("series")\
    .setNumEpochs(10)\
    .setWindow(2)\
    .setStride(1)

groupByBatchOp = GroupByBatchOp()\
    .setGroupByPredicate("id")\
    .setSelectClause("mtable_agg(ts, series) as mtable_agg_series")

deepARPredictBatchOp = DeepARPredictBatchOp()\
            .setPredictNum(2)\
            .setPredictionCol("pred")\
            .setValueCol("mtable_agg_series")

deepARPredictBatchOp\
    .linkFrom(
        deepARTrainBatchOp.linkFrom(source),
        groupByBatchOp.linkFrom(source)
    )\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.operator.batch.timeseries.DeepARPredictBatchOp;
import com.alibaba.alink.operator.batch.timeseries.DeepARTrainBatchOp;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

public class DeepARTrainBatchOpTest {

    @Test
    public void testDeepARTrainBatchOp() throws Exception {
        BatchOperator.setParallelism(1);

        List <Row> data = Arrays.asList(
            Row.of(0, Timestamp.valueOf("2021-11-01 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-02 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-03 00:00:00"), 100.0),
```

```
            Row.of(0, Timestamp.valueOf("2021-11-04 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-05 00:00:00"), 100.0)
        );

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "id int,
ts timestamp, series double");

        DeepARTrainBatchOp deepARTrainBatchOp = new DeepARTrainBatchOp()
            .setTimeCol("ts")
            .setSelectedCol("series")
            .setNumEpochs(10)
            .setWindow(2)
            .setStride(1);

        GroupByBatchOp groupByBatchOp = new GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("mtable_agg(ts, series) as mtable_agg_series");

        DeepARPredictBatchOp deepARPredictBatchOp = new DeepARPredictBatchOp()
            .setPredictNum(2)
            .setPredictionCol("pred")
            .setValueCol("mtable_agg_series");

        deepARPredictBatchOp
            .linkFrom(
                deepARTrainBatchOp.linkFrom(memSourceBatchOp),
                groupByBatchOp.linkFrom(memSourceBatchOp)
            )
            .print();
    }
}
```

运行结果

| id | mtable_agg_series | pred | |----+----------+----------+| 0 | {"data":{"ts":["2021-11-01 00:00:00.0","2021-11-02 00:00:00.0","2021-11-03 00:00:00.0","2021-11-04 00:00:00.0","2021-11-05 00:00:00.0"],"series":[100.0,100.0,100.0,100.0,100.0]},"schema":"ts TIMESTAMP,series DOUBLE"} | {"data":{"ts":["2021-11-06 00:00:00.0","2021-11-07 00:00:00.0"],"series":[31.424224853515625,39.10265350341797]},"schema":"ts TIMESTAMP,series DOUBLE"} |

HoltWinters (HoltWintersBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.HoltWintersBatchOp

Python 类名：HoltWintersBatchOp

功能介绍

给定分组，对每一组的数据使用HoltWinters进行时间序列预测。

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

算法原理

HoltWinters由Holt和Winters提出的三次指数平滑算法，又称holt-winters,

HoltWinters 详细介绍请见链接 https://en.wikipedia.org/wiki/Exponential_smoothing

holt-winters支持2种季节类型： additive 和 multiplicative

- additive seasonal holt-winters

$$
\begin{aligned}
y'_{t+h|t} &= l_t + hb_t + s_{(t+h) \bmod p} \\
l_t &= \alpha(y_t - s_{t-p}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\
b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\
s_t &= \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-p}
\end{aligned}
$$

- multiplicative seasonal holt_winters

$$
\begin{aligned}
y'_{t+h|t} &= (l_t + hb_t)s_{(t+h) \bmod p} \\
l_t &= \alpha(y_t \div s_{t-p}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\
b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\
s_t &= \gamma(y_t \div (l_{t-1} + b_{t-1})) + (1-\gamma)s_{t-p}
\end{aligned}
$$

- 其中，
  - smoothValue（l、b、s）分别表示level，trend，seasonal
  - smoothParameter(α、β、γ)分别表示alpha，beta，gamma
  - t表示当前时刻，h表示要预测h步
  - p表示period或frequency，时间序列的周期

使用方式

- 第一步，将每组数据(时间列和数据列) 聚合成MTable.

```
GroupByBatchOp()
   .setGroupByPredicate("id")
   .setSelectClause("id, mtable_agg(ts, val) as data")
```

- 第二步，使用时间序列方法进行预测，预测结果也是MTable。
- 第三步，使用FlattenMTableBatchOp，将MTable转换成列，

```
FlattenMTableBatchOp()
    .setReservedCols(["id", "predict"])
    .setSelectedCol("predict")
    .setSchemaStr("ts timestamp, val double")
```

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
valueCol	value列，类型为MTable	value列，类型为MTable	String	✓	所选列类型为[M_TABLE]	
alpha	alpha	alpha	Double		[0.0, 1.0]	0.3
beta	beta	beta	Double		[0.0, 1.0]	0.1
doSeasonal	时间是否具有季节性	时间是否具有季节性	Boolean			false
doTrend	时间是否具有趋势性	时间是否具有趋势性	Boolean			false
frequency	时序频率	时序频率	Integer		[1, +inf)	10
gamma	gamma	gamma	Double		[0.0, 1.0]	0.1
levelStart	level初始值	level初始值	Double			
predictNum	预测条数	预测条数	Integer			1
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			

reservedCols	算法保留列名	算法保留列	String[]			null
seasonalStart	seasonal初始值	seasonal初始值	double[]			
seasonalType	季节类型	季节类型	String		"MULTIPLICATIVE", "ADDITIVE"	"ADDITIV
trendStart	trend初始值	trend初始值	Double			
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
            [1,  datetime.datetime.fromtimestamp(1), 10.0],
            [1,  datetime.datetime.fromtimestamp(2), 11.0],
            [1,  datetime.datetime.fromtimestamp(3), 12.0],
            [1,  datetime.datetime.fromtimestamp(4), 13.0],
            [1,  datetime.datetime.fromtimestamp(5), 14.0],
            [1,  datetime.datetime.fromtimestamp(6), 15.0],
            [1,  datetime.datetime.fromtimestamp(7), 16.0],
            [1,  datetime.datetime.fromtimestamp(8), 17.0],
            [1,  datetime.datetime.fromtimestamp(9), 18.0],
            [1,  datetime.datetime.fromtimestamp(10), 19.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val
double', op_type='batch')

source.link(
        GroupByBatchOp()
```

```
            .setGroupByPredicate("id")
            .setSelectClause("id, mtable_agg(ts, val) as data")
        ).link(HoltWintersBatchOp()
            .setValueCol("data")
            .setPredictionCol("pred")
            .setPredictNum(12)
        ).print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.timeseries;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

public class HoltWintersBatchOpTest {
    @Test
    public void test() throws Exception {
        List<Row> mTableData = Arrays.asList(
            Row.of(1, new Timestamp(1), 10.0),
            Row.of(1, new Timestamp(2), 11.0),
            Row.of(1, new Timestamp(3), 12.0),
            Row.of(1, new Timestamp(4), 13.0),
            Row.of(1, new Timestamp(5), 14.0),
            Row.of(1, new Timestamp(6), 15.0),
            Row.of(1, new Timestamp(7), 16.0),
            Row.of(1, new Timestamp(8), 17.0),
            Row.of(1, new Timestamp(9), 18.0),
            Row.of(1, new Timestamp(10), 19.0)
        );

        MemSourceBatchOp source = new MemSourceBatchOp(mTableData, new String[]
{"id", "ts", "val"});

        source.link(
            new GroupByBatchOp()
                .setGroupByPredicate("id")
                .setSelectClause("mtable_agg(ts, val) as data")
        ).link(new HoltWintersBatchOp()
            .setValueCol("data")
```

```
            .setPredictionCol("pred")
            .setPredictNum(12)
        ).print();
    }
}
```

运行结果

id	data	
1	{"data":{"ts":["1970-01-01 08:00:00.001","1970-01-01 08:00:00.002","1970-01-01 08:00:00.003","1970-01-01 08:00:00.004","1970-01-01 08:00:00.005","1970-01-01 08:00:00.006","1970-01-01 08:00:00.007","1970-01-01 08:00:00.008","1970-01-01 08:00:00.009","1970-01-01 08:00:00.01"],"val":[10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0]},"schema":"ts TIMESTAMP,val DOUBLE"}	{"data":{"ts":["1970-01-01 08:00:0 01-01 08:00:00.013","1970-01-01 08:00:00.015","1970-01-01 08:00 08:00:00.017","1970-01-01 08:00 08:00:00.019","1970-01-01 08:00 08:00:00.021","1970-01-01 08:00 [19.0,19.0,19.0,19.0,19.0,19.0,19 TIMESTAMP,val DOUBLE"}

LSTNet预测 (LSTNetPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.LSTNetPredictBatchOp

Python 类名：LSTNetPredictBatchOp

功能介绍

使用 LSTNet 进行时间序列训练和预测。

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
valueCol	value列，类型为MTable	value列，类型为MTable	String	✓	所选列类型为[M_TABLE]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictNum	预测条数	预测条数	Integer			1
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
    [0, datetime.datetime.fromisoformat("2021-11-01 00:00:00"), 100.0],
    [0, datetime.datetime.fromisoformat("2021-11-02 00:00:00"), 200.0],
    [0, datetime.datetime.fromisoformat("2021-11-03 00:00:00"), 300.0],
    [0, datetime.datetime.fromisoformat("2021-11-04 00:00:00"), 400.0],
    [0, datetime.datetime.fromisoformat("2021-11-06 00:00:00"), 500.0],
    [0, datetime.datetime.fromisoformat("2021-11-07 00:00:00"), 600.0],
    [0, datetime.datetime.fromisoformat("2021-11-08 00:00:00"), 700.0],
    [0, datetime.datetime.fromisoformat("2021-11-09 00:00:00"), 800.0],
    [0, datetime.datetime.fromisoformat("2021-11-10 00:00:00"), 900.0],
    [0, datetime.datetime.fromisoformat("2021-11-11 00:00:00"), 800.0],
    [0, datetime.datetime.fromisoformat("2021-11-12 00:00:00"), 700.0],
    [0, datetime.datetime.fromisoformat("2021-11-13 00:00:00"), 600.0],
    [0, datetime.datetime.fromisoformat("2021-11-14 00:00:00"), 500.0],
    [0, datetime.datetime.fromisoformat("2021-11-15 00:00:00"), 400.0],
    [0, datetime.datetime.fromisoformat("2021-11-16 00:00:00"), 300.0],
    [0, datetime.datetime.fromisoformat("2021-11-17 00:00:00"), 200.0],
    [0, datetime.datetime.fromisoformat("2021-11-18 00:00:00"), 100.0],
    [0, datetime.datetime.fromisoformat("2021-11-19 00:00:00"), 200.0],
    [0, datetime.datetime.fromisoformat("2021-11-20 00:00:00"), 300.0],
    [0, datetime.datetime.fromisoformat("2021-11-21 00:00:00"), 400.0],
    [0, datetime.datetime.fromisoformat("2021-11-22 00:00:00"), 500.0],
    [0, datetime.datetime.fromisoformat("2021-11-23 00:00:00"), 600.0],
    [0, datetime.datetime.fromisoformat("2021-11-24 00:00:00"), 700.0],
    [0, datetime.datetime.fromisoformat("2021-11-25 00:00:00"), 800.0],
    [0, datetime.datetime.fromisoformat("2021-11-26 00:00:00"), 900.0],
    [0, datetime.datetime.fromisoformat("2021-11-27 00:00:00"), 800.0],
    [0, datetime.datetime.fromisoformat("2021-11-28 00:00:00"), 700.0],
    [0, datetime.datetime.fromisoformat("2021-11-29 00:00:00"), 600.0],
    [0, datetime.datetime.fromisoformat("2021-11-30 00:00:00"), 500.0],
    [0, datetime.datetime.fromisoformat("2021-12-01 00:00:00"), 400.0],
    [0, datetime.datetime.fromisoformat("2021-12-02 00:00:00"), 300.0],
    [0, datetime.datetime.fromisoformat("2021-12-03 00:00:00"), 200.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, series
double', op_type='batch')

lstNetTrainBatchOp = LSTNetTrainBatchOp()\
    .setTimeCol("ts")\
    .setSelectedCol("series")\
    .setNumEpochs(10)\
```

```
    .setWindow(24)\
    .setHorizon(1)

groupByBatchOp = GroupByBatchOp()\
    .setGroupByPredicate("id")\
    .setSelectClause("mtable_agg(ts, series) as mtable_agg_series")

lstNetPredictBatchOp = LSTNetPredictBatchOp()\
    .setPredictNum(1)\
    .setPredictionCol("pred")\
    .setReservedCols([])\
    .setValueCol("mtable_agg_series")\

lstNetPredictBatchOp\
    .linkFrom(
        lstNetTrainBatchOp.linkFrom(source),
        groupByBatchOp.linkFrom(source.filter("ts >= TO_TIMESTAMP('2021-11-10
00:00:00')"))
    )\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.operator.batch.timeseries.LSTNetPredictBatchOp;
import com.alibaba.alink.operator.batch.timeseries.LSTNetTrainBatchOp;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

public class LSTNetTrainBatchOpTest {

    @Test
    public void testLSTNetTrainBatchOp() throws Exception {
        BatchOperator.setParallelism(1);

        List <Row> data = Arrays.asList(
            Row.of(0, Timestamp.valueOf("2021-11-01 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-02 00:00:00"), 200.0),
            Row.of(0, Timestamp.valueOf("2021-11-03 00:00:00"), 300.0),
            Row.of(0, Timestamp.valueOf("2021-11-04 00:00:00"), 400.0),
```

```
            Row.of(0, Timestamp.valueOf("2021-11-06 00:00:00"), 500.0),
            Row.of(0, Timestamp.valueOf("2021-11-07 00:00:00"), 600.0),
            Row.of(0, Timestamp.valueOf("2021-11-08 00:00:00"), 700.0),
            Row.of(0, Timestamp.valueOf("2021-11-09 00:00:00"), 800.0),
            Row.of(0, Timestamp.valueOf("2021-11-10 00:00:00"), 900.0),
            Row.of(0, Timestamp.valueOf("2021-11-11 00:00:00"), 800.0),
            Row.of(0, Timestamp.valueOf("2021-11-12 00:00:00"), 700.0),
            Row.of(0, Timestamp.valueOf("2021-11-13 00:00:00"), 600.0),
            Row.of(0, Timestamp.valueOf("2021-11-14 00:00:00"), 500.0),
            Row.of(0, Timestamp.valueOf("2021-11-15 00:00:00"), 400.0),
            Row.of(0, Timestamp.valueOf("2021-11-16 00:00:00"), 300.0),
            Row.of(0, Timestamp.valueOf("2021-11-17 00:00:00"), 200.0),
            Row.of(0, Timestamp.valueOf("2021-11-18 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-19 00:00:00"), 200.0),
            Row.of(0, Timestamp.valueOf("2021-11-20 00:00:00"), 300.0),
            Row.of(0, Timestamp.valueOf("2021-11-21 00:00:00"), 400.0),
            Row.of(0, Timestamp.valueOf("2021-11-22 00:00:00"), 500.0),
            Row.of(0, Timestamp.valueOf("2021-11-23 00:00:00"), 600.0),
            Row.of(0, Timestamp.valueOf("2021-11-24 00:00:00"), 700.0),
            Row.of(0, Timestamp.valueOf("2021-11-25 00:00:00"), 800.0),
            Row.of(0, Timestamp.valueOf("2021-11-26 00:00:00"), 900.0),
            Row.of(0, Timestamp.valueOf("2021-11-27 00:00:00"), 800.0),
            Row.of(0, Timestamp.valueOf("2021-11-28 00:00:00"), 700.0),
            Row.of(0, Timestamp.valueOf("2021-11-29 00:00:00"), 600.0),
            Row.of(0, Timestamp.valueOf("2021-11-30 00:00:00"), 500.0),
            Row.of(0, Timestamp.valueOf("2021-12-01 00:00:00"), 400.0),
            Row.of(0, Timestamp.valueOf("2021-12-02 00:00:00"), 300.0),
            Row.of(0, Timestamp.valueOf("2021-12-03 00:00:00"), 200.0)
        );

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "id int,
ts timestamp, series double");

        LSTNetTrainBatchOp lstNetTrainBatchOp = new LSTNetTrainBatchOp()
            .setTimeCol("ts")
            .setSelectedCol("series")
            .setNumEpochs(10)
            .setWindow(24)
            .setHorizon(1);

        GroupByBatchOp groupByBatchOp = new GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("mtable_agg(ts, series) as mtable_agg_series");

        LSTNetPredictBatchOp lstNetPredictBatchOp = new LSTNetPredictBatchOp()
            .setPredictNum(1)
            .setPredictionCol("pred")
            .setReservedCols()
```

```
            .setValueCol("mtable_agg_series");

        lstNetPredictBatchOp
            .linkFrom(
                lstNetTrainBatchOp.linkFrom(memSourceBatchOp),
                groupByBatchOp.linkFrom(memSourceBatchOp.filter("ts >=
TO_TIMESTAMP('2021-11-10 00:00:00')"))
            )
            .print();
    }
}
```

运行结果

pred
{"data":{"ts":["2021-12-04 00:00:00.0"],"series":[441.76019287109375]},"schema":"ts TIMESTAMP,series DOUBLE"}

LSTNet训练 (LSTNetTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.LSTNetTrainBatchOp

Python 类名：LSTNetTrainBatchOp

功能介绍

使用 LSTNet 进行时间序列训练和预测。

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

参数说明

名称	中文名称	描述	类型	是否必须?	取值范
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为TensorFlow中 `Estimator` 的 `model_dir` 传入，需要为所有worker 都能访问到的目录	String	✓	
timeCol	时间戳列(TimeStamp)	时间戳列(TimeStamp)	String	✓	所选列类型为[TIMESTAMI
batchSize	数据批大小	数据批大小	Integer		
horizon	horizon大小	horizon大小	Integer		[1, +inf)
intraOpParallelism	Op 间并发度	Op 间并发度	Integer		
learningRate	学习率	学习率	Double		
numEpochs	epoch数	epoch数	Integer		

numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer		
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer		
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String		

removeCheckpointBeforeTraining	是否在训练前移除checkpoint相关文件	是否在训练前移除checkpoint相关文件用于重新训练，只会删除必要的文件	Boolean		
selectedCol	计算列对应的列名	计算列对应的列名, 默认值是null	String		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		所选列类型为[DENSE_VE SPARSE_VE STRING, VE
window	窗口大小	窗口大小	Integer		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
    [0, datetime.datetime.fromisoformat("2021-11-01 00:00:00"), 100.0],
    [0, datetime.datetime.fromisoformat("2021-11-02 00:00:00"), 200.0],
    [0, datetime.datetime.fromisoformat("2021-11-03 00:00:00"), 300.0],
    [0, datetime.datetime.fromisoformat("2021-11-04 00:00:00"), 400.0],
    [0, datetime.datetime.fromisoformat("2021-11-06 00:00:00"), 500.0],
    [0, datetime.datetime.fromisoformat("2021-11-07 00:00:00"), 600.0],
    [0, datetime.datetime.fromisoformat("2021-11-08 00:00:00"), 700.0],
    [0, datetime.datetime.fromisoformat("2021-11-09 00:00:00"), 800.0],
    [0, datetime.datetime.fromisoformat("2021-11-10 00:00:00"), 900.0],
    [0, datetime.datetime.fromisoformat("2021-11-11 00:00:00"), 800.0],
    [0, datetime.datetime.fromisoformat("2021-11-12 00:00:00"), 700.0],
    [0, datetime.datetime.fromisoformat("2021-11-13 00:00:00"), 600.0],
```

```
    [0, datetime.datetime.fromisoformat("2021-11-14 00:00:00"), 500.0],
    [0, datetime.datetime.fromisoformat("2021-11-15 00:00:00"), 400.0],
    [0, datetime.datetime.fromisoformat("2021-11-16 00:00:00"), 300.0],
    [0, datetime.datetime.fromisoformat("2021-11-17 00:00:00"), 200.0],
    [0, datetime.datetime.fromisoformat("2021-11-18 00:00:00"), 100.0],
    [0, datetime.datetime.fromisoformat("2021-11-19 00:00:00"), 200.0],
    [0, datetime.datetime.fromisoformat("2021-11-20 00:00:00"), 300.0],
    [0, datetime.datetime.fromisoformat("2021-11-21 00:00:00"), 400.0],
    [0, datetime.datetime.fromisoformat("2021-11-22 00:00:00"), 500.0],
    [0, datetime.datetime.fromisoformat("2021-11-23 00:00:00"), 600.0],
    [0, datetime.datetime.fromisoformat("2021-11-24 00:00:00"), 700.0],
    [0, datetime.datetime.fromisoformat("2021-11-25 00:00:00"), 800.0],
    [0, datetime.datetime.fromisoformat("2021-11-26 00:00:00"), 900.0],
    [0, datetime.datetime.fromisoformat("2021-11-27 00:00:00"), 800.0],
    [0, datetime.datetime.fromisoformat("2021-11-28 00:00:00"), 700.0],
    [0, datetime.datetime.fromisoformat("2021-11-29 00:00:00"), 600.0],
    [0, datetime.datetime.fromisoformat("2021-11-30 00:00:00"), 500.0],
    [0, datetime.datetime.fromisoformat("2021-12-01 00:00:00"), 400.0],
    [0, datetime.datetime.fromisoformat("2021-12-02 00:00:00"), 300.0],
    [0, datetime.datetime.fromisoformat("2021-12-03 00:00:00"), 200.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, series
double', op_type='batch')

lstNetTrainBatchOp = LSTNetTrainBatchOp()\
    .setTimeCol("ts")\
    .setSelectedCol("series")\
    .setNumEpochs(10)\
    .setWindow(24)\
    .setHorizon(1)

groupByBatchOp = GroupByBatchOp()\
    .setGroupByPredicate("id")\
    .setSelectClause("mtable_agg(ts, series) as mtable_agg_series")

lstNetPredictBatchOp = LSTNetPredictBatchOp()\
    .setPredictNum(1)\
    .setPredictionCol("pred")\
    .setReservedCols([])\
    .setValueCol("mtable_agg_series")\

lstNetPredictBatchOp\
    .linkFrom(
        lstNetTrainBatchOp.linkFrom(source),
        groupByBatchOp.linkFrom(source.filter("ts >= TO_TIMESTAMP('2021-11-10
00:00:00')")))
```

```
    )\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.operator.batch.timeseries.LSTNetPredictBatchOp;
import com.alibaba.alink.operator.batch.timeseries.LSTNetTrainBatchOp;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

public class LSTNetTrainBatchOpTest {

    @Test
    public void testLSTNetTrainBatchOp() throws Exception {
        BatchOperator.setParallelism(1);

        List <Row> data = Arrays.asList(
            Row.of(0, Timestamp.valueOf("2021-11-01 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-02 00:00:00"), 200.0),
            Row.of(0, Timestamp.valueOf("2021-11-03 00:00:00"), 300.0),
            Row.of(0, Timestamp.valueOf("2021-11-04 00:00:00"), 400.0),
            Row.of(0, Timestamp.valueOf("2021-11-06 00:00:00"), 500.0),
            Row.of(0, Timestamp.valueOf("2021-11-07 00:00:00"), 600.0),
            Row.of(0, Timestamp.valueOf("2021-11-08 00:00:00"), 700.0),
            Row.of(0, Timestamp.valueOf("2021-11-09 00:00:00"), 800.0),
            Row.of(0, Timestamp.valueOf("2021-11-10 00:00:00"), 900.0),
            Row.of(0, Timestamp.valueOf("2021-11-11 00:00:00"), 800.0),
            Row.of(0, Timestamp.valueOf("2021-11-12 00:00:00"), 700.0),
            Row.of(0, Timestamp.valueOf("2021-11-13 00:00:00"), 600.0),
            Row.of(0, Timestamp.valueOf("2021-11-14 00:00:00"), 500.0),
            Row.of(0, Timestamp.valueOf("2021-11-15 00:00:00"), 400.0),
            Row.of(0, Timestamp.valueOf("2021-11-16 00:00:00"), 300.0),
            Row.of(0, Timestamp.valueOf("2021-11-17 00:00:00"), 200.0),
            Row.of(0, Timestamp.valueOf("2021-11-18 00:00:00"), 100.0),
            Row.of(0, Timestamp.valueOf("2021-11-19 00:00:00"), 200.0),
            Row.of(0, Timestamp.valueOf("2021-11-20 00:00:00"), 300.0),
            Row.of(0, Timestamp.valueOf("2021-11-21 00:00:00"), 400.0),
            Row.of(0, Timestamp.valueOf("2021-11-22 00:00:00"), 500.0),
            Row.of(0, Timestamp.valueOf("2021-11-23 00:00:00"), 600.0),
```

```
            Row.of(0, Timestamp.valueOf("2021-11-24 00:00:00"), 700.0),
            Row.of(0, Timestamp.valueOf("2021-11-25 00:00:00"), 800.0),
            Row.of(0, Timestamp.valueOf("2021-11-26 00:00:00"), 900.0),
            Row.of(0, Timestamp.valueOf("2021-11-27 00:00:00"), 800.0),
            Row.of(0, Timestamp.valueOf("2021-11-28 00:00:00"), 700.0),
            Row.of(0, Timestamp.valueOf("2021-11-29 00:00:00"), 600.0),
            Row.of(0, Timestamp.valueOf("2021-11-30 00:00:00"), 500.0),
            Row.of(0, Timestamp.valueOf("2021-12-01 00:00:00"), 400.0),
            Row.of(0, Timestamp.valueOf("2021-12-02 00:00:00"), 300.0),
            Row.of(0, Timestamp.valueOf("2021-12-03 00:00:00"), 200.0)
        );

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "id int,
ts timestamp, series double");

        LSTNetTrainBatchOp lstNetTrainBatchOp = new LSTNetTrainBatchOp()
            .setTimeCol("ts")
            .setSelectedCol("series")
            .setNumEpochs(10)
            .setWindow(24)
            .setHorizon(1);

        GroupByBatchOp groupByBatchOp = new GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("mtable_agg(ts, series) as mtable_agg_series");

        LSTNetPredictBatchOp lstNetPredictBatchOp = new LSTNetPredictBatchOp()
            .setPredictNum(1)
            .setPredictionCol("pred")
            .setReservedCols()
            .setValueCol("mtable_agg_series");

        lstNetPredictBatchOp
            .linkFrom(
                lstNetTrainBatchOp.linkFrom(memSourceBatchOp),
                groupByBatchOp.linkFrom(memSourceBatchOp.filter("ts >=
TO_TIMESTAMP('2021-11-10 00:00:00')"))
            )
            .print();
    }
}
```

运行结果

pred
{"data":{"ts":["2021-12-04 00:00:00.0"],"series":[441.76019287109375]},"schema":"ts TIMESTAMP,series DOUBLE"}

时间序列插值 (LookupValueInTimeSeriesBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.LookupValueInTimeSeriesBatchOp

Python 类名：LookupValueInTimeSeriesBatchOp

功能介绍

在时间序列中查找对应时间的值。

注意事项

- 时间序列列，是特殊的MTable类型，一列是时间，一列是值，参考运行结果的data列。
- 查找的时间在数据列中不存在时，用相邻时刻的值差值得到结果。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
timeCol	时间戳列(TimeStamp)	时间戳列(TimeStamp)	String	✓	所选列类型为[TIMESTAMP]	
timeSeriesCol	时间序列列	时间序列列，是特殊的MTable类型，一列是时间，一列是值	String	✓	所选列类型为[M_TABLE]	
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
            [1,  datetime.datetime.fromtimestamp(1), 10.0],
            [1,  datetime.datetime.fromtimestamp(2), 11.0],
            [1,  datetime.datetime.fromtimestamp(3), 12.0],
            [1,  datetime.datetime.fromtimestamp(4), 13.0],
            [1,  datetime.datetime.fromtimestamp(5), 14.0],
            [1,  datetime.datetime.fromtimestamp(6), 15.0],
            [1,  datetime.datetime.fromtimestamp(7), 16.0],
            [1,  datetime.datetime.fromtimestamp(8), 17.0],
            [1,  datetime.datetime.fromtimestamp(9), 18.0],
            [1,  datetime.datetime.fromtimestamp(10), 19.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val
double', op_type='batch')

source.link(
        GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("id, mtable_agg(ts, val) as data")
        ).link(
            ShiftBatchOp()
                    .setValueCol("data")
                    .setShiftNum(7)
                    .setPredictNum(12)
                    .setPredictionCol("predict")
        ).link(
            FlattenMTableBatchOp()
                    .setReservedCols(["id", "predict"])
                    .setSelectedCol("predict")
                    .setSchemaStr("ts timestamp, val double")
        ).link(
            LookupValueInTimeSeriesBatchOp()
                .setTimeCol("ts")
                .setTimeSeriesCol("predict")
                .setOutputCol("out")
                .setReservedCols(["id","ts"])
        ).print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.timeseries;

import org.apache.flink.types.Row;

import com.alibaba.alink.common.MTable;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

public class LookupValueInTimeSeriesBatchOpTest {
    @Test
    public void test() throws Exception {

        List <Row> mTableData = Arrays.asList(
            Row.of(new Timestamp(1), 10.0),
            Row.of(new Timestamp(2), 11.0),
            Row.of(new Timestamp(3), 12.0),
            Row.of(new Timestamp(4), 13.0),
            Row.of(new Timestamp(5), 14.0),
            Row.of(new Timestamp(6), 15.0),
            Row.of(new Timestamp(7), 16.0),
            Row.of(new Timestamp(8), 17.0),
            Row.of(new Timestamp(9), 18.0),
            Row.of(new Timestamp(10), 19.0)
        );

        MTable mtable = new MTable(mTableData, "ts timestamp, val double");

        MemSourceBatchOp source = new MemSourceBatchOp(
            new Object[][] {
                {1, new Timestamp(5), mtable}
            },
            new String[] {"id", "ts", "data"});

        source
            .link(new LookupValueInTimeSeriesBatchOp()
                .setTimeCol("ts")
                .setTimeSeriesCol("data")
                .setOutputCol("out")
                .setReservedCols("id","ts")
            )
            .print();
    }
}
```

运行结果

id	ts	data	out
1	1970-01-01 08:00:00.005	MTable(10,2)(ts,val) 1970-01-01 08:00:00.001 | 10.000 1970-01-01 08:00:00.002 | 11.000 1970-01-01 08:00:00.003 | 12.000 1970-01-01 08:00:00.004 | 13.000 1970-01-01 08:00:00.005 | 14.000	14.0000

时间序列向量插值 (LookupVectorInTimeSeriesBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.LookupVectorInTimeSeriesBatchOp

Python 类名：LookupVectorInTimeSeriesBatchOp

功能介绍

在时间序列中查找对应时间的值。

注意事项

- 时间序列列，是特殊的MTable类型，一列是时间，一列是值，参考运行结果的data列。
- 查找的时间在数据列中不存在时，用相邻时刻的值差值得到结果。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
timeCol	时间戳列(TimeStamp)	时间戳列(TimeStamp)	String	✓	所选列类型为[TIMESTAMP]	
timeSeriesCol	时间序列列	时间序列列，是特殊的MTable类型，一列是时间，一列是值	String	✓	所选列类型为[M_TABLE]	
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
            [1,  datetime.datetime.fromtimestamp(1), "10.0 10.0"],
            [1,  datetime.datetime.fromtimestamp(2), "11.0 11.0"],
            [1,  datetime.datetime.fromtimestamp(3), "12.0 12.0"],
            [1,  datetime.datetime.fromtimestamp(4), "13.0 13.0"],
            [1,  datetime.datetime.fromtimestamp(5), "14.0 14.0"],
            [1,  datetime.datetime.fromtimestamp(6), "15.0 15.0"],
            [1,  datetime.datetime.fromtimestamp(7), "16.0 16.0"],
            [1,  datetime.datetime.fromtimestamp(8), "17.0 17.0"],
            [1,  datetime.datetime.fromtimestamp(9), "18.0 18.0"],
            [1,  datetime.datetime.fromtimestamp(10), "19.0 19.0"]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val
string', op_type='batch')

source.link(
        GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("id, mtable_agg(ts, val) as data")
        ).link(
            ShiftBatchOp()
                    .setValueCol("data")
                    .setShiftNum(7)
                    .setPredictNum(12)
                    .setPredictionCol("predict")
        ).link(
            FlattenMTableBatchOp()
                    .setReservedCols(["id", "predict"])
                    .setSelectedCol("predict")
                    .setSchemaStr("ts timestamp, val VECTOR")
        ).link(
            LookupVectorInTimeSeriesBatchOp()
                .setTimeCol("ts")
                .setTimeSeriesCol("predict")
                .setOutputCol("out")
                .setReservedCols(["id"])
        ).print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.timeseries;

import org.apache.flink.types.Row;

import com.alibaba.alink.common.MTable;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

import static org.junit.Assert.*;

public class LookupVectorInTimeSeriesBatchOpTest {
    @Test
    public void test() throws Exception {

        List<Row> mTableData = Arrays.asList(
            Row.of(new Timestamp(1), "10.0 21.0"),
            Row.of(new Timestamp(2), "11.0 22.0"),
            Row.of(new Timestamp(3), "12.0 23.0"),
            Row.of(new Timestamp(4), "13.0 24.0"),
            Row.of(new Timestamp(5), "14.0 25.0"),
            Row.of(new Timestamp(6), "15.0 26.0"),
            Row.of(new Timestamp(7), "16.0 27.0"),
            Row.of(new Timestamp(8), "17.0 28.0"),
            Row.of(new Timestamp(9), "18.0 29.0"),
            Row.of(new Timestamp(10), "19.0 30.0")
        );

        MTable mtable = new MTable(mTableData, "ts timestamp, val vector");

        MemSourceBatchOp source = new MemSourceBatchOp(
            new Object[][] {
                {1, new Timestamp(5), mtable}
            },
            new String[] {"id", "ts", "data"});

        source
            .link(new LookupVectorInTimeSeriesBatchOp()
                .setTimeCol("ts")
                .setTimeSeriesCol("data")
                .setOutputCol("out")
            )
            .print();
    }
}
```

运行结果

id	ts	data	out
1	1970-01-01 08:00:00.005	MTable(10,2)(ts,val) 1970-01-01 08:00:00.001 | 10.0 21.0 1970-01-01 08:00:00.002 | 11.0 22.0 1970-01-01 08:00:00.003 | 12.0 23.0 1970-01-01 08:00:00.004 | 13.0 24.0 1970-01-01 08:00:00.005 | 14.0 25.0	14.0 25.0

Prophet (ProphetBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.ProphetBatchOp

Python 类名：ProphetBatchOp

功能介绍

对每一行的MTable数据, 进行Prophet时间序列预测，给出下一时间段的预测结果。

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

算法原理

Prophet是facebook开源的一个时间序列预测算法, github地址：https://github.com/facebook/prophet.

Prophet适用于具有明显的内在规律的数据, 例如：

- 有一定的历史数据，有至少几个月的每小时、每天或每周观察的历史数据
- 有较强的季节性趋势：每周的一些天，每年的一些时间
- 有已知的以不定期的间隔发生的重要节假日（比如国庆节）
- 缺失的历史数据或较大的异常数据的数量在合理范围内
- 对于数据中蕴含的非线性增长的趋势都有一个自然极限或饱和状态

使用方式

- 第一步，将每组数据(时间列和数据列) 聚合成MTable.

```
GroupByBatchOp()
   .setGroupByPredicate("id")
   .setSelectClause("id, mtable_agg(ts, val) as data")
```

- 第二步，使用时间序列方法进行预测，预测结果也是MTable。
- 第三步，使用FlattenMTableBatchOp，将MTable转换成列，

```
FlattenMTableBatchOp()
    .setReservedCols(["id", "predict"])
    .setSelectedCol("predict")
    .setSchemaStr("ts timestamp, val double")
```

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
valueCol	value列，类型为MTable	value列，类型为MTable	String	✓	所选列类型为[M_TABLE]	
predictNum	预测条数	预测条数	Integer			1
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
pythonEnv	Python环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			""
reservedCols	算法保留列名	算法保留列	String[]			null
stanInit	初始值	初始值	String			null
uncertaintySamples	用来计算指标的采样数目	用来计算指标的采样数目，设置成0，不计算指标。	Integer			1000
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

downloader = AlinkGlobalConfiguration.getPluginDownloader()
downloader.downloadPlugin('tf115_python_env_linux')

data = pd.DataFrame([
            [1,  datetime.datetime.fromtimestamp(1), 10.0],
            [1,  datetime.datetime.fromtimestamp(2), 11.0],
            [1,  datetime.datetime.fromtimestamp(3), 12.0],
            [1,  datetime.datetime.fromtimestamp(4), 13.0],
            [1,  datetime.datetime.fromtimestamp(5), 14.0],
            [1,  datetime.datetime.fromtimestamp(6), 15.0],
            [1,  datetime.datetime.fromtimestamp(7), 16.0],
            [1,  datetime.datetime.fromtimestamp(8), 17.0],
            [1,  datetime.datetime.fromtimestamp(9), 18.0],
            [1,  datetime.datetime.fromtimestamp(10), 19.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val
double', op_type='batch')

source.link(
        GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("id, mtable_agg(ts, val) as data")
        ).link(ProphetBatchOp()
            .setValueCol("data")
            .setPredictNum(4)
            .setPredictionCol("pred")
        ).print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.timeseries;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
```

```
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
import java.util.List;

public class ArimaBatchOpTest extends AlinkTestBase {

    @Test
    public void test() throws Exception {
        List <Row> mTableData = Arrays.asList(
            Row.of(1, new Timestamp(1), 10.0),
            Row.of(1, new Timestamp(2), 11.0),
            Row.of(1, new Timestamp(3), 12.0),
            Row.of(1, new Timestamp(4), 13.0),
            Row.of(1, new Timestamp(5), 14.0),
            Row.of(1, new Timestamp(6), 15.0),
            Row.of(1, new Timestamp(7), 16.0),
            Row.of(1, new Timestamp(8), 17.0),
            Row.of(1, new Timestamp(9), 18.0),
            Row.of(1, new Timestamp(10), 19.0)
        );

        MemSourceBatchOp source = new MemSourceBatchOp(mTableData, new String[]
{"id", "ts", "val"});

        source.link(
            new GroupByBatchOp()
                .setGroupByPredicate("id")
                .setSelectClause("mtable_agg(ts, val) as data")
        ).link(new ProphetBatchOp()
                   .setValueCol("data")
                   .setPredictNum(4)
                   .setPredictionCol("pred")
        ).print();
    }
}
```

运行结果

id	data	

1	{"data":{"ts":["1970-01-01 08:00:00.001","1970-01-01 08:00:00.002","1970-01-01 08:00:00.003","1970-01-01 08:00:00.004","1970-01-01 08:00:00.005","1970-01-01 08:00:00.006","1970-01-01 08:00:00.007","1970-01-01 08:00:00.008","1970-01-01 08:00:00.009","1970-01-01 08:00:00.01"],"val":[10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0]},"schema":"ts TIMESTAMP,val DOUBLE"}	{"data":{"ts":["1970-01-01 08:00:0 01-01 08:00:00.013","1970-01-01 08:00:00.015","1970-01-01 08:00 08:00:00.017","1970-01-01 08:00 08:00:00.019","1970-01-01 08:00 08:00:00.021","1970-01-01 08:00 [20.0,21.0,22.0,23.0,24.0,25.0,26 TIMESTAMP,val DOUBLE"}

t

Prophet预测 (ProphetPredictBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.ProphetPredictBatchOp

Python 类名：ProphetPredictBatchOp

功能介绍

指定模型(通过ProphetTrainBatchOp训练得到),对每一行的MTable数据, 进行Prophet时间序列预测，给出下一时间段的预测结果。

算法原理

Prophet是facebook开源的一个时间序列预测算法, github地址：https://github.com/facebook/prophet.

Prophet适用于具有明显的内在规律的数据, 例如：

- 有一定的历史数据，有至少几个月的每小时、每天或每周观察的历史数据
- 有较强的季节性趋势：每周的一些天，每年的一些时间
- 有已知的以不定期的间隔发生的重要节假日（比如国庆节）
- 缺失的历史数据或较大的异常数据的数量在合理范围内
- 对于数据中蕴含的非线性增长的趋势都有一个自然极限或饱和状态

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
valueCol	value列，类型为MTable	value列，类型为MTable	String	✓	所选列类型为[M_TABLE]	
modelFilePath	模型的文件路径	模型的文件路径	String			null
predictNum	预测条数	预测条数	Integer			1

predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
pythonEnv	Python环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			""
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

downloader = AlinkGlobalConfiguration.getPluginDownloader()
downloader.downloadPlugin('tf115_python_env_linux')

data = pd.DataFrame([
            [1,  datetime.datetime.fromtimestamp(1), 10.0],
            [1,  datetime.datetime.fromtimestamp(2), 11.0],
            [1,  datetime.datetime.fromtimestamp(3), 12.0],
            [1,  datetime.datetime.fromtimestamp(4), 13.0],
            [1,  datetime.datetime.fromtimestamp(5), 14.0],
            [1,  datetime.datetime.fromtimestamp(6), 15.0],
            [1,  datetime.datetime.fromtimestamp(7), 16.0],
```

```
            [1,  datetime.datetime.fromtimestamp(8), 17.0],
            [1,  datetime.datetime.fromtimestamp(9), 18.0],
            [1,  datetime.datetime.fromtimestamp(10), 19.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val
double', op_type='batch')

prophetModel = source.link(\
                    ProphetTrainBatchOp()\
                        .setTimeCol("ts")\
                        .setValueCol("val")
                )

ProphetPredictBatchOp()\
    .setValueCol("data")\
    .setPredictNum(4)\
    .setPredictionCol("pred")\
    .linkFrom(
         prophetModel,
         source.link(\
            GroupByBatchOp()\
                .setGroupByPredicate("id")\
                .setSelectClause("id, mtable_agg(ts, val) as data")
            )
    )\
    .print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.timeseries;

import org.apache.flink.types.Row;

import com.alibaba.alink.common.AlinkGlobalConfiguration;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;

public class ProphetBatchOpTest {

    @Test
    public void testModel() throws Exception {
        Row[] rowsData =
```

```
            new Row[] {
                Row.of("1", new Timestamp(117, 11, 1, 0, 0, 0, 0),
9.59076113897809),
                Row.of("1", new Timestamp(117, 11, 2, 0, 0, 0, 0),
8.51959031601596),
                Row.of("2", new Timestamp(117, 11, 3, 0, 0, 0, 0),
9.59076113897809),
                Row.of("1", new Timestamp(117, 11, 4, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 5, 0, 0, 0, 0),
8.51959031601596),
                Row.of("1", new Timestamp(117, 11, 6, 0, 0, 0, 0),
8.07246736935477),
                Row.of("2", new Timestamp(117, 11, 7, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 8, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 9, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 10, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 11, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 12, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 13, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 14, 0, 0, 0, 0),
8.18367658262066),
                Row.of("1", new Timestamp(117, 11, 15, 0, 0, 0, 0),
7.8935720735049),
                Row.of("1", new Timestamp(117, 11, 16, 0, 0, 0, 0),
7.78364059622125),
                Row.of("2", new Timestamp(117, 11, 17, 0, 0, 0, 0),
8.07246736935477),
                Row.of("1", new Timestamp(117, 11, 18, 0, 0, 0, 0),
8.41405243249672),
                Row.of("1", new Timestamp(117, 11, 19, 0, 0, 0, 0),
8.82922635473185),
                Row.of("1", new Timestamp(117, 11, 20, 0, 0, 0, 0),
8.38251828808963),
                Row.of("1", new Timestamp(117, 11, 21, 0, 0, 0, 0),
8.06965530688617),
                Row.of("1", new Timestamp(117, 11, 22, 0, 0, 0, 0),
9.59076113897809),
                Row.of("1", new Timestamp(117, 11, 23, 0, 0, 0, 0),
8.51959031601596),
                Row.of("1", new Timestamp(117, 11, 24, 0, 0, 0, 0),
```

```
8.18367658262066),
                Row.of("1", new Timestamp(117, 11, 25, 0, 0, 0, 0),
8.07246736935477),
                Row.of("1", new Timestamp(117, 11, 26, 0, 0, 0, 0),
7.8935720735049),
                Row.of("1", new Timestamp(117, 11, 27, 0, 0, 0, 0),
7.78364059622125),
                Row.of("1", new Timestamp(117, 11, 28, 0, 0, 0, 0),
8.41405243249672),
                Row.of("1", new Timestamp(117, 11, 29, 0, 0, 0, 0),
8.82922635473185),
                Row.of("1", new Timestamp(117, 12, 1, 0, 0, 0, 0),
8.38251828808963),
                Row.of("1", new Timestamp(117, 12, 2, 0, 0, 0, 0),
8.06965530688617),
                Row.of("2", new Timestamp(117, 12, 3, 0, 0, 0, 0),
8.07246736935477),
                Row.of("2", new Timestamp(117, 12, 4, 0, 0, 0, 0),
7.8935720735049),
                Row.of("2", new Timestamp(117, 12, 5, 0, 0, 0, 0),
7.78364059622125),
                Row.of("2", new Timestamp(117, 12, 6, 0, 0, 0, 0),
8.41405243249672),
                Row.of("2", new Timestamp(117, 12, 7, 0, 0, 0, 0),
8.82922635473185),
                Row.of("2", new Timestamp(117, 12, 8, 0, 0, 0, 0),
8.38251828808963),
                Row.of("2", new Timestamp(117, 12, 9, 0, 0, 0, 0),
8.06965530688617)
            };
        String[] colNames = new String[] {"id", "ts", "val"};

        //train batch model.
        MemSourceBatchOp source = new MemSourceBatchOp(Arrays.asList(rowsData),
colNames);

        ProphetTrainBatchOp model = new ProphetTrainBatchOp()
            .setTimeCol("ts")
            .setValueCol("val");

        source.link(model).print();

        //construct times series by id.
        GroupByBatchOp groupData = new GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("mtable_agg(ts, val) as data");

        ProphetPredictBatchOp prophetPredict = new ProphetPredictBatchOp()
```

```
            .setValueCol("data")
            .setPredictNum(4)
            .setPredictionCol("pred");

        prophetPredict.linkFrom(model, source.link(groupData)).print();
    }
}
```

运行结果

id	data	
1	{"data":{"ts":["1970-01-01 08:00:00.001","1970-01-01 08:00:00.002","1970-01-01 08:00:00.003","1970-01-01 08:00:00.004","1970-01-01 08:00:00.005","1970-01-01 08:00:00.006","1970-01-01 08:00:00.007","1970-01-01 08:00:00.008","1970-01-01 08:00:00.009","1970-01-01 08:00:00.01"],"val":[10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0]},"schema":"ts TIMESTAMP,val DOUBLE"}	{"data":{"ts":["1970-01-01 08:00:0 01-01 08:00:00.013","1970-01-01 08:00:00.015","1970-01-01 08:00 08:00:00.017","1970-01-01 08:00 08:00:00.019","1970-01-01 08:00 08:00:00.021","1970-01-01 08:00 [20.0,21.0,22.0,23.0,24.0,25.0,26 TIMESTAMP,val DOUBLE"}

t

Prophet训练 (ProphetTrainBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.ProphetTrainBatchOp

Python 类名：ProphetTrainBatchOp

功能介绍

进行Prophet训练，得到Prophet模型，可以用来做预测。

算法原理

Prophet是facebook开源的一个时间序列预测算法, github地址：https://github.com/facebook/prophet.

Prophet适用于具有明显的内在规律的数据, 例如：

- 有一定的历史数据，有至少几个月的每小时、每天或每周观察的历史数据
- 有较强的季节性趋势：每周的一些天，每年的一些时间
- 有已知的以不定期的间隔发生的重要节假日（比如国庆节）
- 缺失的历史数据或较大的异常数据的数量在合理范围内
- 对于数据中蕴含的非线性增长的趋势都有一个自然极限或饱和状态

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
timeCol	时间戳列(TimeStamp)	时间戳列(TimeStamp)	String	✓	所选列类型为[TIMESTAMP]	
valueCol	value列，类型为MTable	value列，类型为MTable	String	✓	所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]	

pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			""

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

downloader = AlinkGlobalConfiguration.getPluginDownloader()
downloader.downloadPlugin('tf115_python_env_linux')

data = pd.DataFrame([
            [1,  datetime.datetime.fromtimestamp(1), 10.0],
            [1,  datetime.datetime.fromtimestamp(2), 11.0],
            [1,  datetime.datetime.fromtimestamp(3), 12.0],
            [1,  datetime.datetime.fromtimestamp(4), 13.0],
            [1,  datetime.datetime.fromtimestamp(5), 14.0],
            [1,  datetime.datetime.fromtimestamp(6), 15.0],
            [1,  datetime.datetime.fromtimestamp(7), 16.0],
            [1,  datetime.datetime.fromtimestamp(8), 17.0],
            [1,  datetime.datetime.fromtimestamp(9), 18.0],
            [1,  datetime.datetime.fromtimestamp(10), 19.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val
double', op_type='batch')

prophetModel = source.link(\
                    ProphetTrainBatchOp()\
```

```
                        .setTimeCol("ts")\
                        .setValueCol("val")
                )

ProphetPredictBatchOp()\
    .setValueCol("data")\
    .setPredictNum(4)\
    .setPredictionCol("pred")\
    .linkFrom(
         prophetModel,
         source.link(
            GroupByBatchOp()
                .setGroupByPredicate("id")
                .setSelectClause("id, mtable_agg(ts, val) as data")
            )
    )\
    .print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.timeseries;

import org.apache.flink.types.Row;

import com.alibaba.alink.common.AlinkGlobalConfiguration;
import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;

public class ProphetBatchOpTest {

    @Test
    public void testModel() throws Exception {
        Row[] rowsData =
            new Row[] {
                Row.of("1", new Timestamp(117, 11, 1, 0, 0, 0, 0),
9.59076113897809),
                Row.of("1", new Timestamp(117, 11, 2, 0, 0, 0, 0),
8.51959031601596),
                Row.of("2", new Timestamp(117, 11, 3, 0, 0, 0, 0),
9.59076113897809),
                Row.of("1", new Timestamp(117, 11, 4, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 5, 0, 0, 0, 0),
```

```
8.51959031601596),
                Row.of("1", new Timestamp(117, 11, 6, 0, 0, 0, 0),
8.07246736935477),
                Row.of("2", new Timestamp(117, 11, 7, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 8, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 9, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 10, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 11, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 12, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 13, 0, 0, 0, 0),
8.18367658262066),
                Row.of("2", new Timestamp(117, 11, 14, 0, 0, 0, 0),
8.18367658262066),
                Row.of("1", new Timestamp(117, 11, 15, 0, 0, 0, 0),
7.8935720735049),
                Row.of("1", new Timestamp(117, 11, 16, 0, 0, 0, 0),
7.78364059622125),
                Row.of("2", new Timestamp(117, 11, 17, 0, 0, 0, 0),
8.07246736935477),
                Row.of("1", new Timestamp(117, 11, 18, 0, 0, 0, 0),
8.41405243249672),
                Row.of("1", new Timestamp(117, 11, 19, 0, 0, 0, 0),
8.82922635473185),
                Row.of("1", new Timestamp(117, 11, 20, 0, 0, 0, 0),
8.38251828808963),
                Row.of("1", new Timestamp(117, 11, 21, 0, 0, 0, 0),
8.06965530688617),
                Row.of("1", new Timestamp(117, 11, 22, 0, 0, 0, 0),
9.59076113897809),
                Row.of("1", new Timestamp(117, 11, 23, 0, 0, 0, 0),
8.51959031601596),
                Row.of("1", new Timestamp(117, 11, 24, 0, 0, 0, 0),
8.18367658262066),
                Row.of("1", new Timestamp(117, 11, 25, 0, 0, 0, 0),
8.07246736935477),
                Row.of("1", new Timestamp(117, 11, 26, 0, 0, 0, 0),
7.8935720735049),
                Row.of("1", new Timestamp(117, 11, 27, 0, 0, 0, 0),
7.78364059622125),
                Row.of("1", new Timestamp(117, 11, 28, 0, 0, 0, 0),
8.41405243249672),
                Row.of("1", new Timestamp(117, 11, 29, 0, 0, 0, 0),
```

```
8.82922635473185),
                Row.of("1", new Timestamp(117, 12, 1, 0, 0, 0, 0),
8.38251828808963),
                Row.of("1", new Timestamp(117, 12, 2, 0, 0, 0, 0),
8.06965530688617),
                Row.of("2", new Timestamp(117, 12, 3, 0, 0, 0, 0),
8.07246736935477),
                Row.of("2", new Timestamp(117, 12, 4, 0, 0, 0, 0),
7.8935720735049),
                Row.of("2", new Timestamp(117, 12, 5, 0, 0, 0, 0),
7.78364059622125),
                Row.of("2", new Timestamp(117, 12, 6, 0, 0, 0, 0),
8.41405243249672),
                Row.of("2", new Timestamp(117, 12, 7, 0, 0, 0, 0),
8.82922635473185),
                Row.of("2", new Timestamp(117, 12, 8, 0, 0, 0, 0),
8.38251828808963),
                Row.of("2", new Timestamp(117, 12, 9, 0, 0, 0, 0),
8.06965530688617)
            };
        String[] colNames = new String[] {"id", "ds1", "y1"};

        //train batch model.
        MemSourceBatchOp source = new MemSourceBatchOp(Arrays.asList(rowsData),
colNames);

        ProphetTrainBatchOp model = new ProphetTrainBatchOp()
            .setTimeCol("ds1")
            .setValueCol("y1");

        source.link(model).print();

        //construct times series by id.
        GroupByBatchOp groupData = new GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("mtable_agg(ts, val) as data");

        ProphetPredictBatchOp prophetPredict = new ProphetPredictBatchOp()
            .setValueCol("ts")
            .setPredictNum(4)
            .setPredictionCol("pred");

        prophetPredict.linkFrom(model, source.link(groupData)).print();
    }
}
```

运行结果

id	data	
1	{"data":{"ts":["1970-01-01 08:00:00.001","1970-01-01 08:00:00.002","1970-01-01 08:00:00.003","1970-01-01 08:00:00.004","1970-01-01 08:00:00.005","1970-01-01 08:00:00.006","1970-01-01 08:00:00.007","1970-01-01 08:00:00.008","1970-01-01 08:00:00.009","1970-01-01 08:00:00.01"],"val":[10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0]},"schema":"ts TIMESTAMP,val DOUBLE"}	{"data":{"ts":["1970-01-01 08:00:0 01-01 08:00:00.013","1970-01-01 08:00:00.015","1970-01-01 08:00 08:00:00.017","1970-01-01 08:00 08:00:00.019","1970-01-01 08:00 08:00:00.021","1970-01-01 08:00 [20.0,21.0,22.0,23.0,24.0,25.0,26 TIMESTAMP,val DOUBLE"}

t

Shift (ShiftBatchOp)

Java 类名：com.alibaba.alink.operator.batch.timeseries.ShiftBatchOp

Python 类名：ShiftBatchOp

功能介绍

给定分组，对每一组的数据使用Shift进行时间序列预测,使用ShiftNum之前的数据作为预测结果。

使用方式

使用方式

参考文档 https://www.yuque.com/pinshu/alink_guide/xbp5ky

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
valueCol	value列，类型为MTable	value列，类型为MTable	String	✓	所选列类型为[M_TABLE]	
predictNum	预测条数	预测条数	Integer			1
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
shiftNum	shift个数	shift个数	Integer			7
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

import time, datetime
import numpy as np
import pandas as pd

data = pd.DataFrame([
            [1,  datetime.datetime.fromtimestamp(1), 10.0],
            [1,  datetime.datetime.fromtimestamp(2), 11.0],
            [1,  datetime.datetime.fromtimestamp(3), 12.0],
            [1,  datetime.datetime.fromtimestamp(4), 13.0],
            [1,  datetime.datetime.fromtimestamp(5), 14.0],
            [1,  datetime.datetime.fromtimestamp(6), 15.0],
            [1,  datetime.datetime.fromtimestamp(7), 16.0],
            [1,  datetime.datetime.fromtimestamp(8), 17.0],
            [1,  datetime.datetime.fromtimestamp(9), 18.0],
            [1,  datetime.datetime.fromtimestamp(10), 19.0]
])

source = dataframeToOperator(data, schemaStr='id int, ts timestamp, val
double', op_type='batch')

source.link(
        GroupByBatchOp()
            .setGroupByPredicate("id")
            .setSelectClause("id, mtable_agg(ts, val) as data")
        ).link(ShiftBatchOp()
                    .setValueCol("data")
                    .setShiftNum(7)
                    .setPredictNum(12)
                    .setPredictionCol("predict")
        ).print()
```

Java 代码

```
package com.alibaba.alink.operator.batch.timeseries;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.sql.GroupByBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;

import org.junit.Test;

import java.sql.Timestamp;
import java.util.Arrays;
```

```
import java.util.List;

public class ShiftBatchOpTest {

    @Test
    public void test() throws Exception {
        List <Row> mTableData = Arrays.asList(
            Row.of(1, new Timestamp(1), 10.0),
            Row.of(1, new Timestamp(2), 11.0),
            Row.of(1, new Timestamp(3), 12.0),
            Row.of(1, new Timestamp(4), 13.0),
            Row.of(1, new Timestamp(5), 14.0),
            Row.of(1, new Timestamp(6), 15.0),
            Row.of(1, new Timestamp(7), 16.0),
            Row.of(1, new Timestamp(8), 17.0),
            Row.of(1, new Timestamp(9), 18.0),
            Row.of(1, new Timestamp(10), 19.0)
        );

        MemSourceBatchOp source = new MemSourceBatchOp(mTableData, new String[]
{"id", "ts", "val"});

        source
            .link(
                new GroupByBatchOp()
                    .setGroupByPredicate("id")
                    .setSelectClause("mtable_agg(ts, val) as data")
            )
            .link(
                new ShiftBatchOp()
                    .setGroupCol("id")
                    .setValueCol("data")
                    .setShiftNum(7)
                    .setPredictNum(12)
                    .setPredictionCol("predict")
            )
            .print();
    }
}
```

运行结果

id	data	

1	{"data":{"ts":["1970-01-01 08:00:00.001","1970-01-01 08:00:00.002","1970-01-01 08:00:00.003","1970-01-01 08:00:00.004","1970-01-01 08:00:00.005","1970-01-01 08:00:00.006","1970-01-01 08:00:00.007","1970-01-01 08:00:00.008","1970-01-01 08:00:00.009","1970-01-01 08:00:00.01"],"val":[10.0,11.0,12.0,13.0,14.0,15.0,16.0,17.0,18.0,19.0]},"schema":"ts TIMESTAMP,val DOUBLE"}	{"data":{"ts":["1970-01-01 08:00:0 01-01 08:00:00.013","1970-01-01 08:00:00.015","1970-01-01 08:00 08:00:00.017","1970-01-01 08:00 08:00:00.019","1970-01-01 08:00 08:00:00.021","1970-01-01 08:00 [13.0,14.0,15.0,16.0,17.0,18.0,19 TIMESTAMP,val DOUBLE"}

Agg表查找模型 (AggLookup)

Java 类名：com.alibaba.alink.pipeline.dataproc.AggLookup

Python 类名：AggLookup

功能介绍

需要查找多个值，并统计结果的总和、平均值、最大最小值或拼接查询结果时，可以使用聚合查找，该组件有两个输入，依次是模型数据表和原始数据表。模型数据有两列，依次是String类型和DenseVector类型，原始数据有任意行和列，每列都是String类型。原始数据默认使用空格作为单词的分隔符。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
clause	运算语句	运算语句	String	✓		
delimiter	分隔符	用来分割字符串	String			" "
modelFilePath	模型的文件路径	模型的文件路径	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["the quality of the word vectors increases"],
    ["amount of the training data increases"],
    ["the training speed is significantly improved"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='sentence string')

df2 = pd.DataFrame([
    ["the", "0.6343,0.8561,0.1249,0.4701"],
    ["training", "0.2753,0.2444,0.3699,0.6048"],
    ["of", "0.3160,0.3675,0.1649,0.4116"],
    ["increases", "1.0372,0.6092,0.1050,0.2630"],
    ["word", "0.9911,0.6338,0.4570,0.8451"],
    ["vectors", "0.8780,0.4500,0.5455,0.7495"],
    ["speed", "0.9504,0.3168,0.7484,0.6965"],
    ["significantly", "-0.0465,0.6597,0.0906,0.7137"],
    ["quality", "0.9745,0.7521,0.8874,0.5192"],
    ["is", "0.8221,0.0487,-0.0065,0.4088"],
    ["improved", "0.1910,0.0723,0.8216,0.4367"],
    ["data", "0.8985,0.0117,0.8083,0.9636"],
    ["amount", "0.9786,0.1470,0.7385,0.8856"]
])

modelOp = BatchOperator.fromDataframe(df2, schemaStr="id string, vec string")

AggLookup()\
    .setModelData(modelOp) \
    .setClause("CONCAT(sentence,2) as concat, AVG(sentence) as avg,
SUM(sentence) as sum,MAX(sentence) as max,MIN(sentence) as min") \
    .setDelimiter(" ") \
    .setReservedCols([]) \
    .transform(inOp)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;
```

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.AggLookup;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AggLookupTest {
    @Test
    public void testAggLookup() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("the quality of the word vectors increases"),
            Row.of("amount of the training data increases"),
            Row.of("the training speed is significantly improved")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "sentence string");
        List <Row> df2 = Arrays.asList(
            Row.of("the", "0.6343,0.8561,0.1249,0.4701"),
            Row.of("training", "0.2753,0.2444,0.3699,0.6048"),
            Row.of("of", "0.3160,0.3675,0.1649,0.4116"),
            Row.of("increases", "1.0372,0.6092,0.1050,0.2630"),
            Row.of("word", "0.9911,0.6338,0.4570,0.8451"),
            Row.of("vectors", "0.8780,0.4500,0.5455,0.7495"),
            Row.of("speed", "0.9504,0.3168,0.7484,0.6965"),
            Row.of("significantly", "-0.0465,0.6597,0.0906,0.7137"),
            Row.of("quality", "0.9745,0.7521,0.8874,0.5192"),
            Row.of("is", "0.8221,0.0487,-0.0065,0.4088"),
            Row.of("improved", "0.1910,0.0723,0.8216,0.4367"),
            Row.of("data", "0.8985,0.0117,0.8083,0.9636"),
            Row.of("amount", "0.9786,0.1470,0.7385,0.8856")
        );
        BatchOperator <?> modelOp = new MemSourceBatchOp(df2, "id string, vec
string");
        new AggLookup()
            .setModelData(modelOp)
            .setClause("CONCAT(sentence,2) as concat, AVG(sentence) as avg,
SUM(sentence) as sum,MAX(sentence) as max,MIN(sentence) as min")
            .setDelimiter(" ")
            .transform(inOp)
            .print();
    }
}
```

脚本输出结果

concat

0.6343 0.8561 0.1249 0.4701 0.9745 0.7521 0.8874 0.5192
0.9786 0.147 0.7385 0.8856 0.316 0.3675 0.1649 0.4116
0.6343 0.8561 0.1249 0.4701 0.2753 0.2444 0.3699 0.6048

avg	sum	max	min
0.7807 0.6464 0.3442 0.5326	5.4654 4.5248 2.4096 3.7286	1.0372 0.8561 0.8874 0.8451	0.316 0.3675 0.105 0.263
0.6899 0.3726 0.3852 0.5997	4.1399 2.2359 2.3115 3.5987	1.0372 0.8561 0.8083 0.9636	0.2753 0.0117 0.105 0.263
0.4710 0.3663 0.3581 0.5550	2.8266 2.1980 2.1489 3.3306	0.9504 0.8561 0.8216 0.7137	-0.0465 0.0487 -0.0065 0.4088

缺失值填充 (Imputer)

Java 类名：com.alibaba.alink.pipeline.dataproc.Imputer

Python 类名：Imputer

功能介绍

填充缺失值，生成的模型可以用于其他数据的预处理过程

支持的填充策略包含最大值，最小值，均值和指定数值，默认为均值填充

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
fillValue	填充缺失值	自定义的填充值。当strategy为value时，读取fillValue的值	String			null
modelFilePath	模型的文件路径	模型的文件路径	String			null

outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			fals
strategy	缺失值填充规则	缺失值填充的规则，支持mean，max，min或者value。选择value时，需要读取fillValue的值	String		"MEAN", "MIN", "MAX", "VALUE"	"ME
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1],
            [None, None, None]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]

inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 double')
```

```
sinOp = StreamOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 double')

model = Imputer()\
           .setSelectedCols(selectedColNames)\
           .fit(inOp)

model.transform(inOp).print()

model.transform(sinOp).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.pipeline.dataproc.Imputer;
import com.alibaba.alink.pipeline.dataproc.ImputerModel;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ImputerTrainBatchOpTest {
    @Test
    public void testImputerTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1),
            Row.of(null, null, null)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df_data, "col1 string,
col2 double, col3 int");
        ImputerModel imputerModel = new Imputer()
            .setSelectedCols(selectedColNames)
```

```
            .fit(inOp);
        imputerModel.transform(inOp).print();
        StreamOperator <?> sinOp = new MemSourceStreamOp(df_data, "col1 string,
col2 double, col3 int");
        imputerModel.transform(inOp).print();
        StreamOperator.execute();
    }
}
```

运行结果

col1	col2	col3
a	10.000000	100
b	-2.500000	9
c	100.200000	1
d	-99.900000	100
a	1.400000	1
b	-2.200000	9
c	100.900000	1
null	15.414286	31

缺失值填充模型 (ImputerModel)

Java 类名：com.alibaba.alink.pipeline.dataproc.ImputerModel

Python 类名：ImputerModel

功能介绍

填充缺失值模型，由Imputer生成，可以对数据执行缺失值填充操作

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1],
            [None, None, None]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]

inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 double')
sinOp = StreamOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 double')

model = Imputer()\
           .setSelectedCols(selectedColNames)\
           .fit(inOp)

model.transform(inOp).print()

model.transform(sinOp).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.pipeline.dataproc.Imputer;
import com.alibaba.alink.pipeline.dataproc.ImputerModel;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class ImputerTrainBatchOpTest {
    @Test
    public void testImputerTrainBatchOp() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1),
            Row.of(null, null, null)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df_data, "col1 string,
col2 double, col3 int");
        ImputerModel imputerModel = new Imputer()
            .setSelectedCols(selectedColNames)
            .fit(inOp);
        imputerModel.transform(inOp).print();
        StreamOperator <?> sinOp = new MemSourceStreamOp(df_data, "col1 string,
col2 double, col3 int");
        imputerModel.transform(inOp).print();
        StreamOperator.execute();
    }
}
```

运行结果

col1	col2	col3
a	10.000000	100
b	-2.500000	9
c	100.200000	1
d	-99.900000	100
a	1.400000	1
b	-2.200000	9
c	100.900000	1
null	15.414286	31

IndexToString (IndexToString)

Java 类名：com.alibaba.alink.pipeline.dataproc.IndexToString

Python 类名：IndexToString

功能介绍

基于StringIndexer模型，将一列整数映射为字符串。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelName	模型名字	模型名字	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ["football"],
    ["football"],
    ["football"],
    ["basketball"],
    ["basketball"],
    ["tennis"],
])

train_data = BatchOperator.fromDataframe(df_data, schemaStr='f0 string')
data = StreamOperator.fromDataframe(df_data, schemaStr='f0 string')

stringIndexer = StringIndexer() \
    .setModelName("string_indexer_model") \
    .setSelectedCol("f0") \
    .setOutputCol("f0_indexed") \
    .setStringOrderType("frequency_asc").fit(train_data)

batch_model = stringIndexer.transform(train_data)
indexed = stringIndexer.transform(data)

indexToStrings = IndexToStringPredictStreamOp(batch_model) \
    .setSelectedCol("f0_indexed") \
    .setOutputCol("f0_indxed_unindexed")

indexToStrings.linkFrom(indexed).print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.dataproc.IndexToStringPredictStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.dataproc.StringIndexer;
import com.alibaba.alink.pipeline.dataproc.StringIndexerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class IndexToStringTest {
    @Test
    public void testIndexToString() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("football"),
            Row.of("football"),
            Row.of("football"),
            Row.of("basketball"),
            Row.of("basketball"),
            Row.of("tennis")
        );
        BatchOperator <?> train_data = new MemSourceBatchOp(df_data, "f0
string");
        StreamOperator <?> data = new MemSourceStreamOp(df_data, "f0 string");
        StringIndexerModel stringIndexer = new StringIndexer()
            .setModelName("string_indexer_model")
            .setSelectedCol("f0")
            .setOutputCol("f0_indexed")
            .setStringOrderType("frequency_asc").fit(train_data);
        BatchOperator batch_model = stringIndexer.transform(train_data);
        StreamOperator indexed = stringIndexer.transform(data);
        StreamOperator <?> indexToStrings = new
IndexToStringPredictStreamOp(batch_model)
            .setSelectedCol("f0_indexed")
            .setOutputCol("f0_indxed_unindexed");
        indexToStrings.linkFrom(indexed).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f0_indexed	f0_indxed_unindexed
football	2	football
football	2	football
football	2	football
basketball	1	basketball
basketball	1	basketball
tennis	0	tennis

Json值抽取 (JsonValue)

Java 类名：com.alibaba.alink.pipeline.dataproc.JsonValue

Python 类名：JsonValue

功能介绍

该组件完成json字符串中的信息抽取，按照用户给定的Path 抓取出相应的信息。该组件支持多Path抽取。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonPath	Json 路径数组	用来指定 Json 抽取的内容。	String[]	✓		
outputCols	输出结果列列名数组	输出结果列列名数组，必选	String[]	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		
outputColTypes	输出结果列列类型数组	输出结果列类型数组	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
skipFailed	是否跳过错误	当遇到抽取值为null 时是否跳过	boolean			false
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
```

```
    ["{a:boy,b:{b1:1,b2:2}}"],
    ["{a:girl,b:{b1:1,b2:2}}"]])

batchData = BatchOperator.fromDataframe(df, schemaStr='str string')

JsonValue()\
    .setJsonPath(["$.a","$.b.b1"])\
    .setSelectedCol("str")\
    .setOutputCols(["f0","f1"])\
    .transform(batchData)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.JsonValue;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonValueTest {
    @Test
    public void testJsonValue() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("{a:boy,b:{b1:1,b2:2}}")
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "str string");
        new JsonValue()
            .setJsonPath("$.a", "$.b.b1")
            .setSelectedCol("str")
            .setOutputCols("f0", "f1")
            .transform(batchData)
            .print();
    }
}
```

运行结果

str	f0	f1
{a:boy,b:{b1:1,b2:2}}	boy	1
{a:girl,b:{b1:1,b2:2}}	girl	1

表查找 (Lookup)

Java 类名：com.alibaba.alink.pipeline.dataproc.Lookup

Python 类名：Lookup

功能介绍

支持数据查找功能，支持多个key的查找，并将查找后的结果中的value列添加到待查询数据后面。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	
mapKeyCols	Key列名	模型中对应的查找等值的列名	String[]		
mapValueCols	Values列名	模型中需要拼接到样本中的列名	String[]		
modelFilePath	模型的文件路径	模型的文件路径	String		
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		
modelStreamUpdateMethod	模型更新方法	模型更新方法，可选 COMPLETE（全量更新）或者 INCREMENT（增量更新）	String		"COMPLE "INCREME

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

data_df = pd.DataFrame([
    ["10", 2.0],
    ["1", 2.0],
    ["-3", 2.0],
    ["5", 1.0]
])

inOp = StreamOperator.fromDataframe(data_df, schemaStr='f0 string, f1 double')

data_df = pd.DataFrame([
    ["1", "value1"],
    ["2", "value2"],
    ["5", "value5"]
])

modelOp = BatchOperator.fromDataframe(data_df, schemaStr="key_col string,
value_col string")

Lookup()\
```

```
    .setModelData(modelOp)\
    .setMapKeyCols(["key_col"])\
    .setMapValueCols(["value_col"]) \
    .setSelectedCols(["f0"])\
    .transform(inOp)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.dataproc.Lookup;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LookupTest {
    @Test
    public void testLookup() throws Exception {
        List <Row> data_df = Arrays.asList(
            Row.of("10", 2.0),
            Row.of("1", 2.0),
            Row.of("-3", 2.0),
            Row.of("5", 1.0)
        );
        StreamOperator <?> inOp = new MemSourceStreamOp(data_df, "f0 string, f1
double");
        data_df = Arrays.asList(
            Row.of("1", "value1"),
            Row.of("2", "value2"),
            Row.of("5", "value5")
        );
        BatchOperator <?> modelOp = new MemSourceBatchOp(data_df, "key_col
string, value_col string");
        new Lookup()
            .setModelData(modelOp)
            .setMapKeyCols("key_col")
            .setMapValueCols("value_col")
            .setSelectedCols("f0")
            .transform(inOp)
```

```
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	value_col
10	2.0	null
1	2.0	value1
-3	2.0	null
5	1.0	value5

Redis表查找 (LookupRedis)

Java 类名：com.alibaba.alink.pipeline.dataproc.LookupRedis

Python 类名：LookupRedis

功能介绍

支持数据查找功能，支持多个key的查找，并将查找后的结果中的value列添加到待查询数据后面。

功能类似于 LookUp ， 不同的是被查找的数据存储在 Redis 中。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"， 例如"f0 string, f1 bigint, f2 double"	String	✓		
pluginVersion	插件版本号	插件版本号	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
clusterMode	集群模式	是集群模式还是单机模式	Boolean			false
databaseIndex	数据库索引号	数据库索引号	Long			
pipelineSize	流水线大小	Redis 发送命令流水线的大小	Integer			1
redisIPs	Redis IP	Redis 集群的 IP/端口	String[]			
redisPassword	Redis 密码	Redis 服务器密码	String			
reservedCols	算法保留列名	算法保留列	String[]			null

timeout	超时	关闭连接的超时时间	Integer		

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    ["id001", 123, 45.6, "str"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id string, col0 bigint, col1 double, col2 string')

redisIP = "*"
redisPort = 26379

RedisSinkBatchOp()\
    .setRedisIP(redisIP)\
    .setRedisPort(redisPort)\
    .setKeyCols(["id"])\
    .setPluginVersion("2.9.0")\
    .setValueCols(["col0", "col1", "col2"])\
    .linkFrom(inOp)

BatchOperator.execute()

df2 = pd.DataFrame([
    ["id001"]
])
needToLookup = StreamOperator.fromDataframe(df2, schemaStr="id string")

LookupRedis()\
    .setRedisIP(redisIP)\
    .setRedisPort(redisPort)\
    .setPluginVersion("2.9.0")\
    .setSelectedCols(["id"])\
    .setOutputSchemaStr("col0 bigint, col1 double, col2 string")\
    .transform(needToLookup)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.common.AlinkGlobalConfiguration;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.sink.RedisSinkBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.dataproc.LookupRedis;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.Collections;

public class LookupRedisTest extends AlinkTestBase {
    @Test
    public void map() throws Exception {
        String redisIP = "*";
        int redisPort = 26379;

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(
            Collections.singletonList(Row.of("id001", 123L, 45.6, "str")),
            "id string, col0 bigint, col1 double, col2 string"
        );

        new RedisSinkBatchOp()
            .setRedisIP(redisIP)
            .setRedisPort(redisPort)
            .setKeyCols("id")
            .setPluginVersion("2.9.0")
            .setValueCols("col0", "col1", "col2")
            .linkFrom(memSourceBatchOp);

        BatchOperator.execute();

        MemSourceStreamOp needToLookup = new MemSourceStreamOp(
            Collections.singletonList(Row.of("id001")),
            "id string"
        );

        new LookupRedis()
            .setRedisIP(redisIP)
            .setRedisPort(redisPort)
            .setPluginVersion("2.9.0")
            .setSelectedCols("id")
            .setOutputSchemaStr("col0 bigint, col1 double, col2 string")
            .transform(needToLookup)
            .print();
```

```
        StreamOperator.execute();
    }

}
```

运行结果

| id | col0 | col1 | col2 | |-------+------+---------+------| | id001 | 123 | 45.6000 | str |

绝对值最大化 (MaxAbsScaler)

Java 类名：com.alibaba.alink.pipeline.dataproc.MaxAbsScaler

Python 类名：MaxAbsScaler

功能介绍

- 绝对值最大标准化是对数据按照绝对值最大值进行标准化的组件, 将数据归一到-1和1之间。
- 训练过程，生成绝对值最大标准化模型MaxAbsScalerModel

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]

inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')


sinOp = StreamOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')


model = MaxAbsScaler()\
           .setSelectedCols(selectedColNames)\
           .fit(inOp)

model.transform(inOp).print()

model.transform(sinOp).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
```

```
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.dataproc.MaxAbsScaler;
import com.alibaba.alink.pipeline.dataproc.MaxAbsScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MaxAbsScalerTest {
    @Test
    public void testMaxAbsScaler() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
        StreamOperator <?> sinOp = new MemSourceStreamOp(df, "col1 string, col2
double, col3 int");
        MaxAbsScalerModel model = new MaxAbsScaler()
            .setSelectedCols(selectedColNames)
            .fit(inOp);
        model.transform(inOp).print();
        model.transform(sinOp).print();
        StreamOperator.execute();
    }
}
```

绝对值最大化模型 (MaxAbsScalerModel)

Java 类名：com.alibaba.alink.pipeline.dataproc.MaxAbsScalerModel

Python 类名：MaxAbsScalerModel

功能介绍

- 绝对值最大标准化是对数据按照绝对值最大值进行标准化的组件, 将数据归一到-1和1之间。
- 该组件为绝对值最大标准化模型，由MaxAbsScaler生成，用于预处理其他数据

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]

inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')


sinOp = StreamOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')


model = MaxAbsScaler()\
           .setSelectedCols(selectedColNames)\
           .fit(inOp)

model.transform(inOp).print()

model.transform(sinOp).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.dataproc.MaxAbsScaler;
import com.alibaba.alink.pipeline.dataproc.MaxAbsScalerModel;
import org.junit.Test;

import java.util.Arrays;
```

```
import java.util.List;

public class MaxAbsScalerModelTest {
    @Test
    public void testMaxAbsScalerModel() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
        StreamOperator <?> sinOp = new MemSourceStreamOp(df, "col1 string, col2
double, col3 int");
        MaxAbsScalerModel model = new MaxAbsScaler()
            .setSelectedCols(selectedColNames)
            .fit(inOp);
        model.transform(inOp).print();
        model.transform(sinOp).print();
        StreamOperator.execute();
    }
}
```

归一化 (MinMaxScaler)

Java 类名：com.alibaba.alink.pipeline.dataproc.MinMaxScaler

Python 类名：MinMaxScaler

功能介绍

归一化是对数据进行归一的组件, 将数据归一到minValue和maxValue之间，value最终结果为 (value - min) / (max - min) * (maxValue - minValue) + minValue，最终结果的范围为[minValue, maxValue]。

minValue和maxValue由用户指定，默认为0和1。

该组件为训练组件，生成归一化模型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
max	归一化的上界	归一化的上界	Double			1.0
min	归一化的下界	归一化的下界	Double			0.0

modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]

inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')
```

```
sinOp = StreamOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')

model = MinMaxScaler()\
           .setSelectedCols(selectedColNames)\
           .fit(inOp)

model.transform(inOp).print()

model.transform(sinOp).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.dataproc.MinMaxScaler;
import com.alibaba.alink.pipeline.dataproc.MinMaxScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MinMaxScalerTest {
    @Test
    public void testMinMaxScaler() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
        StreamOperator <?> sinOp = new MemSourceStreamOp(df, "col1 string, col2
double, col3 int");
        MinMaxScalerModel model = new MinMaxScaler()
```

```
            .setSelectedCols(selectedColNames)
            .fit(inOp);
        model.transform(inOp).print();
        model.transform(sinOp).print();
        StreamOperator.execute();
    }
}
```

运行结果

col1	col2	col3
a	0.5473	1.0000
b	0.4851	0.0808
c	0.9965	0.0000
d	0.0000	1.0000
a	0.5045	0.0000
b	0.4866	0.0808
c	1.0000	0.0000
col1	col2	col3
----	----	----
b	0.4866	0.0808
c	1.0000	0.0000
b	0.4851	0.0808
c	0.9965	0.0000
a	0.5045	0.0000
a	0.5473	1.0000
d	0.0000	1.0000

归一化模型 (MinMaxScalerModel)

Java 类名：com.alibaba.alink.pipeline.dataproc.MinMaxScalerModel

Python 类名：MinMaxScalerModel

功能介绍

归一化是对数据进行归一的组件, 将数据归一到minValue和maxValue之间，value最终结果为 (value - min) / (max - min) * (maxValue - minValue) + minValue，最终结果的范围为[minValue, maxValue]。

minValue和maxValue由用户指定，默认为0和1。

该组件为模型组件，用于数据预测。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]

inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')

sinOp = StreamOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')

model = MinMaxScaler()\
           .setSelectedCols(selectedColNames)\
           .fit(inOp)

model.transform(inOp).print()

model.transform(sinOp).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
```

```
import com.alibaba.alink.pipeline.dataproc.MinMaxScaler;
import com.alibaba.alink.pipeline.dataproc.MinMaxScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MinMaxScalerModelTest {
    @Test
    public void testMinMaxScalerModel() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
        StreamOperator <?> sinOp = new MemSourceStreamOp(df, "col1 string, col2
double, col3 int");
        MinMaxScalerModel model = new MinMaxScaler()
            .setSelectedCols(selectedColNames)
            .fit(inOp);
        model.transform(inOp).print();
        model.transform(sinOp).print();
        StreamOperator.execute();
    }
}
```

运行结果

col1	col2	col3
a	0.5473	1.0000
b	0.4851	0.0808
c	0.9965	0.0000
d	0.0000	1.0000
a	0.5045	0.0000
b	0.4866	0.0808

c	1.0000	0.0000
col1	col2	col3
----	----	----
b	0.4866	0.0808
a	0.5473	1.0000
b	0.4851	0.0808
c	0.9965	0.0000
a	0.5045	0.0000
d	0.0000	1.0000
c	1.0000	0.0000

MultiStringIndexer (MultiStringIndexer)

Java 类名：com.alibaba.alink.pipeline.dataproc.MultiStringIndexer

Python 类名：MultiStringIndexer

功能介绍

MultiStringIndexer训练组件的作用是训练一个模型用于将多列字符串映射为整数。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null， "error"表示抛异常	String		"KEEP", "ERR "SKIP"
modelFilePath	模型的文件路径	模型的文件路径	String		
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

stringOrderType	Token排序方法	Token排序方法	String		"RANDOM", "FREQUENC "FREQUENC "ALPHABET_ "ALPHABET_
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ["football"],
    ["football"],
    ["football"],
    ["basketball"],
    ["basketball"],
```

```
    ["tennis"],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='f0 string')

stringindexer = MultiStringIndexer() \
    .setSelectedCols(["f0"]) \
    .setOutputCols(["f0_indexed"]) \
    .setStringOrderType("frequency_asc")

stringindexer.fit(data).transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.MultiStringIndexer;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MultiStringIndexerTest {
    @Test
    public void testMultiStringIndexer() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("football"),
            Row.of("football"),
            Row.of("football"),
            Row.of("basketball"),
            Row.of("basketball"),
            Row.of("tennis")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "f0 string");
        MultiStringIndexer stringindexer = new MultiStringIndexer()
            .setSelectedCols("f0")
            .setOutputCols("f0_indexed")
            .setStringOrderType("frequency_asc");
        stringindexer.fit(data).transform(data).print();
    }
}
```

运行结果

f0	f0_indexed
football	2
football	2
football	2
basketball	1
basketball	1
tennis	0

标准化 (StandardScaler)

Java 类名：com.alibaba.alink.pipeline.dataproc.StandardScaler

Python 类名：StandardScaler

功能介绍

标准化是对数据进行按正态化处理的组件

训练过程计算数据的均值和标准差，生成标准化模型

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
withMean	是否使用均值	是否使用均值，默认使用	Boolean			true
withStd	是否使用标准差	是否使用标准差，默认使用	Boolean			true
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]

inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')
```

```
sinOp = StreamOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')

model = StandardScaler()\
           .setSelectedCols(selectedColNames)\
           .fit(inOp)

model.transform(inOp).print()

model.transform(sinOp).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.dataproc.StandardScaler;
import com.alibaba.alink.pipeline.dataproc.StandardScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StandardScalerTest {
    @Test
    public void testStandardScaler() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
        StreamOperator <?> sinOp = new MemSourceStreamOp(df, "col1 string, col2
double, col3 int");
        StandardScalerModel model = new StandardScaler()
```

```
            .setSelectedCols(selectedColNames)
            .fit(inOp);
        model.transform(inOp).print();
        model.transform(sinOp).print();
        StreamOperator.execute();
    }
}
```

运行结果

col1	col2	col3
a	-0.0784	1.4596
b	-0.2592	-0.4814
c	1.2270	-0.6521
d	-1.6687	1.4596
a	-0.2028	-0.6521
b	-0.2549	-0.4814
c	1.2371	-0.6521
col1	col2	col3
----	----	----
c	1.2371	-0.6521
b	-0.2592	-0.4814
c	1.2270	-0.6521
b	-0.2549	-0.4814
a	-0.0784	1.4596
a	-0.2028	-0.6521
d	-1.6687	1.4596

标准化模型 (StandardScalerModel)

Java 类名：com.alibaba.alink.pipeline.dataproc.StandardScalerModel

Python 类名：StandardScalerModel

功能介绍

标准化是对数据进行按正态化处理的组件

标准化模型，用于数据的标准化的处理过程

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
            ["a", 10.0, 100],
            ["b", -2.5, 9],
            ["c", 100.2, 1],
            ["d", -99.9, 100],
            ["a", 1.4, 1],
            ["b", -2.2, 9],
            ["c", 100.9, 1]
])

colnames = ["col1", "col2", "col3"]
selectedColNames = ["col2", "col3"]

inOp = BatchOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')

sinOp = StreamOperator.fromDataframe(df, schemaStr='col1 string, col2 double,
col3 long')

model = StandardScaler()\
           .setSelectedCols(selectedColNames)\
           .fit(inOp)

model.transform(inOp).print()

model.transform(sinOp).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.dataproc.StandardScaler;
import com.alibaba.alink.pipeline.dataproc.StandardScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class StandardScalerModelTest {
    @Test
    public void testStandardScalerModel() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 10.0, 100),
            Row.of("b", -2.5, 9),
            Row.of("c", 100.2, 1),
            Row.of("d", -99.9, 100),
            Row.of("a", 1.4, 1),
            Row.of("b", -2.2, 9),
            Row.of("c", 100.9, 1)
        );

        String[] selectedColNames = new String[] {"col2", "col3"};
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "col1 string, col2
double, col3 int");
        StreamOperator <?> sinOp = new MemSourceStreamOp(df, "col1 string, col2
double, col3 int");
        StandardScalerModel model = new StandardScaler()
            .setSelectedCols(selectedColNames)
            .fit(inOp);
        model.transform(inOp).print();
        model.transform(sinOp).print();
        StreamOperator.execute();
    }
}
```

运行结果

col1	col2	col3
a	-0.0784	1.4596
b	-0.2592	-0.4814
c	1.2270	-0.6521
d	-1.6687	1.4596
a	-0.2028	-0.6521
b	-0.2549	-0.4814
c	1.2371	-0.6521
col1	col2	col3
----	----	----
c	1.2371	-0.6521

b	-0.2592	-0.4814
c	1.2270	-0.6521
b	-0.2549	-0.4814
a	-0.0784	1.4596
a	-0.2028	-0.6521
d	-1.6687	1.4596

StringIndexer (StringIndexer)

Java 类名：com.alibaba.alink.pipeline.dataproc.StringIndexer

Python 类名：StringIndexer

功能介绍

StringIndexer训练组件的作用是训练一个模型用于将单列字符串映射为整数。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范
selectedCol	选中的列名	计算列对应的列名	String	✓	
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null， "error"表示抛异常	String		"KEEP", "ERR "SKIP"
modelFilePath	模型的文件路径	模型的文件路径	String		
modelName	模型名字	模型名字	String		
outputCol	输出结果列	输出结果列列名，可选，默认null	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

selectedCols	选中的列名数组	计算列对应的列名列表	String[]		
stringOrderType	Token排序方法	Token排序方法	String		"RANDOM", "FREQUENC "FREQUENC "ALPHABET_ "ALPHABET_
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
```

```
    ["football"],
    ["football"],
    ["football"],
    ["basketball"],
    ["basketball"],
    ["tennis"],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='f0 string')

stringindexer = StringIndexer() \
    .setSelectedCol("f0") \
    .setOutputCol("f0_indexed") \
    .setStringOrderType("frequency_asc")

stringindexer.fit(data).transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.StringIndexer;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StringIndexerTest {
    @Test
    public void testStringIndexer() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("football"),
            Row.of("football"),
            Row.of("football"),
            Row.of("basketball"),
            Row.of("basketball"),
            Row.of("tennis")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "f0 string");
        StringIndexer stringindexer = new StringIndexer()
            .setSelectedCol("f0")
            .setOutputCol("f0_indexed")
            .setStringOrderType("frequency_asc");
        stringindexer.fit(data).transform(data).print();
```

```
    }
}
```

运行结果

f0	f0_indexed
football	2
football	2
football	2
basketball	1
basketball	1
tennis	0

张量转向量 (TensorToVector)

Java 类名：com.alibaba.alink.pipeline.dataproc.TensorToVector

Python 类名：TensorToVector

功能介绍

转换张量类型为向量类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
convertMethod	转换方法	张量转换为向量的方法，可取 flatten, sum, mean, max, min.	String		"FLATTEN", "SUM", "MEAN", "MAX", "MIN"	"FLATTEN"
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ['DOUBLE#6#0.0 0.1 1.0 1.1 2.0 2.1']
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr = 'tensor string')

TensorToVector().setSelectedCol("tensor").transform(batch_data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Collections;
import java.util.List;

public class TensorToVectorTest {

    @Test
    public void testTensorToVector() throws Exception {
        List <Row> data = Collections.singletonList(Row.of("DOUBLE#6#0.0 0.1
1.0 1.1 2.0 2.1"));

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "tensor
string");

        new TensorToVector()
            .setSelectedCol("tensor")
            .transform(memSourceBatchOp)
            .print();
    }
}
```

运行结果

tensor
0.0 0.1 1.0 1.1 2.0 2.1

转MTable (ToMTable)

Java 类名：com.alibaba.alink.pipeline.dataproc.ToMTable

Python 类名：ToMTable

功能介绍

将输入列转换为MTable类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
outputCol	输出结果列	输出结果列列名，可选，默认 null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
```

```
    ['{"data":{"col0":[1],"col1":["2"],"label":[0]},"schema":"col0 INT, col1
VARCHAR,label INT"}']
])

data = BatchOperator.fromDataframe(df_data, schemaStr = 'vec string')

ToMTable().setSelectedCol("vec").transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

public class ToVectorDemoTest extends AlinkTestBase {
    @Test
    public void test() throws Exception {
        final String mTableStr = "{\"data\":{\"col0\":[1],\"col1\":
[\"2\"],\"label\":[0]},\"schema\":\"col0 INT, col1 VARCHAR,label INT\"}";

        Row[] rows = new Row[] {
            Row.of(mTableStr)
        };

        MemSourceBatchOp data = new MemSourceBatchOp(
            rows, new String[] {"m_table"}
        );
        new ToMTable().setSelectedCol("m_table").transform(data).print();
    }
}
```

运行结果

vec
{"data":{"col0":[1],"col1":["2"],"label":[0]},"schema":"col0 INT,col1 VARCHAR,label INT"}

转Tensor (ToTensor)

Java 类名：com.alibaba.alink.pipeline.dataproc.ToTensor

Python 类名：ToTensor

功能介绍

将指定列转为 Alink 的张量类型。

如果指定列为 String 类型，并且值为 Alink 张量或者向量 toString 的结果，那么张量类型和形状将自动获取。 否则的话，需要指定张量类型和张量形状。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
tensorDataType	要转换的张量数据类型	要转换的张量数据类型。	String		"FLOAT", "DOUBLE", "INT", "LONG", "BOOLEAN", "BYTE", "UBYTE", "STRING"	

tensorShape	张量形状	张量的形状，数组类型。	Long[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame(["FLOAT#6#0.0 0.1 1.0 1.1 2.0 2.1 "])
source = BatchOperator.fromDataframe(df, schemaStr='vec string')

toTensor = ToTensor() \
    .setSelectedCol("vec") \
    .setTensorShape([2, 3]) \
    .setTensorDataType("float")
toTensor.transform(source).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.ToTensor;
import org.junit.Test;

public class ToTensorTest {

    @Test
    public void testToTensor() throws Exception {
        Row[] rows = new Row[] {
            Row.of("FLOAT#6#0.0 0.1 1.0 1.1 2.0 2.1 ")
        };
        MemSourceBatchOp source = new MemSourceBatchOp(rows, new String[]
{"vec"});
```

```
        ToTensor toTensor = new ToTensor()
            .setSelectedCol("vec")
            .setTensorShape(2, 3)
            .setTensorDataType("float");
        toTensor.transform(source).print();
    }
}
```

运行结果

vec
FLOAT#2,3#0.0 0.1 1.0 1.1 2.0 2.1

转向量 (ToVector)

Java 类名：com.alibaba.alink.pipeline.dataproc.ToVector

Python 类名：ToVector

功能介绍

将输入列转换为向量类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
vectorType	要转换的Vector类型。	要转换的Vector类型。	String		"DENSE", "SPARSE"	null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df_data = pd.DataFrame([
    ['1 0 3 4']
])

data = BatchOperator.fromDataframe(df_data, schemaStr = 'vec string')

ToVector().setSelectedCol("vec").setVectorType("SPARSE").transform(data).print(
)
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.common.linalg.VectorType;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

public class ToVectorTest extends AlinkTestBase {
    @Test
    public void test() throws Exception {
        final String vecStr = "1 0 3 4";
        Row[] rows = new Row[] {
            Row.of(vecStr)
        };
        MemSourceBatchOp data = new MemSourceBatchOp(
            rows, new String[] {"vec"}
        );
        new ToVector()
            .setSelectedCol("vec")
            .setVectorType(VectorType.SPARSE)
            .transform(data)
            .print();
    }
}
```

运行结果

vec
$4$0:1.0 2:3.0 3:4.0

向量转张量 (VectorToTensor)

Java 类名：com.alibaba.alink.pipeline.dataproc.VectorToTensor

Python 类名：VectorToTensor

功能介绍

转换向量类型为张量类型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
tensorDataType	要转换的张量数据类型	要转换的张量数据类型。	String		"FLOAT", "DOUBLE", "INT", "LONG", "BOOLEAN", "BYTE", "UBYTE", "STRING"	
tensorShape	张量形状	张量的形状，数组类型。	Long[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ['0.0 0.1 1.0 1.1 2.0 2.1']
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr = 'vec string')

VectorToTensor().setSelectedCol("vec").transform(batch_data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Collections;
import java.util.List;

public class VectorToTensorTest {

    @Test
    public void testVectorToTensor() throws Exception {
        List <Row> data = Collections.singletonList(Row.of("0.0 0.1 1.0 1.1 2.0
2.1"));

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "vec
string");

        new VectorToTensor()
            .setSelectedCol("vec")
```

```
                .transform(memSourceBatchOp)
                .print();
        }

}
```

运行结果

vec
DOUBLE#6#0.0 0.1 1.0 1.1 2.0 2.1

列数据转CSV (ColumnsToCsv)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.ColumnsToCsv

Python 类名：ColumnsToCsv

功能介绍

Pipeline组件，将数据格式从 Columns 转成 Csv

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = ColumnsToCsv()\
    .setSelectedCols(["f0", "f1"])\
    .setReservedCols(["row"])\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.ColumnsToCsv;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ColumnsToCsvTest {
    @Test
    public void testColumnsToCsv() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new ColumnsToCsv()
            .setSelectedCols("f0", "f1")
            .setReservedCols("row")
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	csv
1	1.0,2.0
2	4.0,8.0

列数据转JSON (ColumnsToJson)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.ColumnsToJson

Python 类名：ColumnsToJson

功能介绍

Pipeline组件，将数据格式从 Columns 转成 Json

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])
```

```
data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = ColumnsToJson()\
    .setSelectedCols(["f0", "f1"])\
    .setReservedCols(["row"])\
    .setJsonCol("json")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.ColumnsToJson;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ColumnsToJsonTest {
    @Test
    public void testColumnsToJson() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new ColumnsToJson()
            .setSelectedCols("f0", "f1")
            .setReservedCols("row")
            .setJsonCol("json")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	json

1	{"f0":"1.0","f1":"2.0"}
2	{"f0":"4.0","f1":"8.0"}

列数据转KV (ColumnsToKv)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.ColumnsToKv

Python 类名：ColumnsToKv

功能介绍

将数据格式从 Columns 转成 Kv

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
kvCol	KV列名	KV列的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = ColumnsToKv()\
    .setSelectedCols(["f0", "f1"])\
    .setReservedCols(["row"])\
    .setKvCol("kv")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.ColumnsToKv;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ColumnsToKvTest {
    @Test
    public void testColumnsToKv() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new ColumnsToKv()
            .setSelectedCols("f0", "f1")
```

```
            .setReservedCols("row")
            .setKvCol("kv")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	kv
1	f0:1.0,f1:2.0
2	f0:4.0,f1:8.0

列数据转向量 (ColumnsToVector)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.ColumnsToVector

Python 类名：ColumnsToVector

功能介绍

将数据格式从 Columns 转成 Vector

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
vectorCol	向量列名	向量列对应的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
vectorSize	向量长度	向量长度	Long			-1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = ColumnsToVector()\
    .setSelectedCols(["f0", "f1"])\
    .setReservedCols(["row"])\
    .setVectorCol("vec")\
    .setVectorSize(5)\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.ColumnsToVector;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ColumnsToVectorTest {
    @Test
    public void testColumnsToVector() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator <?> op = new ColumnsToVector()
            .setSelectedCols("f0", "f1")
            .setReservedCols("row")
            .setVectorCol("vec")
            .setVectorSize(5)
            .transform(data);
```

```
        op.print();
    }
}
```

运行结果

row	vec
1	$5$1.0 2.0
2	$5$4.0 8.0

CSV转列数据 (CsvToColumns)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.CsvToColumns

Python 类名：CsvToColumns

功能介绍

将数据格式从 Csv 转成 Columns

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = CsvToColumns()\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .setReservedCols(["row"])\
    .setSchemaStr("f0 double, f1 double")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.CsvToColumns;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CsvToColumnsTest {
    @Test
    public void testCsvToColumns() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new CsvToColumns()
```

```
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .setReservedCols("row")
            .setSchemaStr("f0 double, f1 double")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	f0	f1
1	1.0	2.0
2	4.0	8.0

CSV转JSON (CsvToJson)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.CsvToJson

Python 类名：CsvToJson

功能介绍

将数据格式从 Csv 转成 Json

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
jsonCol	JSON列名	JSON列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = CsvToJson()\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .setReservedCols(["row"])\
    .setJsonCol("json")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.CsvToJson;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CsvToJsonTest {
    @Test
    public void testCsvToJson() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
```

```
double, f1 double");
        BatchOperator op = new CsvToJson()
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .setReservedCols("row")
            .setJsonCol("json")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	json
1	{"f0":"1.0","f1":"2.0"}
2	{"f0":"4.0","f1":"8.0"}

CSV转KV (CsvToKv)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.CsvToKv

Python 类名：CsvToKv

功能介绍

将数据格式从 Csv 转成 Kv

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
kvCol	KV列名	KV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"

quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = CsvToKv()\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .setReservedCols(["row"])\
    .setKvCol("kv")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.CsvToKv;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class CsvToKvTest {
    @Test
    public void testCsvToKv() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new CsvToKv()
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .setReservedCols("row")
            .setKvCol("kv")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	kv
1	f0:1.0,f1:2.0
2	f0:4.0,f1:8.0

CSV转向量 (CsvToVector)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.CsvToVector

Python 类名：CsvToVector

功能介绍

将数据格式从 Csv 转成 Vector

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null
vectorSize	向量长度	向量长度	Long			-1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = CsvToVector()\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .setReservedCols(["row"])\
    .setVectorCol("vec")\
    .setVectorSize(5)\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.CsvToVector;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CsvToVectorTest {
    @Test
    public void testCsvToVector() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
```

```
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new CsvToVector()
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .setReservedCols("row")
            .setVectorCol("vec")
            .setVectorSize(5)
            .transform(data);
        op.print();
    }
}
```

运行结果

row	vec
1	$5$1.0 2.0
2	$5$4.0 8.0

JSON转列数据 (JsonToColumns)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.JsonToColumns

Python 类名：JsonToColumns

功能介绍

将数据格式从 Json 转成 Columns

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
```

```
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = JsonToColumns()\
    .setJsonCol("json")\
    .setReservedCols(["row"])\
    .setSchemaStr("f0 double, f1 double")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.JsonToColumns;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonToColumnsTest {
    @Test
    public void testJsonToColumns() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new JsonToColumns()
            .setJsonCol("json")
            .setReservedCols("row")
            .setSchemaStr("f0 double, f1 double")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	f0	f1
1	1.0	2.0
2	4.0	8.0

JSON转CSV (JsonToCsv)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.JsonToCsv

Python 类名：JsonToCsv

功能介绍

将数据格式从 Json 转成 Csv

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
jsonCol	JSON列名	JSON列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = JsonToCsv()\
    .setJsonCol("json")\
    .setReservedCols(["row"])\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.JsonToCsv;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonToCsvTest {
    @Test
    public void testJsonToCsv() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
```

```
double, f1 double");
        BatchOperator op = new JsonToCsv()
            .setJsonCol("json")
            .setReservedCols("row")
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	csv
1	1.0,2.0
2	4.0,8.0

JSON转KV (JsonToKv)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.JsonToKv

Python 类名：JsonToKv

功能介绍

将数据格式从 Json 转成 Kv

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓		
kvCol	KV列名	KV列的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = JsonToKv()\
    .setJsonCol("json")\
    .setReservedCols(["row"])\
    .setKvCol("kv")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.JsonToKv;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonToKvTest {
    @Test
    public void testJsonToKv() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new JsonToKv()
            .setJsonCol("json")
            .setReservedCols("row")
            .setKvCol("kv")
            .transform(data);
```

```
        op.print();
    }
}
```

运行结果

row	kv
1	f1:1.0,f2:2.0
2	f2:4.0,f4:8.0

JSON转向量 (JsonToVector)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.JsonToVector

Python 类名：JsonToVector

功能介绍

将数据格式从 Json 转成 Vector

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null
vectorSize	向量长度	向量长度	Long			-1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"0":"1.0","1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
```

```
    ['2', '{"0":"4.0","1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = JsonToVector()\
    .setJsonCol("json")\
    .setReservedCols(["row"])\
    .setVectorCol("vec")\
    .setVectorSize(5)\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.JsonToVector;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class JsonToVectorTest {
    @Test
    public void testJsonToVector() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"0\":\"1.0\",\"1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new JsonToVector()
            .setJsonCol("json")
            .setReservedCols("row")
            .setVectorCol("vec")
            .setVectorSize(5)
            .transform(data);
        op.print();
    }
}
```

运行结果

row	vec
1	$5$1.0 2.0
2	$5$4.0 8.0

KV转列数据 (KvToColumns)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.KvToColumns

Python 类名：KvToColumns

功能介绍

将数据格式从 Kv 转成 Columns

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
kvCol	KV列名	KV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = KvToColumns()\
    .setKvCol("kv")\
    .setReservedCols(["row"])\
    .setSchemaStr("f0 double, f1 double")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.KvToColumns;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KvToColumnsTest {
    @Test
    public void testKvToColumns() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new KvToColumns()
            .setKvCol("kv")
```

```
            .setReservedCols("row")
            .setSchemaStr("f0 double, f1 double")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	f0	f1
1	1.0	2.0
2	4.0	8.0

KV转CSV (KvToCsv)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.KvToCsv

Python 类名：KvToCsv

功能介绍

将数据格式从 Kv 转成 Csv

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
kvCol	KV列名	KV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"

quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = KvToCsv()\
    .setKvCol("kv")\
    .setReservedCols(["row"])\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.KvToCsv;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class KvToCsvTest {
    @Test
    public void testKvToCsv() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new KvToCsv()
            .setKvCol("kv")
            .setReservedCols("row")
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	csv
1	1.0,2.0
2	4.0,8.0

KV转JSON (KvToJson)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.KvToJson

Python 类名：KvToJson

功能介绍

将数据格式从 Kv 转成 Json

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓		
kvCol	KV列名	KV列的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', 'f0:1.0,f1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', 'f0:4.0,f1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = KvToJson()\
    .setKvCol("kv")\
    .setReservedCols(["row"])\
    .setJsonCol("json")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.KvToJson;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KvToJsonTest {
    @Test
    public void testKvToJson() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"f0:1.0,f1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new KvToJson()
            .setKvCol("kv")
            .setReservedCols("row")
            .setJsonCol("json")
            .transform(data);
```

```
        op.print();
    }
}
```

运行结果

row	json
1	{"f0":"1.0","f1":"2.0"}
2	{"f0":"4.0","f1":"8.0"}

KV转向量 (KvToVector)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.KvToVector

Python 类名：KvToVector

功能介绍

将数据格式从 Kv 转成 Vector

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
kvCol	KV列名	KV列的列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null
vectorSize	向量长度	向量长度	Long			-1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', '0:1.0,1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', '0:4.0,1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = KvToVector()\
    .setKvCol("kv")\
    .setReservedCols(["row"])\
    .setVectorCol("vec")\
    .setVectorSize(5)\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.KvToVector;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KvToVectorTest {
    @Test
    public void testKvToVector() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"0:1.0,1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new KvToVector()
```

```
            .setKvCol("kv")
            .setReservedCols("row")
            .setVectorCol("vec")
            .setVectorSize(5)
            .transform(data);
        op.print();
    }
}
```

运行结果

row	vec
1	$5$1.0 2.0
2	$5$4.0 8.0

向量转列数据 (VectorToColumns)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.VectorToColumns

Python 类名：VectorToColumns

功能介绍

将数据格式从 Vector 转成 Columns

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', '0:1.0,1:2.0',
```

```
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', '0:4.0,1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = VectorToColumns()\
    .setVectorCol("vec")\
    .setReservedCols(["row"])\
    .setSchemaStr("f0 double, f1 double")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.VectorToColumns;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorToColumnsTest {
    @Test
    public void testVectorToColumns() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"0:1.0,1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new VectorToColumns()
            .setVectorCol("vec")
            .setReservedCols("row")
            .setSchemaStr("f0 double, f1 double")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	f0	f1
1	1.0	2.0
2	4.0	8.0

向量转CSV (VectorToCsv)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.VectorToCsv

Python 类名：VectorToCsv

功能介绍

将数据格式从 Vector 转成 Csv

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
csvCol	CSV列名	CSV列的列名	String	✓		
schemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]", 例如"f0 string, f1 bigint, f2 double"	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓		
csvFieldDelimiter	字段分隔符	字段分隔符	String			","
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
quoteChar	引号字符	引号字符	Character			"\""
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', '0:1.0,1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', '0:4.0,1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = VectorToCsv()\
    .setVectorCol("vec")\
    .setReservedCols(["row"])\
    .setCsvCol("csv")\
    .setSchemaStr("f0 double, f1 double")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.VectorToCsv;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorToCsvTest {
    @Test
    public void testVectorToCsv() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"0:1.0,1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
```

```
double, f1 double");
        BatchOperator op = new VectorToCsv()
            .setVectorCol("vec")
            .setReservedCols("row")
            .setCsvCol("csv")
            .setSchemaStr("f0 double, f1 double")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	csv
1	1.0,2.0
2	4.0,8.0

向量转JSON (VectorToJson)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.VectorToJson

Python 类名：VectorToJson

功能介绍

将数据格式从 Vector 转成 Json

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
jsonCol	JSON列名	JSON列的列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', '0:1.0,1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', '0:4.0,1:8.0',
'4.0,8.0', 4.0, 8.0]])
```

```
data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = VectorToJson()\
    .setVectorCol("vec")\
    .setReservedCols(["row"])\
    .setJsonCol("json")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.VectorToJson;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorToJsonTest {
    @Test
    public void testVectorToJson() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"0:1.0,1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new VectorToJson()
            .setVectorCol("vec")
            .setReservedCols("row")
            .setJsonCol("json")
            .transform(data);
        op.print();
    }
}
```

运行结果

row	json
1	{"1":"1.0","2":"2.0"}

2	{"2":"4.0","2":"8.0"}

向量转KV (VectorToKv)

Java 类名：com.alibaba.alink.pipeline.dataproc.format.VectorToKv

Python 类名：VectorToKv

功能介绍

将数据格式从 Vector 转成 Kv

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
kvCol	KV列名	KV列的列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓		
handleInvalid	解析异常处理策略	解析异常处理策略，可选为ERROR（抛出异常）或者SKIP（输出NULL）	String		"ERROR", "SKIP"	"ERROR"
kvColDelimiter	分隔符	当输入数据为稀疏格式时，key-value对之间的分隔符	String			","
kvValDelimiter	分隔符	当输入数据为稀疏格式时，key和value的分割符	String			":"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ['1', '{"f0":"1.0","f1":"2.0"}', '$3$0:1.0 1:2.0', '0:1.0,1:2.0',
'1.0,2.0', 1.0, 2.0],
    ['2', '{"f0":"4.0","f1":"8.0"}', '$3$0:4.0 1:8.0', '0:4.0,1:8.0',
'4.0,8.0', 4.0, 8.0]])

data = BatchOperator.fromDataframe(df, schemaStr="row string, json string, vec
string, kv string, csv string, f0 double, f1 double")

op = VectorToKv()\
    .setVectorCol("vec")\
    .setReservedCols(["row"])\
    .setKvCol("kv")\
    .transform(data)

op.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.format.VectorToKv;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorToKvTest {
    @Test
    public void testVectorToKv() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1", "{\"f0\":\"1.0\",\"f1\":\"2.0\"}", "$3$0:1.0 1:2.0",
"0:1.0,1:2.0", "1.0,2.0", 1.0, 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "row string, json string, vec string, kv string, csv string, f0
double, f1 double");
        BatchOperator op = new VectorToKv()
            .setVectorCol("vec")
            .setReservedCols("row")
            .setKvCol("kv")
            .transform(data);
```

```
        op.print();
    }
}
```

运行结果

row	kv
1	1:1.0,2:2.0
2	1:4.0,2:8.0

向量聚合 (VectorAssembler)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorAssembler

Python 类名：VectorAssembler

功能介绍

数据结构转换，将多列数据（可以是向量列也可以是数值列）转化为一列向量数据。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
```

```
    ["0", "$6$1:2.0 2:3.0 5:4.3", "3.0 2.0 3.0"],
    ["1", "$8$1:2.0 2:3.0 7:4.3", "3.0 2.0 3.0"],
    ["2", "$8$1:2.0 2:3.0 7:4.3", "2.0 3.0"]
])
data = BatchOperator.fromDataframe(df, schemaStr="id string, c0 string, c1
string")

res = VectorAssembler()\
            .setSelectedCols(["c0", "c1"])\
            .setOutputCol("table2vec")
res.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorAssembler;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorAssemblerTest {
    @Test
    public void testVectorAssembler() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("0", "$6$1:2.0 2:3.0 5:4.3", "3.0 2.0 3.0"),
            Row.of("1", "$8$1:2.0 2:3.0 7:4.3", "3.0 2.0 3.0"),
            Row.of("2", "$8$1:2.0 2:3.0 7:4.3", "2.0 3.0")
        );
        MemSourceBatchOp data = new MemSourceBatchOp(df, "id string, c0 string,
c1 string");
        VectorAssembler res = new VectorAssembler()
            .setSelectedCols("c0", "c1")
            .setOutputCol("table2vec");
        res.transform(data).print();
    }
}
```

运行结果

id	c0	c1	table2vec
0	$6$1:2.0 2:3.0 5:4.3	3.0 2.0 3.0	$9$1:2.0 2:3.0 5:4.3 6:3.0 7:2.0 8:3.0
1	$8$1:2.0 2:3.0 7:4.3	3.0 2.0 3.0	$11$1:2.0 2:3.0 7:4.3 8:3.0 9:2.0 10:3.0

2	$8$1:2.0 2:3.0 7:4.3	2.0 3.0	$10$1:2.0 2:3.0 7:4.3 8:2.0 9:3.0

二元向量函数 (VectorBiFunction)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorBiFunction

Python 类名：VectorBiFunction

功能介绍

- 对两个向量进行操作的函数，支持minus(减),plus(加),dot(内积),merge(拼接),EuclidDistance(欧式距离),Cosine(cos值), ElementWiseMultiply(点乘)。
- 支持稀疏和稠密两种 Vector。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
biFuncName	函数名字	函数操作名称, 可取minus(减),plus(加),dot(内积),merge(拼接),EuclidDistance(欧式距离),Cosine(cos值), ElementWiseMultiply(点乘).	String	✓	"Minus", "Dot", "Plus", "Merge", "EuclidDistance", "Cosine", "ElementWiseMultiply"	
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		

reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1 2 3", "2 3 4"]
])
data = BatchOperator.fromDataframe(df, schemaStr="vec1 string, vec2 string")
VectorBiFunction() \
        .setSelectedCols(["vec1", "vec2"]) \

.setBiFuncName("minus").setOutputCol("vec_minus").transform(data).print();
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorBiFunction;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;
```

```
import java.util.ArrayList;
import java.util.List;

public class ToVectorBiFunctionTest extends AlinkTestBase {
    @Test
    public void testBiVectorFunction() throws Exception {
        List <Row> df = new ArrayList <>();
        df.add(Row.of("1 2 3", "2 3 4"));
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec1 string, vec2
string");
         new VectorBiFunction()
            .setSelectedCols("vec1", "vec2")
            .setBiFuncName("minus")
            .setOutputCol("vec_minus").transform(data).print();
    }
}
```

运行结果

vec1	vec2	vec_minus
1 2 3	2 3 4	-1.0 -1.0 -1.0

向量元素依次相乘 (VectorElementwiseProduct)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorElementwiseProduct

Python 类名：VectorElementwiseProduct

功能介绍

Vector 中的每一个非零元素与scalingVector的每一个对应元素乘，返回乘积后的新vector。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
scalingVector	尺度变化向量。	尺度的变化向量。	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1:3 2:4 4:7", 1],
    ["0:3 5:5", 3],
    ["2:4 4:5", 4]
])
```

```
data = BatchOperator.fromDataframe(df, schemaStr="vec string, id bigint")
vecEP = VectorElementwiseProduct().setSelectedCol("vec") \
    .setOutputCol("vec1") \
    .setScalingVector("$8$1:3.0 3:3.0 5:4.6")
vecEP.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorElementwiseProduct;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorElementwiseProductTest {
    @Test
    public void testVectorElementwiseProduct() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:3 2:4 4:7", 1),
            Row.of("0:3 5:5", 3),
            Row.of("2:4 4:5", 4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string, id
int");
        VectorElementwiseProduct vecEP = new
VectorElementwiseProduct().setSelectedCol("vec")
            .setOutputCol("vec1")
            .setScalingVector("$8$1:3.0 3:3.0 5:4.6");
        vecEP.transform(data).print();
    }
}
```

运行结果

vec	id	vec1
1:3,2:4,4:7	1	1:9.0 2:0.0 4:0.0
0:3,5:5	3	0:0.0 5:23.0
2:4,4:5	4	2:0.0 4:0.0

向量函数 (VectorFunction)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorFunction

Python 类名：VectorFunction

功能介绍

- 获取一个向量的最大值、最小值，或者最大值、最小值的索引，或者对向量做尺度变换, 求NormL2, 求NormL1, 求NormL2Square, Normalize。
- 支持稀疏和稠密两种 Vector。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
funcName	函数名字	函数操作名称, 可取max（最大值）, min（最小值）, argMax（最大值索引）, argMin（最小值索引）, scale（尺度变换）, NormL2, NormL1, NormL2Square, Normalize	String	✓	"Max", "Min", "ArgMax", "ArgMin", "Scale", "NormL2", "NormL1", "NormL2Square", "Normalize"	
selectedCol	选中的列名	计算列对应的列名	String	✓		
WithVariable	变量	函数中变量	String			
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null

reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1,"16.3, 1.1, 1.1"],
    [2,"16.8, 1.4, 1.5"],
    [3,"19.2, 1.7, 1.8"],
    [4,"10.0, 1.7, 1.7"],
    [5,"19.5, 1.8, 1.9"],
    [6,"20.9, 1.8, 1.8"],
    [7,"21.1, 1.9, 1.8"],
    [8,"20.9, 2.0, 2.1"],
    [9,"20.3, 2.3, 2.4"],
    [10,"22.0, 2.4, 2.5"]
])
opData = BatchOperator.fromDataframe(df, schemaStr="id bigint, vec string")
result = VectorFunction().setSelectedCol("vec")\
.setOutputCol("out").setFuncName("max").transform(opData)
result.collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;
import com.alibaba.alink.pipeline.dataproc.vector.VectorFunction;
import com.alibaba.alink.operator.stream.BatchOperator;
import com.alibaba.alink.operator.stream.source.MemSourceBatchOp;
import com.alibaba.alink.testutil.AlinkTestBase;
import org.junit.Test;

import java.util.ArrayList;
import java.util.List;

public class VectorFunctionTest extends AlinkTestBase {

    @Test
    public void testVectorFunction() throws Exception {
        List <Row> df = new ArrayList <>();
        df.add(Row.of(1, "16.3, 1.1, 1.1"));
        df.add(Row.of(2, "16.8, 1.4, 1.5"));
        df.add(Row.of(3, "19.2, 1.7, 1.8"));
        df.add(Row.of(4, "10.0, 1.7, 1.7"));
        df.add(Row.of(5, "19.5, 1.8, 1.9"));
        df.add(Row.of(6, "20.9, 1.8, 1.8"));
        df.add(Row.of(7, "21.1, 1.9, 1.8"));
        df.add(Row.of(8, "20.9, 2.0, 2.1"));
        df.add(Row.of(9, "20.3, 2.3, 2.4"));
        df.add(Row.of(10, "22.0, 2.4, 2.5"));

        BatchOperator<?> streamData =  new MemSourceBatchOp(df, "id int, vec
string");

        new VectorFunction().setSelectedCol("vec")

.setOutputCol("out").setFuncName("max").transform(streamData).print();
    }
}
```

运行结果

id	vec	out
1	16.3, 1.1, 1.1	16.3
2	16.8, 1.4, 1.5	16.8
3	19.2, 1.7, 1.8	19.2
4	10.0, 1.7, 1.7	10.0
5	19.5, 1.8, 1.9	19.5

6	20.9, 1.8, 1.8	20.9
7	21.1, 1.9, 1.8	21.1
8	20.9, 2.0, 2.1	20.9
9	20.3, 2.3, 2.4	20.3
10	22.0, 2.4, 2.5	22.0

向量缺失值填充 (VectorImputer)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorImputer

Python 类名：VectorImputer

功能介绍

- 训练Vector 缺失值填充组件的模型，输出模型。
- 填充策略支持最大值，最小值，均值和默认值4种策略，默认为均值。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认
selectedCol	选中的列名	计算列对应的列名	String	✓		
fillValue	填充缺失值	自定义的填充值。当strategy为value时，读取fillValue的值	Double			null
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			fals
strategy	缺失值填充规则	缺失值填充的规则，支持mean，max，min或者value。选择value时，需要读取fillValue的值	String		"MEAN", "MIN", "MAX", "VALUE"	"ME
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1:3,2:4,4:7", 1],
    ["1:3,2:NaN", 3],
    ["2:4,4:5", 4]])
data = BatchOperator.fromDataframe(df, schemaStr="vec string, id bigint")
vecFill = VectorImputer().setSelectedCol("vec").setOutputCol("vec1")
model = vecFill.fit(data)
model.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorImputer;
import com.alibaba.alink.pipeline.dataproc.vector.VectorImputerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorImputerTest {
    @Test
    public void testVectorImputer() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:3,2:4,4:7", 1),
```

```
            Row.of("1:3,2:NaN", 3)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string, id
int");
        VectorImputer vecFill = new
VectorImputer().setSelectedCol("vec").setOutputCol("vec1");
        VectorImputerModel model = vecFill.fit(data);
        model.transform(data).print();
    }
}
```

运行结果

vec	id	vec1
1:3,2:4,4:7	1	1:3.0 2:4.0 4:7.0
1:3,2:NaN	3	1:3.0 2:4.0
2:4,4:5	4	2:4.0 4:5.0

向量缺失值填充模型 (VectorImputerModel)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorImputerModel

Python 类名：VectorImputerModel

功能介绍

- Vector缺失值填充模型
- 模型由VectorImputer生成，用于预处理其他数据

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ["1:3,2:4,4:7", 1],
    ["1:3,2:NaN", 3],
    ["2:4,4:5", 4]])
data = BatchOperator.fromDataframe(df, schemaStr="vec string, id bigint")
vecFill = VectorImputer().setSelectedCol("vec").setOutputCol("vec1")
model = vecFill.fit(data)
model.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorImputer;
import com.alibaba.alink.pipeline.dataproc.vector.VectorImputerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorImputerModelTest {
    @Test
    public void testVectorImputerModel() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:3,2:4,4:7", 1),
            Row.of("1:3,2:NaN", 3)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string, id
int");
        VectorImputer vecFill = new
VectorImputer().setSelectedCol("vec").setOutputCol("vec1");
        VectorImputerModel model = vecFill.fit(data);
        model.transform(data).print();
    }
}
```

运行结果

vec	id	vec1
1:3,2:4,4:7	1	1:3.0 2:4.0 4:7.0

1:3,2:NaN	3	1:3.0 2:4.0
2:4,4:5	4	2:4.0 4:5.0

向量元素两两相乘 (VectorInteraction)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorInteraction

Python 类名：VectorInteraction

功能介绍

对两个vector 中的元素两两相乘，并组成一个新的向量。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

备注：选择列的数目必须为两列

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$8$1:3,2:4,4:7", "$8$1:3,2:4,4:7"],
    ["$8$0:3,5:5", "$8$1:2,2:4,4:7"],
    ["$8$2:4,4:5", "$8$1:3,2:3,4:7"]
])
data = BatchOperator.fromDataframe(df, schemaStr="vec1 string, vec2 string")
vecInter =
```

```
VectorInteraction().setSelectedCols(["vec1","vec2"]).setOutputCol("vec_product"
)
vecInter.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorInteraction;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorInteractionTest {
    @Test
    public void testVectorInteraction() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("$8$1:3,2:4,4:7", "$8$1:3,2:4,4:7"),
            Row.of("$8$0:3,5:5", "$8$1:2,2:4,4:7"),
            Row.of("$8$2:4,4:5", "$8$1:3,2:3,4:7")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec1 string, vec2
string");
        VectorInteraction vecInter = new
VectorInteraction().setSelectedCols("vec1", "vec2").setOutputCol(
            "vec_product");
        vecInter.transform(data).print();
    }
}
```

运行结果

vec1	vec2	vec_product
$8$1:3,2:4,4:7	$8$1:3,2:4,4:7	$64$9:9.0 10:12.0 12:21.0 17:12.0 18:16.0 20:28.0 33:21.0 34:28.0 36:49.0
$8$0:3,5:5	$8$1:2,2:4,4:7	$64$8:6.0 13:10.0 16:12.0 21:20.0 32:21.0 37:35.0
$8$2:4,4:5	$8$1:3,2:3,4:7	$64$10:12.0 12:15.0 18:12.0 20:15.0 34:28.0 36:35.0

向量绝对值最大化 (VectorMaxAbsScaler)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorMaxAbsScaler

Python 类名：VectorMaxAbsScaler

功能介绍

- vector绝对值最大标准化是对vector数据按照最大绝对值进行标准化的组件, 将数据归一到-1和1之间。
- 生成最大绝对值标准化模型VectorMaxAbsScalerModel

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
])
data = BatchOperator.fromDataframe(df, schemaStr="col string, vec string")

res = VectorMaxAbsScaler()\
           .setSelectedCol("vec")
model = res.fit(data)
model.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorMaxAbsScaler;
import com.alibaba.alink.pipeline.dataproc.vector.VectorMaxAbsScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorMaxAbsScalerTest {
    @Test
    public void testVectorMaxAbsScaler() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vec
string");
        VectorMaxAbsScaler res = new VectorMaxAbsScaler()
```

```
            .setSelectedCol("vec");
        VectorMaxAbsScalerModel model = res.fit(data);
        model.transform(data).print();
    }
}
```

运行结果

col1	vec
c	1.0,0.01
b	-0.024777006937561942,0.09
d	-0.9900891972249752,1.0
a	0.09910802775024777,1.0
b	-0.02180376610505451,0.09
c	0.9930624380574826,0.01
a	0.013875123885034686,0.01

向量绝对值最大化模型 (VectorMaxAbsScalerModel)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorMaxAbsScalerModel

Python 类名：VectorMaxAbsScalerModel

功能介绍

- vector绝对值最大标准化是对vector数据按照最大绝对值进行标准化的组件, 将数据归一到-1和1之间。
- 模型由VectorMaxAbsScaler训练产生

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
])
data = BatchOperator.fromDataframe(df, schemaStr="col string, vec string")

res = VectorMaxAbsScaler()\
           .setSelectedCol("vec")
model = res.fit(data)
model.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorMaxAbsScaler;
import com.alibaba.alink.pipeline.dataproc.vector.VectorMaxAbsScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorMaxAbsScalerModelTest {
    @Test
    public void testVectorMaxAbsScalerModel() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vec
string");
        VectorMaxAbsScaler res = new VectorMaxAbsScaler()
```

```
            .setSelectedCol("vec");
        VectorMaxAbsScalerModel model = res.fit(data);
        model.transform(data).print();
    }
}
```

运行结果

col1	vec
c	1.0,0.01
b	-0.024777006937561942,0.09
d	-0.9900891972249752,1.0
a	0.09910802775024777,1.0
b	-0.02180376610505451,0.09
c	0.9930624380574826,0.01
a	0.013875123885034686,0.01

向量归一化 (VectorMinMaxScaler)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorMinMaxScaler

Python 类名：VectorMinMaxScaler

功能介绍

- vector归一化是对vector数据进行归一的组件, 将数据归一到min和max之间。
- 计算公式为x_scaled = (x - eMin) / (eMax - eMin) * (maxV - minV) + minV 其中maxV和minV为用户设定的, 默认值为1和0
- 该组件提供训练功能，生成的Vector归一化模型供预测使用

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
max	归一化的上界	归一化的上界	Double			1.0
min	归一化的下界	归一化的下界	Double			0.0

modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
])
data = BatchOperator.fromDataframe(df, schemaStr="col string, vec string")
res = VectorMinMaxScaler()\
           .setSelectedCol("vec")
model = res.fit(data)
model.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorMinMaxScaler;
import com.alibaba.alink.pipeline.dataproc.vector.VectorMinMaxScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorMinMaxScalerTest {
    @Test
    public void testVectorMinMaxScaler() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vec
string");
        VectorMinMaxScaler res = new VectorMinMaxScaler()
            .setSelectedCol("vec");
        VectorMinMaxScalerModel model = res.fit(data);
        model.transform(data).print();
    }
}
```

运行结果

col1	vec
a	0.5473107569721115,1.0
b	0.4850597609561753,0.08080808080808081
c	0.9965139442231076,0.0
d	0.0,1.0
a	0.5044820717131474,0.0
b	0.4865537848605578,0.08080808080808081

c	1.0,0.0

向量归一化模型 (VectorMinMaxScalerModel)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorMinMaxScalerModel

Python 类名：VectorMinMaxScalerModel

功能介绍

- vector归一化是对vector数据进行归一的组件, 将数据归一到min和max之间。
- 计算公式为x_scaled = (x - eMin) / (eMax - eMin) * (maxV - minV) + minV 其中maxV和minV为用户设定的, 默认值为1和0
- 该组件提供预测功能，对输入的数据进行归一化处理

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
])
data = BatchOperator.fromDataframe(df, schemaStr="col string, vec string")
res = VectorMinMaxScaler()\
           .setSelectedCol("vec")
model = res.fit(data)
model.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorMinMaxScaler;
import com.alibaba.alink.pipeline.dataproc.vector.VectorMinMaxScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorMinMaxScalerModelTest {
    @Test
    public void testVectorMinMaxScalerModel() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vec
string");
        VectorMinMaxScaler res = new VectorMinMaxScaler()
            .setSelectedCol("vec");
```

```
        VectorMinMaxScalerModel model = res.fit(data);
        model.transform(data).print();
    }
}
```

运行结果

col1	vec
a	0.5473107569721115,1.0
b	0.4850597609561753,0.08080808080808081
c	0.9965139442231076,0.0
d	0.0,1.0
a	0.5044820717131474,0.0
b	0.4865537848605578,0.08080808080808081
c	1.0,0.0

向量标准化 (VectorNormalizer)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorNormalizer

Python 类名：VectorNormalizer

功能介绍

对 Vector 进行正则化操作。

指定参数范数的阶，例如p = 2, 对于向量，计算向量的平方和再开二次方记为norm，最终计算结果为

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
p	范数的阶	范数的阶，默认2	Double			2.0
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1:3 2:4 4:7", 1],
    ["0:3 5:5", 3],
    ["2:4 4:5", 4]
])
```

```
data = BatchOperator.fromDataframe(df, schemaStr="vec string, id bigint")
VectorNormalizer().setSelectedCol("vec").setOutputCol("vec_norm").transform(dat
a).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorNormalizer;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorNormalizerTest {
    @Test
    public void testVectorNormalizer() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:3 2:4 4:7", 1),
            Row.of("0:3 5:5", 3),
            Row.of("2:4 4:5", 4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string, id
int");
        new
VectorNormalizer().setSelectedCol("vec").setOutputCol("vec_norm").transform(dat
a).print();
    }
}
```

运行结果

vec	id	vec_norm
1:3,2:4,4:7	1	1:0.34874291623145787 2:0.46499055497527714 4:0.813733471206735
0:3,5:5	3	0:0.5144957554275265 5:0.8574929257125441
2:4,4:5	4	2:0.6246950475544243 4:0.7808688094430304

向量多项式展开 (VectorPolynomialExpand)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorPolynomialExpand

Python 类名：VectorPolynomialExpand

功能介绍

对 Vector 进行多项式展开，组成一个新的Vector。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
degree	多项式阶数	多项式的阶数，默认2	Integer		[1, +inf)	2
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["$8$1:3,2:4,4:7"],
    ["$8$2:4,4:5"]
])
data = BatchOperator.fromDataframe(df, schemaStr="vec string")
```

```
VectorPolynomialExpand().setSelectedCol("vec").setOutputCol("vec_out").transfor
m(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorPolynomialExpand;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorPolynomialExpandTest {
    @Test
    public void testVectorPolynomialExpand() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("$8$1:3,2:4,4:7"),
            Row.of("$8$2:4,4:5")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string");
        new
VectorPolynomialExpand().setSelectedCol("vec").setOutputCol("vec_out").transfor
m(data).print();
    }
}
```

运行结果

vec	vec_out
$8$1:3,2:4,4:7	$44$2:3.0 4:9.0 5:4.0 7:12.0 8:16.0 14:7.0 16:21.0 17:28.0 19:49.0
$8$2:4,4:5	$44$5:4.0 8:16.0 14:5.0 17:20.0 19:25.0

向量长度检验 (VectorSizeHint)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorSizeHint

Python 类名：VectorSizeHint

功能介绍

取出Vector 的size进行检测，并进行处理。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
size	向量大小	用于判断向量的大小是否和设置的一致	Integer	✓		
handleInvalidMethod	处理无效值的方法	处理无效值的方法，可取 error, skip	String		"ERROR", "SKIP"	"ERROR"
outputCol	输出结果列	输出结果列列名，可选，默认 null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ["$8$1:3,2:4,4:7"],
    ["$8$2:4,4:5"]
])
data = BatchOperator.fromDataframe(df, schemaStr="vec string")
model =
VectorSizeHint().setSelectedCol("vec").setOutputCol("vec_hint").setHandleInvali
dMethod("SKIP").setSize(8)
model.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorSizeHint;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorSizeHintTest {
    @Test
    public void testVectorSizeHint() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("$8$1:3,2:4,4:7"),
            Row.of("$8$2:4,4:5")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string");
        VectorSizeHint model = new
VectorSizeHint().setSelectedCol("vec").setOutputCol("vec_hint")
            .setHandleInvalidMethod("SKIP").setSize(8);
        model.transform(data).print();
    }
}
```

运行结果

vec	vec_hint
$8$1:3,2:4,4:7	$8$1:3.0 2:4.0 4:7.0
$8$2:4,4:5	$8$2:4.0 4:5.0

向量切片 (VectorSlicer)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorSlicer

Python 类名：VectorSlicer

功能介绍

取出Vector 中的若干列，组成一个新的Vector。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
indices	需要被提取的索引数组	需要被提取的索引数组	int[]			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1:3 2:4 4:7", 1],
    ["0:3 5:5", 3],
    ["2:4 4:5", 4]
])
```

```
data = BatchOperator.fromDataframe(df, schemaStr="vec string, id bigint")
vecSlice =
VectorSlicer().setSelectedCol("vec").setOutputCol("vec_slice").setIndices([1,2,
3])
vecSlice.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorSlicer;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorSlicerTest {
    @Test
    public void testVectorSlicer() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:3 2:4 4:7", 1),
            Row.of("0:3 5:5", 3),
            Row.of("2:4 4:5", 4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "vec string, id
int");
        VectorSlicer vecSlice = new
VectorSlicer().setSelectedCol("vec").setOutputCol("vec_slice").setIndices(
            new int[] {1, 2, 3});
        vecSlice.transform(data).print();
    }
}
```

运行结果

vec	id	vec_slice
1:3,2:4,4:7	1	$3$0:3.0 1:4.0
0:3,5:5	3	3
2:4,4:5	4	$3$1:4.0

向量标准化 (VectorStandardScaler)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorStandardScaler

Python 类名：VectorStandardScaler

功能介绍

- Vector标准化是对Vector数据进行按正态化处理的组件
- 该组件生成VectorStandardScalerModel，Vector标准化模型

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

withMean	是否使用均值	是否使用均值，默认使用	Boolean			true
withStd	是否使用标准差	是否使用标准差，默认使用	Boolean			true
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
])
data = BatchOperator.fromDataframe(df, schemaStr="col string, vector string")
model = VectorStandardScaler().setSelectedCol("vector").fit(data)
model.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorStandardScaler;
import com.alibaba.alink.pipeline.dataproc.vector.VectorStandardScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorStandardScalerTest {
```

```
    @Test
    public void testVectorStandardScaler() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vector
string");
        VectorStandardScalerModel model = new
VectorStandardScaler().setSelectedCol("vector").fit(data);
        model.transform(data).print();
    }
}
```

运行结果

col1	vec
a	-0.07835182408093559,1.4595814453461897
c	1.2269606224811418,-0.6520885789229323
b	-0.2549018445693762,-0.4814485769617911
a	-0.20280511721213143,-0.6520885789229323
c	1.237090541689495,-0.6520885789229323
b	-0.25924323851581327,-0.4814485769617911
d	-1.6687491397923802,1.4595814453461897

向量标准化模型 (VectorStandardScalerModel)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorStandardScalerModel

Python 类名：VectorStandardScalerModel

功能介绍

- Vector标准化是对Vector数据进行按正态化处理的组件
- 该组件为Vector标准化模型，可用于对数据做标准化处理

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ["a", "10.0, 100"],
    ["b", "-2.5, 9"],
    ["c", "100.2, 1"],
    ["d", "-99.9, 100"],
    ["a", "1.4, 1"],
    ["b", "-2.2, 9"],
    ["c", "100.9, 1"]
])
data = BatchOperator.fromDataframe(df, schemaStr="col string, vector string")
model = VectorStandardScaler().setSelectedCol("vector").fit(data)
model.transform(data).collectToDataframe()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.dataproc.vector.VectorStandardScaler;
import com.alibaba.alink.pipeline.dataproc.vector.VectorStandardScalerModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorStandardScalerModelTest {
    @Test
    public void testVectorStandardScalerModel() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", "10.0, 100"),
            Row.of("b", "-2.5, 9"),
            Row.of("c", "100.2, 1"),
            Row.of("d", "-99.9, 100"),
            Row.of("a", "1.4, 1"),
            Row.of("b", "-2.2, 9"),
            Row.of("c", "100.9, 1")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "col string, vector
string");
        VectorStandardScalerModel model = new
VectorStandardScaler().setSelectedCol("vector").fit(data);
        model.transform(data).print();
```

```
    }
}
```

运行结果

col1	vec
a	-0.07835182408093559,1.4595814453461897
c	1.2269606224811418,-0.6520885789229323
b	-0.2549018445693762,-0.4814485769617911
a	-0.20280511721213143,-0.6520885789229323
c	1.237090541689495,-0.6520885789229323
b	-0.25924323851581327,-0.4814485769617911
d	-1.6687491397923802,1.4595814453461897

VectorToColumnsLegacy (VectorToColumnsLegacy)

Java 类名：com.alibaba.alink.pipeline.dataproc.vector.VectorToColumnsLegacy

Python 类名：VectorToColumnsLegacy

功能介绍

将向量转为表，向量的每一维数据都转为表的列。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCols	输出结果列列名数组	输出结果列列名数组，必选	String[]	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

data = np.array([["a", "10.0, 100"],\
    ["b", "-2.5, 9"],\
    ["c", "100.2, 1"],\
    ["d", "-99.9, 100"],\
    ["a", "1.4, 1"],\
    ["b", "-2.2, 9"],\
    ["c", "100.9, 1"]])
df = pd.DataFrame({"col" : data[:,0], "vec" : data[:,1]})
data = dataframeToOperator(df, schemaStr="col string, vec
string",op_type="batch")
VectorToColumnsLegacy().setSelectedCol("vec").setOutputCols(["f0",
"f1"]).transform(data).collectToDataframe()
```

结果

	col	vec	f0	f1
0	a	10.0, 100	10.0	100.0
1	b	-2.5, 9	-2.5	9.0
2	c	100.2, 1	100.2	1.0
3	d	-99.9, 100	-99.9	100.0
4	a	1.4, 1	1.4	1.0
5	b	-2.2, 9	-2.2	9.0
6	c	100.9, 1	100.9	1.0

SQL操作：Select (Select)

Java 类名：com.alibaba.alink.pipeline.sql.Select

Python 类名：Select

功能介绍

提供 SQL 的 SELECT 语句功能。

使用方式

该组件提供与 SelectBatch/StreamOp 相近的功能。

该组件既可以单独使用（直接调用 `transform` 方法），也可以置于构建 `Pipeline` 中使用。 前一种场景的使用方式跟 SelectBatch/StreamOp 一致。

后一种场景时，有两点需要特别注意：

1. 语句需要严格输入 1 行输出 1 行，即满足 `map` 的功能；
2. 因为内部实现实际采用的是 Calcite 而非 Flink，因此可能不支持某些 Flink 内置方法；经过测试可以使用的内置方法可以见测试代码 。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
clause	运算语句	运算语句	String	✓		

代码示例

Python 代码

```
URL = "https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/iris.csv"
SCHEMA_STR = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string";
data = CsvSourceBatchOp().setFilePath(URL).setSchemaStr(SCHEMA_STR)
select = Select().setClause("category as label")
select.transform(data).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.sql.Select;
```

```
import org.junit.Test;

public class SelectTest {
    @Test
    public void testSelect() throws Exception {
        String URL = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv";
        String SCHEMA_STR
            = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string";
        BatchOperator <?> data = new
CsvSourceBatchOp().setFilePath(URL).setSchemaStr(SCHEMA_STR);
        Select select = new Select().setClause("category as label");
        select.transform(data).print();
    }
}
```

运行结果

label
Iris-versicolor
Iris-setosa
Iris-setosa
Iris-setosa
Iris-virginica
...
Iris-setosa
Iris-versicolor
Iris-virginica
Iris-setosa
Iris-versicolor

二值化 (Binarizer)

Java 类名：com.alibaba.alink.pipeline.feature.Binarizer

Python 类名：Binarizer

功能介绍

给定一个阈值，将连续变量二值化（大于等于阈值转为1，小于阈值转为0）。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
threshold	二值化阈值	二值化阈值	Double			0.0
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.1, True, "2", "A"],
    [1.1, False, "2", "B"],
    [1.1, True, "1", "B"],
    [2.2, True, "1", "A"]
])
```

```
inOp = BatchOperator.fromDataframe(df, schemaStr='double double, bool boolean,
number int, str string')
binarizer = Binarizer().setSelectedCol("double").setThreshold(2.0)
binarizer.transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.feature.Binarizer;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class BinarizerTest {
    @Test
    public void testBinarizer() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.1, true, 2, "A"),
            Row.of(1.1, false, 2, "B"),
            Row.of(1.1, true, 1, "B"),
            Row.of(2.2, true, 1, "A")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "double double, bool
boolean, number int, str string");
        Binarizer binarizer = new
Binarizer().setSelectedCol("double").setThreshold(2.0);
        binarizer.transform(inOp).print();
    }
}
```

运行结果

double	bool	number	str
0.0000	true	2	A
0.0000	false	2	B
0.0000	true	1	B
1.0000	true	1	A

分箱 (Binning)

Java 类名：com.alibaba.alink.pipeline.feature.Binning

Python 类名：Binning

功能介绍

使用等频，等宽或自动分箱的方法对数据进行分箱操作，输入为连续或离散的特征数据，输出为每个特征的分箱规则，在分箱组件中点键右键选择**我要分箱**，可以对分箱结果进行查看和编辑。

算法简介

信用评分卡模型在国外是一种成熟的预测方法，尤其在信用风险评估以及金融风险控制领域更是得到了比较广泛的使用，其原理是将模型变量WOE编码方式离散化之后运用logistic回归模型进行建模，其中本文介绍的分箱算法就是这里说的WOE编码过程。

分箱通常是指对连续变量进行区间划分，将连续变量划分成几个区间变量，主要目的是为了避免“过拟合”，使评分结果更具有稳健性和预测性。这里有个典型的例子就是：用决策树进行评分看起来效果不错，但往往都会因为”过拟合”，模型无法实际应用。

分箱支持等频、等宽两种方法，输入为连续或离散的特征数据，输出为每个特征的分箱规则，分箱结果支持合并、拆分等。

算法原理

分箱方法

等频分箱

等频分箱是对特征数据进行排序，按分位点的方式选取用户指定的N个分位点作为分箱边界，若相邻分位点相同则将两个分箱合并，因此分箱结果中有可能少于用户指定的分箱个数。

等宽分箱

等宽分箱是对特征数据按最大值和最小值等平均分成N份，在每一等份的边界作为分箱的边界。

离散变量分箱

字符串类型变量即为离散变量，离散变量的分箱不使用上述的三种分箱方法，对于离散变量的每一种取值会单独分成一个分箱，仅当某个取值小于用户指定的最小阈值时，该取值会被统一归入ELSE分箱中。

相关公式

WOE的计算公式如下:

$$ WOE_i=ln(\dfrac{py_i}{pn_i}) $$

其中，py_i是这个组中正样本个数占所有正样本个数的比例，pn_i是这个组中负样本个数占所有负样本个数的比例。

IV的计算公式如下：

$$ IV_i=(py_i-pn_i)*WOE_i $$

实际使用时，一般用IV评估变量的重要性，IV越大，变量对模型的影响越大，在单个变量内部，再利用WOE判断每个bin的区分度如何，WOE越大，区分度越大。

分箱个数针对连续变量而言，支持三种类型输入:

```
i. 输入数值如5，这个数值会对每一列起作用，每列数值型都会被分成5组。

ii. 输入数组如5,6,7; 这个数组与选择列中数值列的长度必须保持一致，这种输入与第一种/第三种输入是互斥的。

iii. 输入key-value字符串如age:11,pay:7,bill:11，其中key和value用":"分割，不同特征用","分隔，这种输入与第一种输入一同起作用，用key-value定义的变量会采用key-value中的值，其余变量采用第一种输入中的数值。
```

离散值个数阈值针对离散变量而言，指数据中出现次数低于这个值的组会被分到ELSE箱子中，同 **分箱个数** 的输入方法一致。

参数说明

名称	中文名称	描述	类型	是否必须?	取
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	
binningMethod	连续特征分箱方法	连续特征分箱方法	String		"QUANTILE
defaultWoe	默认Woe，在woe为Nan或NULL时替换	默认Woe，在woe为Nan或NULL时替换	Double		
discreteThresholds	离散个数阈值	离散个数阈值，低于该阈值的离散样本将不会单独成一个组别。	Integer		
discreteThresholdsArray	离散个数阈值	离散个数阈值，每一列对应数组中一个元素。	Integer[]		

discreteThresholdsMap	离散分箱离散为ELSE的最小阈值, 形式如col0:3, col1:4	离散分箱离散为ELSE的最小阈值，形式如col0:3, col1:4。	String		
dropLast	是否删除最后一个元素	删除最后一个元素是为了保证线性无关性。默认true	Boolean		
encode	编码方法	编码方法	String		"WOE", "VE "ASSEMBL "INDEX"
fromUserDefined	是否读取用户自定义JSON	是否读取用户自定义JSON, true则为用户自定义分箱，false则按参数配置分箱。	Boolean		
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null， "error"表示抛异常	String		"KEEP", "E
labelCol	标签列名	输入表中的标签列名	String		
leftOpen	是否左开右闭	左开右闭为true，左闭右开为false	Boolean		
modelFilePath	模型的文件路径	模型的文件路径	String		
numBuckets	quantile个数	quantile个数，对所有列有效。	Integer		
numBucketsArray	quantile个数	quantile个数，每一列对应数组中一个元素。	Integer[]		
numBucketsMap	用户定义的bucket个数, 形式如col0:3, col1:4	用户定义的bucket个数, 形式如col0:3, col1:4	String		

outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
positiveLabelValueString	正样本	正样本对应的字符串格式。	String		
reservedCols	算法保留列名	算法保留列	String[]		
userDefinedBin	用户定义的bin的json	用户定义的bin的json	String		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    [1.0, True, 0, "A", 1],
    [2.1, False, 2, "B", 1],
    [1.1, True, 3, "C", 1],
    [2.2, True, 1, "E", 0],
    [0.1, True, 2, "A", 0],
    [1.5, False, -4, "D", 1],
    [1.3, True, 1, "B", 0],
    [0.2, True, -1, "A", 1],
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='f0 double, f1 boolean, f2
int, f3 string, label int')

binning = Binning().setEncode("INDEX").setSelectedCols(["f0", "f1", "f2",
"f3"]).setLabelCol("label").setPositiveLabelValueString("1")
binning.fit(inOp1).transform(inOp1).print()
```

运行结果

```
   f0  f1  f2  f3  label
0   0   1   0   3      1
1   1   0   1   0      1
2   0   1   1   1      1
3   1   1   0   2      0
4   0   1   1   3      0
5   1   0   0   4      1
6   0   1   0   0      0
7   0   1   0   3      1
```

分桶 (Bucketizer)

Java 类名：com.alibaba.alink.pipeline.feature.Bucketizer

Python 类名：Bucketizer

功能介绍

给定切分点，将连续变量分桶，需要选择需要进行切分的单列或多列，同时给出选中每列的切分点，每列切分点都是一个double数组，需要严格递增。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
cutsArray	多列的切分点	多列的切分点	double[][]			
dropLast	是否删除最后一个元素	删除最后一个元素是为了保证线性无关性。默认true	Boolean			true
encode	编码方法	编码方法	String		"VECTOR", "ASSEMBLED_VECTOR", "INDEX"	"INDEX"
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null, "error"表示抛异常	String		"KEEP", "ERROR", "SKIP"	"KEEP"

leftOpen	是否左开右闭	左开右闭为true，左闭右开为false	Boolean			true
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.1, True, "2", "A"],
    [1.1, False, "2", "B"],
    [1.1, True, "1", "B"],
    [2.2, True, "1", "A"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='double double, bool boolean,
number int, str string')
bucketizer = Bucketizer().setSelectedCols(["double"]).setCutsArray([[2.0]])
bucketizer.transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;
```

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.feature.Bucketizer;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class BucketizerTest {
    @Test
    public void testBucketizer() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.1, true, 2, "A"),
            Row.of(1.1, false, 2, "B"),
            Row.of(1.1, true, 1, "B"),
            Row.of(2.2, true, 1, "A")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "double double, bool
boolean, number int, str string");
        double[] cutsArray = {2.0};
        Bucketizer bucketizer = new
Bucketizer().setSelectedCols("double").setCutsArray(cutsArray);
        bucketizer.transform(inOp).print();
    }
}
```

运行结果

double	bool	number	str
0	true	2	A
0	false	2	B
0	true	1	B
1	true	1	A

C45编码 (C45Encoder)

Java 类名：com.alibaba.alink.pipeline.feature.C45Encoder

Python 类名：C45Encoder

功能介绍

使用C45模型，对数据进行特征转换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
```

```
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(df, schemaStr=' f0 double, f1 string,
f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(df, schemaStr=' f0 double, f1
string, f2 int, f3 int, label int')

c45EncoderModel = C45Encoder()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setPredictionCol("encoded_features")\
    .fit(batchSource)

c45EncoderModel.transform(batchSource).print()
c45EncoderModel.transform(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.C45Encoder;
import com.alibaba.alink.pipeline.feature.C45EncoderModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class C45EncoderTest {
    @Test
    public void testC45Encoder() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
```

```
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, " f0
double, f1 string, f2 int, f3 int, label "
            + "int");
        C45EncoderModel c45EncoderModel = new C45Encoder()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setPredictionCol("encoded_features")
            .fit(batchSource);
        c45EncoderModel.transform(batchSource).print();
        c45EncoderModel.transform(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$2$0:1.0
2.0000	B	1	1	0	$2$0:1.0
3.0000	C	2	2	1	$2$1:1.0
4.0000	D	3	3	1	$2$1:1.0

Cart编码 (CartEncoder)

Java 类名：com.alibaba.alink.pipeline.feature.CartEncoder

Python 类名：CartEncoder

功能介绍

使用Cart模型，对数据进行特征转换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
```

```
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(df, schemaStr=' f0 double, f1 string,
f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(df, schemaStr=' f0 double, f1
string, f2 int, f3 int, label int')

cartEncoderModel = CartEncoder()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setPredictionCol("encoded_features")\
    .fit(batchSource)

cartEncoderModel.transform(batchSource).print()
cartEncoderModel.transform(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.CartEncoder;
import com.alibaba.alink.pipeline.feature.CartEncoderModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CartEncoderTest {
    @Test
    public void testCartEncoder() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
```

```
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, " f0
double, f1 string, f2 int, f3 int, label "
            + "int");
        CartEncoderModel cartEncoderModel = new CartEncoder()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setPredictionCol("encoded_features")
            .fit(batchSource);
        cartEncoderModel.transform(batchSource).print();
        cartEncoderModel.transform(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$2$0:1.0
2.0000	B	1	1	0	$2$0:1.0
3.0000	C	2	2	1	$2$1:1.0
4.0000	D	3	3	1	$2$1:1.0

Cart回归编码 (CartRegEncoder)

Java 类名：com.alibaba.alink.pipeline.feature.CartRegEncoder

Python 类名：CartRegEncoder

功能介绍

使用Cart回归模型，对数据进行特征转换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
```

```
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(df, schemaStr=' f0 double, f1 string,
f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(df, schemaStr=' f0 double, f1
string, f2 int, f3 int, label int')

cartRegEncoderModel = CartRegEncoder()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setPredictionCol("encoded_features")\
    .fit(batchSource)

cartRegEncoderModel.transform(batchSource).print()
cartRegEncoderModel.transform(streamSource).print()

StreamOperator.execute()
```

Java代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.CartRegEncoder;
import com.alibaba.alink.pipeline.feature.CartRegEncoderModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CartRegEncoderTest {
    @Test
    public void testCartRegEncoder() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
```

```
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, " f0
double, f1 string, f2 int, f3 int, label "
            + "int");
        CartRegEncoderModel cartRegEncoderModel = new CartRegEncoder()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setPredictionCol("encoded_features")
            .fit(batchSource);

        cartRegEncoderModel.transform(batchSource).print();
        cartRegEncoderModel.transform(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$2$0:1.0
2.0000	B	1	1	0	$2$0:1.0
3.0000	C	2	2	1	$2$1:1.0
4.0000	D	3	3	1	$2$1:1.0

离散余弦变换 (DCT)

Java 类名：com.alibaba.alink.pipeline.feature.DCT

Python 类名：DCT

功能介绍

对数据进行离散余弦变换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
inverse	是否为逆变换	是否为逆变换，false表示正变换，true表示逆变换。默认正变换。	Boolean			false
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    ["-0.6264538 0.1836433"],
```

```
    ["-0.8356286 1.5952808"],
    ["0.3295078 -0.8204684"],
    ["0.4874291 0.7383247"],
    ["0.5757814 -0.3053884"],
    ["1.5117812 0.3898432"],
    ["-0.6212406 -2.2146999"],
    ["11.1249309 9.9550664"],
    ["9.9838097 10.9438362"],
    ["10.8212212 10.5939013"],
    ["10.9189774 10.7821363"],
    ["10.0745650 8.0106483"],
    ["10.6198257 9.9438713"],
    ["9.8442045 8.5292476"],
    ["9.5218499 10.4179416"],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='features string')

dct = DCT().setSelectedCol("features").setOutputCol("result")

dct.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.feature.DCT;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DCTTest {
    @Test
    public void testDCT() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("-0.6264538 0.1836433"),
            Row.of("-0.8356286 1.5952808"),
            Row.of("0.3295078 -0.8204684"),
            Row.of("0.4874291 0.7383247"),
            Row.of("0.5757814 -0.3053884"),
            Row.of("1.5117812 0.3898432"),
            Row.of("-0.6212406 -2.2146999"),
            Row.of("11.1249309 9.9550664"),
            Row.of("9.9838097 10.9438362"),
```

```
            Row.of("10.8212212 10.5939013"),
            Row.of("10.9189774 10.7821363"),
            Row.of("10.0745650 8.0106483"),
            Row.of("10.6198257 9.9438713"),
            Row.of("9.8442045 8.5292476"),
            Row.of("9.5218499 10.4179416")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "features
string");
        DCT dct = new DCT().setSelectedCol("features").setOutputCol("result");
        dct.transform(data).print();
    }
}
```

运行结果

features	result
-0.6264538 0.1836433	-0.31311430733060563 -0.5728251528295567
-0.8356286 1.5952808	0.5371552219632794 -1.7189125211901217
0.3295078 -0.8204684	-0.34716156955541605 0.8131559692231375
0.4874291 0.7383247	0.866738824045179 -0.17740998012986753
0.5757814 -0.3053884	0.19119672388537412 0.6230811409567939
1.5117812 0.3898432	1.3446515085097996 0.7933299678708727
-0.6212406 -2.2146999	-2.005312758591568 1.126745876574769
11.1249309 9.9550664	14.905809038224113 0.8272191210194105
9.9838097 10.9438362	14.798080330160849 -0.6788412482687869
10.8212212 10.5939013	15.142778339690611 0.1607394427886475
10.9189774 10.7821363	15.345004656570287 0.09676126975502636
10.0745650 8.0106483	12.788176963635138 1.4594094943741613
10.6198257 9.9438713	14.54072959496546 0.4779719400128842
9.8442045 8.5292476	12.991992573716212 0.9298149409580412
9.5218499 10.4179416	14.099561785095878 -0.6336325176349823

决策树编码 (DecisionTreeEncoder)

Java 类名：com.alibaba.alink.pipeline.feature.DecisionTreeEncoder

Python 类名：DecisionTreeEncoder

功能介绍

使用决策树模型，对数据进行特征转换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

treeType	模型中树的类型	模型中树的类型，三种选项可选：树为一种方式 gini, infoGain, infoGainRatio	String		"GINI", "INFOGAIN", "INFOGAINRAT
weightCol	权重列名	权重列对应的列名	String		所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLI FLOAT, INTEGI LONG, SHORT
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(df, schemaStr=' f0 double, f1 string,
f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(df, schemaStr=' f0 double, f1
string, f2 int, f3 int, label int')

decisionTreeEncoderModel = DecisionTreeEncoder()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setPredictionCol("encoded_features")\
    .fit(batchSource)

decisionTreeEncoderModel.transform(batchSource).print()
decisionTreeEncoderModel.transform(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.DecisionTreeEncoder;
import com.alibaba.alink.pipeline.feature.DecisionTreeEncoderModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DecisionTreeEncoderTest {
    @Test
    public void testDecisionTreeEncoder() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
```

```
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        DecisionTreeEncoderModel decisionTreeEncoderModel = new
DecisionTreeEncoder()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setPredictionCol("encoded_features")
            .fit(batchSource);
        decisionTreeEncoderModel.transform(batchSource).print();
        decisionTreeEncoderModel.transform(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$2$0:1.0
2.0000	B	1	1	0	$2$0:1.0
3.0000	C	2	2	1	$2$1:1.0
4.0000	D	3	3	1	$2$1:1.0

决策树回归编码 (DecisionTreeRegEncoder)

Java 类名：com.alibaba.alink.pipeline.feature.DecisionTreeRegEncoder

Python 类名：DecisionTreeRegEncoder

功能介绍

使用决策树回归模型，对数据进行特征转换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

treeType	模型中树的类型	模型中树的类型，三种选项可选：树为一种方式 gini, infoGain, infoGainRatio	String		"GINI", "INFOGAIN", "INFOGAINRAT
weightCol	权重列名	权重列对应的列名	String		所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLI FLOAT, INTEGI LONG, SHORT
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

decisionTreeRegEncoderModel = DecisionTreeRegEncoder()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setPredictionCol("encoded_features")\
    .fit(batchSource)

decisionTreeRegEncoderModel.transform(batchSource).print()
decisionTreeRegEncoderModel.transform(streamSource).print()

StreamOperator.execute()
```

Java代码

```
package javatest.com.alibaba.alink.pipeline.feature;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.DecisionTreeRegEncoder;
import com.alibaba.alink.pipeline.feature.DecisionTreeRegEncoderModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DecisionTreeRegEncoderTest {
    @Test
    public void testDecisionTreeRegEncoder() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
```

```
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        DecisionTreeRegEncoderModel decisionTreeEncoderModel = new
DecisionTreeRegEncoder()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setPredictionCol("encoded_features")
            .fit(batchSource);
        decisionTreeEncoderModel.transform(batchSource).print();
        decisionTreeEncoderModel.transform(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$2$0:1.0
2.0000	B	1	1	0	$2$0:1.0
3.0000	C	2	2	1	$2$1:1.0
4.0000	D	3	3	1	$2$1:1.0

等宽离散化 (EqualWidthDiscretizer)

Java 类名：com.alibaba.alink.pipeline.feature.EqualWidthDiscretizer

Python 类名：EqualWidthDiscretizer

功能介绍

等宽离散可以计算选定数值列的分位点，每个区间都有相同的组距，也就是数据范围/组数，通过训练可以得到一系列分为点， 然后使用这些分位点进行预测。 其中可以所有列使用同一个分组数量，也可以每一列对应一个分组数量。预测结果可以是特征值或一系列0/1离散特征。

编码结果

Encode ——> INDEX

预测结果为单个token的index

Encode ——> VECTOR

预测结果为稀疏向量:

```
1. dropLast为true,向量中非零元个数为0或者1
2. dropLast为false,向量中非零元个数必定为1
```

Encode ——> ASSEMBLED_VECTOR

预测结果为稀疏向量,是预测选择列中,各列预测为VECTOR时,按照选择顺序ASSEMBLE的结果。

向量维度

Encode ——> Vector

$$ vectorSize = numBuckets - dropLast(true: 1, false: 0) + (handleInvalid: keep(1), skip(0), error(0)) $$

```
numBuckets: 训练参数

dropLast: 预测参数

handleInvalid: 预测参数
```

Token index

Encode ——> Vector

```
1. 正常数据: 唯一的非零元为数据所在的bucket,若 dropLast为true, 最大的bucket的值会被丢掉,
预测结果为全零元
```

```
2. null:
    2.1 handleInvalid为keep: 唯一的非零元为:numBuckets - dropLast(true: 1, false:
0)
    2.2 handleInvalid为skip: null
    2.3 handleInvalid为error: 报错
```

参数说明

名称	中文名称	描述	类型	是否必须?	取值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	
dropLast	是否删除最后一个元素	删除最后一个元素是为了保证线性无关性。默认true	Boolean		
encode	编码方法	编码方法	String		"VECTOR", "ASSEMBL "INDEX"
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null， "error"表示抛异常	String		"KEEP", "EF
leftOpen	是否左开右闭	左开右闭为true，左闭右开为false	Boolean		
modelFilePath	模型的文件路径	模型的文件路径	String		
numBuckets	quantile个数	quantile个数，对所有列有效。	Integer		
numBucketsArray	quantile个数	quantile个数，每一列对应数组中一个元素。	Integer[]		
outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		

reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1, 1.1],
    ["b", -2, 0.9],
    ["c", 100, -0.01],
    ["d", -99, 100.9],
    ["a", 1, 1.1],
    ["b", -2, 0.9],
    ["c", 100, -0.01],
    ["d", -99, 100.9]
])

batchSource =  BatchOperator.fromDataframe(df,schemaStr="f_string string,
f_long long, f_double double")
streamSource = StreamOperator.fromDataframe(df,schemaStr="f_string string,
```

```
f_long long, f_double double")

EqualWidthDiscretizer().setSelectedCols(['f_long',
'f_double']).setNumBuckets(5).fit(batchSource).transform(batchSource).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.feature.EqualWidthDiscretizer;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class EqualWidthDiscretizerTest {
    @Test
    public void testEqualWidthDiscretizer() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1, 1.1),
            Row.of("b", -2, 0.9),
            Row.of("c", 100, -0.01),
            Row.of("d", -99, 100.9),
            Row.of("a", 1, 1.1),
            Row.of("b", -2, 0.9),
            Row.of("c", 100, -0.01),
            Row.of("d", -99, 100.9)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, "f_string
string, f_int int, f_double double");
        new EqualWidthDiscretizer().setSelectedCols("f_int",
"f_double").setNumBuckets(5).fit(batchSource).transform(
            batchSource).print();
    }
}
```

运行结果

f_string	f_int	f_double
a	2	0
b	2	0
c	4	0

d	0	4
a	2	0
b	2	0
c	4	0
d	0	4

特征哈希 (FeatureHasher)

Java 类名：com.alibaba.alink.pipeline.feature.FeatureHasher

Python 类名：FeatureHasher

功能介绍

将多个特征组合成一个特征向量。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
categoricalCols	离散特征列名	离散特征列名	String[]			
numFeatures	向量维度	生成向量长度	Integer			262144
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df1 = pd.DataFrame([
    [1.1, True, "2", "A"],
    [1.1, False, "2", "B"],
    [1.1, True, "1", "B"],
    [2.2, True, "1", "A"]
```

```
])

inOp = BatchOperator.fromDataframe(df1, schemaStr='double double, bool boolean,
number int, str string')
binarizer = FeatureHasher().setSelectedCols(["double", "bool", "number",
"str"]).setOutputCol("output").setNumFeatures(200)
binarizer.transform(inOp).print()

df2 = pd.DataFrame([
    [1.1, True, "2", "A"],
    [1.1, False, "2", "B"],
    [1.1, True, "1", "B"],
    [2.2, True, "1", "A"]
])

inOp1 = BatchOperator.fromDataframe(df2, schemaStr='double double, bool
boolean, number int, str string')
inOp2 = StreamOperator.fromDataframe(df2, schemaStr='double double, bool
boolean, number int, str string')

hasher = FeatureHasherBatchOp().setSelectedCols(["double", "bool", "number",
"str"]).setOutputCol("output").setNumFeatures(200)
hasher.linkFrom(inOp1).print()

hasher = FeatureHasherStreamOp().setSelectedCols(["double", "bool", "number",
"str"]).setOutputCol("output").setNumFeatures(200)
hasher.linkFrom(inOp2).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.feature.FeatureHasherBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.feature.FeatureHasherStreamOp;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.FeatureHasher;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FeatureHasherTest {
```

```
    @Test
    public void testFeatureHasher() throws Exception {
        List <Row> df1 = Arrays.asList(
            Row.of(1.1, true, 2, "A"),
            Row.of(1.1, false, 2, "B"),
            Row.of(1.1, true, 1, "B"),
            Row.of(2.2, true, 1, "A")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df1, "double double, bool
boolean, number int, str string");
        FeatureHasher binarizer = new FeatureHasher().setSelectedCols("double",
"bool", "number", "str").setOutputCol(
            "output").setNumFeatures(200);
        binarizer.transform(inOp).print();
        List <Row> df2 = Arrays.asList(
            Row.of(1.1, true, 2, "A"),
            Row.of(1.1, false, 2, "B"),
            Row.of(1.1, true, 1, "B"),
            Row.of(2.2, true, 1, "A")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df2, "double double,
bool boolean, number int, str string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df2, "double double,
bool boolean, number int, str string");
        BatchOperator <?> hasher1 = new
FeatureHasherBatchOp().setSelectedCols("double", "bool", "number", "str")
            .setOutputCol("output").setNumFeatures(200);
        hasher1.linkFrom(inOp1).print();
        StreamOperator <?> hasher2 = new
FeatureHasherStreamOp().setSelectedCols("double", "bool", "number", "str")
            .setOutputCol("output").setNumFeatures(200);
        hasher2.linkFrom(inOp2).print();
        StreamOperator.execute();
    }
}
```

运行结果

double	bool	number	str	output
1.1000	true	2	A	$200$13:2.0 38:1.1 45:1.0 195:1.0
1.1000	false	2	B	$200$13:2.0 30:1.0 38:1.1 76:1.0
1.1000	true	1	B	$200$13:1.0 38:1.1 76:1.0 195:1.0
2.2000	true	1	A	$200$13:1.0 38:2.2 45:1.0 195:1.0

double	bool	number	str	output
1.1000	true	2	A	$200$13:2.0 38:1.1 45:1.0 195:1.0
1.1000	false	2	B	$200$13:2.0 30:1.0 38:1.1 76:1.0
1.1000	true	1	B	$200$13:1.0 38:1.1 76:1.0 195:1.0
2.2000	true	1	A	$200$13:1.0 38:2.2 45:1.0 195:1.0

double	bool	number	str	output
2.2000	true	1	A	$200$13:1.0 38:2.2 45:1.0 195:1.0
1.1000	true	1	B	$200$13:1.0 38:1.1 76:1.0 195:1.0
1.1000	true	2	A	$200$13:2.0 38:1.1 45:1.0 195:1.0
1.1000	false	2	B	$200$13:2.0 30:1.0 38:1.1 76:1.0

GBDT编码 (GbdtEncoder)

Java 类名：com.alibaba.alink.pipeline.feature.GbdtEncoder

Python 类名：GbdtEncoder

功能介绍

使用Gbdt模型，对数据进行特征转换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
criteria	树分裂的策略	树分裂的策略，可以为PAI, XGBOOST	String		"PAI", "XGBOOST
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
featureImportanceType	特征重要性类型	特征重要性类型（默认为GAIN）	String		"WEIGHT", "GAIN", "COVER"
featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double		
gamma	xgboost中的l2正则项	xgboost中的l2正则项	Double		
lambda	xgboost中的l1正则项	xgboost中的l1正则项	Double		

learningRate	学习率	学习率（默认为0.3）	Double		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
minSumHessianPerLeaf	叶子节点最小Hessian值	叶子节点最小Hessian值（默认为0）	Double		
modelFilePath	模型的文件路径	模型的文件路径	String		
newtonStep	是否使用二阶梯度	是否使用二阶梯度	Boolean		
numTrees	模型中树的棵数	模型中树的棵数	Integer		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		

reservedCols	算法保留列名	算法保留列	String[]		
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型[BIGDECIMBIGINTEGEBYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(df, schemaStr=' f0 double, f1 string,
f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(df, schemaStr=' f0 double, f1
string, f2 int, f3 int, label int')

gbdtEncoderModel = GbdtEncoder()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setPredictionCol("encoded_features")\
    .fit(batchSource)

gbdtEncoderModel.transform(batchSource).print()
gbdtEncoderModel.transform(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.GbdtEncoder;
import com.alibaba.alink.pipeline.feature.GbdtEncoderModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GbdtEncoderTest {
    @Test
    public void testGbdtEncoder() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
```

```
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, " f0
double, f1 string, f2 int, f3 int, label "
            + "int");
        GbdtEncoderModel gbdtEncoderModel = new GbdtEncoder()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setPredictionCol("encoded_features")
            .fit(batchSource);
        gbdtEncoderModel.transform(batchSource).print();
        gbdtEncoderModel.transform(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$123$0:1.0 2:1.0 4:1.0 6:1.0 8:1.0 10:1.0 12:1.0 14:1.0 16:1.0 18:1.0 20:1.0 22:1.0 24:1.0 26:1.0 28:1.0 30:1.0 32:1.0 34:1.0 36:1.0 38:1.0 40:1.0 42:1.0 44:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0 120:1.0 121:1.0 122:1.0

2.0000	B	1	1	0	$123$0:1.0 2:1.0 4:1.0 6:1.0 8:1.0 10:1.0 12:1.0 14:1.0 16:1.0 18:1.0 20:1.0 22:1.0 24:1.0 26:1.0 28:1.0 30:1.0 32:1.0 34:1.0 36:1.0 38:1.0 40:1.0 42:1.0 44:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0 120:1.0 121:1.0 122:1.0
3.0000	C	2	2	1	$123$1:1.0 3:1.0 5:1.0 7:1.0 9:1.0 11:1.0 13:1.0 15:1.0 17:1.0 19:1.0 21:1.0 23:1.0 25:1.0 27:1.0 29:1.0 31:1.0 33:1.0 35:1.0 37:1.0 39:1.0 41:1.0 43:1.0 45:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0 120:1.0 121:1.0 122:1.0
4.0000	D	3	3	1	$123$1:1.0 3:1.0 5:1.0 7:1.0 9:1.0 11:1.0 13:1.0 15:1.0 17:1.0 19:1.0 21:1.0 23:1.0 25:1.0 27:1.0 29:1.0 31:1.0 33:1.0 35:1.0 37:1.0 39:1.0 41:1.0 43:1.0 45:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0 120:1.0 121:1.0 122:1.0

Gbdt回归编码 (GbdtRegEncoder)

Java 类名：com.alibaba.alink.pipeline.feature.GbdtRegEncoder

Python 类名：GbdtRegEncoder

功能介绍

使用Gbdt回归模型，对数据进行特征转换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
criteria	树分裂的策略	树分裂的策略，可以为PAI, XGBOOST	String		"PAI", "XGBOOST
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
featureImportanceType	特征重要性类型	特征重要性类型（默认为GAIN）	String		"WEIGHT", "GAIN", "COVER"
featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double		
gamma	xgboost中的l2正则项	xgboost中的l2正则项	Double		
lambda	xgboost中的l1正则项	xgboost中的l1正则项	Double		

learningRate	学习率	学习率（默认为0.3）	Double		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
minSumHessianPerLeaf	叶子节点最小Hessian值	叶子节点最小Hessian值（默认为0）	Double		
modelFilePath	模型的文件路径	模型的文件路径	String		
newtonStep	是否使用二阶梯度	是否使用二阶梯度	Boolean		
numTrees	模型中树的棵数	模型中树的棵数	Integer		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		

reservedCols	算法保留列名	算法保留列	String[]		
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型[BIGDECIMBIGINTEGEBYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *
```

```
import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])


batchSource = BatchOperator.fromDataframe(df, schemaStr=' f0 double, f1 string,
f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(df, schemaStr=' f0 double, f1
string, f2 int, f3 int, label int')

gbdtRegEncoderModel = GbdtRegEncoder()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setPredictionCol("encoded_features")\
    .fit(batchSource)

gbdtRegEncoderModel.transform(batchSource).print()
gbdtRegEncoderModel.transform(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.GbdtRegEncoder;
import com.alibaba.alink.pipeline.feature.GbdtRegEncoderModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GbdtRegEncoderTest {
    @Test
    public void testGbdtRegEncoder() throws Exception {
        List <Row> df = Arrays.asList(
```

```
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, " f0
double, f1 string, f2 int, f3 int, label "
            + "int");
        GbdtRegEncoderModel gbdtRegEncoderModel = new GbdtRegEncoder()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setPredictionCol("encoded_features")
            .fit(batchSource);
        gbdtRegEncoderModel.transform(batchSource).print();
        gbdtRegEncoderModel.transform(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$120$0:1.0 2:1.0 4:1.0 6:1.0 8:1.0 10:1.0 12:1.0 14:1.0 16:1.0 18:1.0 20:1.0 22:1.0 24:1.0 26:1.0 28:1.0 30:1.0 32:1.0 34:1.0 36:1.0 38:1.0 40:1.0 41:1.0 42:1.0 43:1.0 44:1.0 45:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0

2.0000	B	1	1	0	$120$0:1.0 2:1.0 4:1.0 6:1.0 8:1.0 10:1.0 12:1.0 14:1.0 16:1.0 18:1.0 20:1.0 22:1.0 24:1.0 26:1.0 28:1.0 30:1.0 32:1.0 34:1.0 36:1.0 38:1.0 40:1.0 41:1.0 42:1.0 43:1.0 44:1.0 45:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0
3.0000	C	2	2	1	$120$1:1.0 3:1.0 5:1.0 7:1.0 9:1.0 11:1.0 13:1.0 15:1.0 17:1.0 19:1.0 21:1.0 23:1.0 25:1.0 27:1.0 29:1.0 31:1.0 33:1.0 35:1.0 37:1.0 39:1.0 40:1.0 41:1.0 42:1.0 43:1.0 44:1.0 45:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0
4.0000	D	3	3	1	$120$1:1.0 3:1.0 5:1.0 7:1.0 9:1.0 11:1.0 13:1.0 15:1.0 17:1.0 19:1.0 21:1.0 23:1.0 25:1.0 27:1.0 29:1.0 31:1.0 33:1.0 35:1.0 37:1.0 39:1.0 40:1.0 41:1.0 42:1.0 43:1.0 44:1.0 45:1.0 46:1.0 47:1.0 48:1.0 49:1.0 50:1.0 51:1.0 52:1.0 53:1.0 54:1.0 55:1.0 56:1.0 57:1.0 58:1.0 59:1.0 60:1.0 61:1.0 62:1.0 63:1.0 64:1.0 65:1.0 66:1.0 67:1.0 68:1.0 69:1.0 70:1.0 71:1.0 72:1.0 73:1.0 74:1.0 75:1.0 76:1.0 77:1.0 78:1.0 79:1.0 80:1.0 81:1.0 82:1.0 83:1.0 84:1.0 85:1.0 86:1.0 87:1.0 88:1.0 89:1.0 90:1.0 91:1.0 92:1.0 93:1.0 94:1.0 95:1.0 96:1.0 97:1.0 98:1.0 99:1.0 100:1.0 101:1.0 102:1.0 103:1.0 104:1.0 105:1.0 106:1.0 107:1.0 108:1.0 109:1.0 110:1.0 111:1.0 112:1.0 113:1.0 114:1.0 115:1.0 116:1.0 117:1.0 118:1.0 119:1.0

Id3编码 (Id3Encoder)

Java 类名：com.alibaba.alink.pipeline.feature.Id3Encoder

Python 类名：Id3Encoder

功能介绍

使用ID3模型，对数据进行特征转换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
```

```
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(\
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(\
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

id3EncoderModel = Id3Encoder()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setPredictionCol("encoded_features")\
    .fit(batchSource)

id3EncoderModel.transform(batchSource).print()
id3EncoderModel.transform(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.Id3Encoder;
import com.alibaba.alink.pipeline.feature.Id3EncoderModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class Id3EncoderTest {
    @Test
    public void testId3Encoder() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
```

```
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, " f0
double, f1 string, f2 int, f3 int, label "
            + "int");
        Id3EncoderModel id3EncoderModel = new Id3Encoder()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setPredictionCol("encoded_features")
            .fit(batchSource);
        id3EncoderModel.transform(batchSource).print();
        id3EncoderModel.transform(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$2$0:1.0
2.0000	B	1	1	0	$2$0:1.0
3.0000	C	2	2	1	$2$1:1.0
4.0000	D	3	3	1	$2$1:1.0

多热编码 (MultiHotEncoder)

Java 类名：com.alibaba.alink.pipeline.feature.MultiHotEncoder

Python 类名：MultiHotEncoder

功能介绍

multi-hot编码，也称多热编码，是与独热编码相对应的一种编码方式。该编码对每一个字符串特征列按照指定分隔符进行分割，分割得到的值存在m个可能值，那么经过多热编码后就变成了m个二元特征。对每一字段编码将会把该字段分割后的每一个值映射到唯一的编码。 因此，编码后的数据会变成稀疏数据，输出结果也是kv的稀疏结构。

编码结果

输入

col_0	col_1
"a b"	"1 2"
"b c"	"1 3"
"c d"	"1 4"
"a d"	"3 2"
"d e"	null
NULL	"2 3"

Encode ——> VECTOR

预测结果为稀疏向量:

向量中非零元个数必定为1, 只能是一个稀疏向量`$5$0:1.0 4:1.0`或者`NULL`。

Encode ——> ASSEMBLED_VECTOR

预测结果为稀疏向量,是预测选择列中,各列预测为VECTOR时,按照选择顺序ASSEMBLE的结果。

向量维度

Encode ——> Vector

$$ vectorSize = distinct token Number + enableElse(true: 1, false:0) + (handleInvalid: keep(1), skip(0), error(0)) $$

```
distinct token Number: 训练集中指定列的去重后的token数目

enableElse: 训练时若填写discreteThresholds或discreteThresholdsArray则为true，默认为false

handleInvalid: 预测参数
```

举例

输入列为col_0

```
1. 如果没有填写discreteThresholds，那么enableElse为false，distinct token Number为(a,b,c,d,e)一共5个token
        1.1.1 handleInvalid为keep: vectorSize=(5 + 0 + 1 = 6)
        1.2.2 handleInvalid为skip: vectorSize=(5 + 0 + 0 = 5)
        1.2.3 handleInvalid为error: vectorSize=(5 + 0 + 0 = 5)
2. 如果discreteThresholds为2，那么enableElse为true，distinct token Number为(a,b,c,d,e)一共5个token
        1.1.1 handleInvalid为keep: vectorSize=(5 + 1 + 1 = 7)
        1.2.2 handleInvalid为skip: vectorSize=(5 + 1 + 0 = 6)
        1.2.3 handleInvalid为error: vectorSize=(5 + 1 + 0 = 6)
```

Token index

Encode ——> Vector

```
1. 训练集中出现过的token: 预测值为模型中token对应的token_index

2. 训练集中未出现过的token:
    3.1 enableElse为true
        3.1.1 handleInvalid为keep: 预测值为:distinct token Number + 1
        3.1.2 handleInvalid为skip: 预测值为:distinct token Number
        3.1.3 handleInvalid为error: 预测值为:distinct token Number

    3.2 enableElse为false
        3.2.1 handleInvalid为keep: 预测值为:distinct token Number
        3.2.2 handleInvalid为skip: 无index
        3.2.3 handleInvalid为error: 报错
```

举例

输入列为col_0

1. 如果没有填写discreteThresholds, 假设模型中a,b,c,d,e对应的token index为0,1,2,3,4

1.1 handleInvalid为keep

col_0	Encode为VECTOR的输出

"a b"	$6$0:1.0 1:1.0
"b c"	$6$1:1.0 2:1.0
"c d"	$6$3:1.0 3:1.0
"a d"	$6$0:1.0 3:1.0
"d e"	$6$0:3.0 4:1.0
NULL	NULL

1.2 handleInvalid为skip

col_0	Encode为VECTOR的输出
"a b"	$5$0:1.0 1:1.0
"b c"	$5$1:1.0 2:1.0
"c d"	$5$3:1.0 3:1.0
"a d"	$5$0:1.0 3:1.0
"d e"	$5$0:3.0 4:1.0
NULL	NULL

1.3 handleInvalid为error: 直接报错

参数说明

名称	中文名称	描述	类型	是否必须?	取值范
outputCols	输出结果列列名数组	输出结果列列名数组，必选	String[]	✓	
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	
delimiter	分隔符	用来分割字符串	String		
discreteThresholds	离散个数阈值	离散个数阈值，低于该阈值的离散样本将不会单独成一个组别。	Integer		

discreteThresholdsArray	离散个数阈值	离散个数阈值，每一列对应数组中一个元素。	Integer[]		
encode	编码方法	编码方法	String		"VECTOR", "ASSEMBLED
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null， "error"表示抛异常	String		"KEEP", "ERR "SKIP"
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a b", 1],
    ["b c", 1],
    ["c d", 1],
    ["a d", 2],
    ["d e", 2],
    [None, 1]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='query string, weight long')

multi_hot =
MultiHotEncoder().setSelectedCols(["query"]).setOutputCols(["output"])
multi_hot.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.feature.MultiHotEncoder;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class MultiHotEncoderTest {
```

```
    @Test
    public void testMultiHotEncoder() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a b", 1),
            Row.of("b c", 1),
            Row.of("c d", 1),
            Row.of("a d", 2),
            Row.of("d e", 2),
            Row.of(null, 1)
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "query string, weight
int");
        MultiHotEncoder multi_hot = new
MultiHotEncoder().setSelectedCols("query").setOutputCols("output");
        multi_hot.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

query	weight	output
a b	1	$6$0:1.0 1:1.0
b c	1	$6$1:1.0 2:1.0
c d	1	$6$2:1.0 3:1.0
a d	2	$6$0:1.0 3:1.0
d e	2	$6$3:1.0 4:1.0
null	1	null

独热编码 (OneHotEncoder)

Java 类名：com.alibaba.alink.pipeline.feature.OneHotEncoder

Python 类名：OneHotEncoder

参数说明

名称	中文名称	描述	类型	是否必须?	取值范
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	
discreteThresholds	离散个数阈值	离散个数阈值，低于该阈值的离散样本将不会单独成一个组别。	Integer		
discreteThresholdsArray	离散个数阈值	离散个数阈值，每一列对应数组中一个元素。	Integer[]		
dropLast	是否删除最后一个元素	删除最后一个元素是为了保证线性无关性。默认true	Boolean		
encode	编码方法	编码方法	String		"VECTOR", "ASSEMBLED "INDEX"
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null， "error"表示抛异常	String		"KEEP", "ERR
modelFilePath	模型的文件路径	模型的文件路径	String		

outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1],
    ["b", 1],
    ["c", 1],
    ["e", 2],
    ["a", 2],
    ["b", 1],
    ["c", 2],
    ["d", 2],
    [None, 1]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='query string, weight long')

# one hot train
one_hot = OneHotEncoder().setSelectedCols(["query"]).setOutputCols(["output"])
one_hot.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.feature.OneHotEncoder;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class OneHotEncoderTest {
    @Test
    public void testOneHotEncoder() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a", 1),
            Row.of("b", 1),
            Row.of("c", 1),
            Row.of("e", 2),
            Row.of("a", 2),
            Row.of("b", 1),
            Row.of("c", 2),
            Row.of("d", 2),
            Row.of(null, 1)
        );
```

```
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "query string, weight
int");
        OneHotEncoder one_hot = new
OneHotEncoder().setSelectedCols("query").setOutputCols("output");
        one_hot.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

query	weight	output
a	1	$5$0:1.0
b	1	$5$1:1.0
c	1	$5$2:1.0
e	2	5
a	2	$5$0:1.0
b	1	$5$1:1.0
c	2	$5$2:1.0
d	2	$5$3:1.0
null	1	$5$4:1.0

主成分分析 (PCA)

Java 类名：com.alibaba.alink.pipeline.feature.PCA

Python 类名：PCA

功能介绍

主成分分析，是考察多个变量间相关性一种多元统计方法，研究如何通过少数几个主成分来揭示多个变量间的内部结构，即从原始变量中导出少数几个主成分，使它们尽可能多地保留原始变量的信息，且彼此间互不相关，作为新的综合指标。详细介绍请见维基百科链接wiki。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默
k	降维后的维度	降维后的维度	Integer	✓	[1, +inf)	
predictionCol	预测结果列名	预测结果列名	String	✓		
calculationType	计算类型	计算类型，包含"CORR", "COV"两种。	String		"CORR", "COV"	"CO
modelFilePath	模型的文件路径	模型的文件路径	String			null

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			fals
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
        [0.0,0.0,0.0],
        [0.1,0.2,0.1],
        [0.2,0.2,0.8],
        [9.0,9.5,9.7],
        [9.1,9.1,9.6],
        [9.2,9.3,9.9]
])

# batch source
inOp = BatchOperator.fromDataframe(df, schemaStr='x1 double, x2 double, x3
double')

pca = PCA().setK(2).setSelectedCols(["x1","x2","x3"]).setPredictionCol("pred")

# train
```

```
model = pca.fit(inOp)

# batch predict
model.transform(inOp).print()

# stream predict
inStreamOp = StreamOperator.fromDataframe(df, schemaStr='x1 double, x2 double,
x3 double')

model.transform(inStreamOp).print()

StreamOperator.execute()
```

Java 代码

```
package javatest.com.alibaba.alink.pipeline.feature;

import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.PCA;
import com.alibaba.alink.pipeline.feature.PCAModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class PcaTest {

    @Test
    public void testPca() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0.0, 0.0, 0.0),
            Row.of(0.1, 0.2, 0.1),
            Row.of(0.2, 0.2, 0.8),
            Row.of(9.0, 9.5, 9.7),
            Row.of(9.1, 9.1, 9.6),
            Row.of(9.2, 9.3, 9.9)
        );

        BatchOperator <?> inOp = new MemSourceBatchOp(df, "x1 double, x2
double, x3 double");
        MemSourceStreamOp inStreamOp = new MemSourceStreamOp(df, "x1 double, x2
double, x3 double");
```

```
        PCA pca = new PCA()
            .setK(2)
            .setSelectedCols(new String[] {"x1", "x2",
"x3"}).setPredictionCol("pred");

        PCAModel model = pca.fit(inOp);

        model.transform(inOp).print();

        model.transform(inStreamOp).print();

        StreamOperator.execute();
    }
}
```

结果

x1	x2	x3	pred
0.0000	0.0000	0.0000	-1.6404909810453345 -0.0251812826908675
0.1000	0.2000	0.1000	-1.5946357760302712 -0.037364200387782764
0.2000	0.2000	0.8000	-1.5048402139720405 0.06485201225195414
9.0000	9.5000	9.7000	1.587547449494739 -0.02506612043660217
9.1000	9.1000	9.6000	1.5421273389336387 0.0022882493013524074
9.2000	9.3000	9.9000	1.6102921826192689 0.020471341961945777

分位数离散化 (QuantileDiscretizer)

Java 类名：com.alibaba.alink.pipeline.feature.QuantileDiscretizer

Python 类名：QuantileDiscretizer

功能介绍

分位点离散可以计算选定列的分位点，然后使用这些分位点进行离散化。 生成选中列对应的q-quantile，其中可以所有列指定一个，也可以每一列对应一个

参数说明

名称	中文名称	描述	类型	是否必须?	取值
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓	
dropLast	是否删除最后一个元素	删除最后一个元素是为了保证线性无关性。默认true	Boolean		
encode	编码方法	编码方法	String		"VECTOR", "ASSEMBL "INDEX"
handleInvalid	未知token处理策略	未知token处理策略。"keep"表示用最大id加1代替, "skip"表示补null， "error"表示抛异常	String		"KEEP", "E
leftOpen	是否左开右闭	左开右闭为true，左闭右开为false	Boolean		
modelFilePath	模型的文件路径	模型的文件路径	String		
numBuckets	quantile个数	quantile个数，对所有列有效。	Integer		
numBucketsArray	quantile个数	quantile个数，每一列对应数组中一个元素。	Integer[]		

outputCols	输出结果列列名数组	输出结果列列名数组，可选，默认null	String[]		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a", 1, 1, 2.0, True],
    ["c", 1, 2, -3.0, True],
    ["a", 2, 2, 2.0, False],
    ["c", 0, 0, 0.0, False]
])
```

```
batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f_string string, f_long long, f_int int, f_double double,
f_boolean boolean')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f_string string, f_long long, f_int int, f_double double,
f_boolean boolean')

QuantileDiscretizer()\
    .setSelectedCols(['f_double'])\
    .setNumBuckets(8)\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()

QuantileDiscretizer()\
    .setSelectedCols(['f_double'])\
    .setNumBuckets(8)\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.QuantileDiscretizer;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class QuantileDiscretizerTest {
    @Test
    public void testQuantileDiscretizer() throws Exception {
        List <Row> sourceFrame = Arrays.asList(
            Row.of("a", 1, 1, 2.0, true),
            Row.of("c", 1, 2, -3.0, true),
            Row.of("a", 2, 2, 2.0, false),
            Row.of("c", 0, 0, 0.0, false)
        );
```

```
        BatchOperator <?> batchSource = new MemSourceBatchOp(sourceFrame,
            "f_string string, f_long int, f_int int, f_double double, f_boolean
boolean");
        StreamOperator <?> streamSource = new MemSourceStreamOp(sourceFrame,
            "f_string string, f_long int, f_int int, f_double double, f_boolean
boolean");
        new
QuantileDiscretizer().setSelectedCols("f_double").setNumBuckets(8).fit(batchSou
rce).transform(batchSource)
            .print();
        new
QuantileDiscretizer().setSelectedCols("f_double").setNumBuckets(8).fit(batchSou
rce).transform(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

f_string	f_long	f_int	f_double	f_boolean
a	1	1	2	true
c	1	2	0	true
a	2	2	2	false
c	0	0	1	false

随机森林编码 (RandomForestEncoder)

Java 类名：com.alibaba.alink.pipeline.feature.RandomForestEncoder

Python 类名：RandomForestEncoder

功能介绍

使用随机森林模型，对数据进行特征转换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double		

maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		

modelFilePath	模型的文件路径	模型的文件路径	String		
numSubsetFeatures	每棵树的特征采样数目	每棵树的特征采样数目	Integer		
numTrees	模型中树的棵数	模型中树的棵数	Integer		[1, +inf)
numTreesOfGini	模型中Cart树的棵数	模型中Cart树的棵数	Integer		
numTreesOfInfoGain	模型中Id3树的棵数	模型中Id3树的棵数	Integer		
numTreesOfInfoGainRatio	模型中C4.5树的棵数	模型中C4.5树的棵数	Integer		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(\
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(\
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

randomforestEncoderModel = RandomForestEncoder()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setPredictionCol("encoded_features")\
    .fit(batchSource)

randomforestEncoderModel.transform(batchSource).print()
randomforestEncoderModel.transform(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.RandomForestEncoder;
import com.alibaba.alink.pipeline.feature.RandomForestEncoderModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RandomForestEncoderTest {
    @Test
    public void testRandomForestEncoder() throws Exception {
```

```
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, " f0
double, f1 string, f2 int, f3 int, label "
            + "int");
        RandomForestEncoderModel randomforestEncoderModel = new
RandomForestEncoder()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setPredictionCol("encoded_features")
            .fit(batchSource);
        randomforestEncoderModel.transform(batchSource).print();
        randomforestEncoderModel.transform(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$18$0:1.0 2:1.0 4:1.0 5:1.0 7:1.0 9:1.0 11:1.0 13:1.0 15:1.0 17:1.0
2.0000	B	1	1	0	$18$0:1.0 2:1.0 4:1.0 5:1.0 7:1.0 9:1.0 11:1.0 13:1.0 15:1.0 17:1.0
3.0000	C	2	2	1	$18$1:1.0 3:1.0 4:1.0 6:1.0 8:1.0 10:1.0 12:1.0 14:1.0 16:1.0 17:1.0
4.0000	D	3	3	1	$18$1:1.0 3:1.0 4:1.0 6:1.0 8:1.0 10:1.0 12:1.0 14:1.0 16:1.0 17:1.0

随机森林回归编码 (RandomForestRegEncoder)

Java 类名：com.alibaba.alink.pipeline.feature.RandomForestRegEncoder

Python 类名：RandomForestRegEncoder

功能介绍

使用随机森林回归模型，对数据进行特征转换。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
numSubsetFeatures	每棵树的特征采样数目	每棵树的特征采样数目	Integer		
numTrees	模型中树的棵数	模型中树的棵数	Integer		[1, +inf)

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
seed	采样种子	采样种子	Long		
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(\
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(\
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

randomforestRegEncoderModel = RandomForestRegEncoder()\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .setPredictionCol("encoded_features")\
    .fit(batchSource)

randomforestRegEncoderModel.transform(batchSource).print()
randomforestRegEncoderModel.transform(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.feature.RandomForestRegEncoder;
import com.alibaba.alink.pipeline.feature.RandomForestRegEncoderModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RandomForestRegEncoderTest {
    @Test
    public void testRandomForestRegEncoder() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, " f0
double, f1 string, f2 int, f3 int, label "
            + "int");

        RandomForestRegEncoderModel randomforestRegEncoderModel = new
RandomForestRegEncoder()
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setPredictionCol("encoded_features")
            .fit(batchSource);
        randomforestRegEncoderModel.transform(batchSource).print();
        randomforestRegEncoderModel.transform(streamSource).print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	encoded_features
1.0000	A	0	0	0	$18$0:1.0 2:1.0 4:1.0 5:1.0 7:1.0 9:1.0 11:1.0 13:1.0 15:1.0 17:1.0
2.0000	B	1	1	0	$18$0:1.0 2:1.0 4:1.0 5:1.0 7:1.0 9:1.0 11:1.0 13:1.0 15:1.0 17:1.0

3.0000	C	2	2	1	$18$1:1.0 3:1.0 4:1.0 6:1.0 8:1.0 10:1.0 12:1.0 14:1.0 16:1.0 17:1.0
4.0000	D	3	3	1	$18$1:1.0 3:1.0 4:1.0 6:1.0 8:1.0 10:1.0 12:1.0 14:1.0 16:1.0 17:1.0

Bert文本嵌入 (BertTextEmbedding)

Java 类名：com.alibaba.alink.pipeline.nlp.BertTextEmbedding

Python 类名：BertTextEmbedding

功能介绍

把文本输入到 BERT 模型，提取某一编码层的 pooled output 作为该句子的 embedding 结果。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		
bertModelName	BERT模型名字	BERT模型名字： Base-Chinese,Base-Multilingual-Cased,Base-Uncased,Base-Cased	String			"Base-Chinese"
doLowerCase	是否将文本转换为小写	是否将文本转换为小写，默认根据模型自动决定	Boolean			null
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4
layer	输出第几层 encoder layer 的结果	输出第几层 encoder layer 的结果， -1 表示最后一层，-2 表示倒数第2层，以此类推	Integer			-1
maxSeqLength	句子截断长度	句子截断长度	Integer			128
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 'An english sentence.'],
    [2, '这是一个中文句子']
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr='f1 bigint, f2
string')

BertTextEmbedding() \
    .setSelectedCol("f2") \
    .setOutputCol("embedding") \
    .setLayer(-2) \
    .transform(batch_data) \
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.common.MLEnvironmentFactory;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.pipeline.nlp.BertTextEmbedding;
import org.junit.Test;

public class BertTextEmbeddingTest {

    @Test
    public void testBertTextEmbedding() throws Exception {
        Row[] rows1 = new Row[] {
            Row.of(1L, "An english sentence."),
            Row.of(2L, "这是一个中文句子"),
        };

        BatchOperator <?> data = BatchOperator.fromTable(
            MLEnvironmentFactory.getDefault().createBatchTable(rows1, new
String[] {"f1", "f2"}));
```

```
            new BertTextEmbedding()
                .setSelectedCol("f2").setOutputCol("embedding").setLayer(-2)
                .setDoLowerCase(true)
                .setIntraOpParallelism(4)
                .transform(data)
                .print();
    }
}
```

运行结果

f1	f2	embedding
1	An english sentence.	-0.4501993 0.06074004 0.121287264 -0.27875 0.3...
2	这是一个中文句子	-0.8317032 0.32284066 -0.12233654 -0.6955824 0...

文本特征生成 (DocCountVectorizer)

Java 类名：com.alibaba.alink.pipeline.nlp.DocCountVectorizer

Python 类名：DocCountVectorizer

功能介绍

根据文本中词语的特征信息，将每条文本转化为稀疏向量。

使用方式

文本内容列（SelectedCol）中的内容用于统计词语的统计信息，需要是用空格分隔的词语。

将文本转换为稀疏向量时，每个唯一的词语将对应向量中的一个唯一的索引值。 而向量中对应索引的值表示这个词语在文本中的特征信息，可以通过参数 featureType 来选择不同的特征。

在转换时，所使用的词语集合还可以通过参数来进行筛选：

- maxDF/minDF：根据包含词语的文本次数（DF）进行筛选（当设置值在[0,1)区间时，表示占总文本数的比例）；
- minTF：仅在预测单条文本时起作用，根据词语在当前文本中的出现的次数进行筛选（当设置值在[0,1)区间时，表示占当前文本总次数的比例）；
- vocabSize：根据词语在所有文本中出现的总次数排序，只使用前 vocabSize 个词语。

参数说明

名称	中文名称	描述	类型	是否必须?	
selectedCol	选中的列名	计算列对应的列名	String	✓	
featureType	特征类型	生成特征向量的类型，支持IDF/WORD_COUNT/TF_IDF/Binary/TF	String		"IDF "WC "TF_ "BIN

maxDF	最大词频	如果一个词出现的文档次数大于maxDF,这个词不会被包含在字典中。maxDF可以是具体的词频也可以是整体词频的比例，如果minDF在[0,1)区间，会被认为是比例。	Double		
minDF	最小文档词频	如果一个词出现的文档次数小于minDF,这个词不会被包含在字典中。minTF可以是具体的词频也可以是整体词频的比例，如果minDF在[0,1)区间，会被认为是比例。	Double		
minTF	最低词频	最低词频，如果词频小于minTF,这个词会被忽略掉。minTF可以是具体的词频也可以是整体词频的比例，如果minTF在[0,1)区间，会被认为是比例。	Double		
modelFilePath	模型的文件路径	模型的文件路径	String		
outputCol	输出结果列	输出结果列列名，可选，默认null	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

vocabSize	字典库大小	字典库大小，如果总词数目大于这个值，那个文档频率低的词会被过滤掉。	Integer		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集（ 国内版 ）全十二册馆藏书']
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, text string')

pipeline = Pipeline() \
    .add(Segment().setSelectedCol("text")) \
    .add(DocCountVectorizer().setSelectedCol("text"))

pipeline.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.Pipeline;
import com.alibaba.alink.pipeline.nlp.DocCountVectorizer;
import com.alibaba.alink.pipeline.nlp.Segment;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DocCountVectorizerTest {
    @Test
    public void testDocCountVectorizer() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集（ 国内版 ）全十二册馆藏书")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text
string");
        Pipeline pipeline = new Pipeline()
            .add(new Segment().setSelectedCol("text"))
            .add(new DocCountVectorizer().setSelectedCol("text"));
```

```
        pipeline.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

输出数据

id	text
0	$37$10:1.0 14:1.0 18:1.0 25:1.0 29:1.0 34:1.0
1	$37$0:1.0 1:1.0 4:1.0 7:1.0 13:1.0 17:1.0 22:1.0 26:1.0 29:1.0 33:1.0 35:1.0
2	$37$5:1.0 6:1.0 12:1.0 19:1.0 20:1.0 23:1.0 26:1.0 28:1.0 29:1.0 31:1.0 36:2.0
3	$37$8:1.0 9:1.0 16:1.0 29:1.0 32:1.0
4	$37$0:1.0 1:1.0 2:1.0 3:1.0 11:1.0 15:1.0 21:1.0 24:1.0 27:1.0 29:1.0 30:1.0

文本哈希特征生成 (DocHashCountVectorizer)

Java 类名：com.alibaba.alink.pipeline.nlp.DocHashCountVectorizer

Python 类名：DocHashCountVectorizer

功能介绍

根据文本中词语的特征信息，将每条文本转化为固定长度的稀疏向量。

在转换时，每个词语会通过哈希函数映射到稀疏向量的一个索引值，映射到同一个索引值的多个词语将看作同一个词语来统计特征信息。

使用方式

文本内容列（SelectedCol）中的内容用于统计词语的统计信息，需要是用空格分隔的词语。

将文本转换为稀疏向量时，需要指定向量维度（numFeatures）。 每个词语会通过哈希函数映射到一个 [0, numFeatures) 内的索引值，映射到同一个索引值的多个词语将看作同一个词语来统计特征信息。 而向量中对应索引的值表示对应的词语在文本中的特征信息，可以通过参数 featureType 来选择不同的特征。

在转换时，所使用的词语集合还可以通过参数来进行筛选：

- maxDF/minDF：根据包含词语的文本次数（DF）进行筛选（当设置值在[0,1)区间时，表示占总文本数的比例）；
- minTF：仅在预测单条文本时起作用，根据词语在当前文本中的出现的次数进行筛选（当设置值在[0,1)区间时，表示占当前文本总次数的比例）。

参数说明

名称	中文名称	描述	类型	是否必须?	
selectedCol	选中的列名	计算列对应的列名	String	✓	
featureType	特征类型	生成特征向量的类型，支持IDF/WORD_COUNT/TF_IDF/Binary/TF	String		"IDF "WC "TF_ "BIN

minDF	最小文档词频	如果一个词出现的文档次数小于minDF,这个词不会被包含在字典中。minTF可以是具体的词频也可以是整体词频的比例，如果minDF在[0,1)区间，会被认为是比例。	Double		
minTF	最低词频	最低词频，如果词频小于minTF,这个词会被忽略掉。minTF可以是具体的词频也可以是整体词频的比例，如果minTF在[0,1)区间，会被认为是比例。	Double		
modelFilePath	模型的文件路径	模型的文件路径	String		
numFeatures	向量维度	生成向量长度	Integer		
outputCol	输出结果列	输出结果列列名，可选，默认null	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969'],
```

```
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集（ 国内版 ）全十二册馆藏书']
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, text string')

pipeline = Pipeline() \
    .add(Segment().setSelectedCol("text")) \
    .add(DocHashCountVectorizer().setSelectedCol("text"))

pipeline.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.Pipeline;
import com.alibaba.alink.pipeline.nlp.DocHashCountVectorizer;
import com.alibaba.alink.pipeline.nlp.Segment;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DocHashCountVectorizerTest {
    @Test
    public void testDocHashCountVectorizer() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集（ 国内版 ）全十二册馆藏书")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text
string");
        Pipeline pipeline = new Pipeline()
            .add(new Segment().setSelectedCol("text"))
            .add(new DocHashCountVectorizer().setSelectedCol("text"));
        pipeline.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

输出数据

id	text
0	$262144$10121:1.0 64444:1.0 119456:1.0 206232:1.0 210357:1.0 256946:1.0
1	$262144$0:6.0 37505:1.0 46743:1.0 93228:1.0 119217:1.0 138080:1.0 141480:1.0 172901:1.0 206232:1.0 216139:1.0 226698:1.0 254628:1.0
2	$262144$40170:1.0 70777:1.0 96509:1.0 126159:1.0 158267:1.0 181703:1.0 206232:1.0 216139:1.0 232884:1.0 250534:2.0 259932:1.0
3	$262144$206232:1.0 214785:1.0 251090:1.0 255656:1.0 261064:1.0
4	$262144$0:4.0 87711:1.0 138080:1.0 162140:1.0 180035:1.0 195777:1.0 206232:1.0 219988:1.0 241122:1.0 254628:1.0 257763:1.0 259051:1.0

NGram (NGram)

Java 类名：com.alibaba.alink.pipeline.nlp.NGram

Python 类名：NGram

功能介绍

根据文本生成对应的 NGram 结果。

算法原理

N-Gram 是一种基于统计语言模型的算法，它将文本里面的内容进行大小为 N 的滑动窗口操作，形成了长度是 N 的片段序列。

使用方式

该组件对于文本内容列（SelectedCol）中的每一行文本进行 N-Gram 处理，产生一行输出。 N 的值通过参数 n 设置。

输入每行文本中各个词语间需要以空格进行分割，可以使用分词（SegmentBatchOp）组件的输出结果列 输出一行中包含多个 N-Gram 结果，各个结果间用空格分隔；单个结果中各个词语间用下划线连接。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
n	nGram长度	nGram长度	Integer			2
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, 'That is an English Book!'],
    [1, 'Do you like math?'],
    [2, 'Have a good day!']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id long, text string')

op = NGram().setSelectedCol("text")
op.transform(inOp1).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.nlp.NGram;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class NGramTest {
    @Test
    public void testNGram() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "That is an English Book!"),
            Row.of(1, "Do you like math?"),
            Row.of(2, "Have a good day!")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        NGram op = new NGram().setSelectedCol("text");
        op.transform(inOp1).print();
    }
}
```

运行结果

id	text
0	That_is is_an an_English English_Book!
1	Do_you you_like like_math?
2	Have_a a_good good_day!

RegexTokenizer (RegexTokenizer)

Java 类名：com.alibaba.alink.pipeline.nlp.RegexTokenizer

Python 类名：RegexTokenizer

功能介绍

通过正则表达式对文本进行切分或者匹配操作。

使用方式

文本列通过参数 selectedCol 指定，切分或者匹配用的正则表达式通过参数 pattern 指定。

当参数 gaps 为 True 时，对文本进行切分操作（类似于分词）； 当参数 gaps 为 为 False 时，将提取匹配正则表达式的词语。

对于处理后的结果，还可以通过参数 minTokenLength 根据长度筛掉词语，或者通过参数 toLowerCase 将所有词语转为小写。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
gaps	切分/匹配	如果gaps为True，pattern用于切分文档；如果gaps为False，会提取出匹配pattern的词。	Boolean			true
minTokenLength	词语最短长度	词语的最短长度，小于这个值的词语会被过滤掉	Integer			1
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
pattern	分隔符/正则匹配符	如果gaps为True，pattern用于切分文档；如果gaps为False，会提取出匹配pattern的词。	String			"\s+"

reservedCols	算法保留列名	算法保留列	String[]			null
toLowerCase	是否转换为小写	转换为小写	Boolean			true
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, 'That is an English Book!'],
    [1, 'Do you like math?'],
    [2, 'Have a good day!']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id long, text string')
op =
RegexTokenizer().setSelectedCol("text").setGaps(False).setToLowerCase(True).set
OutputCol("token").setPattern(
    "\\w+")
op.transform(inOp1).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.nlp.RegexTokenizer;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RegexTokenizerTest {
    @Test
```

```
    public void testRegexTokenizer() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "That is an English Book!"),
            Row.of(1, "Do you like math?"),
            Row.of(2, "Have a good day!")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        RegexTokenizer op = new
RegexTokenizer().setSelectedCol("text").setGaps(false).setToLowerCase(true)
            .setOutputCol("token").setPattern("\\w+");
        op.transform(inOp1).print();
    }
}
```

运行结果

id	text	token
0	That is an English Book!	that is an english book
1	Do you like math?	do you like math
2	Have a good day!	have a good day

分词 (Segment)

Java 类名：com.alibaba.alink.pipeline.nlp.Segment

Python 类名：Segment

功能介绍

对文本进行分词，分词后各个词语间用空格分隔。

使用方式

文本列通过参数 selectedCol 指定。 词典文件可以从这里 查看。 通过参数 userDefinedDict 可以添加额外的词语。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
userDefinedDict	用户自定义字典	用户自定义字典	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读 （ 下册 ） 李宜燮南开大学出版社 9787310003969'],
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集 （ 国内版 ） 全十二册馆藏书']
])
```

```
inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text string')

segment = Segment().setSelectedCol("text").setOutputCol("segment")

segment.transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.nlp.Segment;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SegmentTest {
    @Test
    public void testSegment() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集（ 国内版 ）全十二册馆藏书")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text
string");
        Segment segment = new
Segment().setSelectedCol("text").setOutputCol("segment");
        segment.transform(inOp).print();
    }
}
```

运行结果

id	text	segment
0	二手旧书:医学电磁成像	二手 旧书 : 医学 电磁 成像
1	二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969	二手 美国 文学 选读 （ 下册 ） 李宜燮 南开大学 出版社 9787310003969

2	二手正版图解象棋入门/谢恩思主编/华龄出版社	二手 正版 图解 象棋 入门 / 谢恩 思 主编 / 华龄 出版社
3	二手中国糖尿病文献索引	二手 中国 糖尿病 文献 索引
4	二手郁达夫文集（国内版）全十二册馆藏书	二手 郁达夫 文集 （ 国内 版 ） 全 十二册 馆藏 书

停用词过滤 (StopWordsRemover)

Java 类名：com.alibaba.alink.pipeline.nlp.StopWordsRemover

Python 类名：StopWordsRemover

功能介绍

过滤文本中的噪声词语（例如：的、是、啊等）。

使用方式

文本列通过参数 selectedCol 指定，需要是空格分隔的词语，可以使用分词（Segment）组件的输出结果列。 可以通过参数 caseSensitive 设置过滤时是否大小写敏感。

噪声词表可以从这里 查看。 通过参数 stopWords 可以添加额外的噪声词语。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
caseSensitive	是否大小写敏感	大小写敏感	Boolean			false
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
stopWords	用户自定义停用词表	用户自定义停用词表	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, u'二手旧书:医学电磁成像'],
    [1, u'二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969'],
```

```
    [2, u'二手正版图解象棋入门/谢恩思主编/华龄出版社'],
    [3, u'二手中国糖尿病文献索引'],
    [4, u'二手郁达夫文集（ 国内版 ）全十二册馆藏书']
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text string')

pipeline = Pipeline() \
    .add(Segment().setSelectedCol("text")) \
    .add(StopWordsRemover().setSelectedCol("text"))
pipeline.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.Pipeline;
import com.alibaba.alink.pipeline.nlp.Segment;
import com.alibaba.alink.pipeline.nlp.StopWordsRemover;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StopWordsRemoverTest {
    @Test
    public void testStopWordsRemover() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "二手旧书:医学电磁成像"),
            Row.of(1, "二手美国文学选读（ 下册 ）李宜燮南开大学出版社 9787310003969"),
            Row.of(2, "二手正版图解象棋入门/谢恩思主编/华龄出版社"),
            Row.of(3, "二手中国糖尿病文献索引"),
            Row.of(4, "二手郁达夫文集（ 国内版 ）全十二册馆藏书")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text
string");
        Pipeline pipeline = new Pipeline()
            .add(new Segment().setSelectedCol("text"))
            .add(new StopWordsRemover().setSelectedCol("text"));
        pipeline.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

id	text
0	二手 旧书 医学 电磁 成像
1	二手 美国 文学 选读 下册 李宜燮 南开大学 出版社 9787310003969
2	二手 正版 图解 象棋 入门 谢恩 思 主编 华龄 出版社
3	二手 中国 糖尿病 文献 索引
4	二手 郁达夫 文集 国内 版 全 十二册 馆藏 书

文本分解 (Tokenizer)

Java 类名：com.alibaba.alink.pipeline.nlp.Tokenizer

Python 类名：Tokenizer

功能介绍

对文本按空白符进行切分操作。

使用方式

文本列通过参数 selectedCol 指定，输出列通过 outputCol 指定。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
selectedCol	选中的列名	计算列对应的列名	String	✓		
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
df = pd.DataFrame([
    [0, 'That is an English Book!'],
    [1, 'Do you like math?'],
    [2, 'Have a good day!']
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id long, text string')

op = Tokenizer().setSelectedCol("text")
op.transform(inOp1).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.nlp.Tokenizer;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TokenizerTest {
    @Test
    public void testTokenizer() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "That is an English Book!"),
            Row.of(1, "Do you like math?"),
            Row.of(2, "Have a good day!")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text
string");
        Tokenizer op = new Tokenizer().setSelectedCol("text");
        op.transform(inOp1).print();
    }
}
```

运行结果

id	text
0	that is an english book!
1	do you like math?
2	have a good day!

Word2Vec (Word2Vec)

Java 类名：com.alibaba.alink.pipeline.nlp.Word2Vec

Python 类名：Word2Vec

功能介绍

Word2Vec是Google在2013年开源的一个将词表转为向量的算法，其利用神经网络，可以通过训练，将词映射到K维度空间向量，甚至对于表示词的向量进行操作还能和语义相对应，由于其简单和高效引起了很多人的关注。

Word2Vec的工具包相关链接：https://code.google.com/p/word2vec/

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
selectedCol	选中的列名	计算列对应的列名	String	✓	
alpha	学习率	学习率	Double		
minCount	最小词频	最小词频	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
numIter	迭代次数	迭代次数，默认为1。	Integer		
outputCol	输出结果列	输出结果列列名，可选，默认null	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predMethod	向量组合方法	预测文档向量时，需要用到的方法。支持三种方法：平均（avg），最小（min）和最大（max），默认值为平均	String		"AVG", "SUM", "MIN", "MAX"
randomWindow	是否使用随机窗口	是否使用随机窗口，默认使用	String		
reservedCols	算法保留列名	算法保留列	String[]		

vectorSize	embedding的向量长度	embedding的向量长度	Integer		[1, +inf)
window	窗口大小	窗口大小	Integer		
wordDelimiter	单词分隔符	单词之间的分隔符	String		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["A B C"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='tokens string')
word2vec = Word2Vec().setSelectedCol("tokens").setMinCount(1).setVectorSize(4)
word2vec.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;
```

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.nlp.Word2Vec;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class Word2VecTest {
    @Test
    public void testWord2Vec() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("A B C")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "tokens string");
        Word2Vec word2vec = new
Word2Vec().setSelectedCol("tokens").setMinCount(1).setVectorSize(4);
        word2vec.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

tokens	
0.731000431888515 0.40841702428161525 0.5173676180773374 0.3393047625647364	

Bert文本分类器 (BertTextClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.BertTextClassifier

Python 类名：BertTextClassifier

功能介绍

Bert 文本分类器。

参数说明

名称	中文名称	描述
labelCol	标签列名	输入表中的标签列名
predictionCol	预测结果列名	预测结果列名
textCol	文本列	文本列
batchSize	数据批大小	数据批大小
bertModelName	BERT模型名字	BERT模型名字： Base-Chinese,Base-Multilingual-Case
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estima` 为所有 worker 都能访问到的目录
customConfigJson	自定义参数	对应 https://github.com/alibaba/EasyTransfer/blob/master/eas 中的config_json
inferBatchSize	推理数据批大小	推理数据批大小
intraOpParallelism	Op 间并发度	Op 间并发度
learningRate	学习率	学习率
maxSeqLength	句子截断长度	句子截断长度

modelFilePath	模型的文件路径	模型的文件路径
numEpochs	epoch 数	epoch 数
numFineTunedLayers	微调层数	微调层数
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设（需要取整），否则为总并发度减去 Worker 角色数。
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设（需要取整），否则为总并发度减去 PS 角色数。
overwriteSink	是否覆写已有数据	是否覆写已有数据
predictionDetailCol	预测详细信息列名	预测详细信息列名
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文且目录名与压缩文件主文件名一致，可以使用 http://, http是目录，那么只能使用本地路径，即 file://。
removeCheckpointBeforeTraining	是否在训练前移除checkpoint相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只
reservedCols	算法保留列名	算法保留列
modelStreamFilePath	模型流的文件路径	模型流的文件路径
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-m见Timestamp.valueOf(String s)

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv"
schemaStr = "label bigint, review string"
```

```
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schemaStr) \
    .setIgnoreFirstLine(True)
data = data.where("review is not null")
data = ShuffleBatchOp().linkFrom(data)

classifier = BertTextClassifier() \
    .setTextCol("review") \
    .setLabelCol("label") \
    .setNumEpochs(0.01) \
    .setNumFineTunedLayers(1) \
    .setMaxSeqLength(128) \
    .setBertModelName("Base-Chinese") \
    .setPredictionCol("pred") \
    .setPredictionDetailCol("pred_detail")
model = classifier.fit(data)
predict = model.transform(data.firstN(300))
predict.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ShuffleBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.classification.BertClassificationModel;
import com.alibaba.alink.pipeline.classification.BertTextClassifier;
import org.junit.Test;

public class BertTextClassifierTest {
    @Test
    public void test() throws Exception {
        String url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv";
        String schemaStr = "label bigint, review string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schemaStr)
            .setIgnoreFirstLine(true);
        data = data.where("review is not null");
        data = new ShuffleBatchOp().linkFrom(data);

        BertTextClassifier classifier = new BertTextClassifier()
            .setTextCol("review")
            .setLabelCol("label")
            .setNumEpochs(0.01)
            .setNumFineTunedLayers(1)
```

```
            .setMaxSeqLength(128)
            .setBertModelName("Base-Chinese")
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail");
        BertClassificationModel model = classifier.fit(data);
        BatchOperator <?> predict = model.transform(data.firstN(300));
        predict.print();
    }
}
```

运行结果

label	review	pred	pred_detail
1	知道网线接口在哪儿吗？比高家庄的地道口还隐蔽。在床头柜后面！想不道吧？看你怎么用。1：自带4米以上网线；2：自带小板凳：3：房间的垃圾桶可翻过来坐.....	1	{"0":0.07187604904174805,"1":0.928123950958252}
1	1.酒店承诺接机，飞机本来应该在晚上10：00到，但是因为稍晚再加取行李，10：30分才出来，酒店的服务生已经在出口举牌等了，虽然只有我一个人，但仍很热心，态度很好的送我回了酒店。很感谢！（酒店在市中心，机场离市区蛮远，酒店的接机真的很方便。）2.酒店楼很高（比想象的），价格合理，很干净不知道官方是怎么评级的，个人觉得是不错的3星补充点评2008年6月23日：:-)已经1个多月了，想起来还是觉得酒店不错看来以后如果我机会再去HHHT，肯定还会住这里的	1	{"0":0.03521829843521118,"1":0.9647817015647888}
1	超赞！虽然窗正对中环，但一点也不觉得吵。房间很干净、整齐，给人很舒服的感觉，服务也很好，价格也不贵，非常满意！	1	{"0":0.162497878074646,"1":0.837502121925354}

1	吸取网友们的经验，我们预定的是度假区高层海景大床房，于6月21-23日入住。很高兴亚太的服务还不错，完全按照我们的要求安排了房间。服务员在走廊里遇到客人也会问好；晚上在酒店吃烧烤，服务也不错。总的来说，酒店的环境和海滨浴场都不错，泳池造型很美，水也清澈，晚上水池的灯光也很漂亮，可能是淡季的原因，海滨浴场人很少，特别安静，适合度假。美中不足是房间的设施比较陈旧，电视太小，风扇和空调都有噪音，电话机也很老旧，拖鞋看起来不太干净，好在我们自带了拖鞋。再有就是早餐要提出批评，虽然种类很多，但味道实在不敢恭维，作为一个老五星酒店，房间设施陈旧可以理解，更新起来成本过高，但早餐质量应该尽快改进，这样才能与星级相配。另外，北京的五星酒店都会在入住的第一天送水果，甚至蜈支洲岛上的观海木楼都送了水果，可惜亚太却没有，这也是对客人表示欢迎的一种服务吧，建议亚太考虑一下。总的来说，酒店还是很舒服，给我们留下了比较好的印象。	1	{"0":0.04930460453033447,"1":0.9506953954696655}

1	携程这次不错，订的行政房给我免费升级到了8号楼的山景房。对于酒店，总结一下：超5星的环境4星的房间3星的设施和服务酒店分栋依山而建，酒店里有专门的散步路线图，要把整个酒店的山景从容走完至少1小时，期间山水桥路，鸟叫虫鸣，空气清新，心旷神怡！8号楼是酒店最好的一栋楼了（价格最贵），进入房间，房间宽大，落地窗外就是山景翠色。但其大床一看就知不会很舒服，床很低，床垫质量不好，床上用品虽然干净但明显成色已经旧了灰扑扑的颜色，摸上去硬邦邦的，家具电器以及卫生间用品都很一般。68一位早餐品种还是比较丰富，吃饱没问题，精致美味就谈不上了。各处的服务员态度还是很好但服务意识和服务质量应提高，酒店有宽阔的停车场，不过无论住店客人均收费。宾馆反馈2008年7月10日：您好!感谢您对红珠山宾馆提出的宝贵意见,我宾馆现已经将布草全部更换,您的意见我们已经交给相关部门进行处理,我们诚恳地期待您的下次光临!	1	{"0":0.0854041576385498,"1":0.9145958423614502}
...	...	...	...

0	我这次是第5次住在长春的雁鸣湖大酒店。昨晚夜里停电。深夜我睡着了。我的钱包被内贼进入我的房间，偷了我近1000元和4张信用卡。。。我的证件和外币，数码相机等都在房间的保险箱里，原封不动。我打了好几个小时的长途电话来处理我的信用卡的冻结。我报案了，这个4星酒店的保安摄像探头竟然坏了，没有修理！保安还查房卡入门时间，就是没有其他人在深夜进入我的房间。难道内贼不会用其他高明的方式进入吗？我的羽绒服也被这个内贼放在地上！我醒来时没有多想！近中午时我才发觉钱包少了现金和信用卡！还有，这家酒店的态度很差！没有同情心！我之前授权的2000元，我打了国际电话，银行说两天前我入酒店的2000元授权了，可是酒店的财务不领情，说中国银行没有授权。我又打了国际电话，我的银行说通过了！这家4星级的酒店不负责，认为不可能发生，我报案了，我下次再也不住这个1星不到的服务态度，很可耻！我还要把这个事件说给那些想定这个酒店的住客。酒店为何停电，摄像头坏得也太凑巧了来让大家知道这种内贼行为是要强力打击的。好了，不说了！！！千元丢了小事。酒店的处理态度我很反感！我强力告诉大家和提醒其他人不要到该酒店！	1	{"0":0.025804638862609863,"1":0.9741953611373901}

1	该酒店对去溧阳公务或旅游的人都很适合，自助早餐很丰富，酒店内部环境和服务很好。唯一的不足是酒店大门口在晚上时太乱，各种车辆和人在门口挤成一团。补充点评2008年5月9日：房间淋浴水压不稳，一会热、一会冷，很不好调整。宾馆反馈2008年5月13日：非常感谢您选择入住金陵溧阳宾馆。您给予我们的肯定与赞赏让我们倍受鼓舞，也使我们更加自信地去做好每一天的服务工作。正是有许多像您一样的宾客给予我们不断的鼓励和赞赏，酒店的服务品质才能得以不断提升。对于酒店大门口的秩序和房间淋浴水的问题我们已做出了相应的措施。再次向您表示我们最衷心的感谢！我们期待您的再次光临！	1	{"0":0.09373688697814941,"1":0.9062631130218506}
1	房间设备太破,连喷头都是不好用,空调几乎感觉不到,虽然我开了最大另外就是设备维修不及时,洗澡用品感觉都是廉价货,味道很奇怪的洗头液等等...总体感觉服务还可以,设备招待所水平...	1	{"0":0.15379667282104492,"1":0.8462033271789551}
1	在桐乡是不错的选择，大堂很大。打车方便。	1	{"0":0.14478212594985962,"1":0.8552178740501404}
1	商务大床房，房间很大，床有2M宽，整体感觉经济实惠不错!	1	{"0":0.0821835994720459,"1":0.9178164005279541}

Bert文本对分类器 (BertTextPairClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.BertTextPairClassifier

Python 类名：BertTextPairClassifier

功能介绍

Bert 文本对分类器。

参数说明

名称	中文名称	描述
labelCol	标签列名	输入表中的标签列名
predictionCol	预测结果列名	预测结果列名
textCol	文本列	文本列
textPairCol	文本对列	文本对列
batchSize	数据批大小	数据批大小
bertModelName	BERT模型名字	BERT模型名字： Base-Chinese,Base-Multilingual-Case
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estima` 为所有 worker 都能访问到的目录
customConfigJson	自定义参数	对应 https://github.com/alibaba/EasyTransfer/blob/master/eas 中的config_json
inferBatchSize	推理数据批大小	推理数据批大小
intraOpParallelism	Op 间并发度	Op 间并发度
learningRate	学习率	学习率

maxSeqLength	句子截断长度	句子截断长度
modelFilePath	模型的文件路径	模型的文件路径
numEpochs	epoch 数	epoch 数
numFineTunedLayers	微调层数	微调层数
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设 （需要取整），否则为总并发度减去 Worker 角色数。
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设 （需要取整），否则为总并发度减去 PS 角色数。
overwriteSink	是否覆写已有数据	是否覆写已有数据
predictionDetailCol	预测详细信息列名	预测详细信息列名
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文 且目录名与压缩文件主文件名一致，可以使用 http://, http 是目录，那么只能使用本地路径，即 file://。
removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只
reservedCols	算法保留列名	算法保留列
modelStreamFilePath	模型流的文件路径	模型流的文件路径
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-m 见Timestamp.valueOf(String s)

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv"
schemaStr = "f_quality bigint, f_id_1 string, f_id_2 string, f_string_1 string,
f_string_2 string"
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schemaStr) \
    .setFieldDelimiter("\t") \
    .setIgnoreFirstLine(True) \
    .setQuoteChar(None)
data = ShuffleBatchOp().linkFrom(data)

classifier = BertTextPairClassifier() \

.setTextCol("f_string_1").setTextPairCol("f_string_2").setLabelCol("f_quality")
\
    .setNumEpochs(0.1) \
    .setMaxSeqLength(32) \
    .setNumFineTunedLayers(1) \
    .setBertModelName("Base-Uncased") \
    .setPredictionCol("pred") \
    .setPredictionDetailCol("pred_detail")
model = classifier.fit(data)
predict = model.transform(data.firstN(300))
predict.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ShuffleBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.classification.BertClassificationModel;
import com.alibaba.alink.pipeline.classification.BertTextClassifier;
import org.junit.Test;

public class BertTextClassifierTest {
    @Test
    public void test() throws Exception {
        String url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv";
        String schemaStr = "label bigint, review string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schemaStr)
            .setIgnoreFirstLine(true);
        data = data.where("review is not null");
        data = new ShuffleBatchOp().linkFrom(data);
```

```
        BertTextClassifier classifier = new BertTextClassifier()
            .setTextCol("review")
            .setLabelCol("label")
            .setNumEpochs(0.01)
            .setNumFineTunedLayers(1)
            .setMaxSeqLength(128)
            .setBertModelName("Base-Chinese")
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail");
        BertClassificationModel model = classifier.fit(data);
        BatchOperator <?> predict = model.transform(data.firstN(300));
        predict.print();
    }
}
```

运行结果

f_quality	f_id_1	f_id_2	f_string_1	f_string_2	pred	
0	218017	218035	Application Intelligence will be included as p...	The new application intelligence features will...	1	{"0":0.2033517360
1	1642169	1642368	The new 25-member Governing Council 's first m...	Its first decisions were to scrap all holidays...	1	{"0":0.2033517360
1	3399091	3399055	Also in Mosul , rebel gunmen on Friday assassi...	Near a mosque in the northern town of Mosul , ...	1	{"0":0.2033517360
0	2583299	2583319	" We 're still confident that Gephardt will ge...	Whether or not we get to the two-thirds , we '...	1	{"0":0.2033517360
1	1568540	1568627	Monday , the CIA said analysts concluded that ...	The CIA on Monday said voice and sound analyst...	1	{"0":0.2033517360

...	...	...	...	...	...	...
1	1805639	1805436	Printer maker Lexmark International Inc. spurt...	Other gainers included Lexmark , which rose $ ...	1	{"0":0.2367869615
1	2182211	2182122	It ended a diplomatic drought between the two ...	The contact between the delegations ended a di...	1	{"0":0.2367869615
1	774666	774871	An unclear number of people were killed and mo...	Four people were killed and 50 injured in the ...	1	{"0":0.2367869615
0	2582380	2582198	Shaklee spokeswoman Jenifer Thompson said the ...	Shaklee spokeswoman Jenifer Thompson referred ...	1	{"0":0.2367869615
1	427232	427141	After three months , Atkins dieters had lost a...	Three months into the study , the Atkins group...	1	{"0":0.2367869615

C45决策树分类 (C45)

Java 类名：com.alibaba.alink.pipeline.classification.C45

Python 类名：C45

功能介绍

- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		

reservedCols	算法保留列名	算法保留列	String[]		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')


C45()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()


C45()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()


StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.classification.C45;
import org.junit.Test;
```

```
import java.util.Arrays;
import java.util.List;

public class C45Test {
    @Test
    public void testC45() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(df, " f0 double,
f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource
            = new MemSourceStreamOp(df, " f0 double, f1 string, f2 int, f3 int,
label int");
        new C45()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(batchSource)
            .print();
        new C45()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}

4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

CART决策树分类 (Cart)

Java 类名：com.alibaba.alink.pipeline.classification.Cart

Python 类名：Cart

功能介绍

- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		

reservedCols	算法保留列名	算法保留列	String[]		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

Cart()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()

Cart()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.classification.Cart;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class CartTest {
    @Test
    public void testCart() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );
        BatchOperator <?> batchSource
            = new MemSourceBatchOp(df, " f0 double, f1 string, f2 int, f3 int,
label int");
        StreamOperator <?> streamSource
            = new MemSourceStreamOp(df, " f0 double, f1 string, f2 int, f3 int,
label int");
        new Cart()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(batchSource)
            .print();
        new Cart()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

决策树分类器 (DecisionTreeClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.DecisionTreeClassifier

Python 类名：DecisionTreeClassifier

功能介绍

- 决策树支持多种树模型
- id3，cart，c4.5
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		

maxDepth	树的深度限制	树的深度限制	Integer		
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		

predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
reservedCols	算法保留列名	算法保留列	String[]		
treeType	模型中树的类型	模型中树的类型，三种选项可选：树为一种方式 gini, infoGain, infoGainRatio	String		"GINI", "INFOGAIN", "INFOGAINRAT
weightCol	权重列名	权重列对应的列名	String		所选列类型为 [BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE FLOAT, INTEGE LONG, SHORT
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

DecisionTreeClassifier()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()

DecisionTreeClassifier()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
```

```
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.classification.DecisionTreeClassifier;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DecisionTreeClassifierTest {
    @Test
    public void testDecisionTreeClassifier() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(
            df, " f0 double, f1 string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource
            = new MemSourceStreamOp(df, " f0 double, f1 string, f2 int, f3 int,
label int");
        new DecisionTreeClassifier()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(batchSource)
            .print();
        new DecisionTreeClassifier()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}

2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

FM分类 (FmClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.FmClassifier

Python 类名：FmClassifier

功能介绍

FM即因子分解机（Factor Machine），它的特点是考虑了特征之间的相互作用，是一种非线性模型。该组件使用FM模型解决分类问题。

算法原理

FM模型是线性模型的升级，是在线性表达式后面加入了新的交叉项特征及对应的权值，FM模型的表达式如下所示：

$$y = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

这里我们使用 Adagrad 优化算法求解该模型。算法原理细节可以参考文献[1]。

算法使用

FM算法是推荐领域被验证的效果较好的推荐方案之一，在电商、广告、视频、信息流、游戏的推荐领域有广泛应用。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献

[1] S. Rendle, "Factorization Machines," 2010 IEEE International Conference on Data Mining, 2010, pp. 995-1000, doi: 10.1109/ICDM.2010.127.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	

predictionCol	预测结果列名	预测结果列名	String	✓	
batchSize	迭代数据batch size	数据batch size	Integer		
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
initStdev	初始化参数的标准差	初始化参数的标准差	Double		
lambda0	常数项正则化系数	常数项正则化系数	Double		
lambda1	线性项正则化系数	线性项正则化系数	Double		
lambda2	二次项正则化系数	二次项正则化系数	Double		
learnRate	学习率	学习率	Double		
modelFilePath	模型的文件路径	模型的文件路径	String		
numEpochs	epoch数	epoch数	Integer		
numFactor	因子数	因子数	Integer		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
reservedCols	算法保留列名	算法保留列	String[]		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL BIGINTEGER BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
withIntercept	是否有常数项	是否有常数项，默认true	Boolean		
withLinearItem	是否含有线性项	是否含有线性项	Boolean		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
       ["1:1.1 3:2.0", 1.0],
       ["2:2.1 10:3.1", 1.0],
       ["1:1.2 5:3.2", 0.0],
       ["3:1.2 7:4.2", 0.0]
])

input = BatchOperator.fromDataframe(df, schemaStr='kv string, label double')
test = StreamOperator.fromDataframe(df, schemaStr='kv string, label double')
# load data
fm =
FmClassifier().setVectorCol("kv").setLabelCol("label").setPredictionCol("pred")
model = fm.fit(input)
model.transform(test).print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.classification.FmClassifier;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmClassifierTest {
    @Test
    public void testFmClassifier() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:1.1 3:2.0", 1.0),
            Row.of("2:2.1 10:3.1", 1.0),
            Row.of("1:1.2 5:3.2", 0.0),
            Row.of("3:1.2 7:4.2", 0.0)
        );
        BatchOperator <?> input = new MemSourceBatchOp(df, "kv string, label
double");
        StreamOperator <?> test = new MemSourceStreamOp(df, "kv string, label
double");
        FmClassifier fm = new
FmClassifier().setVectorCol("kv").setLabelCol("label").setPredictionCol("pred")
;

        fm.fit(input)
            .transform(test)
            .print();

        StreamOperator.execute();
    }
}
```

运行结果

kv	label	pred
1:1.1 3:2.0	1.0	1.0
2:2.1 10:3.1	1.0	1.0
1:1.2 5:3.2	0.0	0.0
3:1.2 7:4.2	0.0	0.0

GBDT分类器 (GbdtClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.GbdtClassifier

Python 类名：GbdtClassifier

功能介绍

- gbdt(Gradient Boosting Decision Trees)二分类，是经典的基于boosting的有监督学习模型，可以用来解决二分类问题
- 支持连续特征和离散特征
- 支持数据采样和特征采样
- 目标分类必须是两个

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
criteria	树分裂的策略	树分裂的策略，可以为PAI, XGBOOST	String		"PAI", "XGBOOST
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
featureImportanceType	特征重要性类型	特征重要性类型（默认为GAIN）	String		"WEIGHT", "GAIN", "COVER"
featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double		

gamma	xgboost中的l2正则项	xgboost中的l2正则项	Double		
lambda	xgboost中的l1正则项	xgboost中的l1正则项	Double		
learningRate	学习率	学习率（默认为0.3）	Double		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
minSumHessianPerLeaf	叶子节点最小Hessian值	叶子节点最小Hessian值（默认为0）	Double		
modelFilePath	模型的文件路径	模型的文件路径	String		
newtonStep	是否使用二阶梯度	是否使用二阶梯度	Boolean		

numTrees	模型中树的棵数	模型中树的棵数	Integer		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
reservedCols	算法保留列名	算法保留列	String[]		
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型[BIGDECIMBIGINTEGEBYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

参数建议

对于训练效果来说，比较重要的参数是 树的棵树+学习率、叶子节点最小样本数、单颗树最大深度、特征采样比例。

单个离散特征的取值种类数不能超过256，否则会出错。

代码示例

```
df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

GbdtClassifier()\
    .setLearningRate(1.0)\
    .setNumTrees(3)\
    .setMinSamplesPerLeaf(1)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()

GbdtClassifier()\
    .setLearningRate(1.0)\
    .setNumTrees(3)\
    .setMinSamplesPerLeaf(1)\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
```

```
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()

StreamOperator.execute()
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":0.9849144946061075,"1":0.01508550539389248}
2.0000	B	1	1	0	0	{"0":0.9849144946061075,"1":0.01508550539389248}
3.0000	C	2	2	1	1	{"0":0.015085505393892529,"1":0.9849144946061075}
4.0000	D	3	3	1	1	{"0":0.015085505393892529,"1":0.9849144946061075}

ID3决策树分类 (Id3)

Java 类名：com.alibaba.alink.pipeline.classification.Id3

Python 类名：Id3

功能介绍

- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		

reservedCols	算法保留列名	算法保留列	String[]		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

Id3()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()

Id3()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.classification.Id3;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class Id3Test {
    @Test
    public void testId3() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource
            = new MemSourceBatchOp(df, " f0 double, f1 string, f2 int, f3 int,
label int");
        StreamOperator <?> streamSource
            = new MemSourceStreamOp(df, " f0 double, f1 string, f2 int, f3 int,
label int");

        new Id3()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(batchSource)
            .print();
        new Id3()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}
3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}

4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

KerasSequential分类器 (KerasSequentialClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.KerasSequentialClassifier

Python 类名：KerasSequentialClassifier

功能介绍

构建一个 Keras 的 Sequential 模型， 训练分类模型。

通过 layers 参数指定构成 Sequential 模型的网络层，Alink 会自动在最开始添加 Input 层，在最后添加 Dense 层和激活层，得到完整的模型用于训练。

指定 layers 参数时，使用的是 Python 语句，例如

```
"Conv1D(256, 5, padding='same', activation='relu')",
"Conv1D(128, 5, padding='same', activation='relu')",
"Dropout(0.1)",
"MaxPooling1D(pool_size=8)",
"Conv1D(128, 5, padding='same', activation='relu')",
"Conv1D(128, 5, padding='same', activation='relu')",
"Flatten()"
```

`tf.keras.layers` 内的网络层已经提前 import，可以直接使用。 使用的 TensorFlow 版本是 2.3.1。

参数说明

名称	中文名称	描述	类型	是
labelCol	标签列名	输入表中的标签列名	String	✓
layers	各 layer 的描述	各 layer 的描述，使用 Python 语法，例如 "Conv1D(256, 5, padding='same', activation='relu')"	String[]	✓
predictionCol	预测结果列名	预测结果列名	String	✓
tensorCol	tensor列	tensor列	String	✓
batchSize	数据批大小	数据批大小	Integer	

bestMetric	最优指标	判断模型最优时用的指标，仅在总并发度为 1 时起作用。都支持的有：loss； 二分类还支持：auc, precision, recall, binary_accuracy, false_negatives, false_positives, true_negatives, true_positives；多分类还支持：sparse_categorical_accuracy；回归还支持：mean_absolute_error, mean_absolute_percentage_error, mean_squared_error, mean_squared_logarithmic_error, root_mean_squared_error	String	
checkpointFilePath	保存 checkpoint 的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estimator` 的 `model_dir` 传入，需要为所有 worker 都能访问到的目录	String	
inferBatchSize	推理数据批大小	推理数据批大小	Integer	
intraOpParallelism	Op 间并发度	Op 间并发度	Integer	
learningRate	学习率	学习率	Double	
modelFilePath	模型的文件路径	模型的文件路径	String	
numEpochs	epoch数	epoch数	Integer	
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer	
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer	
optimizer	优化器	优化器，使用 Python 语法，例如 "Adam(learning_rate=0.1)"	String	
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean	

predictionDetailCol	预测详细信息列名	预测详细信息列名	String	
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String	
removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只会删除必要的文件	Boolean	
reservedCols	算法保留列名	算法保留列	String[]	
saveBestOnly	是否导出最优的 checkpoint	是否导出最优的 checkpoint，仅在总并发度为 1 时生效	Boolean	
saveCheckpointsEpochs	每隔多少 epochs 保存 checkpoints	每隔多少 epochs 保存 checkpoints	Double	
saveCheckpointsSecs	每隔多少秒保存 checkpoints	每隔多少秒保存 checkpoints	Double	
validationSplit	验证集比例	验证集比例，仅在总并发度为 1 时生效	Double	
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String	
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer	
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String	

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
source = CsvSourceBatchOp() \
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/random_tensor.csv") \
    .setSchemaStr("tensor string, label int")

source = ToTensorBatchOp() \
    .setSelectedCol("tensor") \
    .setTensorDataType("DOUBLE") \
    .setTensorShape([200, 3]) \
    .linkFrom(source)

trainer = KerasSequentialClassifier() \
    .setTensorCol("tensor") \
    .setLabelCol("label") \
    .setLayers([
        "Conv1D(256, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Dropout(0.1)",
        "MaxPooling1D(pool_size=8)",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Flatten()"
    ]) \
    .setOptimizer("Adam()") \
    .setNumEpochs(1) \
    .setPredictionCol("pred") \
    .setPredictionDetailCol("pred_detail") \
    .setReservedCols(["label"])

model = trainer.fit(source)
prediction = model.transform(source)
prediction.lazyPrint(10)
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ToTensorBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import
com.alibaba.alink.pipeline.classification.KerasSequentialClassificationModel;
import com.alibaba.alink.pipeline.classification.KerasSequentialClassifier;
import org.junit.Test;

public class KerasSequentialClassifierTest {
```

```
    @Test
    public void testKerasSequentialClassifier() throws Exception {
        BatchOperator<?> source = new CsvSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/random_tensor.csv")
            .setSchemaStr("tensor string, label int");

        source = new ToTensorBatchOp()
            .setSelectedCol("tensor")
            .setTensorDataType("DOUBLE")
            .setTensorShape(200, 3)
            .linkFrom(source);

        KerasSequentialClassifier trainer = new KerasSequentialClassifier()
            .setTensorCol("tensor")
            .setLabelCol("label")
            .setLayers(new String[] {
                "Conv1D(256, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Dropout(0.1)",
                "MaxPooling1D(pool_size=8)",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Flatten()"
            })
            .setOptimizer("Adam()")
            .setNumEpochs(1)
            .setPredictionCol("pred")
            .setPredictionDetailCol("pred_detail")
            .setReservedCols("label");

        KerasSequentialClassificationModel model = trainer.fit(source);
        BatchOperator <?> prediction = model.transform(source);
        prediction.lazyPrint(10);
        BatchOperator.execute();
    }
}
```

运行结果

label	pred	pred_detail
0	0	{"0":0.636155836712713,"1":0.36384416328728697}
1	0	{"0":0.6334926095655181,"1":0.3665073904344819}
1	0	{"0":0.6381823204965642,"1":0.3618176795034358}

1	0	{"0":0.6376416296248051,"1":0.362358370375195}
1	0	{"0":0.6345856985385896,"1":0.36541430146141035}
1	0	{"0":0.6357593109428179,"1":0.364240689057182}
0	0	{"0":0.6404387449594703,"1":0.3595612550405296}
1	0	{"0":0.6372702905549685,"1":0.36272970944503136}
0	0	{"0":0.635502012172225,"1":0.36449798782777487}
0	0	{"0":0.6262401788033837,"1":0.37375982119661644}

最近邻分类 (KnnClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.KnnClassifier

Python 类名：KnnClassifier

功能介绍

KNN (K Nearest Neighbor）是一种分类算法。 KNN算法的核心思想是如果一个样本在特征空间中的k个最相邻的样本中的大多数属于某一个类别; 则该样本也属于这个类别，并具有这个类别上样本的特性。

KNN的训练与一般机器学习模型的训练过程不同：在KNN训练中我们只进行一些字典表的预处理，而在预测过程中才会进行计算预测每个数据点的类别。 因此，KNN的训练和预测通常同时使用，一般不单独使用。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE"
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
k	topK	topK	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		

predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
reservedCols	算法保留列名	算法保留列	String[]		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)
```

```
df = pd.DataFrame([
  [1, "0,0,0"],
  [1, "0.1,0.1,0.1"],
  [1, "0.2,0.2,0.2"],
  [0, "9,9,9"],
  [0, "9.1,9.1,9.1"],
  [0, "9.2,9.2,9.2"]
])

dataSource = BatchOperator.fromDataframe(df, schemaStr="label int, vec string")
knn = KnnClassifier().setVectorCol("vec") \
    .setPredictionCol("pred") \
    .setLabelCol("label") \
    .setK(3)

model = knn.fit(dataSource)
model.transform(dataSource).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.classification.KnnClassifier;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KnnClassifierTest {
    @Test
    public void testKnnClassifier() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1, "0,0,0"),
            Row.of(1, "0.1,0.1,0.1"),
            Row.of(1, "0.2,0.2,0.2"),
            Row.of(0, "9,9,9"),
            Row.of(0, "9.1,9.1,9.1"),
            Row.of(0, "9.2,9.2,9.2")
        );
        BatchOperator <?> dataSource = new MemSourceBatchOp(df, "label int, vec
string");
        KnnClassifier knn = new KnnClassifier().setVectorCol("vec")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setK(3);
```

```
        knn.fit(dataSource)
            .transform(dataSource)
            .print();
    }
}
```

运行结果

label	vec	pred
1	0,0,0	1
1	0.1,0.1,0.1	1
1	0.2,0.2,0.2	1
0	9,9,9	0
0	9.1,9.1,9.1	0
0	9.2,9.2,9.2	0

线性支持向量机 (LinearSvm)

Java 类名：com.alibaba.alink.pipeline.classification.LinearSvm

Python 类名：LinearSvm

功能介绍

线性SVM算法是经典的二分类算法，通过对打标签样本集合训练得到模型，使用模型预测样本的标签。逻辑回归组件支持稀疏、稠密两种数据格式。

算法原理

SVM使用铰链损失函数（hinge loss）计算经验风险（empirical risk）并在求解系统中加入了正则化项以优化结构风险（structural risk）， 是一个具有稀疏性和稳健性的分类器。

算法使用

SVM在各领域的模式识别问题中有应用，包括人像识别、文本分类、手写字符识别、生物信息学等。

文献

[1] Vapnik, V．Statistical learning theory. 1998 (Vol. 3). . New York, NY：Wiley，1998：Chapter 10-11, pp.401-492.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	

epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
l1	L1 正则化系数	L1 正则化系数，默认为 0。	Double		[0.0, +inf)
l2	正则化系数	L2 正则化系数，默认为 0。	Double		[0.0, +inf)
maxIter	最大迭代步数	最大迭代步数，默认为 100	Integer		[1, +inf)
modelFilePath	模型的文件路径	模型的文件路径	String		
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
reservedCols	算法保留列名	算法保留列	String[]		
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]

withIntercept	是否有常数项	是否有常数项，默认true	Boolean		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 2]
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int,
label int')

colnames = ["f0","f1"]
svm =
LinearSvm().setFeatureCols(colnames).setLabelCol("label").setPredictionCol("pre
d")
model = svm.fit(batchData)
model.transform(batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.classification.LinearSvm;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LinearSvmTest {
    @Test
    public void testLinearSvm() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
```

```
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 2)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data, "f0 int, f1
int, label int");
        LinearSvm svm = new LinearSvm().setFeatureCols("f0",
"f1").setLabelCol("label").setPredictionCol("pred");
        svm.fit(batchData)
            .transform(batchData)
            .print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	2	2

逻辑回归 (LogisticRegression)

Java 类名：com.alibaba.alink.pipeline.classification.LogisticRegression

Python 类名：LogisticRegression

功能介绍

逻辑回归算法是经典的二分类算法，通过对打标签样本集合训练得到模型，使用模型预测样本的标签。逻辑回归组件支持稀疏、稠密两种数据格式。

算法原理

面对二分类问题，建立代价函数，然后通过优化方法迭代求解出最优的模型参数，然后测试验证我们这个求解的模型的好坏。Logistic回归虽然名字里带“回归”， 但是它实际上是一种分类方法，主要用于二分类问题（即输出只有两种，分别代表两个类别）回归模型中，y是一个定性变量，比如y=0或1， logistic方法主要应用于研究某些事件发生的概率。

算法使用

常用于数据挖掘，疾病自动诊断，经济预测等领域。例如，探讨引发疾病的危险因素，并根据危险因素预测疾病发生的概率等。以心脏病病情分析为例，选择两组人群， 一组是心脏病组，一组是非心脏病组，两组人群必定具有不同的属性及身体指标。因此因变量就为是否有心脏病，值为“是”或“否”，自变量就可以包括很多了， 如年龄、性别、最大心跳数、血压、胆固醇、空腹血糖等。自变量既可以是连续的，也可以是分类的。然后通过logistic回归分析，可以得到自变量的权重， 从而可以大致了解到底哪些因素是心脏病的危险因素。同时根据该权值可以根据危险因素预测一个人心脏病的可能性。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Wright, R. E. (1995). Logistic regression. In L. G. Grimm & P. R. Yarnold (Eds.), Reading and understanding multivariate statistics (pp. 217–244). American Psychological Association. [2] https://baike.baidu.com/item/logistic%E5%9B%9E%E5%BD%92

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围

labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)

modelFilePath	模型的文件路径	模型的文件路径	String		
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
reservedCols	算法保留列名	算法保留列	String[]		
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		

weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
withIntercept	是否有常数项	是否有常数项，默认true	Boolean		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 2]
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int,
label int')

colnames = ["f0","f1"]
lr =
LogisticRegression().setFeatureCols(colnames).setLabelCol("label").setPredictio
nCol("pred")
model = lr.fit(batchData)
model.transform(batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
```

```
import com.alibaba.alink.pipeline.classification.LogisticRegression;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LogisticRegressionTest {
    @Test
    public void testLogisticRegression() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 2)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data, "f0 int, f1
int, label int");
        LogisticRegression lr = new LogisticRegression().setFeatureCols("f0",
"f1").setLabelCol("label")
            .setPredictionCol("pred");
        lr.fit(batchData)
            .transform(batchData)
            .print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	2	2

多层感知机分类 (MultilayerPerceptronClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.MultilayerPerceptronClassifier

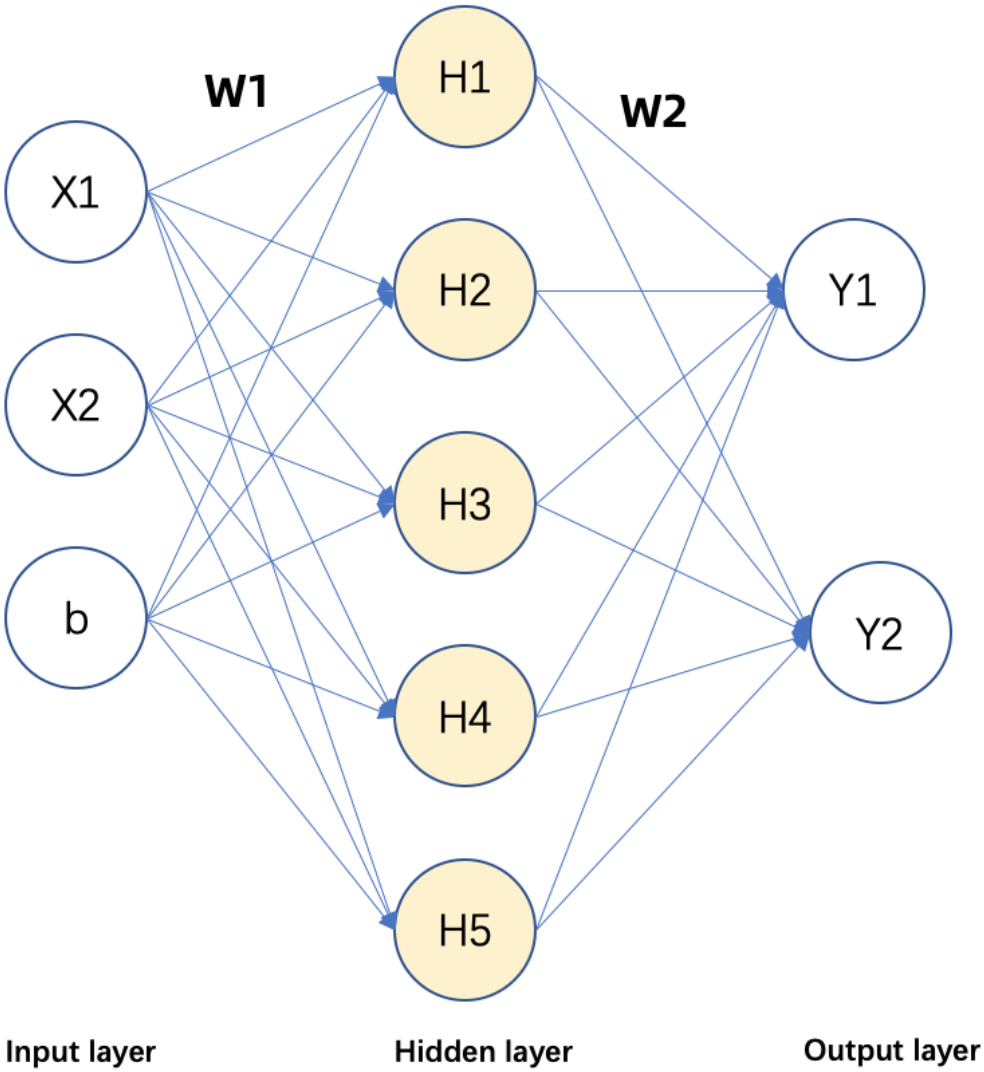
Python 类名：MultilayerPerceptronClassifier

功能介绍

多层感知机（MLP，Multilayer Perceptron）也被称作人工神经网络（ANN，Artificial Neural Network），经常用来进行多分类问题的训练预测。

算法原理

多层感知机算法除了输入输出层外，它中间可以有多个隐层，最简单的MLP只含一个隐层，即三层的结构，如下图：



从上图可以看到，多层感知机层与层之间是全连接的。多层感知机最左边是输入层，中间是隐藏层，最后是输出层。 其中输出层对应的是各个分类标签，输出层 的每一个节点对应每一个标签的出现的概率。

算法使用

多层感知机主要用于多分类问题，类似文字识别，语音识别，文本分析等问题。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献

[1]Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences MW Gardner, SR Dorling - Atmospheric environment, 1998 - Elsevier.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默
labelCol	标签列名	输入表中的标签列名	String	✓		
layers	神经网络层大小	神经网络层大小	int[]	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
blockSize	数据分块大小，默认值64	数据分块大小，默认值64	Integer			6
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1 6
featureCols	特征列名数组	特征列名数组，默认全选	String[]			n
initialWeights	初始权重值	初始权重值	DenseVector			n
l1	L1 正则化系数	L1 正则化系数，默认为 0。	Double		[0.0, +inf)	0
l2	正则化系数	L2 正则化系数，默认为 0。	Double		[0.0, +inf)	0
maxIter	最大迭代步数	最大迭代步数，默认为 100	Integer		[1, +inf)	1

modelFilePath	模型的文件路径	模型的文件路径	String			n
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			fa
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			n
vectorCol	向量列名	向量列对应的列名，默认值是null	String			n
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			n
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			1
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			n

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [5,2,3.5,1,'Iris-versicolor'],
    [5.1,3.7,1.5,0.4,'Iris-setosa'],
    [6.4,2.8,5.6,2.2,'Iris-virginica'],
    [6,2.9,4.5,1.5,'Iris-versicolor'],
    [4.9,3,1.4,0.2,'Iris-setosa'],
    [5.7,2.6,3.5,1,'Iris-versicolor'],
    [4.6,3.6,1,0.2,'Iris-setosa'],
    [5.9,3,4.2,1.5,'Iris-versicolor'],
    [6.3,2.8,5.1,1.5,'Iris-virginica'],
    [4.7,3.2,1.3,0.2,'Iris-setosa'],
    [5.1,3.3,1.7,0.5,'Iris-setosa'],
    [5.5,2.4,3.8,1.1,'Iris-versicolor'],
])

data = BatchOperator.fromDataframe(df, schemaStr='sepal_length double,
sepal_width double, petal_length double, petal_width double, category string')

mlpc = MultilayerPerceptronClassifier() \
    .setFeatureCols(["sepal_length", "sepal_width", "petal_length",
"petal_width"]) \
    .setLabelCol("category") \
    .setLayers([4, 5, 3]) \
    .setMaxIter(20) \
    .setPredictionCol("pred_label") \
    .setPredictionDetailCol("pred_detail")

mlpc.fit(data).transform(data).firstN(4).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import
com.alibaba.alink.pipeline.classification.MultilayerPerceptronClassifier;
import org.junit.Test;
```

```
import java.util.Arrays;
import java.util.List;

public class MultilayerPerceptronClassifierTest {
    @Test
    public void testMultilayerPerceptronClassifier() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(5.0, 2.0, 3.5, 1.0, "Iris-versicolor"),
            Row.of(5.1, 3.7, 1.5, 0.4, "Iris-setosa"),
            Row.of(6.4, 2.8, 5.6, 2.2, "Iris-virginica"),
            Row.of(6.0, 2.9, 4.5, 1.5, "Iris-versicolor"),
            Row.of(4.9, 3.0, 1.4, 0.2, "Iris-setosa"),
            Row.of(5.7, 2.6, 3.5, 1.0, "Iris-versicolor"),
            Row.of(4.6, 3.6, 1.0, 0.2, "Iris-setosa"),
            Row.of(5.9, 3.0, 4.2, 1.5, "Iris-versicolor"),
            Row.of(6.3, 2.8, 5.1, 1.5, "Iris-virginica"),
            Row.of(4.7, 3.2, 1.3, 0.2, "Iris-setosa"),
            Row.of(5.1, 3.3, 1.7, 0.5, "Iris-setosa"),
            Row.of(5.5, 2.4, 3.8, 1.1, "Iris-versicolor")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df,
            "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string");
        MultilayerPerceptronClassifier mlpc = new
MultilayerPerceptronClassifier()
            .setFeatureCols("sepal_length", "sepal_width", "petal_length",
"petal_width")
            .setLabelCol("category")
            .setLayers(new int[] {4, 5, 3})
            .setMaxIter(20)
            .setPredictionCol("pred_label")
            .setPredictionDetailCol("pred_detail");
        mlpc.fit(data).transform(data).firstN(4).print();
    }
}
```

运行结果

sepal_length	sepal_width	petal_length	petal_width	category	p
5.0000	2.0000	3.5000	1.0000	Iris-versicolor	Iris-versicolor
5.1000	3.7000	1.5000	0.4000	Iris-setosa	Iris-versicolor
6.4000	2.8000	5.6000	2.2000	Iris-virginica	Iris-versicolor
6.0000	2.9000	4.5000	1.5000	Iris-versicolor	Iris-versicolor
4.9000	3.0000	1.4000	0.2000	Iris-setosa	Iris-versicolor

5.7000	2.6000	3.5000	1.0000	Iris-versicolor	Iris-versicolor
4.6000	3.6000	1.0000	0.2000	Iris-setosa	Iris-setosa
5.9000	3.0000	4.2000	1.5000	Iris-versicolor	Iris-versicolor
6.3000	2.8000	5.1000	1.5000	Iris-virginica	Iris-versicolor
4.7000	3.2000	1.3000	0.2000	Iris-setosa	Iris-versicolor
5.1000	3.3000	1.7000	0.5000	Iris-setosa	Iris-versicolor
5.5000	2.4000	3.8000	1.1000	Iris-versicolor	Iris-versicolor

朴素贝叶斯 (NaiveBayes)

Java 类名：com.alibaba.alink.pipeline.classification.NaiveBayes

Python 类名：NaiveBayes

功能介绍

训练一个朴素贝叶斯模型用于多分类任务。

算法原理

朴素贝叶斯算法基于贝叶斯定理和一个"朴素"的假设：各特征间两两条件独立。

通过贝叶斯定理可以在给定特征$(x*1,\cdots,x_n)$时计算类别为y的概率：*$P(y|x_1,\cdots,x_n)=\frac{P(y)P(x_1,\cdots,x_n|y)}{P(x_1,\cdots,x_n) }$，而通过特征间两两独立的假设可以将上面公式简化为：$P(y|x_1,\cdots,x_n)=\frac{P(y)\prod{i = 1}^{n}P(x_i|y)}{P(x_1,\cdots,x_n)}$。

对于连续型特征$x*i$，通常假设 $P(x_i|y)$满足高斯分布 (μ_y,σ_y)，参数可以通过对训练数据进行最大似然估计得到。对于离散型特征x_i，$P(x_i|y)=\frac{N{yi} + \alpha}{N*{y}+\alpha n}$，其中 $N{yi}$ 表示类别为 y特征x_i共同出现的样本数， N_y表示类别 y 的样本数，α 是平滑系数。

使用方式

为了训练朴素贝叶斯模型，需要指定参数特征列名（featureCols）和标签列名（labelCol）。 特征列名中，数值类型的列默认看作连续型特征处理，如果需要强制作为离散型特征处理，需要将这些列的列名添加到参数离散特征列名（categoricalCol）中。 平滑因子可以通过参数 smoothing 指定，默认为不平滑。

组件还支持设置每条样本的权重，通过参数权重列（weightCol）指定。

文献索引

H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	

labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		

reservedCols	算法保留列名	算法保留列	String[]		
smoothing	算法参数	光滑因子，默认为0.0	Double		[0.0, +inf)
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
       [1.0, 1.0, 0.0, 1.0, 1],
       [1.0, 0.0, 1.0, 1.0, 1],
       [1.0, 0.0, 1.0, 1.0, 1],
       [0.0, 1.0, 1.0, 0.0, 0],
       [0.0, 1.0, 1.0, 0.0, 0],
       [0.0, 1.0, 1.0, 0.0, 0],
       [0.0, 1.0, 1.0, 0.0, 0],
       [1.0, 1.0, 1.0, 1.0, 1],
       [0.0, 1.0, 1.0, 0.0, 0]
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='f0 double, f1
double, f2 double, f3 double, label int')

colnames = ["f0","f1","f2", "f3"]
```

```
ns = NaiveBayesTrainBatchOp().setFeatureCols(colnames).setLabelCol("label")
model = batchData.link(ns)

predictor = NaiveBayesPredictBatchOp().setPredictionCol("pred")
predictor.linkFrom(model, batchData).print()
colnames = ["f0","f1","f2", "f3"]
# pipeline model
ns =
NaiveBayes().setFeatureCols(colnames).setLabelCol("label").setPredictionCol("pr
ed")
model = ns.fit(batchData)
model.transform(batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.classification.NaiveBayes;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class NaiveBayesTest {
    @Test
    public void testNaiveBayes() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1.0, 1.0, 0.0, 1.0, 1),
            Row.of(1.0, 0.0, 1.0, 1.0, 1),
            Row.of(1.0, 0.0, 1.0, 1.0, 1),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(0.0, 1.0, 1.0, 0.0, 0),
            Row.of(1.0, 1.0, 1.0, 1.0, 1),
            Row.of(0.0, 1.0, 1.0, 0.0, 0)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data,
            "f0 double, f1 double, f2 double, f3 double, label int");
        NaiveBayes ns = new NaiveBayes()
            .setFeatureCols("f0", "f1", "f2", "f3")
            .setLabelCol("label")
            .setPredictionCol("pred");

        ns.fit(batchData)
```

```
            .transform(batchData)
            .print();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred
1.0	1.0	0.0	1.0	1	1
1.0	0.0	1.0	1.0	1	1
1.0	0.0	1.0	1.0	1	1
0.0	1.0	1.0	0.0	0	0
0.0	1.0	1.0	0.0	0	0
0.0	1.0	1.0	0.0	0	0
0.0	1.0	1.0	0.0	0	0
1.0	1.0	1.0	1.0	1	1
0.0	1.0	1.0	0.0	0	0

朴素贝叶斯文本分类器 (NaiveBayesTextClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.NaiveBayesTextClassifier

Python 类名：NaiveBayesTextClassifier

功能介绍

训练一个朴素贝叶斯文本分类模型用于多分类任务。

算法原理

朴素贝叶斯算法基于贝叶斯定理和一个"朴素"的假设：各特征间两两条件独立。

通过贝叶斯定理可以在给定特征$(x*1,\cdots,x_n)$时计算类别为y的概率：*$P(y|x_1,\cdots,x_n)=\frac{P(y)P(x_1,\cdots,x_n|y)}{P(x_1,\cdots,x_n) }$，而通过特征间两两独立的假设可以将上面公式简化为：$P(y|x_1,\cdots,x_n)=\frac{P(y)\prod{i = 1}^{n}P(x_i|y)}{P(x_1,\cdots,x_n)}$。

在朴素贝叶斯用于文本分类时，文本中的词语（token）对应一个特征。 类别c的概率可以估算为 $\hat P(c)=\frac{Nc}{N}$，*其中 N_c 是类别为c的总文本数，N的总文本数。 词语t_i在类别c所包含的文本出现的频次用 $T{cti}$表示，那么可以用于估算概率 $\hat P(t|c)= \frac{T{cti}}{\sum{t'\in V}T_{ct'}}$。*

与一般的朴素贝叶斯分类模型类似，可以添加平滑系数来解决T_{ct_i} 为 0 时的问题。

上面描述的是多项分布时的模型，即$T*{ct_i}$ 可以去大于1的值。如果考虑的是二项分布，那么$T*{ct_i}$只能取值0或者1。

使用方式

为了训练朴素贝叶斯模型，需要指定参数向量列名（vectorCol）和标签列名（labelCol）。 通过参数模型类型可以设置使用多项分布或者二项分布。 平滑因子可以通过参数 smoothing 指定，默认为不平滑。

组件还支持设置每条样本的权重，通过参数权重列（weightCol）指定。

文献索引

Naive Bayes text classification: https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围

labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
vectorCol	向量列名	向量列对应的列名	String	✓	
modelFilePath	模型的文件路径	模型的文件路径	String		
modelType	模型类型	取值为 Multinomial 或 Bernoulli	String		"Multinomial", "Bernoulli"
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		

reservedCols	算法保留列名	算法保留列	String[]		
smoothing	算法参数	光滑因子，默认为1.0	Double		[0.0, +inf)
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
          ["$31$0:1.0 1:1.0 2:1.0 30:1.0","1.0  1.0  1.0  1.0", '1'],
          ["$31$0:1.0 1:1.0 2:0.0 30:1.0","1.0  1.0  0.0  1.0", '1'],
          ["$31$0:1.0 1:0.0 2:1.0 30:1.0","1.0  0.0  1.0  1.0", '1'],
          ["$31$0:1.0 1:0.0 2:1.0 30:1.0","1.0  0.0  1.0  1.0", '1'],
          ["$31$0:0.0 1:1.0 2:1.0 30:0.0","0.0  1.0  1.0  0.0", '0'],
          ["$31$0:0.0 1:1.0 2:1.0 30:0.0","0.0  1.0  1.0  0.0", '0'],
          ["$31$0:0.0 1:1.0 2:1.0 30:0.0","0.0  1.0  1.0  0.0", '0']
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='sv string, dv
string, label string')
# pipeline
model =
NaiveBayesTextClassifier().setVectorCol("sv").setLabelCol("label").setReservedC
```

```
ols(["sv", "label"]).setPredictionCol("pred")
model.fit(batchData).transform(batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.classification.NaiveBayesTextClassifier;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class NaiveBayesTextClassifierTest {
    @Test
    public void testNaiveBayesTextClassifier() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("$31$0:1.0 1:1.0 2:1.0 30:1.0", "1.0  1.0  1.0  1.0", "1"),
            Row.of("$31$0:1.0 1:1.0 2:0.0 30:1.0", "1.0  1.0  0.0  1.0", "1"),
            Row.of("$31$0:1.0 1:0.0 2:1.0 30:1.0", "1.0  0.0  1.0  1.0", "1"),
            Row.of("$31$0:1.0 1:0.0 2:1.0 30:1.0", "1.0  0.0  1.0  1.0", "1"),
            Row.of("$31$0:0.0 1:1.0 2:1.0 30:0.0", "0.0  1.0  1.0  0.0", "0"),
            Row.of("$31$0:0.0 1:1.0 2:1.0 30:0.0", "0.0  1.0  1.0  0.0", "0"),
            Row.of("$31$0:0.0 1:1.0 2:1.0 30:0.0", "0.0  1.0  1.0  0.0", "0")
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data, "sv string,
dv string, label string");
        NaiveBayesTextClassifier model = new
NaiveBayesTextClassifier().setVectorCol("sv").setLabelCol("label")
            .setReservedCols("sv", "label").setPredictionCol("pred");
        model.fit(batchData).transform(batchData).print();
    }
}
```

运行结果

sv	label	pred
"$31$0:1.0 1:1.0 2:1.0 30:1.0"	1	1
"$31$0:1.0 1:1.0 2:0.0 30:1.0"	1	1
"$31$0:1.0 1:0.0 2:1.0 30:1.0"	1	1
"$31$0:1.0 1:0.0 2:1.0 30:1.0"	1	1

"$31$0:0.0 1:1.0 2:1.0 30:0.0"	0	0
"$31$0:0.0 1:1.0 2:1.0 30:0.0"	0	0
"$31$0:0.0 1:1.0 2:1.0 30:0.0"	0	0

OneVsRest (OneVsRest)

Java 类名：com.alibaba.alink.pipeline.classification.OneVsRest

Python 类名：OneVsRest

功能介绍

本组件用One VS Rest策略进行多分类。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
numClass	类别数	多分类的类别数，必选	Integer	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

URL = "https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/iris.csv";
SCHEMA_STR = "sepal_length double, sepal_width double, petal_length double,
```

```
petal_width double, category string";
data = CsvSourceBatchOp().setFilePath(URL).setSchemaStr(SCHEMA_STR)

lr = LogisticRegression() \
    .setFeatureCols(["sepal_length", "sepal_width", "petal_length",
"petal_width"]) \
    .setLabelCol("category") \
    .setPredictionCol("pred_result") \
    .setMaxIter(100)

oneVsRest = OneVsRest().setClassifier(lr).setNumClass(3)
model = oneVsRest.fit(data)
model.setPredictionCol("pred_result").setPredictionDetailCol("pred_detail")
model.transform(data).print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.classification.LogisticRegression;
import com.alibaba.alink.pipeline.classification.OneVsRest;
import com.alibaba.alink.pipeline.classification.OneVsRestModel;
import org.junit.Test;

public class OneVsRestTest {
    @Test
    public void testOneVsRest() throws Exception {
        String URL = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/iris.csv";
        String SCHEMA_STR
            = "sepal_length double, sepal_width double, petal_length double,
petal_width double, category string";
        BatchOperator <?> data = new
CsvSourceBatchOp().setFilePath(URL).setSchemaStr(SCHEMA_STR);
        LogisticRegression lr = new LogisticRegression()
            .setFeatureCols("sepal_length", "sepal_width", "petal_length",
"petal_width")
            .setLabelCol("category")
            .setPredictionCol("pred_result")
            .setMaxIter(100);
        OneVsRest oneVsRest = new OneVsRest().setClassifier(lr).setNumClass(3);
        OneVsRestModel model = oneVsRest.fit(data);

model.setPredictionCol("pred_result").setPredictionDetailCol("pred_detail");
        model.transform(data).print();
    }
}
```

运行结果

sepal_length	sepal_width	petal_length	petal_width	category	pred_result	
6.7000	3.1000	4.4000	1.4000	Iris-versicolor	Iris-versicolor	{"Iris-versi virgir 5,"Iris 12}
5.4000	3.0000	4.5000	1.5000	Iris-versicolor	Iris-versicolor	{"Iris-versi virgir setos
5.4000	3.9000	1.7000	0.4000	Iris-setosa	Iris-setosa	{"Iris-versi virgir setos
5.0000	3.4000	1.6000	0.4000	Iris-setosa	Iris-setosa	{"Iris-versi virgir setos
5.6000	3.0000	4.5000	1.5000	Iris-versicolor	Iris-versicolor	{"Iris-versi virgir setos
...	...	...	...	...	...	...

感知机 (Perceptron)

Java 类名：com.alibaba.alink.pipeline.classification.Perceptron

Python 类名：Perceptron

功能介绍

感知机(perceptron)是二类分类的线性分类模型，其输入为实例的特征向量，输出为实例的类别。

算法原理

感知机对应于输入空间中将实例划分为正负两类的分离超平面，通过超平面将正负样本分离，属于判别模型。具体原理可参考文献。

算法使用

该算法经常应用二分类问题中，该算法支持稀疏和稠密两种输入样本。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Gallant, Stephen I. "Perceptron-based learning algorithms." IEEE Transactions on neural networks 1.2 (1990): 179-191.

[2] 李航. 统计学习方法[M]. 北京：清华大学出版社，2012。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	

epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)
modelFilePath	模型的文件路径	模型的文件路径	String		
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
reservedCols	算法保留列名	算法保留列	String[]		
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]

withIntercept	是否有常数项	是否有常数项，默认true	Boolean		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 2]
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int,
label int')
colnames = ["f0","f1"]
per =
Perceptron().setFeatureCols(colnames).setLabelCol("label").setPredictionCol("pr
ed")
model = per.fit(batchData)
model.transform(batchData).print()
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	2	2

随机森林分类器 (RandomForestClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.RandomForestClassifier

Python 类名：RandomForestClassifier

功能介绍

- 随机森林是一种常用的树模型，由于bagging的过程，可以避免过拟合
- 随机森林组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double		

maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		

modelFilePath	模型的文件路径	模型的文件路径	String		
numSubsetFeatures	每棵树的特征采样数目	每棵树的特征采样数目	Integer		
numTrees	模型中树的棵数	模型中树的棵数	Integer		[1, +inf)
numTreesOfGini	模型中Cart树的棵数	模型中Cart树的棵数	Integer		
numTreesOfInfoGain	模型中Id3树的棵数	模型中Id3树的棵数	Integer		
numTreesOfInfoGainRatio	模型中C4.5树的棵数	模型中C4.5树的棵数	Integer		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
reservedCols	算法保留列名	算法保留列	String[]		

subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr=' f0 double, f1 string, f2 int, f3 int, label int')

RandomForestClassifier()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()

RandomForestClassifier()\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.classification.RandomForestClassifier;
```

```
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RandomForestClassifierTest {
    @Test
    public void testRandomForestClassifier() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource
            = new MemSourceBatchOp(df, " f0 double, f1 string, f2 int, f3 int,
label int");
        StreamOperator <?> streamSource =
            new MemSourceStreamOp(df, " f0 double, f1 string, f2 int, f3 int,
label int");
        new RandomForestClassifier()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(batchSource)
            .print();
        new RandomForestClassifier()
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

f0	f1	f2	f3	label	pred	pred_detail
1.0000	A	0	0	0	0	{"0":1.0,"1":0.0}
2.0000	B	1	1	0	0	{"0":1.0,"1":0.0}

3.0000	C	2	2	1	1	{"0":0.0,"1":1.0}
4.0000	D	3	3	1	1	{"0":0.0,"1":1.0}

Softmax (Softmax)

Java 类名：com.alibaba.alink.pipeline.classification.Softmax

Python 类名：Softmax

功能介绍

Softmax算法是Logistic回归算法的推广，Logistic回归主要是用来处理二分类问题，而Softmax回归则是处理多分类问题。

算法原理

面对多分类问题，建立代价函数，然后通过优化方法迭代求解出最优的模型参数。具体原理可参考文献。

算法使用

该算法经常应用到多分类问题中，类似情感分析、手写字识别等问题都可以使用Softmax算法，该算法支持稀疏和稠密两种输入样本。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Brown, Peter F., et al. "Class-based n-gram models of natural language." Computational linguistics 18.4 (1992): 467-480.

[2] Goodman, Joshua. "Classes for fast maximum entropy training." 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221). Vol. 1. IEEE, 2001.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	

predictionCol	预测结果列名	预测结果列名	String	✓	
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)
modelFilePath	模型的文件路径	模型的文件路径	String		

optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
reservedCols	算法保留列名	算法保留列	String[]		
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		

weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
withIntercept	是否有常数项	是否有常数项，默认true	Boolean		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
       [2, 1, 1],
       [3, 2, 1],
       [4, 3, 2],
       [2, 4, 1],
       [2, 2, 1],
       [4, 3, 2],
       [1, 2, 1],
       [5, 3, 3]
])

batchData = BatchOperator.fromDataframe(df_data, schemaStr='f0 int, f1 int,
label int')
dataTest = batchData
colnames = ["f0","f1"]
softmax =
Softmax().setFeatureCols(colnames).setLabelCol("label").setPredictionCol("pred"
)
model = softmax.fit(batchData)

model.transform(dataTest).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
```

```
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.classification.Softmax;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SoftmaxTest {
    @Test
    public void testSoftmax() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1),
            Row.of(5, 3, 3)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df_data, "f0 int, f1
int, label int");
        Softmax softmax = new Softmax().setFeatureCols("f0",
"f1").setLabelCol("label").setPredictionCol("pred");
        softmax.fit(batchData).transform(batchData).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1
3	2	1	1
4	3	2	2
2	4	1	1
2	2	1	1
4	3	2	2
1	2	1	1
5	3	3	3

XGBoost二分类训练 (XGBoostClassifier)

Java 类名：com.alibaba.alink.pipeline.classification.XGBoostClassifier

Python 类名：XGBoostClassifier

功能介绍

XGBoost 组件是在开源社区的基础上进行包装，使功能和 PAI 更兼容，更易用。 XGBoost 算法在 Boosting 算法的基础上进行了扩展和升级，具有较好的易用性和鲁棒性，被广泛用在各种机器学习生产系统和竞赛领域。 当前支持分类，回归和排序。

参数说明

名称	中文名称	描述	类型	是否必须?	
labelCol	标签列名	输入表中的标签列名	String	✓	
numRound	树的棵树	树的棵树	Integer	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
alpha	L1 正则项	L1 正则项	Double		
baseScore	Base score	Base score	Double		
colSampleByLevel	每个树列采样	每个树列采样	Double		
colSampleByNode	每个结点列采样	每个结点采样	Double		
colSampleByTree	每个树列采样	每个树列采样	Double		
eta	学习率	学习率	Double		
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
gamma	结点分裂最小损失变化	节点分裂最小损失变化	Double		
growPolicy	GrowPolicy	GrowPolicy	String		"DE "LC
interactionConstraints	interaction constraints	interaction constraints	String		
lambda	L2 正则项	L2 正则项	Double		
maxBin	最大结点个数	最大结点个数	Integer		

maxDeltaStep	Delta step	Delta step	Double		
maxDepth	最大深度	最大深度	Integer		
maxLeaves	最大结点个数	最大结点个数	Integer		
minChildWeight	结点的最小权重	结点的最小权重	Double		
modelFilePath	模型的文件路径	模型的文件路径	String		
monotoneConstraints	monotone constraints	monotone constraints	String		
numClass	标签类别个数	标签类别个数， 多分类时有效	Integer		
objective	objective	objective	String		"BI "BI "BI "MU "MU
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
pluginVersion	插件版本号	插件版本号	String		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
processType	ProcessType	ProcessType	String		"DE
refreshLeaf	RefreshLeaf	RefreshLeaf	Integer		
reservedCols	算法保留列名	算法保留列	String[]		
runningMode	运行模式	XGBoost的运行模型，ICQ速度快，但使用内存多，TRIAVIAL速度略慢，但是节省内存，按照流式方式处理。由于训练数据本身在XGBoost运行时已经被缓存进内存，所以存两份和存一份数据的资源消耗和速度对比，还需要进一步的测试。	String		"IC
samplingMethod	采样方法	采样方法	String		"UN "GF
scalePosWeight	ScalePosWeight	ScalePosWeight	Double		

singlePrecisionHistogram	single precision histogram	single precision histogram	Boolean		
sketchEps	SketchEps	SketchEps	Double		
subSample	样本采样比例	样本采样比例	Double		
treeMethod	构建树的方法	构建树的方法	String		"AU "AP
updater	Updater	Updater	String		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    [0, 1, 1.1, 1.0],
    [1, -2, 0.9, 2.0],
    [0, 100, -0.01, 3.0],
    [1, -99, 0.1, 4.0],
    [0, 1, 1.1, 5.0],
    [1, -2, 0.9, 6.0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='y int, x1 int, x2 double, x3 double'
```

```
)

streamSource = StreamOperator.fromDataframe(
    df, schemaStr='y int, x1 int, x2 double, x3 double'
)

xgboostClassifier = XGBoostClassifier() \
    .setNumRound(1)\
    .setPluginVersion('1.5.1')\
    .setLabelCol('y')\
    .setPredictionDetailCol('pred_detail')\
    .setPredictionCol('pred')

xgboostClassifier.fit(batchSource).transform(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.classification.XGBoostClassifier;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class XGBoostClassifierTest {

    @Test
    public void testXGBoostClassifier() throws Exception {
        List <Row> data = Arrays.asList(
            Row.of(0, 1, 1.1, 1.0),
            Row.of(1, -2, 0.9, 2.0),
            Row.of(0, 100, -0.01, 3.0),
            Row.of(1, -99, 0.1, 4.0),
            Row.of(0, 1, 1.1, 5.0),
            Row.of(1, -2, 0.9, 6.0)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(data, "y int, x1
int, x2 double, x3 double");
        StreamOperator <?> streamSource = new MemSourceStreamOp(data, "y int,
```

```
x1 int, x2 double, x3 double");

        XGBoostClassifier xgBoostClassifier = new XGBoostClassifier()
            .setNumRound(1)
            .setPluginVersion("1.5.1")
            .setLabelCol("y")
            .setPredictionDetailCol("pred_detail")
            .setPredictionCol("pred");

        xgBoostClassifier.fit(batchSource).transform(streamSource).print();

        StreamOperator.execute();
    }
}
```

生存回归 (AftSurvivalRegression)

Java 类名：com.alibaba.alink.pipeline.regression.AftSurvivalRegression

Python 类名：AftSurvivalRegression

功能介绍

在生存分析领域，加速失效时间模型(accelerated failure time model,AFT 模型)可以作为比例风险模型的替代模型。生存回归组件支持稀疏、稠密两种数据格式。

算法原理

AFT模型将线性回归模型的建模方法引人到生存分析的领域， 将生存时间的对数作为反应变量，研究多协变量与对数生存时间之间的回归关系，在形式上，模型与一般的线性回归模型相似。对回归系数的解释也与一般的线性回归模型相似，较之Cox模型， AFT模型对分析结果的解释更加简单、直观且易于理解，并且可以预测个体的生存时间。

算法使用

生存回归分析是研究特定事件的发生与时间的关系的回归。这里特定事件可以是：病人死亡、病人康复、用户流失、商品下架等。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Wei, Lee-Jen. "The accelerated failure time model: a useful alternative to the Cox regression model in survival analysis." Statistics in medicine 11.14　15 (1992): 1871-1879.

[2] https://spark.apache.org/docs/latest/ml-classification-regression.html#survival-regression

名称	中文名称	描述	类型	是否必须?	取值范围	
censorCol	生存列名	生存列名	String	✓		
labelCol	标签列名	输入表中的标签列名	String	✓		

predictionCol	预测结果列名	预测结果列名	String	✓		
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)	1.0E-6
featureCols	特征列名数组	特征列名数组，默认全选	String[]			null
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)	0.0
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
modelFilePath	模型的文件路径	模型的文件路径	String			null

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
quantileProbabilities	分位数概率数组	分位数概率数组	double[]			[0.01,0.
reservedCols	算法保留列名	算法保留列	String[]			null
vectorCol	向量列名	向量列对应的列名，默认值是null	String			null
withIntercept	是否有常数项	是否有常数项，默认true	Boolean			true

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [1.218, 1.0, "1.560,-0.605"],
    [2.949, 0.0, "0.346,2.158"],
    [3.627, 0.0, "1.380,0.231"],
    [0.273, 1.0, "0.520,1.151"],
    [4.199, 0.0, "0.795,-0.226"]])

data = BatchOperator.fromDataframe(df, schemaStr="label double, censor double,
features string")

reg = AftSurvivalRegression()\
            .setVectorCol("features")\
            .setLabelCol("label")\
            .setCensorCol("censor")\
            .setPredictionCol("result")

pipeline = Pipeline().add(reg)
model = pipeline.fit(data)

model.save().lazyPrint(10)
model.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.Pipeline;
import com.alibaba.alink.pipeline.PipelineModel;
import com.alibaba.alink.pipeline.regression.AftSurvivalRegression;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AftSurvivalRegressionTest {
    @Test
    public void testAftSurvivalRegression() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.218, 1.0, "1.560,-0.605"),
            Row.of(2.949, 0.0, "0.346,2.158"),
            Row.of(3.627, 0.0, "1.380,0.231"),
            Row.of(0.273, 1.0, "0.520,1.151")
```

```
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "label double, censor
double, features string");
        AftSurvivalRegression reg = new AftSurvivalRegression()
            .setVectorCol("features")
            .setLabelCol("label")
            .setCensorCol("censor")
            .setPredictionCol("result");
        Pipeline pipeline = new Pipeline().add(reg);
        PipelineModel model = pipeline.fit(data);
        model.save().lazyPrint(10);
        model.transform(data).print();
    }
}
```

运行结果

模型

id	
-1	{"stages":"[{\"identifier\":null,\"params\":null,\"schemaIndices\":[1,0,2],\"colNames\":null,\"parent\":-1},{\"id
1	{"hasInterceptItem":"true","vectorCol":"\"features\"","modelName":"\"AFTSurvivalRegTrainBatchOp\"","la
1	{"featureColNames":null,"featureColTypes":null,"coefVector":{"data":[-29.487324590178716,24.4277301 [1.5843984652595493,3.6032723097911044,0.4,1.4794299745122195,0.9580954096270979,1.6,1.377

结果

label	censor	features	result
1.2180	1.0000	1.560,-0.605	1.6231
2.9490	0.0000	0.346,2.158	2933.1642
3.6270	0.0000	1.380,0.231	1524.2502
0.2730	1.0000	0.520,1.151	0.2706

Bert文本对回归 (BertTextPairRegressor)

Java 类名：com.alibaba.alink.pipeline.regression.BertTextPairRegressor

Python 类名：BertTextPairRegressor

功能介绍

Bert 文本对回归。

参数说明

名称	中文名称	描述
labelCol	标签列名	输入表中的标签列名
predictionCol	预测结果列名	预测结果列名
textCol	文本列	文本列
textPairCol	文本对列	文本对列
batchSize	数据批大小	数据批大小
bertModelName	BERT模型名字	BERT模型名字： Base-Chinese,Base-Multilingual-Case
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estima` 为所有 worker 都能访问到的目录
customConfigJson	自定义参数	对应 https://github.com/alibaba/EasyTransfer/blob/master/eas 中的config_json
inferBatchSize	推理数据批大小	推理数据批大小
intraOpParallelism	Op 间并发度	Op 间并发度
learningRate	学习率	学习率

maxSeqLength	句子截断长度	句子截断长度
modelFilePath	模型的文件路径	模型的文件路径
numEpochs	epoch 数	epoch 数
numFineTunedLayers	微调层数	微调层数
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设 （需要取整），否则为总并发度减去 Worker 角色数。
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设 （需要取整），否则为总并发度减去 PS 角色数。
overwriteSink	是否覆写已有数据	是否覆写已有数据
predictionDetailCol	预测详细信息列名	预测详细信息列名
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文 且目录名与压缩文件主文件名一致，可以使用 http://, http 是目录，那么只能使用本地路径，即 file://。
removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只
reservedCols	算法保留列名	算法保留列
modelStreamFilePath	模型流的文件路径	模型流的文件路径
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-m 见Timestamp.valueOf(String s)

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv"
schemaStr = "f_quality double, f_id_1 string, f_id_2 string, f_string_1 string,
f_string_2 string"
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schemaStr) \
    .setFieldDelimiter("\t") \
    .setIgnoreFirstLine(True) \
    .setQuoteChar(None)
data = ShuffleBatchOp().linkFrom(data)

regressor = BertTextPairRegressor() \

.setTextCol("f_string_1").setTextPairCol("f_string_2").setLabelCol("f_quality")
\
    .setNumEpochs(0.1) \
    .setMaxSeqLength(32) \
    .setNumFineTunedLayers(1) \
    .setBertModelName("Base-Uncased") \
    .setPredictionCol("pred")
model = regressor.fit(data)
predict = model.transform(data.firstN(300))
predict.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ShuffleBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.regression.BertRegressionModel;
import com.alibaba.alink.pipeline.regression.BertTextPairRegressor;
import org.junit.Test;

public class BertTextPairRegressorTest {
    @Test
    public void test() throws Exception {
        String url = "http://alink-algo-packages.oss-cn-hangzhou-
zmf.aliyuncs.com/data/MRPC/train.tsv";
        String schemaStr = "f_quality double, f_id_1 string, f_id_2 string,
f_string_1 string, f_string_2 string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schemaStr)
            .setFieldDelimiter("\t")
            .setIgnoreFirstLine(true)
            .setQuoteChar(null);
```

```
        data = new ShuffleBatchOp().linkFrom(data);

        BertTextPairRegressor regressor = new BertTextPairRegressor()

.setTextCol("f_string_1").setTextPairCol("f_string_2").setLabelCol("f_quality")
            .setNumEpochs(0.1)
            .setMaxSeqLength(32)
            .setNumFineTunedLayers(1)
            .setBertModelName("Base-Uncased")
            .setPredictionCol("pred");
        BertRegressionModel model = regressor.fit(data);
        BatchOperator <?> predict = model.transform(data.firstN(300));
        predict.print();
    }
}
```

运行结果

FIELD1	f_quality	f_id_1	f_id_2	f_string_1	f_string_2
0	1.0	2224622	2224687	The indictment ``strikes at one of the very to...	The newly unsealed 32-count indictment alleges...
1	0.0	58543	58516	The tech-heavy Nasdaq Stock Markets composite ...	The Nasdaq Composite index , full of technolog...
2	1.0	1048622	1048759	Rescue workers had to scrounge for boats to re...	Wagner and Wolford both said that rescue worke...
3	1.0	2733687	2733747	Asked about the controversy , Bloomberg said ,...	" I didn 't see either one today , but if they...
4	0.0	3270421	3270325	At 10 p.m. , Odette was centered about 295 mil...	At 10 p.m. Thursday , Odette was about 295 mil...
...	...	...	...	...	...
295	0.0	2187868	2187826	" They were brutally beaten , and it 's really...	" It 's a wonder that Porchia was the only dec...
296	0.0	257789	257726	Knight had 29 interceptions in his six years w...	Knight has 29 interceptions in six seasons , a...

297	1.0	3128160	3128278	It will appear in the next few weeks on the We...	Details of the research will appear in a futur...
298	1.0	1125898	1125882	Following the ATP 's notification by Hewitt 's...	Following ATP 's notification by Hewitt 's law...
299	0.0	3041807	3041719	Details of the research appear in the Nov. 5 i...	The results , published in the Journal of the ...

Bert文本回归 (BertTextRegressor)

Java 类名：com.alibaba.alink.pipeline.regression.BertTextRegressor

Python 类名：BertTextRegressor

功能介绍

Bert 文本回归。

参数说明

名称	中文名称	描述
labelCol	标签列名	输入表中的标签列名
predictionCol	预测结果列名	预测结果列名
textCol	文本列	文本列
batchSize	数据批大小	数据批大小
bertModelName	BERT模型名字	BERT模型名字： Base-Chinese,Base-Multilingual-Case
checkpointFilePath	保存checkpoint的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estima` 为所有 worker 都能访问到的目录
customConfigJson	自定义参数	对应 https://github.com/alibaba/EasyTransfer/blob/master/eas 中的config_json
inferBatchSize	推理数据批大小	推理数据批大小
intraOpParallelism	Op 间并发度	Op 间并发度
learningRate	学习率	学习率
maxSeqLength	句子截断长度	句子截断长度

modelFilePath	模型的文件路径	模型的文件路径
numEpochs	epoch 数	epoch 数
numFineTunedLayers	微调层数	微调层数
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未i （需要取整），否则为总并发度减去 Worker 角色数。
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未i （需要取整），否则为总并发度减去 PS 角色数。
overwriteSink	是否覆写已有数据	是否覆写已有数据
predictionDetailCol	预测详细信息列名	预测详细信息列名
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文 且目录名与压缩文件主文件名一致，可以使用 http://, http 是目录，那么只能使用本地路径，即 file://。
removeCheckpointBeforeTraining	是否在训练前移除checkpoint相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只
reservedCols	算法保留列名	算法保留列
modelStreamFilePath	模型流的文件路径	模型流的文件路径
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-m 见Timestamp.valueOf(String s)

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv"
schemaStr = "label double, review string"
```

```
data = CsvSourceBatchOp() \
    .setFilePath(url) \
    .setSchemaStr(schemaStr) \
    .setIgnoreFirstLine(True)
data = data.where("review is not null")
data = ShuffleBatchOp().linkFrom(data)

regressor = BertTextRegressor() \
    .setTextCol("review") \
    .setLabelCol("label") \
    .setNumEpochs(0.01) \
    .setNumFineTunedLayers(1) \
    .setMaxSeqLength(128) \
    .setBertModelName("Base-Chinese") \
    .setPredictionCol("pred")
model = regressor.fit(data)
predict = model.transform(data.firstN(300))
predict.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ShuffleBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.regression.BertRegressionModel;
import com.alibaba.alink.pipeline.regression.BertTextRegressor;
import org.junit.Test;

public class BertTextRegressorTest {
    @Test
    public void test() throws Exception {
        String url = "http://alink-test.oss-cn-beijing.aliyuncs.com/jiqi-
temp/tf_ut_files/ChnSentiCorp_htl_small.csv";
        String schemaStr = "label double, review string";
        BatchOperator <?> data = new CsvSourceBatchOp()
            .setFilePath(url)
            .setSchemaStr(schemaStr)
            .setIgnoreFirstLine(true);
        data = data.where("review is not null");
        data = new ShuffleBatchOp().linkFrom(data);

        BertTextRegressor regressor = new BertTextRegressor()
            .setTextCol("review")
            .setLabelCol("label")
            .setNumEpochs(0.01)
            .setNumFineTunedLayers(1)
            .setMaxSeqLength(128)
```

```
            .setBertModelName("Base-Chinese")
            .setPredictionCol("pred");
        BertRegressionModel model = regressor.fit(data);
        BatchOperator <?> predict = model.transform(data.firstN(300));
        predict.print();
    }
}
```

运行结果

	label	review	pred
0	0.0	出差入住的酒店,订了个三人间.房间没空调,冷得要死,而且被子很潮.火车站旁,步行可到.据说当...	0.711733
1	1.0	我已经住过两次了。比较满意，特别是接机服务，一个人也接，及时周到。宽带速度也比较快。但是没有...	0.861643
2	1.0	酒店的设施稍微老了一点，是四星级酒店的一个附楼。但是服务不错，而且酒店的价格非常吸引人。下次...	0.933909
3	1.0	价格比比较不错的酒店。这次免费升级了，感谢前台服务员。房子还好，地毯是新的，比上次的好些。早...	0.863195
4	1.0	前台小姑娘很有意思，经常入住，都熟了，呵呵	0.521080
...	...	...	...
113	1.0	1。酒店比较新，装潢和设施还不错，只是房间有些油漆味。2。早餐还可以，只是品种不是很多。3。...	0.710072
114	1.0	这次去北京，是要去北师大办事，所以特意留意了下附近的宾馆。住了两天，首先该宾馆很好找，离西四...	1.596990
115	0.0	1。我住的是靠马路的标准间。房间内设施简陋，并且的房间玻璃窗户外还有一层幕墙玻璃，而且不能打...	1.823668
116	1.0	房间很清洁，洗手间装修好。早餐非常丰盛，豆腐脑很好吃！交通很方便。隔音不够好，我们的邻居夜里...	0.972452
117	1.0	价格偏高,好象连云港这地方的酒店都偏贵.早饭不好.房间还不错,窗外风景还行.最重要是房间的窗...	0.744328

CART决策树回归 (CartReg)

Java 类名：com.alibaba.alink.pipeline.regression.CartReg

Python 类名：CartReg

功能介绍

- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		

maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
```

```
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

CartReg()\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()

CartReg()\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.regression.CartReg;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class CartRegTest {
    @Test
    public void testCartReg() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
```

```
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(df, "f0 double, f1
string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, "f0 double,
f1 string, f2 int, f3 int, label int");
        new CartReg()
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(batchSource)
            .print();
        new CartReg()
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred
3.0000	C	2	2	1	1.0000
1.0000	A	0	0	0	0.0000

4.0000	D	3	3	1	1.0000
2.0000	B	1	1	0	0.0000

决策树回归 (DecisionTreeRegressor)

Java 类名：com.alibaba.alink.pipeline.regression.DecisionTreeRegressor

Python 类名：DecisionTreeRegressor

功能介绍

- 本函数支持cart回归
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		

maxDepth	树的深度限制	树的深度限制	Integer		
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		

reservedCols	算法保留列名	算法保留列	String[]		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

DecisionTreeRegressor()\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()

DecisionTreeRegressor()\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.regression.DecisionTreeRegressor;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class DecisionTreeRegressorTest {
```

```
    @Test
    public void testDecisionTreeRegressor() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(df, "f0 double, f1
string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, "f0 double,
f1 string, f2 int, f3 int, label int");
        new DecisionTreeRegressor()
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(batchSource)
            .print();
        new DecisionTreeRegressor()
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred

1.0000	A	0	0	0	0.0000
4.0000	D	3	3	1	1.0000
3.0000	C	2	2	1	1.0000
2.0000	B	1	1	0	0.0000

FM回归 (FmRegressor)

Java 类名：com.alibaba.alink.pipeline.regression.FmRegressor

Python 类名：FmRegressor

功能介绍

FM即因子分解机（Factor Machine），它的特点是考虑了特征之间的相互作用，是一种非线性模型。该组件使用FM模型解决回归问题。

算法原理

FM模型是线性模型的升级，是在线性表达式后面加入了新的交叉项特征及对应的权值，FM模型的表达式如下所示：

$$y = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij} x_i x_j$$

这里我们使用 Adagrad 优化算法求解该模型。算法原理细节可以参考文献[1]。

算法使用

FM算法是推荐领域被验证的效果较好的推荐方案之一，在电商、广告、视频、信息流、游戏的推荐领域有广泛应用。

文献

[1] S. Rendle, "Factorization Machines," 2010 IEEE International Conference on Data Mining, 2010, pp. 995-1000, doi: 10.1109/ICDM.2010.127.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	

batchSize	迭代数据batch size	数据batch size	Integer		
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
initStdev	初始化参数的标准差	初始化参数的标准差	Double		
lambda0	常数项正则化系数	常数项正则化系数	Double		
lambda1	线性项正则化系数	线性项正则化系数	Double		
lambda2	二次项正则化系数	二次项正则化系数	Double		
learnRate	学习率	学习率	Double		
modelFilePath	模型的文件路径	模型的文件路径	String		
numEpochs	epoch数	epoch数	Integer		
numFactor	因子数	因子数	Integer		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		

predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
reservedCols	算法保留列名	算法保留列	String[]		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL BIGINTEGER BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
withIntercept	是否有常数项	是否有常数项，默认true	Boolean		
withLinearItem	是否含有线性项	是否含有线性项	Boolean		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["1:1.1 3:2.0", 1.0],
    ["2:2.1 10:3.1", 1.0],
    ["1:1.2 5:3.2", 0.0],
    ["3:1.2 7:4.2", 0.0]])

input = BatchOperator.fromDataframe(df, schemaStr='kv string, label double')
test = StreamOperator.fromDataframe(df, schemaStr='kv string, label double')

fm = FmRegressor()\
        .setVectorCol("kv")\
        .setLabelCol("label")\
        .setPredictionCol("pred")

model = fm.fit(input)

model.transform(test).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
```

```
import com.alibaba.alink.pipeline.regression.FmRegressionModel;
import com.alibaba.alink.pipeline.regression.FmRegressor;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmRegressorTest {
    @Test
    public void testFmRegressor() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("1:1.1 3:2.0", 1.0),
            Row.of("2:2.1 10:3.1", 1.0),
            Row.of("1:1.2 5:3.2", 0.0)
        );
        BatchOperator <?> input = new MemSourceBatchOp(df, "kv string, label
double");
        StreamOperator <?> test = new MemSourceStreamOp(df, "kv string, label
double");
        FmRegressor fm = new FmRegressor()
            .setVectorCol("kv")
            .setLabelCol("label")
            .setPredictionCol("pred");
        FmRegressionModel model = fm.fit(input);
        model.transform(test).print();
        StreamOperator.execute();
    }
}
```

运行结果

kv	label	pred
1:1.1 3:2.0	1.0	0.473600
2:2.1 10:3.1	1.0	0.755115
1:1.2 5:3.2	0.0	0.005875
3:1.2 7:4.2	0.0	0.004641

GBDT回归 (GbdtRegressor)

Java 类名：com.alibaba.alink.pipeline.regression.GbdtRegressor

Python 类名：GbdtRegressor

功能介绍

- gbdt(Gradient Boosting Decision Trees)回归，是经典的基于boosting的有监督学习模型，可以用来解决回归问题
- 支持连续特征和离散特征
- 支持数据采样和特征采样

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
criteria	树分裂的策略	树分裂的策略，可以为PAI, XGBOOST	String		"PAI", "XGBOOST
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
featureImportanceType	特征重要性类型	特征重要性类型（默认为GAIN）	String		"WEIGHT", "GAIN", "COVER"
featureSubsamplingRatio	每棵树特征采样的比例	每棵树特征采样的比例，范围为(0, 1]。	Double		
gamma	xgboost中的l2正则项	xgboost中的l2正则项	Double		

lambda	xgboost中的l1正则项	xgboost中的l1正则项	Double		
learningRate	学习率	学习率（默认为0.3）	Double		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		
maxDepth	树的深度限制	树的深度限制	Integer		
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
minSumHessianPerLeaf	叶子节点最小Hessian值	叶子节点最小Hessian值（默认为0）	Double		
modelFilePath	模型的文件路径	模型的文件路径	String		
newtonStep	是否使用二阶梯度	是否使用二阶梯度	Boolean		
numTrees	模型中树的棵数	模型中树的棵数	Integer		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型[BIGDECIMBIGINTEGEBYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

参数建议

对于训练效果来说，比较重要的参数是 树的棵树+学习率、叶子节点最小样本数、单颗树最大深度、特征采样比例。

单个离散特征的取值种类数不能超过256，否则会出错。

代码示例

```
df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

GbdtRegressor()\
    .setLearningRate(1.0)\
    .setNumTrees(3)\
    .setMinSamplesPerLeaf(1)\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()

GbdtRegressor()\
    .setLearningRate(1.0)\
    .setNumTrees(3)\
    .setMinSamplesPerLeaf(1)\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()

StreamOperator.execute()
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred	
0	1.0	A	0	0	0	0.0
1	2.0	B	1	1	0	0.0
2	3.0	C	2	2	1	1.0
3	4.0	D	3	3	1	1.0

流预测结果

f0	f1	f2	f3	label	pred
0	1.0	A	0	0 0	0.0
1	3.0	C	2	2 1	1.0
2	2.0	B	1	1 0	0.0
3	4.0	D	3	3 1	1.0

广义线性回归 (GeneralizedLinearRegression)

Java 类名：com.alibaba.alink.pipeline.regression.GeneralizedLinearRegression

Python 类名：GeneralizedLinearRegression

功能介绍

广义线性回归训练和预测

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
epsilon	收敛精度	收敛精度	Double		
family	分布族	分布族，包含gaussian, Binomial, Poisson, Gamma and Tweedie，默认值gaussian。	String		"Gamma", "Binomial", "Gaussian", "Poisson", "Tweedie"

fitIntercept	是否拟合常数项	是否拟合常数项，默认是拟合	Boolean		
link	连接函数	连接函数，包含cloglog, Identity, Inverse, log, logit, power, probit和sqrt，默认值是指数分布族对应的连接函数。	String		"CLogLog", "Identity", "Inverse", "Log", "Logit", "Power", "Probit", "Sqrt"
linkPower	连接函数的超参	连接函数的超参	Double		
linkPredResultCol	连接函数结果的列名	连接函数结果的列名	String		
maxIter	最大迭代步数	最大迭代步数，默认为10。	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		

offsetCol	偏移列	偏移列	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
regParam	l2正则系数	l2正则系数	Double		
reservedCols	算法保留列名	算法保留列	String[]		
variancePower	分布族的超参	分布族的超参，默认值是0.0	Double		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]

numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [1.6094,118.0000,69.0000,1.0000,2.0000],
    [2.3026,58.0000,35.0000,1.0000,2.0000],
    [2.7081,42.0000,26.0000,1.0000,2.0000],
    [2.9957,35.0000,21.0000,1.0000,2.0000],
    [3.4012,27.0000,18.0000,1.0000,2.0000],
    [3.6889,25.0000,16.0000,1.0000,2.0000],
    [4.0943,21.0000,13.0000,1.0000,2.0000],
    [4.3820,19.0000,12.0000,1.0000,2.0000],
    [4.6052,18.0000,12.0000,1.0000,2.0000]
])

source = BatchOperator.fromDataframe(df, schemaStr='u double, lot1 double, lot2
double, offset double, weights double')

featureColNames = ["lot1", "lot2"]
labelColName = "u"

# train
glm = GeneralizedLinearRegression()\
                .setFamily("gamma")\
                .setLink("Log")\
                .setRegParam(0.3)\
                .setMaxIter(5)\
                .setFeatureCols(featureColNames)\
                .setLabelCol(labelColName)\
                .setPredictionCol("pred")

model = glm.fit(source)
predict = model.transform(source)
eval2 = model.evaluate(source)

predict.lazyPrint(10)
eval2.print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.regression.GeneralizedLinearRegression;
import com.alibaba.alink.pipeline.regression.GeneralizedLinearRegressionModel;
import org.junit.Test;
```

```
import java.util.Arrays;
import java.util.List;

public class GeneralizedLinearRegressionTest {
    @Test
    public void testGeneralizedLinearRegression() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.6094, 118.0000, 69.0000, 1.0000, 2.0000),
            Row.of(2.3026, 58.0000, 35.0000, 1.0000, 2.0000),
            Row.of(2.7081, 42.0000, 26.0000, 1.0000, 2.0000),
            Row.of(2.9957, 35.0000, 21.0000, 1.0000, 2.0000),
            Row.of(3.4012, 27.0000, 18.0000, 1.0000, 2.0000),
            Row.of(3.6889, 25.0000, 16.0000, 1.0000, 2.0000),
            Row.of(4.0943, 21.0000, 13.0000, 1.0000, 2.0000),
            Row.of(4.3820, 19.0000, 12.0000, 1.0000, 2.0000),
            Row.of(4.6052, 18.0000, 12.0000, 1.0000, 2.0000)
        );
        BatchOperator <?> source = new MemSourceBatchOp(df,
            "u double, lot1 double, lot2 double, offset double, weights
double");
        String[] featureColNames = new String[] {"lot1", "lot2"};
        String labelColName = "u";
        GeneralizedLinearRegression glm = new GeneralizedLinearRegression()
            .setFamily("gamma")
            .setLink("Log")
            .setRegParam(0.3)
            .setMaxIter(5)
            .setFeatureCols(featureColNames)
            .setLabelCol(labelColName)
            .setPredictionCol("pred");
        GeneralizedLinearRegressionModel model = glm.fit(source);
        BatchOperator <?> predict = model.transform(source);
        BatchOperator <?> eval2 = model.evaluate(source);
        predict.lazyPrint(10);
        eval2.print();
    }
}
```

运行结果

预测结果

u	lot1	lot2	offset	weights	pred
1.6094	118.0000	69.0000	1.0000	2.0000	1.4601
2.3026	58.0000	35.0000	1.0000	2.0000	2.6396

2.7081	42.0000	26.0000	1.0000	2.0000	3.0847
2.9957	35.0000	21.0000	1.0000	2.0000	3.4135
3.4012	27.0000	18.0000	1.0000	2.0000	3.5215
3.6889	25.0000	16.0000	1.0000	2.0000	3.6901
4.0943	21.0000	13.0000	1.0000	2.0000	3.9275
4.3820	19.0000	12.0000	1.0000	2.0000	3.9891
4.6052	18.0000	12.0000	1.0000	2.0000	3.9581

评估结果

{"rank":3,"degreeOfFreedom":6,"residualDegreeOfFreeDom":6,"residualDegreeOfFreedomNull":8,"aic":9702.08
[0.007797743508544688,-0.031175844426488887],"intercept":1.609524324733498,"coefficientStandardErrors
[0.2566303869742456,-0.5880323136505685,14.715031444760141],"pValues":[0.8060371545112832,0.5779

Isotonic回归 (IsotonicRegression)

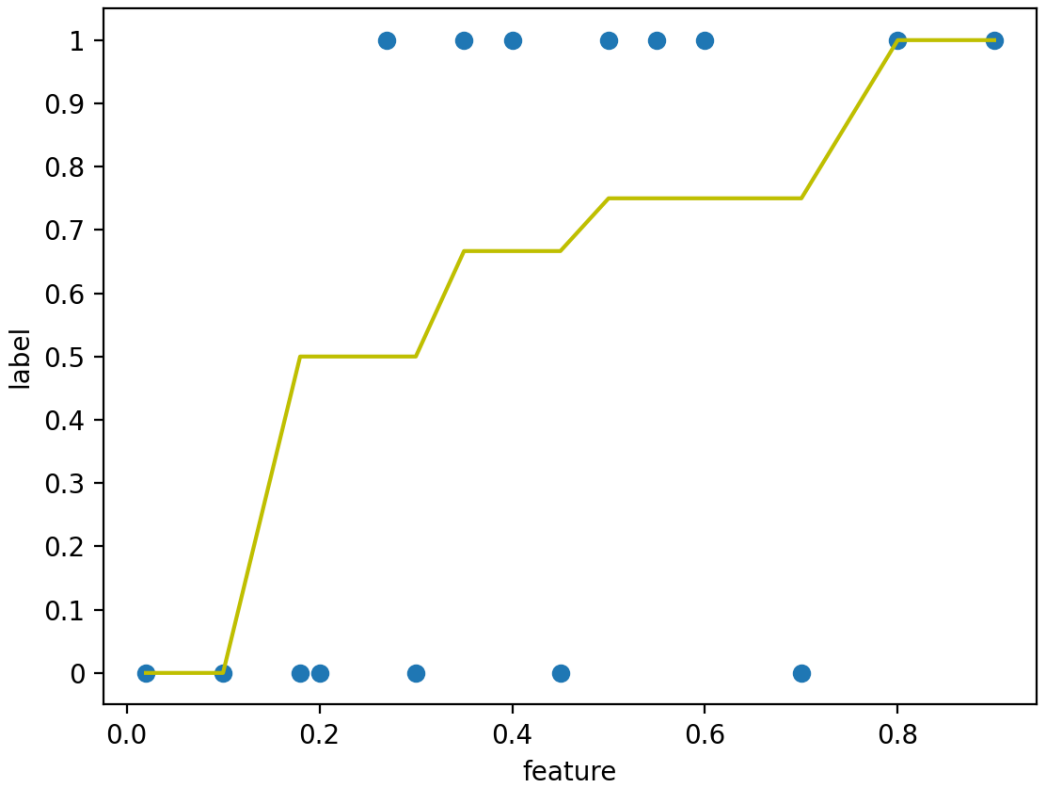
Java 类名：com.alibaba.alink.pipeline.regression.IsotonicRegression

Python 类名：IsotonicRegression

功能介绍

保序回归在观念上是寻找一组非递减的片段连续线性函数（piecewise linear continuous functions），即保序函数，使其与样本尽可能的接近。

保序回归的输入在Alink中称分别为特征（feature）、标签（label）和权重（weight），特征可以是数值或向量，如果是向量还需要设定特征索引（feature index），组件将使用该维进行计算。保序回归的目标是求解一个能使$\textstyle \sum_i{w_i(y_i-\hat{y}_i)^2}$最小的序列$\hat{y}$，若选择保增序，该序列还应满足$X_i<X_j$时$\hat{y}_i\le\hat{y}_j$，若选择保降序满足$X_i<X_j$时$\hat{y}_i\ge\hat{y}_j$。下图中，散点图是训练数据，折线图是得到的保序回归模型，对于训练数据中没有的特征，使用线性插值得到其标签。对应训练和预测代码见示例。



参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
featureCol	特征列名	特征列的名称	String		
featureIndex	训练特征所在维度	训练特征在输入向量的维度索引	Integer		[0, +inf)
isotonic	输出序列是否	输出序列是否递增	Boolean		
modelFilePath	模型的文件路径	模型的文件路径	String		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0.35, 1],
    [0.6, 1],
    [0.55, 1],
    [0.5, 1],
    [0.18, 0],
    [0.1, 1],
    [0.8, 1],
    [0.45, 0],
    [0.4, 1],
    [0.7, 0],
    [0.02, 1],
    [0.3, 0],
    [0.27, 1],
    [0.2, 0],
    [0.9, 1]])
```

```
data = BatchOperator.fromDataframe(df, schemaStr="label double, feature
double")

res = IsotonicRegression()\
            .setFeatureCol("feature")\
            .setLabelCol("label")\
            .setPredictionCol("result")

res.fit(data).transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.regression.IsotonicRegression;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class IsotonicRegressionTest {
    @Test
    public void testIsotonicRegression() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0.02, 0.0),
            Row.of(0.1, 0.0),
            Row.of(0.18, 1.0),
            Row.of(0.2, 0.0),
            Row.of(0.27, 1.0),
            Row.of(0.3, 0.0),
            Row.of(0.35, 1.0),
            Row.of(0.4, 1.0),
            Row.of(0.45, 0.0),
            Row.of(0.5, 1.0),
            Row.of(0.55, 1.0),
            Row.of(0.6, 1.0),
            Row.of(0.7, 0.0),
            Row.of(0.8, 1.0),
            Row.of(0.9, 1.0),
            Row.of(0.98, 1.10)
        );
        List <Row> pred = Arrays.asList(
            Row.of(0.2),
            Row.of(0.32),
```

```
            Row.of(0.4),
            Row.of(0.45),
            Row.of(0.65),
            Row.of(0.9)
        );

        BatchOperator <?> data = new MemSourceBatchOp(df, "feature double,
label double");
        BatchOperator <?> predData = new MemSourceBatchOp(pred, "feature
double");
        IsotonicRegressionModel res = new IsotonicRegression()
            .setFeatureCol("feature")
            .setLabelCol("label")
            .setPredictionCol("predict")
            .fit(data);
        res.transform(predData).print();
    }
}
```

运行结果

feature	predict
0.2000	0.5000
0.3200	0.5667
0.4000	0.6667
0.4500	0.6667
0.6500	0.7500
0.9000	1.0000

KerasSequential回归 (KerasSequentialRegressor)

Java 类名：com.alibaba.alink.pipeline.regression.KerasSequentialRegressor

Python 类名：KerasSequentialRegressor

功能介绍

构建一个 Keras 的 Sequential 模型， 训练回归模型。

通过 layers 参数指定构成 Sequential 模型的网络层，Alink 会自动在最开始添加 Input 层，在最后添加 Dense 层和激活层，得到完整的模型用于训练。

指定 layers 参数时，使用的是 Python 语句，例如

```
"Conv1D(256, 5, padding='same', activation='relu')",
"Conv1D(128, 5, padding='same', activation='relu')",
"Dropout(0.1)",
"MaxPooling1D(pool_size=8)",
"Conv1D(128, 5, padding='same', activation='relu')",
"Conv1D(128, 5, padding='same', activation='relu')",
"Flatten()"
```

`tf.keras.layers` 内的网络层已经提前 import，可以直接使用。 使用的 TensorFlow 版本是 2.3.1。

参数说明

名称	中文名称	描述	类型	是
labelCol	标签列名	输入表中的标签列名	String	✓
layers	各 layer 的描述	各 layer 的描述，使用 Python 语法，例如 "Conv1D(256, 5, padding='same', activation='relu')"	String[]	✓
predictionCol	预测结果列名	预测结果列名	String	✓
tensorCol	tensor列	tensor列	String	✓
batchSize	数据批大小	数据批大小	Integer	

bestMetric	最优指标	判断模型最优时用的指标，仅在总并发度为 1 时起作用。都支持的有：loss； 二分类还支持：auc, precision, recall, binary_accuracy, false_negatives, false_positives, true_negatives, true_positives；多分类还支持：sparse_categorical_accuracy；回归还支持：mean_absolute_error, mean_absolute_percentage_error, mean_squared_error, mean_squared_logarithmic_error, root_mean_squared_error	String	
checkpointFilePath	保存 checkpoint 的路径	用于保存中间结果的路径，将作为 TensorFlow 中 `Estimator` 的 `model_dir` 传入，需要为所有 worker 都能访问到的目录	String	
inferBatchSize	推理数据批大小	推理数据批大小	Integer	
intraOpParallelism	Op 间并发度	Op 间并发度	Integer	
learningRate	学习率	学习率	Double	
modelFilePath	模型的文件路径	模型的文件路径	String	
numEpochs	epoch数	epoch数	Integer	
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer	
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer	
optimizer	优化器	优化器，使用 Python 语法，例如 "Adam(learning_rate=0.1)"	String	
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean	

pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String	
removeCheckpointBeforeTraining	是否在训练前移除 checkpoint 相关文件	是否在训练前移除 checkpoint 相关文件用于重新训练，只会删除必要的文件	Boolean	
reservedCols	算法保留列名	算法保留列	String[]	
saveBestOnly	是否导出最优的 checkpoint	是否导出最优的 checkpoint，仅在总并发度为 1 时生效	Boolean	
saveCheckpointsEpochs	每隔多少 epochs 保存 checkpoints	每隔多少 epochs 保存 checkpoints	Double	
saveCheckpointsSecs	每隔多少秒保存 checkpoints	每隔多少秒保存 checkpoints	Double	
validationSplit	验证集比例	验证集比例，仅在总并发度为 1 时生效	Double	
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String	
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer	
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String	

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
source = CsvSourceBatchOp() \
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/random_tensor.csv") \
    .setSchemaStr("tensor string, label double")

source = ToTensorBatchOp() \
    .setSelectedCol("tensor") \
    .setTensorDataType("DOUBLE") \
    .setTensorShape([200, 3]) \
    .linkFrom(source)

trainer = KerasSequentialRegressor() \
    .setTensorCol("tensor") \
    .setLabelCol("label") \
    .setLayers([
        "Conv1D(256, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Dropout(0.1)",
        "MaxPooling1D(pool_size=8)",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Conv1D(128, 5, padding='same', activation='relu')",
        "Flatten()"
    ]) \
    .setOptimizer("Adam()") \
    .setNumEpochs(1) \
    .setPredictionCol("pred") \
    .setReservedCols(["label"])

model = trainer.fit(source)
prediction = model.transform(source)
prediction.lazyPrint(10)
BatchOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.ToTensorBatchOp;
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.regression.KerasSequentialRegressionModel;
import com.alibaba.alink.pipeline.regression.KerasSequentialRegressor;
import org.junit.Test;

public class KerasSequentialRegressorTest {

    @Test
    public void testKerasSequentialRegressor() throws Exception {
        BatchOperator<?> source = new CsvSourceBatchOp()
```

```
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/random_tensor.csv")
            .setSchemaStr("tensor string, label double");

        source = new ToTensorBatchOp()
            .setSelectedCol("tensor")
            .setTensorDataType("DOUBLE")
            .setTensorShape(200, 3)
            .linkFrom(source);

        KerasSequentialRegressor trainer = new KerasSequentialRegressor()
            .setTensorCol("tensor")
            .setLabelCol("label")
            .setLayers(new String[] {
                "Conv1D(256, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Dropout(0.1)",
                "MaxPooling1D(pool_size=8)",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Conv1D(128, 5, padding='same', activation='relu')",
                "Flatten()"
            })
            .setOptimizer("Adam()")
            .setNumEpochs(1)
            .setPredictionCol("pred")
            .setReservedCols("label");

        KerasSequentialRegressionModel model = trainer.fit(source);
        BatchOperator <?> prediction = model.transform(source);
        prediction.lazyPrint(10);
        BatchOperator.execute();
    }
}
```

运行结果

label	pred
0.0000	0.4580
1.0000	0.4323
1.0000	0.4547
1.0000	0.4381
0.0000	0.4361
1.0000	0.4633

0.0000	0.4565
1.0000	0.4928
1.0000	0.4306
0.0000	0.4359

Lasso回归 (LassoRegression)

Java 类名：com.alibaba.alink.pipeline.regression.LassoRegression

Python 类名：LassoRegression

功能介绍

Lasso回归算法是由1996年Robert Tibshirani首次提出。是一种经典的回归算法。Lasso回归组件支持稀疏、稠密两种数据格式，并且支持带权重样本训练。

算法原理

Lasso回归算法通过构造一个惩罚函数得到一个较为精炼的模型，使得它压缩一些回归系数，即强制系数绝对值之和小于某个固定值；同时设定一些回归系数为零。因此保留了子集收缩的优点，是一种处理具有复共线性数据的有偏估计。

算法使用

Lasso回归模型应用领域和线性回归类似，经常被用来做一些数值型变量的预测，类似房价预测、销售量预测、贷款额度预测、温度预测、适度预测等。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society: Series B (Methodological) 58.1 (1996): 267-288.

[2] https://baike.baidu.com/item/LASSO/20366865?fr=aladdin

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
lambda	希腊字母: lambda	惩罚因子，必选	Double	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	

epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)
modelFilePath	模型的文件路径	模型的文件路径	String		
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMA BIGINTEGE BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
withIntercept	是否有常数项	是否有常数项，默认true	Boolean		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		

modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')
colnames = ["f0","f1"]
lasso = LassoRegression()\
            .setFeatureCols(colnames)\
            .setLambda(0.1)\
            .setLabelCol("label")\
            .setPredictionCol("pred")
model = lasso.fit(batchData)
model.transform(batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.regression.LassoRegression;
import com.alibaba.alink.pipeline.regression.LassoRegressionModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LassoRegressionTest {
    @Test
    public void testLassoRegression() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        String[] colnames = new String[] {"f0", "f1"};
        LassoRegression lasso = new LassoRegression()
            .setFeatureCols(colnames)
            .setLambda(0.1)
            .setLabelCol("label")
            .setPredictionCol("pred");
        LassoRegressionModel model = lasso.fit(batchData);
        model.transform(batchData).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1.1304
3	2	1	1.4047
4	3	2	1.6790

2	4	1	1.1651
2	2	1	1.1420
4	3	2	1.6790
1	2	1	0.8793

线性回归Stepwise (LinearRegStepwise)

Java 类名：com.alibaba.alink.pipeline.regression.LinearRegStepwise

Python 类名：LinearRegStepwise

功能介绍

- Stepwise回归是一个回归算法
- Stepwise回归组件仅支持稠密数据格式

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	
featureCols	特征列名	特征列名，必选	String[]	✓		
labelCol	标签列名	输入表中的标签列名	String	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
method	回归统计的方法	可取值包括：stepwise，forward，backward	String		"Stepwise", "Forward", "Backward"	"

modelFilePath	模型的文件路径	模型的文件路径	String			r
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			f
reservedCols	算法保留列名	算法保留列	String[]			r
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			r

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			1
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			r

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [16.3, 1.1, 1.1],
    [16.8, 1.4, 1.5],
    [19.2, 1.7, 1.8],
    [18.0, 1.7, 1.7],
    [19.5, 1.8, 1.9],
    [20.9, 1.8, 1.8],
    [21.1, 1.9, 1.8],
    [20.9, 2.0, 2.1],
    [20.3, 2.3, 2.4],
    [22.0, 2.4, 2.5]
])

# load data
batchData = BatchOperator.fromDataframe(df, schemaStr='y double, x1 double, x2
double')
```

```
colnames = ["x1", "x2"]

step = LinearRegStepwise()\
        .setFeatureCols(colnames)\
        .setLabelCol("y")\
        .setPredictionCol("pred")

model = step.fit(batchData)
model.transform(batchData).print()
```

运行结果

y	x1	x2	pred
16.3	1.1	1.1	16.380060195635785
16.8	1.4	1.5	17.698344620015032
19.2	1.7	1.8	19.01662904439428
18.0	1.7	1.7	19.01662904439428
19.5	1.8	1.9	19.456057185854025
20.9	1.8	1.8	19.456057185854025
21.1	1.9	1.8	19.89548532731377
20.9	2.0	2.1	20.33491346877352
20.3	2.3	2.4	21.653197893152765
22.0	2.4	2.5	22.092626034612515

线性回归 (LinearRegression)

Java 类名：com.alibaba.alink.pipeline.regression.LinearRegression

Python 类名：LinearRegression

功能介绍

线性回归算法是经典的回归算法，通过对带有回归值的样本集合训练得到回归模型，使用模型预测样本的回归值。线性回归组件支持稀疏、稠密两种数据格式，并且支持带权重样本训练。

算法原理

面对回归类问题，线性回归利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。

算法使用

线性回归模型经常被用来做一些数值型变量的预测，类似房价预测、销售量预测、贷款额度预测、温度预测、适度预测等。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Seber, George AF, and Alan J. Lee. Linear regression analysis. John Wiley & Sons, 2012.

[2] https://baike.baidu.com/item/%E7%BA%BF%E6%80%A7%E5%9B%9E%E5%BD%92/8190345?fr=aladdin

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	

predictionCol	预测结果列名	预测结果列名	String	✓	
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
l1	L1正则化系数	L1 正则化系数，默认为0。	Double		[0.0, +inf)
l2	正则化系数	L2 正则化系数，默认为0。	Double		[0.0, +inf)
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)
modelFilePath	模型的文件路径	模型的文件路径	String		

optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
withIntercept	是否有常数项	是否有常数项，默认true	Boolean		

numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')
colnames = ["f0","f1"]
lr = LinearRegression()\
        .setFeatureCols(colnames)\
        .setLabelCol("label")\
        .setPredictionCol("pred")
model = lr.fit(batchData)
model.transform(batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.regression.LinearRegression;
import com.alibaba.alink.pipeline.regression.LinearRegressionModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LinearRegressionTest {
    @Test
    public void testLinearRegression() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1)
```

```
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        String[] colnames = new String[] {"f0", "f1"};
        LinearRegression lr = new LinearRegression()
            .setFeatureCols(colnames)
            .setLabelCol("label")
            .setPredictionCol("pred");
        LinearRegressionModel model = lr.fit(batchData);
        model.transform(batchData).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	1.0000
3	2	1	1.4118
4	3	2	1.8235
2	4	1	1.1765
2	2	1	1.0588
4	3	2	1.8235
1	2	1	0.7059

线性SVR (LinearSvr)

Java 类名：com.alibaba.alink.pipeline.regression.LinearSvr

Python 类名：LinearSvr

功能介绍

- 线性SVR是一个回归算法
- 线性SVR组件支持稀疏、稠密两种数据格式
- 线性SVR组件支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
C	算法参数	支撑向量回归参数	Double	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)

featureCols	特征列名数组	特征列名数组，默认全选	String[]		
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)
modelFilePath	模型的文件路径	模型的文件路径	String		
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean		

tau	算法参数	支撑向量回归参数	Double		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
withIntercept	是否有常数项	是否有常数项，默认true	Boolean		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int', op_type='batch')

colnames = ["f0","f1"]

lsvr = LinearSvr()\
```

```
            .setFeatureCols(colnames)\
            .setLabelCol("label")\
            .setPredictionCol("pred")

model = lsvr.fit(batchData)

model.transform(batchData).print()
```

运行结果

f0	f1	label	pred
2	1	1	1.000014
3	2	1	1.538474
4	3	2	2.076934
2	4	1	1.138446
2	2	1	1.046158
4	3	2	2.076934
1	2	1	0.553842
5	3	3	2.569250

随机森林回归 (RandomForestRegressor)

Java 类名：com.alibaba.alink.pipeline.regression.RandomForestRegressor

Python 类名：RandomForestRegressor

功能介绍

- 随机森林回归是一种常用的树模型，由于bagging的过程，可以避免过拟合
- 随机森林回归组件支持稠密数据格式
- 支持带样本权重的训练

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
featureCols	特征列名	特征列名，必选	String[]	✓	
labelCol	标签列名	输入表中的标签列名	String	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
categoricalCols	离散特征列名	离散特征列名	String[]		
createTreeMode	创建树的模式。	series表示每个单机创建单颗树，parallel表示并行创建单颗树。	String		
maxBins	连续特征进行分箱的最大个数	连续特征进行分箱的最大个数。	Integer		

maxDepth	树的深度限制	树的深度限制	Integer		
maxLeaves	叶节点的最多个数	叶节点的最多个数	Integer		
maxMemoryInMB	树模型中用来加和统计量的最大内存使用数	树模型中用来加和统计量的最大内存使用数	Integer		
minInfoGain	分裂的最小增益	分裂的最小增益	Double		
minSampleRatioPerChild	子节点占父节点的最小样本比例	子节点占父节点的最小样本比例	Double		
minSamplesPerLeaf	叶节点的最小样本个数	叶节点的最小样本个数	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
numSubsetFeatures	每棵树的特征采样数目	每棵树的特征采样数目	Integer		

numTrees	模型中树的棵数	模型中树的棵数	Integer		[1, +inf)
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
seed	采样种子	采样种子	Long		
subsamplingRatio	每棵树的样本采样比例或采样行数	每棵树的样本采样比例或采样行数，行数上限100w行	Double		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMAL, BIGINTEGER, BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [1.0, "A", 0, 0, 0],
    [2.0, "B", 1, 1, 0],
    [3.0, "C", 2, 2, 1],
    [4.0, "D", 3, 3, 1]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='f0 double, f1 string, f2 int, f3 int, label int')

RandomForestRegressor()\
    .setPredictionCol('pred')\
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(batchSource)\
    .print()

RandomForestRegressor()\
    .setPredictionCol('pred')\
```

```
    .setLabelCol('label')\
    .setFeatureCols(['f0', 'f1', 'f2', 'f3'])\
    .fit(batchSource)\
    .transform(streamSource)\
    .print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.regression.RandomForestRegressor;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RandomForestRegressorTest {
    @Test
    public void testRandomForestRegressor() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(1.0, "A", 0, 0, 0),
            Row.of(2.0, "B", 1, 1, 0),
            Row.of(3.0, "C", 2, 2, 1),
            Row.of(4.0, "D", 3, 3, 1)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(df, "f0 double, f1
string, f2 int, f3 int, label int");
        StreamOperator <?> streamSource = new MemSourceStreamOp(df, "f0 double,
f1 string, f2 int, f3 int, label int");
        new RandomForestRegressor()
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
            .fit(batchSource)
            .transform(batchSource)
            .print();
        new RandomForestRegressor()
            .setPredictionCol("pred")
            .setLabelCol("label")
            .setFeatureCols("f0", "f1", "f2", "f3")
```

```
            .fit(batchSource)
            .transform(streamSource)
            .print();
        StreamOperator.execute();
    }
}
```

运行结果

批预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

流预测结果

f0	f1	f2	f3	label	pred
1.0000	A	0	0	0	0.0000
2.0000	B	1	1	0	0.0000
3.0000	C	2	2	1	1.0000
4.0000	D	3	3	1	1.0000

岭回归 (RidgeRegression)

Java 类名：com.alibaba.alink.pipeline.regression.RidgeRegression

Python 类名：RidgeRegression

功能介绍

岭回归(Ridge regression)算法是一种经典的回归算法。岭回归组件支持稀疏、稠密两种数据格式，并且支持带权重样本训练。

算法原理

岭回归是一种专用于共线性数据分析的有偏估计回归方法，实质上是一种改良的最小二乘估计法，通过放弃最小二乘法的无偏性，以损失部分信息、降低精度为代价获得回归系数更为符合实际、更可靠的回归方法，对病态数据的拟合要强于最小二乘法。

算法使用

岭回归模型应用领域和线性回归类似，经常被用来做一些数值型变量的预测，类似房价预测、销售量预测、贷款额度预测、温度预测、适度预测等。

- 备注： 该组件训练的时候 FeatureCols 和 VectorCol 是两个互斥参数，只能有一个参数来描述算法的输入特征。

文献或出处

[1] Hoerl, Arthur E., and Robert W. Kennard. "Ridge regression: Biased estimation for nonorthogonal problems." Technometrics 12.1 (1970): 55-67.

[2] https://baike.baidu.com/item/%E5%B2%AD%E5%9B%9E%E5%BD%92/554917?fr=aladdin

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
labelCol	标签列名	输入表中的标签列名	String	✓	
lambda	希腊字母：lambda	惩罚因子，必选	Double	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	

epsilon	收敛阈值	迭代方法的终止判断阈值，默认值为 1.0e-6	Double		[0.0, +inf)
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)
modelFilePath	模型的文件路径	模型的文件路径	String		
optimMethod	优化方法	优化问题求解时选择的优化方法	String		"LBFGS", "GD", "Newton", "SGD", "OWLQN"
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
standardization	是否正则化	是否对训练数据做正则化，默认true	Boolean		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
weightCol	权重列名	权重列对应的列名	String		所选列类型为[BIGDECIMA BIGINTEGE BYTE, DOUBLE, FLOAT, INTEGER, LONG, SHORT]
withIntercept	是否有常数项	是否有常数项，默认true	Boolean		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		

modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [2, 1, 1],
    [3, 2, 1],
    [4, 3, 2],
    [2, 4, 1],
    [2, 2, 1],
    [4, 3, 2],
    [1, 2, 1],
    [5, 3, 3]])

batchData = BatchOperator.fromDataframe(df, schemaStr='f0 int, f1 int, label
int')

colnames = ["f0","f1"]

ridge = RidgeRegression()\
            .setFeatureCols(colnames)\
            .setLambda(0.1)\
            .setLabelCol("label")\
            .setPredictionCol("pred")
```

```
model = ridge.fit(batchData)
model.transform(batchData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.regression.RidgeRegression;
import com.alibaba.alink.pipeline.regression.RidgeRegressionModel;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class RidgeRegressionTest {
    @Test
    public void testRidgeRegression() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(2, 1, 1),
            Row.of(3, 2, 1),
            Row.of(4, 3, 2),
            Row.of(2, 4, 1),
            Row.of(2, 2, 1),
            Row.of(4, 3, 2),
            Row.of(1, 2, 1)
        );
        BatchOperator <?> batchData = new MemSourceBatchOp(df, "f0 int, f1 int,
label int");
        String[] colnames = new String[] {"f0", "f1"};
        RidgeRegression ridge = new RidgeRegression()
            .setFeatureCols(colnames)
            .setLambda(0.1)
            .setLabelCol("label")
            .setPredictionCol("pred");
        RidgeRegressionModel model = ridge.fit(batchData);
        model.transform(batchData).print();
    }
}
```

运行结果

f0	f1	label	pred
2	1	1	0.8849

3	2	1	1.2828
4	3	2	1.6807
2	4	1	1.1334
2	2	1	0.9678
4	3	2	1.6807
1	2	1	0.6527

XGBoost回归 (XGBoostRegressor)

Java 类名：com.alibaba.alink.pipeline.regression.XGBoostRegressor

Python 类名：XGBoostRegressor

功能介绍

XGBoost 组件是在开源社区的基础上进行包装，使功能和 PAI 更兼容，更易用。 XGBoost 算法在 Boosting 算法的基础上进行了扩展和升级，具有较好的易用性和鲁棒性，被广泛用在各种机器学习生产系统和竞赛领域。 当前支持分类，回归和排序。

参数说明

名称	中文名称	描述	类型	是否必须?	
labelCol	标签列名	输入表中的标签列名	String	✓	
numRound	树的棵树	树的棵树	Integer	✓	
predictionCol	预测结果列名	预测结果列名	String	✓	
alpha	L1 正则项	L1 正则项	Double		
baseScore	Base score	Base score	Double		
colSampleByLevel	每个树列采样	每个树列采样	Double		
colSampleByNode	每个结点列采样	每个结点采样	Double		
colSampleByTree	每个树列采样	每个树列采样	Double		
eta	学习率	学习率	Double		
featureCols	特征列名数组	特征列名数组，默认全选	String[]		
gamma	结点分裂最小损失变化	节点分裂最小损失变化	Double		
growPolicy	GrowPolicy	GrowPolicy	String		"DE
interactionConstraints	interaction constraints	interaction constraints	String		
lambda	L2 正则项	L2 正则项	Double		
maxBin	最大结点个数	最大结点个数	Integer		

maxDeltaStep	Delta step	Delta step	Double		
maxDepth	最大深度	最大深度	Integer		
maxLeaves	最大结点个数	最大结点个数	Integer		
minChildWeight	结点的最小权重	结点的最小权重	Double		
modelFilePath	模型的文件路径	模型的文件路径	String		
monotoneConstraints	monotone constraints	monotone constraints	String		
numClass	标签类别个数	标签类别个数， 多分类时有效	Integer		
objective	objective	objective	String		"RE "RE "RE "RE "RE "RE
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
pluginVersion	插件版本号	插件版本号	String		
processType	ProcessType	ProcessType	String		"DE
refreshLeaf	RefreshLeaf	RefreshLeaf	Integer		
reservedCols	算法保留列名	算法保留列	String[]		
runningMode	运行模式	XGBoost的运行模型，ICQ速度快，但使用内存多，TRIAVIAL速度略慢，但是节省内存，按照流式方式处理。由于训练数据本身在XGBoost运行时已经被缓存进内存，所以存两份和存一份数据的资源消耗和速度对比，还需要进一步的测试。	String		"IC
samplingMethod	采样方法	采样方法	String		"UN "GF
scalePosWeight	ScalePosWeight	ScalePosWeight	Double		
singlePrecisionHistogram	single precision histogram	single precision histogram	Boolean		

sketchEps	SketchEps	SketchEps	Double		
subSample	样本采样比例	样本采样比例	Double		
treeMethod	构建树的方法	构建树的方法	String		"AU "HI
tweedieVariancePower	学习率	学习率	Double		
updater	Updater	Updater	String		
vectorCol	向量列名	向量列对应的列名，默认值是null	String		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
df = pd.DataFrame([
    [0, 1, 1.1, 1.0],
    [1, -2, 0.9, 2.0],
    [0, 100, -0.01, 3.0],
    [1, -99, 0.1, 4.0],
    [0, 1, 1.1, 5.0],
    [1, -2, 0.9, 6.0]
])

batchSource = BatchOperator.fromDataframe(
    df, schemaStr='y int, x1 double, x2 double, x3 double'
)
```

```
streamSource = StreamOperator.fromDataframe(
    df, schemaStr='y int, x1 double, x2 double, x3 double'
)

xgboostRegressor = XGBoostRegressor() \
    .setNumRound(1)\
    .setPluginVersion('1.5.1')\
    .setLabelCol('y')\
    .setPredictionCol('pred')

xgboostRegressor.fit(batchSource).transform(streamSource).print()

StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.regression.XGBoostRegressor;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class XGBoostRegressorTest {

    @Test
    public void testXGBoostRegressor() throws Exception {
        List <Row> data = Arrays.asList(
            Row.of(0, 1, 1.1, 1.0),
            Row.of(1, -2, 0.9, 2.0),
            Row.of(0, 100, -0.01, 3.0),
            Row.of(1, -99, 0.1, 4.0),
            Row.of(0, 1, 1.1, 5.0),
            Row.of(1, -2, 0.9, 6.0)
        );

        BatchOperator <?> batchSource = new MemSourceBatchOp(data, "y int, x1
int, x2 double, x3 double");
        StreamOperator <?> streamSource = new MemSourceStreamOp(data, "y int,
x1 int, x2 double, x3 double");
```

```
        XGBoostRegressor xgBoostRegressor = new XGBoostRegressor()
            .setNumRound(1)
            .setPluginVersion("1.5.1")
            .setLabelCol("y")
            .setPredictionCol("pred");

        xgBoostRegressor.fit(batchSource).transform(streamSource).print();

        StreamOperator.execute();
    }
}
```

二分K均值聚类 (BisectingKMeans)

Java 类名：com.alibaba.alink.pipeline.clustering.BisectingKMeans

Python 类名：BisectingKMeans

功能介绍

二分k均值算法是k-means聚类算法的一个变体，主要是为了改进k-means算法随机选择初始质心的随机性造成聚类结果不确定性的问题.

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
predictionCol	预测结果列名	预测结果列名	String	✓	
vectorCol	向量列名	向量列对应的列名	String	✓	
distanceType	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN", "COSINE"
k	聚类中心点数目	聚类中心点数目	Integer		

maxIter	最大迭代步数	最大迭代步数，默认为10。	Integer		
minDivisibleClusterSize	最小可分裂的聚类大小	最小可分裂的聚类大小	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
predictionDetailCol	预测详细信息列名	预测详细信息列名	String		
randomSeed	随机数种子	随机数种子	Integer		

reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "0.1,0.1,0.1"],
    [2, "0.2,0.2,0.2"],
    [3, "9 9 9"],
    [4, "9.1 9.1 9.1"],
    [5, "9.2 9.2 9.2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')

kmeans = BisectingKMeans()\
    .setVectorCol("vec")\
    .setK(2)\
    .setPredictionCol("pred")

kmeans.fit(inOp)\
    .transform(inOp)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.clustering.BisectingKMeans;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class BisectingKMeansTest {
    @Test
    public void testBisectingKMeans() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "0.1,0.1,0.1"),
            Row.of(2, "0.2,0.2,0.2"),
```

```
            Row.of(3, "9 9 9"),
            Row.of(4, "9.1 9.1 9.1"),
            Row.of(5, "9.2 9.2 9.2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, vec
string");
        BisectingKMeans kmeans = new BisectingKMeans()
            .setVectorCol("vec")
            .setK(2)
            .setPredictionCol("pred");
        kmeans.fit(inOp)
            .transform(inOp)
            .print();
    }
}
```

运行结果

预测结果

id	vec	pred
0	0 0 0	0
1	0.1,0.1,0.1	0
2	0.2,0.2,0.2	0
3	9 9 9	1
4	9.1 9.1 9.1	1
5	9.2 9.2 9.2	1

高斯混合模型 (GaussianMixture)

Java 类名：com.alibaba.alink.pipeline.clustering.GaussianMixture

Python 类名：GaussianMixture

功能介绍

高斯混合模型聚类

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
predictionCol	预测结果列名	预测结果列名	String	✓		
vectorCol	向量列名	向量列对应的列名	String	✓		
epsilon	收敛阈值	当两轮迭代的中心点距离小于epsilon时，算法收敛。	Double			1.0E-4
k	聚类中心点数量	聚类中心点数量	Integer			2

maxIter	最大迭代步数	最大迭代步数，默认为100	Integer		[1, +inf)	100
modelFilePath	模型的文件路径	模型的文件路径	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
randomSeed	随机数种子	随机数种子	Integer			0
reservedCols	算法保留列名	算法保留列	String[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    ["-0.6264538 0.1836433"],
    ["-0.8356286 1.5952808"],
    ["0.3295078 -0.8204684"],
    ["0.4874291 0.7383247"],
    ["0.5757814 -0.3053884"],
    ["1.5117812 0.3898432"],
    ["-0.6212406 -2.2146999"],
    ["11.1249309 9.9550664"],
    ["9.9838097 10.9438362"],
    ["10.8212212 10.5939013"],
    ["10.9189774 10.7821363"],
    ["10.0745650 8.0106483"],
    ["10.6198257 9.9438713"],
    ["9.8442045 8.5292476"],
    ["9.5218499 10.4179416"],
])

data = BatchOperator.fromDataframe(df, schemaStr='features string')

gmm = GaussianMixture() \
    .setPredictionCol("cluster_id") \
    .setVectorCol("features") \
    .setPredictionDetailCol("cluster_detail") \
    .setEpsilon(0.)

gmm.fit(data).transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.clustering.GaussianMixture;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GaussianMixtureTest {
    @Test
    public void testGaussianMixture() throws Exception {
        List <Row> df_data = Arrays.asList(
```

```
            Row.of("-0.6264538 0.1836433"),
            Row.of("-0.8356286 1.5952808"),
            Row.of("0.3295078 -0.8204684"),
            Row.of("0.4874291 0.7383247"),
            Row.of("0.5757814 -0.3053884"),
            Row.of("1.5117812 0.3898432"),
            Row.of("-0.6212406 -2.2146999"),
            Row.of("11.1249309 9.9550664"),
            Row.of("9.9838097 10.9438362"),
            Row.of("10.8212212 10.5939013"),
            Row.of("10.9189774 10.7821363"),
            Row.of("10.0745650 8.0106483"),
            Row.of("10.6198257 9.9438713"),
            Row.of("9.8442045 8.5292476"),
            Row.of("9.5218499 10.4179416")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "features
string");
        GaussianMixture gmm = new GaussianMixture()
            .setPredictionCol("cluster_id")
            .setVectorCol("features")
            .setPredictionDetailCol("cluster_detail")
            .setEpsilon(0.);
        gmm.fit(data).transform(data).print();
    }
}
```

运行结果

features	cluster_id	cluster_detail
-0.6264538 0.1836433	0	1.0 4.275273913968281E-92
-0.8356286 1.5952808	0	1.0 1.0260377730239899E-92
0.3295078 -0.8204684	0	1.0 1.0970173367545207E-80
0.4874291 0.7383247	0	1.0 3.302173132311E-75
0.5757814 -0.3053884	0	1.0 3.1638113605165424E-76
1.5117812 0.3898432	0	1.0 2.101805230873172E-62
-0.6212406 -2.2146999	0	1.0 6.772270268600749E-97
11.1249309 9.9550664	1	3.156783801247968E-56 1.0
9.9838097 10.9438362	1	1.9024447346702425E-51 1.0
10.8212212 10.5939013	1	2.800973098729604E-56 1.0

10.9189774 10.7821363	1	1.7209132744891298E-57 1.0
10.0745650 8.0106483	1	2.8642696635130495E-43 1.0
10.6198257 9.9438713	1	5.773273991940433E-53 1.0
9.8442045 8.5292476	1	2.5273123050925483E-43 1.0
9.5218499 10.4179416	1	1.7314580596767853E-46 1.0

经纬度K均值聚类 (GeoKMeans)

Java 类名：com.alibaba.alink.pipeline.clustering.GeoKMeans

Python 类名：GeoKMeans

功能介绍

KMeans 是一个经典的聚类算法。

基本思想是：以空间中k个点为中心进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。

本组件主要针对经纬度距离做Kmeans聚类，包括经纬度KMeans，经纬度KMeans预测, 经纬度KMeans流式预测。

经纬度距离（https://en.wikipedia.org/wiki/Haversine_formula)

$$d = 2r \arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)}\right)$$

$$\operatorname{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

输入数据中的经度和纬度使用 `度数` 表示，得到的距离单位为千米(km)。

参数说明

名称	中文名称	描述	类型	是必须
latitudeCol	经度列名	经度列名	String	✓
longitudeCol	纬度列名	纬度列名	String	✓
predictionCol	预测结果列名	预测结果列名	String	✓
epsilon	收敛阈值	当两轮迭代的中心点距离小于epsilon时，算法收敛。	Double	
initMode	中心点初始化方法	初始化中心点的方法，支持"K_MEANS_PARALLEL"和"RANDOM"	String	

initSteps	k-means++初始化迭代步数	k-means初始化中心点时迭代的步数	Integer	
k	聚类中心点数量	聚类中心点数量	Integer	
maxIter	最大迭代步数	最大迭代步数，默认为 50。	Integer	
modelFilePath	模型的文件路径	模型的文件路径	String	
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean	
predictionDetailCol	预测详细信息列名	预测详细信息列名	String	
predictionDistanceCol	预测距离列名	预测距离列名	String	
randomSeed	随机数种子	随机数种子	Integer	
reservedCols	算法保留列名	算法保留列	String[]	
numThreads	组件多线程线程个数	组件多线程线程个数	Integer	
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String	
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer	
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd
```

```
useLocalEnv(1)

df = pd.DataFrame([
    [0, 0],
    [8, 8],
    [1, 2],
    [9, 10],
    [3, 1],
    [10, 7]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='f0 long, f1 long')

kmeans = GeoKMeans()\
    .setLongitudeCol("f0")\
    .setLatitudeCol("f1")\
    .setK(2)\
    .setPredictionCol("pred")

kmeans.fit(inOp1).transform(inOp1).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.clustering.GeoKMeans;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class GeoKMeansTest {
    @Test
    public void testGeoKMeans() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, 0),
            Row.of(8, 8),
            Row.of(1, 2),
            Row.of(9, 10),
            Row.of(3, 1),
            Row.of(10, 7)
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "f0 int, f1 int");
        GeoKMeans kmeans = new GeoKMeans()
            .setLongitudeCol("f0")
```

```
            .setLatitudeCol("f1")
            .setK(2)
            .setPredictionCol("pred");
        kmeans.fit(inOp1).transform(inOp1).print();
    }
}
```

运行结果

f0	f1	pred
0	0	1
8	8	0
1	2	1
9	10	0
3	1	1
10	7	0

K均值聚类 (KMeans)

Java 类名：com.alibaba.alink.pipeline.clustering.KMeans

Python 类名：KMeans

功能介绍

KMeans 是一个经典的聚类算法。

基本思想是：以空间中k个点为中心进行聚类，对最靠近他们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。

参数说明

名称	中文名称	描述	类型	是必须
predictionCol	预测结果列名	预测结果列名	String	✓
vectorCol	向量列名	向量列对应的列名	String	✓
distanceType	距离度量方式	聚类使用的距离类型	String	
epsilon	收敛阈值	当两轮迭代的中心点距离小于epsilon时，算法收敛。	Double	
initMode	中心点初始化方法	初始化中心点的方法，支持"K_MEANS_PARALLEL"和"RANDOM"	String	
initSteps	k-means++初始化迭代步数	k-means初始化中心点时迭代的步数	Integer	
k	聚类中心点数量	聚类中心点数量	Integer	
maxIter	最大迭代步数	最大迭代步数，默认为 50。	Integer	
modelFilePath	模型的文件路径	模型的文件路径	String	

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean	
predictionDetailCol	预测详细信息列名	预测详细信息列名	String	
predictionDistanceCol	预测距离列名	预测距离列名	String	
randomSeed	随机数种子	随机数种子	Integer	
reservedCols	算法保留列名	算法保留列	String[]	
numThreads	组件多线程线程个数	组件多线程线程个数	Integer	
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String	
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer	
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String	

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "0.1,0.1,0.1"],
    [2, "0.2,0.2,0.2"],
    [3, "9 9 9"],
    [4, "9.1 9.1 9.1"],
    [5, "9.2 9.2 9.2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
```

```
kmeans = KMeans()\
    .setVectorCol("vec")\
    .setK(2)\
    .setPredictionCol("pred")

kmeans.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.clustering.KMeans;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class KMeansTest {
    @Test
    public void testKMeans() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "0.1,0.1,0.1"),
            Row.of(2, "0.2,0.2,0.2"),
            Row.of(3, "9 9 9"),
            Row.of(4, "9.1 9.1 9.1"),
            Row.of(5, "9.2 9.2 9.2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, vec
string");
        KMeans kmeans = new KMeans()
            .setVectorCol("vec")
            .setK(2)
            .setPredictionCol("pred");
        kmeans.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

预测结果

id	vec	pred
0	0 0 0	1

1	0.1,0.1,0.1	1
2	0.2,0.2,0.2	1
3	9 9 9	0
4	9.1 9.1 9.1	0
5	9.2 9.2 9.2	0

Kmodes训练 (KModes)

Java 类名：com.alibaba.alink.pipeline.clustering.KModes

Python 类名：KModes

功能介绍

KModes是一种用于离散数据/分类数据(categorical data)的聚类算法。 基本思想是：把n个对象分为k个簇，使簇内具有较小的的相异度(或者称距离)。 距离计算方法：两个字符串比较，相同为0，不同为1。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
featureCols	特征列名	特征列名，必选	String[]	✓		
predictionCol	预测结果列名	预测结果列名	String	✓		
k	聚类中心点数量	聚类中心点数量	Integer			2
modelFilePath	模型的文件路径	模型的文件路径	String			null

numIter	迭代次数	迭代次数，默认为10	Integer			10
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["pc", "Hp.com", 1],
    ["camera", "Hp.com", 1],
    ["digital camera", "Hp.com", 1],
    ["camera", "BestBuy.com", 1],
    ["digital camera", "BestBuy.com", 1],
    ["tv", "BestBuy.com", 1],
    ["flower", "Teleflora.com", 1],
    ["flower", "Orchids.com", 1]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='f0 string, f1 string')

kmodes = KModes()\
    .setFeatureCols(["f0", "f1"])\
    .setK(2)\
```

```
    .setPredictionCol("pred")

kmodes.fit(inOp1).transform(inOp1).print()
```

运行结果

```
               f0             f1  pred
0              pc         Hp.com     1
1          camera         Hp.com     1
2  digital camera         Hp.com     1
3          camera    BestBuy.com     0
4  digital camera    BestBuy.com     0
5              tv    BestBuy.com     0
6          flower  Teleflora.com     0
7          flower    Orchids.com     0
```

LDA (Lda)

Java 类名：com.alibaba.alink.pipeline.clustering.Lda

Python 类名：Lda

功能介绍

LDA是一种文档主题生成模型。LDA是一种非监督机器学习技术，可以用来识别大规模文档集（document collection）或语料库（corpus）中潜藏的主题信息。它采用了词袋（bag of words）的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
predictionCol	预测结果列名	预测结果列名	String	✓	
selectedCol	选中的列名	计算列对应的列名	String	✓	
topicNum	主题个数	主题个数	Integer	✓	
alpha	文章的超参	文章的超参	Double		
beta	词的超参	词的超参	Double		
learningDecay	衰减率	衰减率	Double		
method	优化方法	优化方法, 包含"em"和"online"两种。	String		"Online", "EM"

modelFilePath	模型的文件路径	模型的文件路径	String			
numIter	迭代次数	迭代次数，默认为10	Integer			
onlineLearningOffset	偏移量	偏移量	Double			
optimizeDocConcentration	是否优化alpha	是否优化alpha	Boolean			
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			
predictionDetailCol	预测详细信息列名	预测详细信息列名	String			
randomSeed	随机数种子	随机数种子	Integer			
reservedCols	算法保留列名	算法保留列	String[]			
subsamplingRate	采样率	采样率	Double			
vocabSize	字典库大小	字典库大小，如果总词数目大于这个值，那个文档频率低的词会被过滤掉。	Integer			
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    ["a b b c c c c c e e f f f g h k k k"],
    ["a b b b d e e e h h k"],
    ["a b b b b c f f f f g g g g g g g g i j j"],
    ["a a b d d d g g g g g i i j j j k k k k k k k k k"],
    ["a a a b c d d d d d d d d d e e e g g j k k k"],
    ["a a a a b b d d d e e e e f f f f f g h i j j j j"],
    ["a a b d d d g g g g g i i j j k k k k k k k k k"],
    ["a b c d d d d d d d d d e e f g g j k k k"],
    ["a a a a b b b b d d d e e e e f f g h h h"],
    ["a a b b b b b b b b b c c e e e e g g i i j j j j j j j k k"],
    ["a b c d d d d d d d d d f f g g j j j k k k"],
    ["a a a a b e e e e f f f f f g h h h j"]
])

data = BatchOperator.fromDataframe(df, schemaStr="doc string")

lda = Lda()\
    .setSelectedCol("doc")\
    .setTopicNum(6)\
    .setMethod("online")\
    .setPredictionCol("pred")
```

```
lda.fit(data).transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.clustering.Lda;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class LdaTest {
    @Test
    public void testLda() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of("a b b c c c c c c e e f f f g h k k k"),
            Row.of("a b b b d e e e h h k"),
            Row.of("a b b b b c f f f f g g g g g g g g i j j"),
            Row.of("a a b d d d g g g g g i i j j j k k k k k k k k k"),
            Row.of("a a a b c d d d d d d d d d e e e g g j k k k"),
            Row.of("a a a a b b d d d e e e e f f f f f g h i j j j j"),
            Row.of("a a b d d d g g g g g i i j j k k k k k k k k k"),
            Row.of("a b c d d d d d d d d d e e f g g j k k k"),
            Row.of("a a a a b b b b d d d e e e e f f g h h h"),
            Row.of("a a b b b b b b b b c c e e e g g i i j j j j j j j k k"),
            Row.of("a b c d d d d d d d d d f f g g j j j k k k"),
            Row.of("a a a a b e e e e f f f f f g h h h j")
        );
        BatchOperator <?> data = new MemSourceBatchOp(df, "doc string");
        Lda lda = new Lda()
            .setSelectedCol("doc")
            .setTopicNum(6)
            .setMethod("online")
            .setPredictionCol("pred");
        lda.fit(data).transform(data).print();
    }
}
```

运行结果

模型结果

model_id	
0	{"logPerplexity":"22.332946259667825","betaArray":"[0.2,0.2,0.2,0.2,0.2]","logLikelihood":"-915.
1048576	{"m":5,"n":11,"data": [6135.5227952852865,7454.918734235136,6569.887273287071,7647.590029783137,7459.37
2097152	{"f0":"d","f1":0.36772478012531734,"f2":0}
3145728	{"f0":"k","f1":0.36772478012531734,"f2":1}
4194304	{"f0":"g","f1":0.08004270767353636,"f2":2}
5242880	{"f0":"b","f1":0.0,"f2":3}
6291456	{"f0":"a","f1":0.0,"f2":4}
7340032	{"f0":"e","f1":0.36772478012531734,"f2":5}
8388608	{"f0":"j","f1":0.26236426446749106,"f2":6}
9437184	{"f0":"f","f1":0.4855078157817008,"f2":7}
10485760	{"f0":"c","f1":0.6190392084062235,"f2":8}
11534336	{"f0":"h","f1":0.7731898882334817,"f2":9}
12582912	{"f0":"i","f1":0.7731898882334817,"f2":10}

预测结果

doc	pred
a b b b d e e e h h k	1
a a b d d d g g g g g i i j j j k k k k k k k k k	3
a a a a b b d d d e e e e f f f f f g h i j j j j	3
a a b d d d g g g g g i i j j k k k k k k k k k	1
a a a a b b b b d d d e e e e f f g h h h	3
a b c d d d d d d d d d f f g g j j j k k k	3
a b b c c c c c c e e f f f g h k k k	2
a b b b b c f f f f g g g g g g g g g i j j	0
a a a b c d d d d d d d d d e e e g g j k k k	3
a b c d d d d d d d d d e e f g g j k k k	3
a a b b b b b b b b c c e e e e g g i i j j j j j j j k k	3

a a a a b e e e e f f f f f g h h h j	0

ALS：ItemsPerUser推荐 (AlsItemsPerUserRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.AlsItemsPerUserRecommender

Python 类名：AlsItemsPerUserRecommender

功能介绍

使用ALS (Alternating Lease Square）训练的模型为 user 实时推荐 items。这里的ALS模型可以是隐式模型，也可以是显式模型，输出格式是MTable。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
recommCol	推荐结果列名	推荐结果列名	String	✓	
userCol	User列列名	User列列名	String	✓	
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')
als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)

alsRec = AlsItemsPerUserRecommender() \

.setUserCol("user").setRecommCol("rec").setK(1).setReservedCols(["user"]).setMo
delData(model)

alsRec.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.AlsItemsPerUserRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsItemsPerUserRecommenderTest {
    @Test
    public void testAlsItemsPerUserRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
```

```
        BatchOperator <?> als = new
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
        AlsItemsPerUserRecommender alsRec = new AlsItemsPerUserRecommender()

.setUserCol("user").setRecommCol("rec").setK(1).setReservedCols("user").setMode
lData(model);
        alsRec.transform(data).print();
    }
}
```

运行结果

user	rec
1	{"object":"[1]","rate":"[0.5796224474906921]"}
2	{"object":"[2]","rate":"[0.7668506503105164]"}
2	{"object":"[2]","rate":"[0.7668506503105164]"}
4	{"object":"[1]","rate":"[0.5744813084602356]"}
4	{"object":"[1]","rate":"[0.5744813084602356]"}
4	{"object":"[1]","rate":"[0.5744813084602356]"}

ALS：打分推荐 (AlsRateRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.AlsRateRecommender

Python 类名：AlsRateRecommender

功能介绍

使用ALS (Alternating Lease Square）训练的模型对（user，item）输入流对进行实时评分预测。这里的ALS模型可以是隐式模型，也可以是显式模型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
reservedCols	算法保留列名	算法保留列	String[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')
```

```
als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)

alsRate =
AlsRateRecommender().setUserCol("user").setItemCol("item").setRecommCol("rating
") \
        .setRecommCol("predicted_rating").setModelData(model)

alsRate.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.AlsRateRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsRateRecommenderTest {
    @Test
    public void testAlsRateRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> als = new
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
        AlsRateRecommender alsRate = new
AlsRateRecommender().setUserCol("user").setItemCol("item").setRecommCol(
            "rating")
```

```
            .setRecommCol("predicted_rating").setModelData(model);
        alsRate.transform(data).print();
    }
}
```

运行结果

user	item	rating	predicted_rating
1	1	0.6000	0.5810
2	2	0.8000	0.7669
2	3	0.6000	0.5809
4	1	0.6000	0.5753
4	2	0.3000	0.2989
4	3	0.4000	0.3833

ALS：相似items推荐 (AlsSimilarItemsRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.AlsSimilarItemsRecommender

Python 类名：AlsSimilarItemsRecommender

功能介绍

使用ALS (Alternating Lease Square）训练的模型对相似的item的进行实时推荐。这里的ALS模型可以是隐式模型，也可以是显式模型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
itemCol	Item列列名	Item列列名	String	✓	
recommCol	推荐结果列名	推荐结果列名	String	✓	
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')
```

```
sdata = StreamOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)
alsRec = AlsSimilarItemsRecommender().setModelData(model) \
    .setItemCol("item").setRecommCol("rec").setK(1).setReservedCols(["item"])

alsRec.transform(data).print();
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.recommendation.AlsSimilarItemsRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsSimilarItemsRecommenderTest {
    @Test
    public void testAlsSimilarItemsRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        StreamOperator <?> sdata = new MemSourceStreamOp(df_data, "user int,
item int, rating double");
        BatchOperator <?> als = new
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
```

```
        AlsSimilarItemsRecommender alsRec = new
AlsSimilarItemsRecommender().setModelData(model)

.setItemCol("item").setRecommCol("rec").setK(1).setReservedCols("item");
        alsRec.transform(data).print();
    }
}
```

运行结果

item	rec
1	{"object":"[3]","score":"[0.8821980357170105]"}
2	{"object":"[3]","score":"[0.9917739629745483]"}
3	{"object":"[2]","score":"[0.9917739629745483]"}
1	{"object":"[3]","score":"[0.8821980357170105]"}
2	{"object":"[3]","score":"[0.9917739629745483]"}
3	{"object":"[2]","score":"[0.9917739629745483]"}

ALS：相似users推荐 (AlsSimilarUsersRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.AlsSimilarUsersRecommender

Python 类名：AlsSimilarUsersRecommender

功能介绍

使用ALS (Alternating Lease Square）训练的模型对相似的user的进行实时推荐。这里的ALS模型可以是隐式模型，也可以是显式模型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
recommCol	推荐结果列名	推荐结果列名	String	✓	
userCol	User列列名	User列列名	String	✓	
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		

reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])
```

```
data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')
sdata = StreamOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)
alsRec = AlsSimilarUsersRecommender().setModelData(model) \
    .setUserCol("user").setRecommCol("rec").setK(1).setReservedCols(["user"])

alsRec.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.recommendation.AlsSimilarUsersRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsSimilarUsersRecommenderTest {
    @Test
    public void testAlsSimilarUsersRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        StreamOperator <?> sdata = new MemSourceStreamOp(df_data, "user int,
item int, rating double");
        BatchOperator <?> als = new
```

```
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
        AlsSimilarUsersRecommender alsRec = new
AlsSimilarUsersRecommender().setModelData(model)

.setUserCol("user").setRecommCol("rec").setK(1).setReservedCols("user");
        alsRec.transform(data).print();
    }
}
```

运行结果

user	rec
1	{"object":"[4]","score":"[0.2515771985054016]"}
2	{"object":"[1]","score":"[0.17212671041488647]"}
2	{"object":"[1]","score":"[0.17212671041488647]"}
4	{"object":"[1]","score":"[0.2515771985054016]"}
4	{"object":"[1]","score":"[0.2515771985054016]"}
4	{"object":"[1]","score":"[0.2515771985054016]"}

ALS：UsersPerItem推荐 (AlsUsersPerItemRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.AlsUsersPerItemRecommender

Python 类名：AlsUsersPerItemRecommender

功能介绍

使用ALS (Alternating Lease Square）训练的模型为item 实时推荐users。这里的ALS模型可以是隐式模型，也可以是显式模型。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
itemCol	Item列列名	Item列列名	String	✓	
recommCol	推荐结果列名	推荐结果列名	String	✓	
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

als =
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating") \
    .setNumIter(10).setRank(10).setLambda(0.01)

model = als.linkFrom(data)
alsRec = AlsUsersPerItemRecommender().setModelData(model) \
    .setItemCol("item").setRecommCol("rec").setK(1).setReservedCols(["item"])

alsRec.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.AlsTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.AlsUsersPerItemRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class AlsUsersPerItemRecommenderTest {
    @Test
    public void testAlsUsersPerItemRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> als = new
AlsTrainBatchOp().setUserCol("user").setItemCol("item").setRateCol("rating")
```

```
            .setNumIter(10).setRank(10).setLambda(0.01);
        BatchOperator model = als.linkFrom(data);
        AlsUsersPerItemRecommender alsRec = new
AlsUsersPerItemRecommender().setModelData(model)

.setItemCol("item").setRecommCol("rec").setK(1).setReservedCols("item");
        alsRec.transform(data).print();
    }
}
```

运行结果

user	rec
1	{"object":"[1]","rate":"[0.5796224474906921]"}
2	{"object":"[2]","rate":"[0.7668506503105164]"}
3	{"object":"[2]","rate":"[0.5810791850090027]"}
1	{"object":"[1]","rate":"[0.5796224474906921]"}
2	{"object":"[2]","rate":"[0.7668506503105164]"}
3	{"object":"[2]","rate":"[0.5810791850090027]"}

FM：ItemsPerUser推荐 (FmItemsPerUserRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.FmItemsPerUserRecommender

Python 类名：FmItemsPerUserRecommender

功能介绍

使用Fm推荐模型，为user推荐item list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
recommCol	推荐结果列名	推荐结果列名	String	✓	
userCol	User列列名	User列列名	String	✓	
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = FmRecommTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setNumFactor(20)\
    .setRateCol("rating").linkFrom(data);

rec = FmItemsPerUserRecommender()\
    .setUserCol("user")\
    .setK(1).setReservedCols(["user"])\
    .setRecommCol("prediction_result")\
    .setModelData(model);

rec.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.FmRecommTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.FmItemsPerUserRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmItemsPerUserRecommenderTest {
    @Test
    public void testFmItemsPerUserRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new FmRecommTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setNumFactor(20)
            .setRateCol("rating").linkFrom(data);
        FmItemsPerUserRecommender rec = new FmItemsPerUserRecommender()
            .setUserCol("user")
            .setK(1).setReservedCols("user")
            .setRecommCol("prediction_result")
            .setModelData(model);
        rec.transform(data).print();
    }
}
```

运行结果

user	prediction_result
1	{"object":"[1]","rate":"[0.5829579830169678]"}
2	{"object":"[2]","rate":"[0.576914370059967]"}
2	{"object":"[2]","rate":"[0.576914370059967]"}
4	{"object":"[1]","rate":"[0.5055253505706787]"}
4	{"object":"[1]","rate":"[0.5055253505706787]"}
4	{"object":"[1]","rate":"[0.5055253505706787]"}

FM：打分推荐 (FmRateRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.FmRateRecommender

Python 类名：FmRateRecommender

功能介绍

Fm 推荐打分是使用Fm推荐模型，预测user对item的评分。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
reservedCols	算法保留列名	算法保留列	String[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')
```

```
model = FmRecommTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setNumFactor(20)\
    .setRateCol("rating").linkFrom(data);

fmRate = FmRateRecommender()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRecommCol("prediction_result")\
    .setModelData(model)

fmRate.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.FmRecommTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.FmRateRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmRateRecommenderTest {
    @Test
    public void testFmRateRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new FmRecommTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setNumFactor(20)
            .setRateCol("rating").linkFrom(data);
        FmRateRecommender fmRate = new FmRateRecommender()
```

```
            .setUserCol("user")
            .setItemCol("item")
            .setRecommCol("prediction_result")
            .setModelData(model);
        fmRate.transform(data).print();
    }
}
```

运行结果

user	item	rating	prediction_result
1	1	0.6	0.582958
2	2	0.8	0.576914
2	3	0.6	0.508942
4	1	0.6	0.505525
4	2	0.3	0.372908
4	3	0.4	0.347927

FM：UsersPerItem推荐 (FmUsersPerItemRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.FmUsersPerItemRecommender

Python 类名：FmUsersPerItemRecommender

功能介绍

使用Fm推荐模型，为item推荐user list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
itemCol	Item列列名	Item列列名	String	✓	
recommCol	推荐结果列名	推荐结果列名	String	✓	
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = FmRecommTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setNumFactor(20)\
    .setRateCol("rating").linkFrom(data);

rec = FmUsersPerItemRecommender()\
    .setItemCol("item")\
    .setK(1).setReservedCols(["item"])\
    .setRecommCol("prediction_result")\
    .setModelData(model);

rec.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.FmRecommTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.FmUsersPerItemRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class FmUsersPerItemRecommenderTest {
    @Test
    public void testFmUsersPerItemRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new FmRecommTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setNumFactor(20)
            .setRateCol("rating").linkFrom(data);
        FmUsersPerItemRecommender rec = new FmUsersPerItemRecommender()
            .setItemCol("item")
            .setK(1).setReservedCols("item")
            .setRecommCol("prediction_result")
            .setModelData(model);
        rec.transform(data).print();
    }
}
```

运行结果

item	prediction_result
1	{"object":"[1]","rate":"[0.5829579830169678]"}
2	{"object":"[2]","rate":"[0.576914370059967]"}
3	{"object":"[1]","rate":"[0.5055253505706787]"}
1	{"object":"[1]","rate":"[0.5829579830169678]"}
2	{"object":"[2]","rate":"[0.576914370059967]"}
3	{"object":"[1]","rate":"[0.5055253505706787]"}

ItemCf：ItemsPerUser推荐 (ItemCfItemsPerUserRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.ItemCfItemsPerUserRecommender

Python 类名：ItemCfItemsPerUserRecommender

功能介绍

用ItemCF模型 为user推荐item list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
recommCol	推荐结果列名	推荐结果列名	String	✓	
userCol	User列列名	User列列名	String	✓	
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = ItemCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = ItemCfItemsPerUserRecommender()\
    .setUserCol("user")\
    .setReservedCols(["user"])\
    .setK(1)\
    .setRecommCol("prediction_result")\
    .setModelData(model)

predictor.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.ItemCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.ItemCfItemsPerUserRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ItemCfItemsPerUserRecommenderTest {
    @Test
    public void testItemCfItemsPerUserRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new ItemCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        ItemCfItemsPerUserRecommender predictor = new
ItemCfItemsPerUserRecommender()
            .setUserCol("user")
            .setReservedCols("user")
            .setK(1)
            .setRecommCol("prediction_result")
            .setModelData(model);
        predictor.transform(data).print();
    }
}
```

运行结果

user	prediction_result
1	{"item":"[3]","score":"[0.23533936216582085]"}
2	{"item":"[3]","score":"[0.38953648389671724]"}
2	{"item":"[3]","score":"[0.38953648389671724]"}
4	{"item":"[2]","score":"[0.17950184794838112]"}
4	{"item":"[2]","score":"[0.17950184794838112]"}
4	{"item":"[2]","score":"[0.17950184794838112]"}

ItemCf：打分推荐 (ItemCfRateRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.ItemCfRateRecommender

Python 类名：ItemCfRateRecommender

功能介绍

ItemCF 打分是使用ItemCF模型，预测user对item的评分。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
reservedCols	算法保留列名	算法保留列	String[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')
```

```
sdata = StreamOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = ItemCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = ItemCfRateRecommender()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRecommCol("prediction_result")\
    .setModelData(model)

predictor.transform(sdata).print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.ItemCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import com.alibaba.alink.pipeline.recommendation.ItemCfRateRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ItemCfRateRecommenderTest {
    @Test
    public void testItemCfRateRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        StreamOperator <?> sdata = new MemSourceStreamOp(df_data, "user int,
item int, rating double");
```

```
        BatchOperator <?> model = new ItemCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        ItemCfRateRecommender predictor = new ItemCfRateRecommender()
            .setUserCol("user")
            .setItemCol("item")
            .setRecommCol("prediction_result")
            .setModelData(model);
        predictor.transform(sdata).print();
        StreamOperator.execute();
    }
}
```

运行结果

user	item	rating	prediction_result
1	1	0.6000	0.0000
4	1	0.6000	0.3612
2	3	0.6000	0.8000
4	2	0.3000	0.4406
2	2	0.8000	0.6000
4	3	0.4000	0.3861

ItemCf：相似items推荐 (ItemCfSimilarItemsRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.ItemCfSimilarItemsRecommender

Python 类名：ItemCfSimilarItemsRecommender

功能介绍

用itemCF模型为item推荐相似的item list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
itemCol	Item列列名	Item列列名	String	✓	
recommCol	推荐结果列名	推荐结果列名	String	✓	
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')
```

```
model = ItemCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = ItemCfSimilarItemsRecommender()\
    .setItemCol("item")\
    .setReservedCols(["item"])\
    .setK(1)\
    .setRecommCol("prediction_result")\
    .setModelData(model)

predictor.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.ItemCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.ItemCfSimilarItemsRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ItemCfSimilarItemsRecommenderTest {
    @Test
    public void testItemCfSimilarItemsRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new ItemCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        ItemCfSimilarItemsRecommender predictor = new
ItemCfSimilarItemsRecommender()
```

```
            .setItemCol("item")
            .setReservedCols("item")
            .setK(1)
            .setRecommCol("prediction_result")
            .setModelData(model);
        predictor.transform(data).print();
    }
}
```

运行结果

item	prediction_result
1	{"item":"[3]","similarities":"[0.3922322702763681]"}
2	{"item":"[3]","similarities":"[0.9738412097417931]"}
3	{"item":"[2]","similarities":"[0.9738412097417931]"}
1	{"item":"[3]","similarities":"[0.3922322702763681]"}
2	{"item":"[3]","similarities":"[0.9738412097417931]"}
3	{"item":"[2]","similarities":"[0.9738412097417931]"}

ItemCf：UsersPerItem推荐 (ItemCfUsersPerItemRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.ItemCfUsersPerItemRecommender

Python 类名：ItemCfUsersPerItemRecommender

功能介绍

用ItemCF模型 为item推荐user list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
itemCol	Item列列名	Item列列名	String	✓	
recommCol	推荐结果列名	推荐结果列名	String	✓	
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = ItemCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = ItemCfUsersPerItemRecommender()\
    .setItemCol("item")\
    .setReservedCols(["item"])\
    .setK(1)\
    .setRecommCol("prediction_result")\
    .setModelData(model)

predictor.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.ItemCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.ItemCfUsersPerItemRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class ItemCfUsersPerItemRecommenderTest {
    @Test
    public void testItemCfUsersPerItemRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new ItemCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        ItemCfUsersPerItemRecommender predictor = new
ItemCfUsersPerItemRecommender()
            .setItemCol("item")
            .setReservedCols("item")
            .setK(1)
            .setRecommCol("prediction_result")
            .setModelData(model);
        predictor.transform(data).print();
    }
}
```

运行结果

item	prediction_result
1	{"user":"[2]","score":"[0.21698238771519462]"}
2	{"user":"[2]","score":"[0.29215236292253793]"}
3	{"user":"[2]","score":"[0.38953648389671724]"}
1	{"user":"[2]","score":"[0.21698238771519462]"}
2	{"user":"[2]","score":"[0.29215236292253793]"}
3	{"user":"[2]","score":"[0.38953648389671724]"}

推荐组件：精排 (RecommendationRanking)

Java 类名：com.alibaba.alink.pipeline.recommendation.RecommendationRanking

Python 类名：RecommendationRanking

功能介绍

该组件功能是对召回的结果进行排序，并输出排序后的TopK个object，此处排序算法用户可以通过创建PipelineModel的方式定制，具体使用方式参见代码示例。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
mTableCol	MTable列名	召回列表列	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
rankingCol	用来排序的得分列	用来排序的得分列	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
topN	前N的数据	挑选最近的N个数据	Integer		[1, +inf)	10
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null

modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String			nul

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

import pandas as pd

data = pd.DataFrame([["u6", "0.0 1.0", 0.0, 1.0, 1, "{\"data\":{\"iid\":
[18,19,88]},\"schema\":\"iid INT\"}"]])
predData = BatchOperator.fromDataframe(data, schemaStr='uid string, uf string,
f0 double, f1 double, labels int, ilist string')
predData = predData.link(ToMTableBatchOp().setSelectedCol("ilist"))
data = pd.DataFrame([
            ["u0", "1.0 1.0", 1.0, 1.0, 1, 18],
            ["u1", "1.0 1.0", 1.0, 1.0, 0, 19],
            ["u2", "1.0 0.0", 1.0, 0.0, 1, 88],
            ["u3", "1.0 0.0", 1.0, 0.0, 0, 18],
            ["u4", "0.0 1.0", 0.0, 1.0, 1, 88],
            ["u5", "0.0 1.0", 0.0, 1.0, 0, 19],
            ["u6", "0.0 1.0", 0.0, 1.0, 1, 88]]);
trainData = BatchOperator.fromDataframe(data, schemaStr='uid string, uf string,
f0 double, f1 double, labels int, iid string')
oneHotCols = ["uid", "f0", "f1", "iid"]
multiHotCols = ["uf"]
pipe = Pipeline() \
    .add( \
        OneHotEncoder() \
            .setSelectedCols(oneHotCols) \
            .setOutputCols(["ovec"])) \
```

```
    .add( \
        MultiHotEncoder().setDelimiter(" ") \
            .setSelectedCols(multiHotCols) \
            .setOutputCols(["mvec"])) \
    .add( \
        VectorAssembler() \
            .setSelectedCols(["ovec", "mvec"]) \
            .setOutputCol("vec")) \
    .add(
        LogisticRegression() \
            .setVectorCol("vec") \
            .setLabelCol("labels") \
            .setReservedCols(["uid", "iid"]) \
            .setPredictionDetailCol("detail") \
            .setPredictionCol("pred")) \
    .add( \
        JsonValue() \
            .setSelectedCol("detail") \
            .setJsonPath(["$.1"]) \
            .setOutputCols(["score"]))
lrModel = pipe.fit(trainData)
rank = RecommendationRanking()\
            .setModelData(lrModel.save())\
            .setMTableCol("ilist")\
            .setOutputCol("il")\
            .setTopN(2)\
            .setRankingCol("score")\
            .setReservedCols(["uid", "labels"])
rank.transform(predData).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.Pipeline;
import com.alibaba.alink.pipeline.classification.LogisticRegression;
import com.alibaba.alink.pipeline.dataproc.JsonValue;
import com.alibaba.alink.pipeline.dataproc.vector.VectorAssembler;
import com.alibaba.alink.pipeline.feature.MultiHotEncoder;
import com.alibaba.alink.pipeline.feature.OneHotEncoder;
import org.junit.Test;

import java.util.Arrays;

public class RecommendationRankingTest {
```

```
    @Test
    public void test() throws Exception {

        Row[] predArray = new Row[] {
            Row.of("u6", "0.0 1.0", 0.0, 1.0, 1, "{\"data\":{\"iid\":
[18,19,88]},"
                + "\"schema\":\"iid INT\"}")
        };

        Row[] trainArray = new Row[] {
            Row.of("u0", "1.0 1.0", 1.0, 1.0, 1, 18),
            Row.of("u1", "1.0 1.0", 1.0, 1.0, 0, 19),
            Row.of("u2", "1.0 0.0", 1.0, 0.0, 1, 88),
            Row.of("u3", "1.0 0.0", 1.0, 0.0, 1, 18),
            Row.of("u4", "0.0 1.0", 0.0, 1.0, 1, 88),
            Row.of("u5", "0.0 1.0", 0.0, 1.0, 1, 19),
            Row.of("u6", "0.0 1.0", 0.0, 1.0, 1, 88)
        };
        BatchOperator <?> trainData = new
MemSourceBatchOp(Arrays.asList(trainArray),
                new String[] {"uid", "uf", "f0", "f1", "labels", "iid"});
        BatchOperator <?> predData =  new
MemSourceBatchOp(Arrays.asList(predArray),
                new String[] {"uid", "uf", "f0", "f1", "labels", "ilist"});

        String[] oneHotCols = new String[] {"uid", "f0", "f1", "iid"};
        String[] multiHotCols = new String[] {"uf"};

        Pipeline pipe = new Pipeline()
            .add(
                new OneHotEncoder()
                    .setSelectedCols(oneHotCols)
                    .setOutputCols("ovec"))
            .add(
                new MultiHotEncoder().setDelimiter(" ")
                    .setSelectedCols(multiHotCols)
                    .setOutputCols("mvec"))
            .add(
                new VectorAssembler()
                    .setSelectedCols("ovec", "mvec")
                    .setOutputCol("vec"))
            .add(
                new LogisticRegression()
                    .setVectorCol("vec")
                    .setLabelCol("labels")
                    .setReservedCols("uid", "iid")
                    .setPredictionDetailCol("detail")
```

```
                    .setPredictionCol("pred"))
            .add(
                new JsonValue()
                    .setSelectedCol("detail")
                    .setJsonPath("$.1")
                    .setOutputCols("score"));
        RecommendationRanking rank = new RecommendationRanking()
            .setModelData(pipe.fit(trainData).save())
            .setMTableCol("ilist")
            .setOutputCol("ilist")
            .setTopN(2)
            .setRankingCol("score")
            .setReservedCols("uid", "labels");
        rank.transform(predData).print();
    }
}
```

运行结果

uid	uf	f0	f1	labels	ilist
u6	0.0 1.0	0.0000	1.0000	1	{"data":{"iid":[18,88],"score": [0.9999999999999553,0.9999999999999472]},"schema":" INT,score DOUBLE"}

Swing推荐 (SwingSimilarItemsRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.SwingSimilarItemsRecommender

Python 类名：SwingSimilarItemsRecommender

功能介绍

Swing 是一种被广泛使用的item召回算法

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
itemCol	Item列列名	Item列列名	String	✓	
recommCol	推荐结果列名	推荐结果列名	String	✓	
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
      ["a1", "11L", 2.2],
      ["a1", "12L", 2.0],
      ["a2", "11L", 2.0],
      ["a2", "12L", 2.0],
      ["a3", "12L", 2.0],
      ["a3", "13L", 2.0],
      ["a4", "13L", 2.0],
      ["a4", "14L", 2.0],
      ["a5", "14L", 2.0],
      ["a5", "15L", 2.0],
```

```
       ["a6", "15L", 2.0],
       ["a6", "16L", 2.0],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user string, item
string, rating double')

model = SwingTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .linkFrom(data)

predictor = SwingSimilarItemsRecommender()\
    .setItemCol("item")\
    .setRecommCol("prediction_result")\
    .setModelData(model)

predictor.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.pipeline.recommendation.SwingSimilarItemsRecommender;
import com.alibaba.alink.operator.batch.recommendation.SwingTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class SwingSimilarItemsRecommenderTest {
    @Test
    public void testSwingSimilarItemsRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of("a1", "11L", 2.2),
            Row.of("a1", "12L", 2.0),
            Row.of("a2", "11L", 2.0),
            Row.of("a2", "12L", 2.0),
            Row.of("a3", "12L", 2.0),
            Row.of("a3", "13L", 2.0),
            Row.of("a4", "13L", 2.0),
            Row.of("a4", "14L", 2.0),
            Row.of("a5", "14L", 2.0),
            Row.of("a5", "15L", 2.0),
            Row.of("a6", "15L", 2.0),
```

```
            Row.of("a6", "16L", 2.0)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user string,
item string, rating double");
        BatchOperator <?> model = new SwingTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .linkFrom(data);
        SwingSimilarItemsRecommender <?> predictor = new
SwingSimilarItemsRecommender()
            .setItemCol("item")
            .setRecommCol("prediction_result")
            .setModelData(model);
        predictor.transform(data).print();
    }
}
```

运行结果

user	item	rating	prediction_result
a1	11L	2.2000	{"item":"[\"12L\"]","score":"[0.12805642187595367]"}
a1	12L	2.0000	{"item":"[\"11L\"]","score":"[0.11662912368774414]"}
a2	11L	2.0000	{"item":"[\"12L\"]","score":"[0.12805642187595367]"}
a2	12L	2.0000	{"item":"[\"11L\"]","score":"[0.11662912368774414]"}
a3	12L	2.0000	{"item":"[\"11L\"]","score":"[0.11662912368774414]"}
a3	13L	2.0000	null
a4	13L	2.0000	null
a4	14L	2.0000	null
a5	14L	2.0000	null
a5	15L	2.0000	null
a6	15L	2.0000	null
a6	16L	2.0000	null

UserCf：ItemsPerUser推荐 (UserCfItemsPerUserRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.UserCfItemsPerUserRecommender

Python 类名：UserCfItemsPerUserRecommender

功能介绍

用UserCF模型 为user推荐item list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
recommCol	推荐结果列名	推荐结果列名	String	✓	
userCol	User列列名	User列列名	String	✓	
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = UserCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = UserCfItemsPerUserRecommender()\
    .setUserCol("user")\
    .setReservedCols(["user"])\
    .setK(1)\
    .setRecommCol("prediction_result")\
    .setModelData(model)

predictor.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.UserCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.UserCfItemsPerUserRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class UserCfItemsPerUserRecommenderTest {
    @Test
    public void testUserCfItemsPerUserRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new UserCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        UserCfItemsPerUserRecommender predictor = new
UserCfItemsPerUserRecommender()
            .setUserCol("user")
            .setReservedCols("user")
            .setK(1)
            .setRecommCol("prediction_result")
            .setModelData(model);
        predictor.transform(data).print();
    }
}
```

运行结果

user	prediction_result
1	{"item":"[1]","score":"[0.4609327677584255]"}
2	{"item":"[1]","score":"[0.1843731071033702]"}
2	{"item":"[1]","score":"[0.1843731071033702]"}
4	{"item":"[2]","score":"[0.16388720631410686]"}
4	{"item":"[2]","score":"[0.16388720631410686]"}
4	{"item":"[2]","score":"[0.16388720631410686]"}

UserCf：打分推荐 (UserCfRateRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.UserCfRateRecommender

Python 类名：UserCfRateRecommender

功能介绍

UserCF 打分是使用UserCF模型，预测user对item的评分。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
itemCol	Item列列名	Item列列名	String	✓		
recommCol	推荐结果列名	推荐结果列名	String	✓		
userCol	User列列名	User列列名	String	✓		
modelFilePath	模型的文件路径	模型的文件路径	String			null
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			false
reservedCols	算法保留列名	算法保留列	String[]			null

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			null
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			10
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			null

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')
```

```
model = UserCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = UserCfRateRecommender()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRecommCol("prediction_result")\
    .setModelData(model)

predictor.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.UserCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.UserCfRateRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class UserCfRateRecommenderTest {
    @Test
    public void testUserCfRateRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new UserCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        UserCfRateRecommender predictor = new UserCfRateRecommender()
            .setUserCol("user")
            .setItemCol("item")
```

```
            .setRecommCol("prediction_result")
            .setModelData(model);
        predictor.transform(data).print();
    }
}
```

运行结果

user	item	rating	prediction_result
1	1	0.6000	0.6000
2	2	0.8000	0.3000
2	3	0.6000	0.4000
4	1	0.6000	0.6000
4	2	0.3000	0.8000
4	3	0.4000	0.6000

UserCf：相似users推荐 (UserCfSimilarUsersRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.UserCfSimilarUsersRecommender

Python 类名：UserCfSimilarUsersRecommender

功能介绍

用UserCF模型为user推荐相似的user list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
recommCol	推荐结果列名	推荐结果列名	String	✓	
userCol	User列列名	User列列名	String	✓	
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')
```

```
model = UserCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = UserCfSimilarUsersRecommender()\
    .setUserCol("user")\
    .setReservedCols(["user"])\
    .setK(1)\
    .setRecommCol("prediction_result")\
    .setModelData(model)

predictor.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.UserCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.UserCfSimilarUsersRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class UserCfSimilarUsersRecommenderTest {
    @Test
    public void testUserCfSimilarUsersRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new UserCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        UserCfSimilarUsersRecommender predictor = new
UserCfSimilarUsersRecommender()
```

```
            .setUserCol("user")
            .setReservedCols("user")
            .setK(1)
            .setRecommCol("prediction_result")
            .setModelData(model);
        predictor.transform(data).print();
    }
}
```

运行结果

user	prediction_result
1	{"user":"[4]","similarities":"[0.7682212795973759]"}
2	{"user":"[4]","similarities":"[0.6145770236779007]"}
2	{"user":"[4]","similarities":"[0.6145770236779007]"}
4	{"user":"[1]","similarities":"[0.7682212795973759]"}
4	{"user":"[1]","similarities":"[0.7682212795973759]"}
4	{"user":"[1]","similarities":"[0.7682212795973759]"}

UserCf：UsersPerItem推荐 (UserCfUsersPerItemRecommender)

Java 类名：com.alibaba.alink.pipeline.recommendation.UserCfUsersPerItemRecommender

Python 类名：UserCfUsersPerItemRecommender

功能介绍

用UserCF模型 为item推荐user list。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
itemCol	Item列列名	Item列列名	String	✓	
recommCol	推荐结果列名	推荐结果列名	String	✓	
excludeKnown	排除已知的关联	推荐结果中是否排除训练数据中已知的关联	Boolean		
initRecommCol	初始推荐列列名	初始推荐列列名	String		所选列类型为[M_TABLE]
k	推荐TOP数量	推荐TOP数量	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		

overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df_data = pd.DataFrame([
    [1, 1, 0.6],
    [2, 2, 0.8],
```

```
    [2, 3, 0.6],
    [4, 1, 0.6],
    [4, 2, 0.3],
    [4, 3, 0.4],
])

data = BatchOperator.fromDataframe(df_data, schemaStr='user bigint, item
bigint, rating double')

model = UserCfTrainBatchOp()\
    .setUserCol("user")\
    .setItemCol("item")\
    .setRateCol("rating").linkFrom(data);

predictor = UserCfUsersPerItemRecommender()\
    .setItemCol("item")\
    .setReservedCols(["item"])\
    .setK(1)\
    .setRecommCol("prediction_result")\
    .setModelData(model)

predictor.transform(data).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.recommendation.UserCfTrainBatchOp;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.recommendation.UserCfUsersPerItemRecommender;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class UserCfUsersPerItemRecommenderTest {
    @Test
    public void testUserCfUsersPerItemRecommender() throws Exception {
        List <Row> df_data = Arrays.asList(
            Row.of(1, 1, 0.6),
            Row.of(2, 2, 0.8),
            Row.of(2, 3, 0.6),
            Row.of(4, 1, 0.6),
            Row.of(4, 2, 0.3),
            Row.of(4, 3, 0.4)
        );
```

```
        BatchOperator <?> data = new MemSourceBatchOp(df_data, "user int, item
int, rating double");
        BatchOperator <?> model = new UserCfTrainBatchOp()
            .setUserCol("user")
            .setItemCol("item")
            .setRateCol("rating").linkFrom(data);
        UserCfUsersPerItemRecommender predictor = new
UserCfUsersPerItemRecommender()
            .setItemCol("item")
            .setReservedCols("item")
            .setK(1)
            .setRecommCol("prediction_result")
            .setModelData(model);
        predictor.transform(data).print();
    }
}
```

运行结果

item	prediction_result
1	{"user":"[1]","score":"[0.23046638387921276]"}
2	{"user":"[4]","score":"[0.2458308094711603]"}
3	{"user":"[4]","score":"[0.1843731071033702]"}
1	{"user":"[1]","score":"[0.23046638387921276]"}
2	{"user":"[4]","score":"[0.2458308094711603]"}
3	{"user":"[4]","score":"[0.1843731071033702]"}

(BayesOptimSearchCV)

Java 类名：com.alibaba.alink.pipeline.tuning.BayesOptimSearchCV

Python 类名：BayesOptimSearchCV

功能介绍

bayesoptimsearch是通过构造参数的分布，对其中的每一组输入参数的组很分别进行训练，预测，评估。取得评估参数最优的模型，作为最终的返回模型

cv为交叉验证，将数据切分为k-folds，对每k-1份数据做训练，对剩余一份数据做预测和评估，得到一个评估结果。

此函数用cv方法得到每一个grid对应参数的评估结果，得到最优模型

参数说明

名称	中文名称	描述	类型	是否必须?	默认值
NumFolds	折数	交叉验证的参数，数据的折数（大于等于2）	Integer		10
ParamDist	参数分布	指定搜索的参数的分布	ParamDist	✓	---
Estimator	Estimator	用于调优的Estimator	Estimator	✓	---
TuningEvaluator	评估指标	用于选择最优模型的评估指标	TuningEvaluator	✓	---

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
def adult(url):
    data = (
        CsvSourceBatchOp()
        .setFilePath('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')
        .setSchemaStr(
            'age bigint, workclass string, fnlwgt bigint,'
            'education string, education_num bigint,'
            'marital_status string, occupation string,'
```

```
            'relationship string, race string, sex string,'
            'capital_gain bigint, capital_loss bigint,'
            'hours_per_week bigint, native_country string,'
            'label string'
        )
    )
    return data


def adult_train():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')


def adult_test():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_test.csv')


def adult_numerical_feature_strs():
    return [
        "age", "fnlwgt", "education_num",
        "capital_gain", "capital_loss", "hours_per_week"
    ]


def adult_categorical_feature_strs():
    return [
        "workclass", "education", "marital_status",
        "occupation", "relationship", "race", "sex",
        "native_country"
    ]


def adult_features_strs():
    feature = adult_numerical_feature_strs()
    feature.extend(adult_categorical_feature_strs())

    return feature


def rf_grid_search_cv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
        .setFeatureCols(featureCols)
        .setCategoricalCols(categoryFeatureCols)
        .setLabelCol(label)
        .setPredictionCol('prediction')
```

```
        .setPredictionDetailCol('prediction_detail')
    )
    paramDist = (
        ParamDist()
        .addDist(rf, 'NUM_TREES', ValueDist.uniform(1., 10.))
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
        .setLabelCol(label)
        .setPredictionDetailCol("prediction_detail")
        .setTuningBinaryClassMetric(metric)
    )
    cv = (
        GridSearchCV()
        .setEstimator(rf)
        .setParamGrid(paramGrid)
        .setTuningEvaluator(tuningEvaluator)
        .setNumFolds(2)
    )

    return cv


def rf_grid_search_tv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
        .setFeatureCols(featureCols)
        .setCategoricalCols(categoryFeatureCols)
        .setLabelCol(label)
        .setPredictionCol('prediction')
        .setPredictionDetailCol('prediction_detail')
    )
    paramDist = (
        ParamDist()
        .addDist(rf, 'NUM_TREES', ValueDist.uniform(1., 10.))
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
        .setLabelCol(label)
        .setPredictionDetailCol("prediction_detail")
        .setTuningBinaryClassMetric(metric)
    )
    cv = (
        GridSearchTVSplit()
        .setEstimator(rf)
        .setParamGrid(paramGrid)
        .setTuningEvaluator(tuningEvaluator)
    )
```

```
    return cv


def tuningcv(cv_estimator, input):
    return cv_estimator.fit(input)


def tuningtv(tv_estimator, input):
    return tv_estimator.fit(input)


def main():
    print('rf cv tuning')
    model = tuningcv(
        rf_grid_search_cv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

    print(model.getReport())

    print('rf tv tuning')
    model = tuningtv(
        rf_grid_search_tv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

    print(model.getReport())
main()
```

运行结果

```
rf cv tuning
com.alibaba.alink.pipeline.tuning.GridSearchCV
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8922549257899725
```

```
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8920255970548456
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8944982480437225
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8923867598288401
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9012141767959505
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
```

```
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8993774036693788
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8981738808130779
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9029671873892725
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.905228896323363
} ]
rf tv tuning
com.alibaba.alink.pipeline.tuning.GridSearchTVSplit
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
```

```
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.9022694229691741
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8963559966080328
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9041948454957178
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8982021117392784
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
```

```
  } ],
  "metric" : 0.9031851535310546
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9034443322241488
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8993474753000145
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9090250137144916
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9129786771786127
} ]
```

(BayesOptimSearchTVSplit)

Java 类名：com.alibaba.alink.pipeline.tuning.BayesOptimSearchTVSplit

Python 类名：BayesOptimSearchTVSplit

功能介绍

bayesoptimsearchtv是通过构造参数的分布，对其中的每一组输入参数的组很分别进行训练，预测，评估。取得评估参数最优的模型，作为最终的返回模型

tv为训练验证，将数据按照比例切分为两份，对其中一份数据做训练，对剩余一份数据做预测和评估，得到一个评估结果。

此函数用tv方法得到每一个grid对应参数的评估结果，得到最优模型

参数说明

名称	中文名称	描述	类型	是否必须?	默认值
trainRatio	训练集比例	训练集与验证集的划分比例，取值范围为(0, 1]。	Double		0.8
ParamDist	参数分布	指定搜索的参数的分布	ParamDist	✓	---
Estimator	Estimator	用于调优的Estimator	Estimator	✓	---
TuningEvaluator	评估指标	用于选择最优模型的评估指标	TuningEvaluator	✓	---

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
def adult(url):
    data = (
        CsvSourceBatchOp()
        .setFilePath('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')
        .setSchemaStr(
            'age bigint, workclass string, fnlwgt bigint,'
            'education string, education_num bigint,'
            'marital_status string, occupation string,'
```

```
            'relationship string, race string, sex string,'
            'capital_gain bigint, capital_loss bigint,'
            'hours_per_week bigint, native_country string,'
            'label string'
        )
    )
    return data


def adult_train():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')


def adult_test():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_test.csv')


def adult_numerical_feature_strs():
    return [
        "age", "fnlwgt", "education_num",
        "capital_gain", "capital_loss", "hours_per_week"
    ]


def adult_categorical_feature_strs():
    return [
        "workclass", "education", "marital_status",
        "occupation", "relationship", "race", "sex",
        "native_country"
    ]


def adult_features_strs():
    feature = adult_numerical_feature_strs()
    feature.extend(adult_categorical_feature_strs())

    return feature


def rf_grid_search_cv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
        .setFeatureCols(featureCols)
        .setCategoricalCols(categoryFeatureCols)
        .setLabelCol(label)
        .setPredictionCol('prediction')
```

```
        .setPredictionDetailCol('prediction_detail')
    )
    paramDist = (
        ParamDist()
        .addDist(rf, 'NUM_TREES', ValueDist.uniform(1., 10.))
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
        .setLabelCol(label)
        .setPredictionDetailCol("prediction_detail")
        .setTuningBinaryClassMetric(metric)
    )
    cv = (
        GridSearchCV()
        .setEstimator(rf)
        .setParamDist(paramDist)
        .setTuningEvaluator(tuningEvaluator)
        .setNumFolds(2)
    )

    return cv


def rf_grid_search_tv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
        .setFeatureCols(featureCols)
        .setCategoricalCols(categoryFeatureCols)
        .setLabelCol(label)
        .setPredictionCol('prediction')
        .setPredictionDetailCol('prediction_detail')
    )
    paramDist = (
        ParamDist()
        .addDist(rf, 'NUM_TREES', ValueDist.uniform(1., 10.))
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
        .setLabelCol(label)
        .setPredictionDetailCol("prediction_detail")
        .setTuningBinaryClassMetricå(metric)
    )
    cv = (
        GridSearchTVSplit()
        .setEstimator(rf)
        .setParamDist(paramDist)
        .setTuningEvaluator(tuningEvaluator)
    )
```

```
    return cv


def tuningcv(cv_estimator, input):
    return cv_estimator.fit(input)


def tuningtv(tv_estimator, input):
    return tv_estimator.fit(input)


def main():
    print('rf cv tuning')
    model = tuningcv(
        rf_grid_search_cv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

    print(model.getReport())

    print('rf tv tuning')
    model = tuningtv(
        rf_grid_search_tv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

    print(model.getReport())
main()
```

运行结果

```
rf cv tuning
com.alibaba.alink.pipeline.tuning.GridSearchCV
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8922549257899725
```

```
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8920255970548456
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8944982480437225
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8923867598288401
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9012141767959505
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
```

```
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8993774036693788
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8981738808130779
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9029671873892725
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.905228896323363
} ]
rf tv tuning
com.alibaba.alink.pipeline.tuning.GridSearchTVSplit
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
```

```
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.9022694229691741
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8963559966080328
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9041948454957178
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8982021117392784
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
```

```
  } ],
  "metric" : 0.9031851535310546
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9034443322241488
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8993474753000145
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9090250137144916
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9129786771786127
} ]
```

网格搜索CV (GridSearchCV)

Java 类名：com.alibaba.alink.pipeline.tuning.GridSearchCV

Python 类名：GridSearchCV

功能介绍

gridsearch是通过参数数组组成的网格，对其中的每一组输入参数的组很分别进行训练，预测，评估。取得评估参数最优的模型，作为最终的返回模型

cv为交叉验证，将数据切分为k-folds，对每k-1份数据做训练，对剩余一份数据做预测和评估，得到一个评估结果。

此函数用cv方法得到每一个grid对应参数的评估结果，得到最优模型

参数说明

名称	中文名称	描述	类型	是否必须?	默认值
NumFolds	折数	交叉验证的参数，数据的折数（大于等于2）	Integer		10
ParamGrid	参数网格	指定参数的网格	ParamGrid	✓	---
Estimator	Estimator	用于调优的Estimator	Estimator	✓	---
TuningEvaluator	评估指标	用于选择最优模型的评估指标	TuningEvaluator	✓	---

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

def adult(url):
    data = (
        CsvSourceBatchOp()
        .setFilePath('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')
```

```
        .setSchemaStr(
            'age bigint, workclass string, fnlwgt bigint,'
            'education string, education_num bigint,'
            'marital_status string, occupation string,'
            'relationship string, race string, sex string,'
            'capital_gain bigint, capital_loss bigint,'
            'hours_per_week bigint, native_country string,'
            'label string'
        )
    )
    return data


def adult_train():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')


def adult_test():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_test.csv')


def adult_numerical_feature_strs():
    return [
        "age", "fnlwgt", "education_num",
        "capital_gain", "capital_loss", "hours_per_week"
    ]


def adult_categorical_feature_strs():
    return [
        "workclass", "education", "marital_status",
        "occupation", "relationship", "race", "sex",
        "native_country"
    ]


def adult_features_strs():
    feature = adult_numerical_feature_strs()
    feature.extend(adult_categorical_feature_strs())

    return feature


def rf_grid_search_cv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
```

```
        .setFeatureCols(featureCols)
        .setCategoricalCols(categoryFeatureCols)
        .setLabelCol(label)
        .setPredictionCol('prediction')
        .setPredictionDetailCol('prediction_detail')
    )
    paramGrid = (
        ParamGrid()
        .addGrid(rf, 'SUBSAMPLING_RATIO', [1.0, 0.99, 0.98])
        .addGrid(rf, 'NUM_TREES', [3, 6, 9])
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
        .setLabelCol(label)
        .setPredictionDetailCol("prediction_detail")
        .setTuningBinaryClassMetric(metric)
    )
    cv = (
        GridSearchCV()
        .setEstimator(rf)
        .setParamGrid(paramGrid)
        .setTuningEvaluator(tuningEvaluator)
        .setNumFolds(2)
        .enableLazyPrintTrainInfo("TrainInfo")
    )

    return cv


def rf_grid_search_tv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
        .setFeatureCols(featureCols)
        .setCategoricalCols(categoryFeatureCols)
        .setLabelCol(label)
        .setPredictionCol('prediction')
        .setPredictionDetailCol('prediction_detail')
    )
    paramGrid = (
        ParamGrid()
        .addGrid(rf, 'SUBSAMPLING_RATIO', [1.0, 0.99, 0.98])
        .addGrid(rf, 'NUM_TREES', [3, 6, 9])
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
        .setLabelCol(label)
        .setPredictionDetailCol("prediction_detail")
        .setTuningBinaryClassMetric(metric)
```

```
    )
    cv = (
        GridSearchTVSplit()
        .setEstimator(rf)
        .setParamGrid(paramGrid)
        .setTuningEvaluator(tuningEvaluator)
        .enableLazyPrintTrainInfo("TrainInfo")
    )

    return cv


def tuningcv(cv_estimator, input):
    return cv_estimator.fit(input)


def tuningtv(tv_estimator, input):
    return tv_estimator.fit(input)


def main():
    print('rf cv tuning')
    model = tuningcv(
        rf_grid_search_cv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

    print('rf tv tuning')
    model = tuningtv(
        rf_grid_search_tv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

main()
```

Java 代码

```
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.classification.RandomForestClassifier;
import com.alibaba.alink.pipeline.tuning.BinaryClassificationTuningEvaluator;
import com.alibaba.alink.pipeline.tuning.GridSearchCV;
import com.alibaba.alink.pipeline.tuning.GridSearchCVModel;
import com.alibaba.alink.pipeline.tuning.ParamGrid;
import org.junit.Test;
```

```
public class GridSearchCVTest {
    @Test
    public void testGridSearchCV() throws Exception {
        String[] featureCols = new String[] {
            "age", "fnlwgt", "education_num",
            "capital_gain", "capital_loss", "hours_per_week",
            "workclass", "education", "marital_status",
            "occupation", "relationship", "race", "sex",
            "native_country"
        };
        String[] categoryFeatureCols = new String[] {
            "workclass", "education", "marital_status",
            "occupation", "relationship", "race", "sex",
            "native_country"
        };
        String label = "label";
        CsvSourceBatchOp data = new CsvSourceBatchOp()
            .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv")
            .setSchemaStr(
                "age bigint, workclass string, fnlwgt bigint, education string,
education_num bigint, marital_status "
                    + "string, occupation string, relationship string, race
string, sex string, capital_gain bigint, "
                    + "capital_loss bigint, hours_per_week bigint,
native_country string, label string");
        RandomForestClassifier rf = new RandomForestClassifier()
            .setFeatureCols(featureCols)
            .setCategoricalCols(categoryFeatureCols)
            .setLabelCol(label)
            .setPredictionCol("prediction")
            .setPredictionDetailCol("prediction_detail");
        ParamGrid paramGrid = new ParamGrid()
            .addGrid(rf, RandomForestClassifier.SUBSAMPLING_RATIO, new Double[]
{1.0, 0.99, 0.98})
            .addGrid(rf, RandomForestClassifier.NUM_TREES, new Integer[] {3, 6,
9});
        BinaryClassificationTuningEvaluator tuningEvaluator = new
BinaryClassificationTuningEvaluator()
            .setLabelCol(label)
            .setPredictionDetailCol("prediction_detail")
            .setTuningBinaryClassMetric("AUC");
        GridSearchCV cv = new GridSearchCV()
            .setEstimator(rf)
            .setParamGrid(paramGrid)
            .setTuningEvaluator(tuningEvaluator)
            .setNumFolds(2)
            .enableLazyPrintTrainInfo("TrainInfo");
```

```
        GridSearchCVModel model = cv.fit(data);
    }
}
```

运行结果

TrainInfo Metric information: Metric name: AUC Larger is better: true Tuning information: | AUC| stage| param|value| stage 2| param 2|value 2| |----------|----------|--------|-----|----------|----------|-------| | 0.912327454540025|RandomForestClassifier|numTrees| 9|RandomForestClassifier|subsamplingRatio| 0.98| |0.9113181022628927|RandomForestClassifier|numTrees| 9|RandomForestClassifier|subsamplingRatio| 1.0| |0.9109408773009041|RandomForestClassifier|numTrees| 9|RandomForestClassifier|subsamplingRatio| 0.99| |0.9084745064874684|RandomForestClassifier|numTrees| 6|RandomForestClassifier|subsamplingRatio| 1.0| |0.9066321684664669|RandomForestClassifier|numTrees| 6|RandomForestClassifier|subsamplingRatio| 0.98| |0.9045123178682739|RandomForestClassifier|numTrees| 6|RandomForestClassifier|subsamplingRatio| 0.99| |0.8908957160768797|RandomForestClassifier|numTrees| 3|RandomForestClassifier|subsamplingRatio| 1.0| |0.8903604608878586|RandomForestClassifier|numTrees| 3|RandomForestClassifier|subsamplingRatio| 0.98| | 0.888885807369439|RandomForestClassifier|numTrees| 3|RandomForestClassifier|subsamplingRatio| 0.99|

运行结果

```
rf cv tuning
com.alibaba.alink.pipeline.tuning.GridSearchCV
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8922549257899725
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8920255970548456
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
```

```
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8944982480437225
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8923867598288401
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9012141767959505
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8993774036693788
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
```

```
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8981738808130779
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9029671873892725
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.905228896323363
} ]
rf tv tuning
com.alibaba.alink.pipeline.tuning.GridSearchTVSplit
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.9022694229691741
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
```

```
    "paramValue" : 0.99
  } ],
  "metric" : 0.8963559966080328
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9041948454957178
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8982021117392784
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9031851535310546
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9034443322241488
}, {
```

```
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8993474753000145
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9090250137144916
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9129786771786127
} ]
```

网格搜索TV (GridSearchTVSplit)

Java 类名：com.alibaba.alink.pipeline.tuning.GridSearchTVSplit

Python 类名：GridSearchTVSplit

功能介绍

gridsearch是通过参数数组组成的网格，对其中的每一组输入参数的组很分别进行训练，预测，评估。取得评估参数最优的模型，作为最终的返回模型

tv为训练验证，将数据按照比例切分为两份，对其中一份数据做训练，对剩余一份数据做预测和评估，得到一个评估结果。

此函数用tv方法得到每一个grid对应参数的评估结果，得到最优模型

参数说明

名称	中文名称	描述	类型	是否必须?	默认值
trainRatio	训练集比例	训练集与验证集的划分比例，取值范围为(0, 1]。	Double		0.8
ParamGrid	参数网格	指定参数的网格	ParamGrid	✓	---
Estimator	Estimator	用于调优的Estimator	Estimator	✓	---
TuningEvaluator	评估指标	用于选择最优模型的评估指标	TuningEvaluator	✓	---

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

def adult(url):
    data = (
        CsvSourceBatchOp()
            .setFilePath('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')
```

```
            .setSchemaStr(
            'age bigint, workclass string, fnlwgt bigint,'
            'education string, education_num bigint,'
            'marital_status string, occupation string,'
            'relationship string, race string, sex string,'
            'capital_gain bigint, capital_loss bigint,'
            'hours_per_week bigint, native_country string,'
            'label string'
        )
    )
    return data

def adult_train():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')

def adult_test():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_test.csv')

def adult_numerical_feature_strs():
    return [
        "age", "fnlwgt", "education_num",
        "capital_gain", "capital_loss", "hours_per_week"
    ]

def adult_categorical_feature_strs():
    return [
        "workclass", "education", "marital_status",
        "occupation", "relationship", "race", "sex",
        "native_country"
    ]

def adult_features_strs():
    feature = adult_numerical_feature_strs()
    feature.extend(adult_categorical_feature_strs())

    return feature

def rf_grid_search_cv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
            .setFeatureCols(featureCols)
            .setCategoricalCols(categoryFeatureCols)
            .setLabelCol(label)
            .setPredictionCol('prediction')
            .setPredictionDetailCol('prediction_detail')
    )
```

```
    paramGrid = (
        ParamGrid()
            .addGrid(rf, 'SUBSAMPLING_RATIO', [1.0, 0.99, 0.98])
            .addGrid(rf, 'NUM_TREES', [3, 6, 9])
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
            .setLabelCol(label)
            .setPredictionDetailCol("prediction_detail")
            .setTuningBinaryClassMetric(metric)
    )
    cv = (
        GridSearchCV()
            .setEstimator(rf)
            .setParamGrid(paramGrid)
            .setTuningEvaluator(tuningEvaluator)
            .setNumFolds(2)
            .enableLazyPrintTrainInfo("TrainInfo")
    )

    return cv

def rf_grid_search_tv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
            .setFeatureCols(featureCols)
            .setCategoricalCols(categoryFeatureCols)
            .setLabelCol(label)
            .setPredictionCol('prediction')
            .setPredictionDetailCol('prediction_detail')
    )
    paramGrid = (
        ParamGrid()
            .addGrid(rf, 'SUBSAMPLING_RATIO', [1.0, 0.99, 0.98])
            .addGrid(rf, 'NUM_TREES', [3, 6, 9])
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
            .setLabelCol(label)
            .setPredictionDetailCol("prediction_detail")
            .setTuningBinaryClassMetric(metric)
    )
    cv = (
        GridSearchTVSplit()
            .setEstimator(rf)
            .setParamGrid(paramGrid)
            .setTuningEvaluator(tuningEvaluator)
            .enableLazyPrintTrainInfo("TrainInfo")
```

```
    )

    return cv

def tuningcv(cv_estimator, input):
    return cv_estimator.fit(input)

def tuningtv(tv_estimator, input):
    return tv_estimator.fit(input)

def main():
    print('rf cv tuning')
    model = tuningcv(
        rf_grid_search_cv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

    print('rf tv tuning')
    model = tuningtv(
        rf_grid_search_tv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

main()
```

Java 代码

```
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.classification.RandomForestClassifier;
import com.alibaba.alink.pipeline.tuning.BinaryClassificationTuningEvaluator;
import com.alibaba.alink.pipeline.tuning.GridSearchTVSplit;
import com.alibaba.alink.pipeline.tuning.GridSearchTVSplitModel;
import com.alibaba.alink.pipeline.tuning.ParamGrid;
import org.junit.Test;

public class GridSearchTVSplitTest {
    @Test
    public void testGridSearchTVSplit() throws Exception {
        String[] featureCols = new String[] {
            "age", "fnlwgt", "education_num",
            "capital_gain", "capital_loss", "hours_per_week",
            "workclass", "education", "marital_status",
            "occupation", "relationship", "race", "sex",
            "native_country"
        };
```

```
        String[] categoryFeatureCols = new String[] {
            "workclass", "education", "marital_status",
            "occupation", "relationship", "race", "sex",
            "native_country"
        };
        String label = "label";
        CsvSourceBatchOp data = new CsvSourceBatchOp()
            .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv")
            .setSchemaStr(
                "age bigint, workclass string, fnlwgt bigint, education string,
education_num bigint, marital_status "
                    + "string, occupation string, relationship string, race
string, sex string, capital_gain bigint, "
                    + "capital_loss bigint, hours_per_week bigint,
native_country string, label string");
        RandomForestClassifier rf = new RandomForestClassifier()
            .setFeatureCols(featureCols)
            .setCategoricalCols(categoryFeatureCols)
            .setLabelCol(label)
            .setPredictionCol("prediction")
            .setPredictionDetailCol("prediction_detail");
        ParamGrid paramGrid = new ParamGrid()
            .addGrid(rf, RandomForestClassifier.SUBSAMPLING_RATIO, new Double[]
{1.0, 0.99, 0.98})
            .addGrid(rf, RandomForestClassifier.NUM_TREES, new Integer[] {3, 6,
9});
        BinaryClassificationTuningEvaluator tuningEvaluator = new
BinaryClassificationTuningEvaluator()
            .setLabelCol(label)
            .setPredictionDetailCol("prediction_detail")
            .setTuningBinaryClassMetric("AUC");
        GridSearchTVSplit cv = new GridSearchTVSplit()
            .setEstimator(rf)
            .setParamGrid(paramGrid)
            .setTuningEvaluator(tuningEvaluator)
            .setTrainRatio(0.8)
            .enableLazyPrintTrainInfo("TrainInfo");
        GridSearchTVSplitModel model = cv.fit(data);
    }
}
```

运行结果

TrainInfo Metric information: Metric name: AUC Larger is better: true Tuning information: | AUC| stage| param|value| stage 2| param 2|value 2| |----------------|--------------------|--------|-----|--------------------|----------------|-------| |0.9146840488554084|RandomForestClassifier|numTrees| 9|RandomForestClassifier|subsamplingRatio| 0.99| |0.9125303612307454|RandomForestClassifier|numTrees| 9|RandomForestClassifier|subsamplingRatio|

1.0| |0.9098037018824784|RandomForestClassifier|numTrees| 6|RandomForestClassifier|subsamplingRatio|
0.98| |0.9091959987727252|RandomForestClassifier|numTrees| 6|RandomForestClassifier|subsamplingRatio|
0.99| |0.9078221190139827|RandomForestClassifier|numTrees| 9|RandomForestClassifier|subsamplingRatio|
0.98| |0.9025009982071417|RandomForestClassifier|numTrees| 6|RandomForestClassifier|subsamplingRatio|
1.0| |0.9015572276497046|RandomForestClassifier|numTrees| 3|RandomForestClassifier|subsamplingRatio|
1.0| | 0.893774245901412|RandomForestClassifier|numTrees| 3|RandomForestClassifier|subsamplingRatio|
0.98| | 0.892706752181599|RandomForestClassifier|numTrees| 3|RandomForestClassifier|subsamplingRatio|
0.99|

运行结果

```
rf cv tuning
com.alibaba.alink.pipeline.tuning.GridSearchCV
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8922549257899725
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8920255970548456
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8944982480437225
}, {
  "param" : [ {
```

```
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8923867598288401
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9012141767959505
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8993774036693788
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8981738808130779
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
```

```
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9029671873892725
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.905228896323363
} ]
rf tv tuning
com.alibaba.alink.pipeline.tuning.GridSearchTVSplit
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.9022694229691741
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8963559966080328
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
```

```
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9041948454957178
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8982021117392784
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9031851535310546
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9034443322241488
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8993474753000145
```

```
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9090250137144916
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9129786771786127
} ]
```

随机搜索CV (RandomSearchCV)

Java 类名：com.alibaba.alink.pipeline.tuning.RandomSearchCV

Python 类名：RandomSearchCV

功能介绍

randomsearch是通过随机参数，对其中的每一组输入参数的组很分别进行训练，预测，评估。取得评估参数最优的模型，作为最终的返回模型

cv为交叉验证，将数据切分为k-folds，对每k-1份数据做训练，对剩余一份数据做预测和评估，得到一个评估结果。

此函数用cv方法得到每一个grid对应参数的评估结果，得到最优模型

参数说明

名称	中文名称	描述	类型	是否必须?	默认值
NumFolds	折数	交叉验证的参数，数据的折数（大于等于2）	Integer		10
ParamDist	参数分布	指定搜索的参数的分布	ParamDist	✓	---
Estimator	Estimator	用于调优的Estimator	Estimator	✓	---
TuningEvaluator	评估指标	用于选择最优模型的评估指标	TuningEvaluator	✓	---

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

def adult(url):
    data = (
        CsvSourceBatchOp()
        .setFilePath('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')
```

```
        .setSchemaStr(
            'age bigint, workclass string, fnlwgt bigint,'
            'education string, education_num bigint,'
            'marital_status string, occupation string,'
            'relationship string, race string, sex string,'
            'capital_gain bigint, capital_loss bigint,'
            'hours_per_week bigint, native_country string,'
            'label string'
        )
    )
    return data


def adult_train():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')


def adult_test():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_test.csv')


def adult_numerical_feature_strs():
    return [
        "age", "fnlwgt", "education_num",
        "capital_gain", "capital_loss", "hours_per_week"
    ]


def adult_categorical_feature_strs():
    return [
        "workclass", "education", "marital_status",
        "occupation", "relationship", "race", "sex",
        "native_country"
    ]


def adult_features_strs():
    feature = adult_numerical_feature_strs()
    feature.extend(adult_categorical_feature_strs())

    return feature


def rf_grid_search_cv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
```

```
        .setFeatureCols(featureCols)
        .setCategoricalCols(categoryFeatureCols)
        .setLabelCol(label)
        .setPredictionCol('prediction')
        .setPredictionDetailCol('prediction_detail')
    )
    paramDist = (
        ParamDist()
        .addDist(rf, 'NUM_TREES', ValueDist.randInteger(1, 10))
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
        .setLabelCol(label)
        .setPredictionDetailCol("prediction_detail")
        .setTuningBinaryClassMetric(metric)
    )
    cv = (
        RandomSearchCV()
        .setEstimator(rf)
        .setParamDist(paramDist)
        .setTuningEvaluator(tuningEvaluator)
        .setNumFolds(2)
    )

    return cv


def rf_grid_search_tv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
        .setFeatureCols(featureCols)
        .setCategoricalCols(categoryFeatureCols)
        .setLabelCol(label)
        .setPredictionCol('prediction')
        .setPredictionDetailCol('prediction_detail')
    )
    paramDist = (
        ParamDist()
        .addDist(rf, 'NUM_TREES', ValueDist.randInteger(1, 10))
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
        .setLabelCol(label)
        .setPredictionDetailCol("prediction_detail")
        .setTuningBinaryClassMetric(metric)
    )
    cv = (
        RandomSearchTVSplit()
```

```
        .setEstimator(rf)
        .setParamDist(paramDist)
        .setTuningEvaluator(tuningEvaluator)
    )

    return cv


def tuningcv(cv_estimator, input):
    return cv_estimator.enableLazyPrintTrainInfo("CVTrainInfo").fit(input)


def tuningtv(tv_estimator, input):
    return tv_estimator.enableLazyPrintTrainInfo("TVTrainInfo").fit(input)


def main():
    print('rf cv tuning')
    model = tuningcv(
        rf_grid_search_cv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

    print('rf tv tuning')
    model = tuningtv(
        rf_grid_search_tv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

main()
```

Java 代码

```
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.classification.RandomForestClassifier;
import com.alibaba.alink.pipeline.tuning.BinaryClassificationTuningEvaluator;
import com.alibaba.alink.pipeline.tuning.ParamDist;
import com.alibaba.alink.pipeline.tuning.RandomSearchCV;
import com.alibaba.alink.pipeline.tuning.RandomSearchCVModel;
import com.alibaba.alink.pipeline.tuning.ValueDist;
import org.junit.Test;

public class RandomSearchCVTest {
    @Test
    public void testRandomSearchCV() throws Exception {
```

```
        String[] featureCols = new String[] {
            "age", "fnlwgt", "education_num",
            "capital_gain", "capital_loss", "hours_per_week",
            "workclass", "education", "marital_status",
            "occupation", "relationship", "race", "sex",
            "native_country"
        };
        String[] categoryFeatureCols = new String[] {
            "workclass", "education", "marital_status",
            "occupation", "relationship", "race", "sex",
            "native_country"
        };
        String label = "label";
        CsvSourceBatchOp data = new CsvSourceBatchOp()
            .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv")
            .setSchemaStr(
                "age bigint, workclass string, fnlwgt bigint, education string,
education_num bigint, marital_status "
                    + "string, occupation string, relationship string, race
string, sex string, capital_gain bigint, "
                    + "capital_loss bigint, hours_per_week bigint,
native_country string, label string");
        RandomForestClassifier rf = new RandomForestClassifier()
            .setFeatureCols(featureCols)
            .setCategoricalCols(categoryFeatureCols)
            .setLabelCol(label)
            .setPredictionCol("prediction")
            .setPredictionDetailCol("prediction_detail");
        ParamDist paramDist = new ParamDist()
            .addDist(rf, RandomForestClassifier.NUM_TREES,
ValueDist.randInteger(1, 10));
        BinaryClassificationTuningEvaluator tuningEvaluator = new
BinaryClassificationTuningEvaluator()
            .setLabelCol(label)
            .setPredictionDetailCol("prediction_detail")
            .setTuningBinaryClassMetric("AUC");
        RandomSearchCV cv = new RandomSearchCV()
            .setEstimator(rf)
            .setParamDist(paramDist)
            .setTuningEvaluator(tuningEvaluator)
            .setNumFolds(2)
            .enableLazyPrintTrainInfo("TrainInfo");
        RandomSearchCVModel model = cv.fit(data);
    }
}
```

运行结果

TrainInfo Metric information: Metric name: AUC Larger is better: true Tuning information: | AUC| stage| param|value| |------------------|----------------------|--------|-----| |0.9134148020313496|RandomForestClassifier|numTrees| 10| |0.9123992401477525|RandomForestClassifier|numTrees| 10| |0.9107724678432794|RandomForestClassifier|numTrees| 8| | 0.905703319906151|RandomForestClassifier|numTrees| 6| |0.9052924036494705|RandomForestClassifier|numTrees| 7| |0.8927397325721704|RandomForestClassifier|numTrees| 3| |0.8887150253364192|RandomForestClassifier|numTrees| 3| | 0.885191174049819|RandomForestClassifier|numTrees| 3| |0.8837444737636566|RandomForestClassifier|numTrees| 2| |0.8774725529763574|RandomForestClassifier|numTrees| 2|

运行结果

```
rf cv tuning
com.alibaba.alink.pipeline.tuning.GridSearchCV
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8922549257899725
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8920255970548456
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
```

```
  } ],
  "metric" : 0.8944982480437225
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8923867598288401
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9012141767959505
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8993774036693788
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8981738808130779
}, {
  "param" : [ {
```

```
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9029671873892725
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.905228896323363
} ]
rf tv tuning
com.alibaba.alink.pipeline.tuning.GridSearchTVSplit
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.9022694229691741
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8963559966080328
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
```

```
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9041948454957178
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8982021117392784
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9031851535310546
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9034443322241488
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
```

```
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8993474753000145
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9090250137144916
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9129786771786127
} ]
```

随机搜索TV (RandomSearchTVSplit)

Java 类名：com.alibaba.alink.pipeline.tuning.RandomSearchTVSplit

Python 类名：RandomSearchTVSplit

功能介绍

randomsearch是通过随机参数，对其中的每一组输入参数的组很分别进行训练，预测，评估。取得评估参数最优的模型，作为最终的返回模型

tv为训练验证，将数据按照比例切分为两份，对其中一份数据做训练，对剩余一份数据做预测和评估，得到一个评估结果。

此函数用tv方法得到每一个grid对应参数的评估结果，得到最优模型

参数说明

名称	中文名称	描述	类型	是否必须?	默认值
trainRatio	训练集比例	训练集与验证集的划分比例，取值范围为(0, 1]。	Double		0.8
ParamDist	参数分布	指定搜索的参数的分布	ParamDist	✓	---
Estimator	Estimator	用于调优的Estimator	Estimator	✓	---
TuningEvaluator	评估指标	用于选择最优模型的评估指标	TuningEvaluator	✓	---

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)


def adult(url):
    data = (
        CsvSourceBatchOp()
        .setFilePath('https://alink-test-data.oss-cn-
```

```
hangzhou.aliyuncs.com/adult_train.csv')
        .setSchemaStr(
            'age bigint, workclass string, fnlwgt bigint,'
            'education string, education_num bigint,'
            'marital_status string, occupation string,'
            'relationship string, race string, sex string,'
            'capital_gain bigint, capital_loss bigint,'
            'hours_per_week bigint, native_country string,'
            'label string'
        )
    )
    return data


def adult_train():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv')


def adult_test():
    return adult('https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_test.csv')


def adult_numerical_feature_strs():
    return [
        "age", "fnlwgt", "education_num",
        "capital_gain", "capital_loss", "hours_per_week"
    ]


def adult_categorical_feature_strs():
    return [
        "workclass", "education", "marital_status",
        "occupation", "relationship", "race", "sex",
        "native_country"
    ]


def adult_features_strs():
    feature = adult_numerical_feature_strs()
    feature.extend(adult_categorical_feature_strs())

    return feature


def rf_grid_search_cv(featureCols, categoryFeatureCols, label, metric):
    rf = (
```

```
        RandomForestClassifier()
        .setFeatureCols(featureCols)
        .setCategoricalCols(categoryFeatureCols)
        .setLabelCol(label)
        .setPredictionCol('prediction')
        .setPredictionDetailCol('prediction_detail')
    )
    paramDist = (
        ParamDist()
        .addDist(rf, 'NUM_TREES', ValueDist.randInteger(1, 10))
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
        .setLabelCol(label)
        .setPredictionDetailCol("prediction_detail")
        .setTuningBinaryClassMetric(metric)
    )
    cv = (
        RandomSearchCV()
        .setEstimator(rf)
        .setParamDist(paramDist)
        .setTuningEvaluator(tuningEvaluator)
        .setNumFolds(2)
        .enableLazyPrintTrainInfo("TrainInfo")
    )

    return cv


def rf_grid_search_tv(featureCols, categoryFeatureCols, label, metric):
    rf = (
        RandomForestClassifier()
        .setFeatureCols(featureCols)
        .setCategoricalCols(categoryFeatureCols)
        .setLabelCol(label)
        .setPredictionCol('prediction')
        .setPredictionDetailCol('prediction_detail')
    )
    paramDist = (
        ParamDist()
        .addDist(rf, 'NUM_TREES', ValueDist.randInteger(1, 10))
    )
    tuningEvaluator = (
        BinaryClassificationTuningEvaluator()
        .setLabelCol(label)
        .setPredictionDetailCol("prediction_detail")
        .setTuningBinaryClassMetric(metric)
    )
```

```
    cv = (
        RandomSearchTVSplit()
        .setEstimator(rf)
        .setParamDist(paramDist)
        .setTuningEvaluator(tuningEvaluator)
        .enableLazyPrintTrainInfo("TrainInfo")
    )

    return cv


def tuningcv(cv_estimator, input):
    return cv_estimator.fit(input)


def tuningtv(tv_estimator, input):
    return tv_estimator.fit(input)


def main():
    print('rf cv tuning')
    model = tuningcv(
        rf_grid_search_cv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )

    print('rf tv tuning')
    model = tuningtv(
        rf_grid_search_tv(adult_features_strs(),
                          adult_categorical_feature_strs(), 'label', 'AUC'),
        adult_train()
    )
main()
```

Java 代码

```
import com.alibaba.alink.operator.batch.source.CsvSourceBatchOp;
import com.alibaba.alink.pipeline.classification.RandomForestClassifier;
import com.alibaba.alink.pipeline.tuning.BinaryClassificationTuningEvaluator;
import com.alibaba.alink.pipeline.tuning.ParamDist;
import com.alibaba.alink.pipeline.tuning.RandomSearchTVSplit;
import com.alibaba.alink.pipeline.tuning.RandomSearchTVSplitModel;
import com.alibaba.alink.pipeline.tuning.ValueDist;
import org.junit.Test;

public class RandomSearchTVSplitTest {
```

```
    @Test
    public void testRandomSearchTVSplit() throws Exception {
        String[] featureCols = new String[] {
            "age", "fnlwgt", "education_num",
            "capital_gain", "capital_loss", "hours_per_week",
            "workclass", "education", "marital_status",
            "occupation", "relationship", "race", "sex",
            "native_country"
        };
        String[] categoryFeatureCols = new String[] {
            "workclass", "education", "marital_status",
            "occupation", "relationship", "race", "sex",
            "native_country"
        };
        String label = "label";
        CsvSourceBatchOp data = new CsvSourceBatchOp()
            .setFilePath("https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/adult_train.csv")
            .setSchemaStr(
                "age bigint, workclass string, fnlwgt bigint, education string,
education_num bigint, marital_status "
                    + "string, occupation string, relationship string, race
string, sex string, capital_gain bigint, "
                    + "capital_loss bigint, hours_per_week bigint,
native_country string, label string");
        RandomForestClassifier rf = new RandomForestClassifier()
            .setFeatureCols(featureCols)
            .setCategoricalCols(categoryFeatureCols)
            .setLabelCol(label)
            .setPredictionCol("prediction")
            .setPredictionDetailCol("prediction_detail");
        ParamDist paramDist = new ParamDist()
            .addDist(rf, RandomForestClassifier.NUM_TREES,
ValueDist.randInteger(1, 10));
        BinaryClassificationTuningEvaluator tuningEvaluator = new
BinaryClassificationTuningEvaluator()
            .setLabelCol(label)
            .setPredictionDetailCol("prediction_detail")
            .setTuningBinaryClassMetric("AUC");
        RandomSearchTVSplit cv = new RandomSearchTVSplit()
            .setEstimator(rf)
            .setParamDist(paramDist)
            .setTuningEvaluator(tuningEvaluator)
            .setTrainRatio(0.8)
            .enableLazyPrintTrainInfo("TrainInfo");
        RandomSearchTVSplitModel model = cv.fit(data);
    }
}
```

运行结果

TrainInfo Metric information: Metric name: AUC Larger is better: true Tuning information: | AUC| stage| param|value| |------------------|----------------------|--------|-----| |0.9121169398031198|RandomForestClassifier|numTrees| 8| |0.9105096451486404|RandomForestClassifier|numTrees| 7| |0.9105087086051442|RandomForestClassifier|numTrees| 6| |0.9098174499836453|RandomForestClassifier|numTrees| 6| |0.9089294943807537|RandomForestClassifier|numTrees| 4| |0.8910848199717841|RandomForestClassifier|numTrees| 2| |0.8862271106520978|RandomForestClassifier|numTrees| 2| |0.8748876808857913|RandomForestClassifier|numTrees| 2| | 0.858989501722944|RandomForestClassifier|numTrees| 1| |0.8553973913661752|RandomForestClassifier|numTrees| 1|

运行结果

```
rf cv tuning
com.alibaba.alink.pipeline.tuning.GridSearchCV
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8922549257899725
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8920255970548456
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
```

```
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8944982480437225
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8923867598288401
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9012141767959505
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.8993774036693788
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8981738808130779
```

```
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9029671873892725
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.905228896323363
} ]
rf tv tuning
com.alibaba.alink.pipeline.tuning.GridSearchTVSplit
[ {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.9022694229691741
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.8963559966080328
}, {
```

```
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 3
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9041948454957178
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8982021117392784
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9031851535310546
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 6
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9034443322241488
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
```

```
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 1.0
  } ],
  "metric" : 0.8993474753000145
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.99
  } ],
  "metric" : 0.9090250137144916
}, {
  "param" : [ {
    "stage" : "RandomForestClassifier",
    "paramName" : "numTrees",
    "paramValue" : 9
  }, {
    "stage" : "RandomForestClassifier",
    "paramName" : "subsamplingRatio",
    "paramValue" : 0.98
  } ],
  "metric" : 0.9129786771786127
} ]
```

字符串近似最近邻 (StringApproxNearestNeighbor)

Java 类名：com.alibaba.alink.pipeline.similarity.StringApproxNearestNeighbor

Python 类名：StringApproxNearestNeighbor

功能介绍

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

SimhashHamming（SimHash_Hamming_Distance)相似度=1-距离/64.0，应选择metric的参数为SIMHASH_HAMMING_SIM。

MinHash应选择metric的参数为MINHASH_SIM。

Jaccard应选择metric的参数为JACCARD_SIM。

参数说明

名称	中文名称	描述	类型	是否必须?	取值
idCol	id列名	id列名	String	✓	
selectedCol	选中的列名	计算列对应的列名	String	✓	
metric	距离类型	用于计算的距离类型	String		"SIMHASH_H "SIMHASH_H "MINHASH_J "JACCARD_S
modelFilePath	模型的文件路径	模型的文件路径	String		
numBucket	分桶个数	分桶个数	Integer		
numHashTables	哈希表个数	哈希表的数目	Integer		

outputCol	输出结果列	输出结果列列名，可选，默认null	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
radius	radius值	radius值	Double		
reservedCols	算法保留列名	算法保留列	String[]		
seed	采样种子	采样种子	Long		
topN	TopN的值	TopN的值	Integer		[1, +inf)
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "abcde", "aabce"],
    [1, "aacedw", "aabbed"],
    [2, "cdefa", "bbcefa"],
    [3, "bdefh", "ddeac"],
    [4, "acedm", "aeefbc"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')

pipeline =
StringApproxNearestNeighbor().setIdCol("id").setSelectedCol("text1").setMetric(
"SIMHASH_HAMMING_SIM").setTopN(3)

pipeline.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.similarity.StringApproxNearestNeighbor;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StringApproxNearestNeighborTest {
    @Test
    public void testStringApproxNearestNeighbor() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "abcde", "aabce"),
            Row.of(1, "aacedw", "aabbed"),
            Row.of(2, "cdefa", "bbcefa"),
            Row.of(3, "bdefh", "ddeac"),
            Row.of(4, "acedm", "aeefbc")
        );
```

```
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        StringApproxNearestNeighbor pipeline = new
StringApproxNearestNeighbor().setIdCol("id").setSelectedCol("text1")
            .setMetric("SIMHASH_HAMMING_SIM").setTopN(3);
        pipeline.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

id	text1	text2
0	{"ID":"[0,1,4]","METRIC":"[1.0,0.96875,0.921875]"}	aabce
1	{"ID":"[1,0,4]","METRIC":"[1.0,0.96875,0.921875]"}	aabbed
2	{"ID":"[2,4,1]","METRIC":"[1.0,0.9375,0.890625]"}	bbcefa
3	{"ID":"[3,4,2]","METRIC":"[1.0,0.828125,0.828125]"}	ddeac
4	{"ID":"[4,2,1]","METRIC":"[1.0,0.9375,0.921875]"}	aeefbc

字符串最近邻 (StringNearestNeighbor)

Java 类名：com.alibaba.alink.pipeline.similarity.StringNearestNeighbor

Python 类名：StringNearestNeighbor

功能介绍

本算法支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine四种相似度精确计算方式，通过选择metric参数可计算不同的相似度。

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

Levenshtein（Levenshtein Distance), 相似度=(1-距离)/length，length为两个字符长度的最大值，应选metric的参数为LEVENSHTEIN_SIM。

LCS（Longest Common SubString), 相似度=(1-距离)/length，length为两个字符长度的最大值，应选择metric的参数为LCS_SIM。

SSK（String Subsequence Kernel）支持相似度计算，应选择metric的参数为SSK。

Cosine（Cosine）支持相似度计算，应选择metric的参数为COSINE。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范
idCol	id列名	id列名	String	✓	
selectedCol	选中的列名	计算列对应的列名	String	✓	
lambda	匹配字符权重	匹配字符权重，SSK中使用	Double		
metric	距离类型	用于计算的距离类型	String		"LEVENSHTE "LEVENSHTE "LCS_SIM", " "SSK", "COSI

modelFilePath	模型的文件路径	模型的文件路径	String		
outputCol	输出结果列	输出结果列列名，可选，默认null	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
radius	radius值	radius值	Double		
reservedCols	算法保留列名	算法保留列	String[]		
topN	TopN的值	TopN的值	Integer		[1, +inf)
windowSize	窗口大小	窗口大小	Integer		[1, +inf)
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "abcde", "aabce"],
    [1, "aacedw", "aabbed"],
    [2, "cdefa", "bbcefa"],
    [3, "bdefh", "ddeac"],
    [4, "acedm", "aeefbc"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2 string')

pipeline = StringNearestNeighbor().setIdCol("id").setSelectedCol("text1").setMetric("LEVENSHTEIN_SIM").setTopN(3)

pipeline.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.similarity.StringNearestNeighbor;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class StringNearestNeighborTest {
    @Test
    public void testStringNearestNeighbor() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "abcde", "aabce"),
            Row.of(1, "aacedw", "aabbed"),
            Row.of(2, "cdefa", "bbcefa"),
            Row.of(3, "bdefh", "ddeac"),
            Row.of(4, "acedm", "aeefbc")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        StringNearestNeighbor pipeline = new
StringNearestNeighbor().setIdCol("id").setSelectedCol("text1").setMetric(
            "LEVENSHTEIN_SIM").setTopN(3);
        pipeline.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

id	text1	text2
0	{"ID":"[0,1,4]","METRIC":"[1.0,0.5,0.4]"}	aabce
1	{"ID":"[1,4,0]","METRIC":"[1.0,0.6666666666666667,0.5]"}	aabbed
2	{"ID":"[2,3,0]","METRIC":"[1.0,0.6,0.19999999999999996]"}	bbcefa
3	{"ID":"[3,2,0]","METRIC":"[1.0,0.6,0.19999999999999996]"}	ddeac
4	{"ID":"[4,1,0]","METRIC":"[1.0,0.6666666666666667,0.4]"}	aeefbc

字符串两两相似度计算 (StringSimilarityPairwise)

Java 类名：com.alibaba.alink.pipeline.similarity.StringSimilarityPairwise

Python 类名：StringSimilarityPairwise

功能介绍

字符相似度是计算两两文章或者句子之间的相似度： 支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine，SimHashHamming，MinHash和Jaccard七种相似度计算方式，通过选择metric参数可计算不同的相似度。

Levenshtein（Levenshtein Distance）支持距离和相似度两种方式，相似度=(1-距离)/length，length为两个字符长度的最大值，距离应选择metric的参数为LEVENSHTEIN，相似度应选metric的参数为LEVENSHTEIN_SIM。

LCS（Longest Common SubString）支持距离和相似度两种参数，相似度=(1-距离)/length，length为两个字符长度的最大值，距离应选择metric的参数为LCS，相似度应选择metric的参数为LCS_SIM。

SSK（String Subsequence Kernel）支持相似度计算，应选择metric的参数为SSK。

Cosine（Cosine）支持相似度计算，应选择metric的参数为COSINE。

SimhashHamming（SimHash_Hamming_Distance），支持距离和相似度两种方式，相似度=1-距离/64.0，距离应选择metric的参数为SIMHASH_HAMMING，相似度应选择metric的参数为SIMHASH_HAMMING_SIM。

MinHash 支持相似度计算，应选择metric的参数为MINHASH_SIM。

Jaccard 支持相似度计算，应选择metric的参数为JACCARD_SIM。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		

lambda	匹配字符权重	匹配字符权重，SSK中使用	Double			0.5
metric	度量类型	计算距离时，可以取不同的度量	String		"LEVENSHTEIN", "LEVENSHTEIN_SIM", "LCS", "LCS_SIM", "SSK", "COSINE", "SIMHASH_HAMMING", "SIMHASH_HAMMING_SIM", "JACCARD_SIM"	"LEVENSHTEI
numBucket	分桶个数	分桶个数	Integer			10
numHashTables	哈希表个数	哈希表的数目	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
seed	采样种子	采样种子	Long			0
windowSize	窗口大小	窗口大小	Integer		[1, +inf)	2

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "abcde", "aabce"],
    [1, "aacedw", "aabbed"],
    [2, "cdefa", "bbcefa"],
    [3, "bdefh", "ddeac"],
    [4, "acedm", "aeefbc"]
])

inOp1 = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')
inOp2 = StreamOperator.fromDataframe(df, schemaStr='id long, text1 string,
text2 string')

stringSimilarityPairwise = StringSimilarityPairwise().setSelectedCols(["text1",
"text2"]).setMetric("LEVENSHTEIN").setOutputCol("output")
stringSimilarityPairwise.transform(inOp1).print()
stringSimilarityPairwise.transform(inOp2).print()
StreamOperator.execute()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.operator.stream.StreamOperator;
```

```
import com.alibaba.alink.pipeline.similarity.StringSimilarityPairwise;
import com.alibaba.alink.operator.stream.source.MemSourceStreamOp;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class StringSimilarityPairwiseTest {
    @Test
    public void testStringSimilarityPairwise() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "abcde", "aabce"),
            Row.of(1, "aacedw", "aabbed"),
            Row.of(2, "cdefa", "bbcefa"),
            Row.of(3, "bdefh", "ddeac"),
            Row.of(4, "acedm", "aeefbc")
        );
        BatchOperator <?> inOp1 = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        StreamOperator <?> inOp2 = new MemSourceStreamOp(df, "id int, text1
string, text2 string");
        StringSimilarityPairwise stringSimilarityPairwise = new
StringSimilarityPairwise().setSelectedCols("text1", "text2").setMetric(
            "LEVENSHTEIN").setOutputCol("output");
        stringSimilarityPairwise.transform(inOp1).print();

        stringSimilarityPairwise.transform(inOp2).print();
        StreamOperator.execute();
    }
}
```

运行结果

id	text1	text2	output
0	abcde	aabce	2.0000
1	aacedw	aabbed	3.0000
2	cdefa	bbcefa	3.0000
3	bdefh	ddeac	3.0000
4	acedm	aeefbc	4.0000

文本近似最近邻 (TextApproxNearestNeighbor)

Java 类名：com.alibaba.alink.pipeline.similarity.TextApproxNearestNeighbor

Python 类名：TextApproxNearestNeighbor

功能介绍

文本相似度是在字符串相似度的基础上，基于词，计算两两文章或者句子之间的相似度，文章或者句子需要以空格分割的文本，计算方式和字符串相似度类似：支持SimHashHamming，MinHash和Jaccard三种近似相似度计算方式，通过选择metric参数可计算不同的相似度。

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

SimhashHamming（SimHash_Hamming_Distance)相似度=1-距离/64.0，应选择metric的参数为SIMHASH_HAMMING_SIM。

MinHash应选择metric的参数为MINHASH_SIM。

Jaccard应选择metric的参数为JACCARD_SIM。

参数说明

名称	中文名称	描述	类型	是否必须?	取值
idCol	id列名	id列名	String	✓	
selectedCol	选中的列名	计算列对应的列名	String	✓	
metric	距离类型	用于计算的距离类型	String		"SIMHASH_H "SIMHASH_H "MINHASH_J "JACCARD_S
modelFilePath	模型的文件路径	模型的文件路径	String		
numBucket	分桶个数	分桶个数	Integer		

numHashTables	哈希表个数	哈希表的数目	Integer		
outputCol	输出结果列	输出结果列列名，可选，默认null	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
radius	radius值	radius值	Double		
reservedCols	算法保留列名	算法保留列	String[]		
seed	采样种子	采样种子	Long		
topN	TopN的值	TopN的值	Integer		[1, +inf)
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "a b c d e", "a a b c e"],
    [1, "a a c e d w", "a a b b e d"],
    [2, "c d e f a", "b b c e f a"],
    [3, "b d e f h", "d d e a c"],
    [4, "a c e d m", "a e e f b c"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')

pipeline =
TextApproxNearestNeighbor().setIdCol("id").setSelectedCol("text1").setMetric("S
IMHASH_HAMMING_SIM").setTopN(3)

pipeline.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.similarity.TextApproxNearestNeighbor;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class TextApproxNearestNeighborTest {
    @Test
    public void testTextApproxNearestNeighbor() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "a b c d e", "a a b c e"),
            Row.of(1, "a a c e d w", "a a b b e d"),
            Row.of(2, "c d e f a", "b b c e f a"),
            Row.of(3, "b d e f h", "d d e a c"),
```

```
            Row.of(4, "a c e d m", "a e e f b c")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        TextApproxNearestNeighbor pipeline = new
TextApproxNearestNeighbor().setIdCol("id").setSelectedCol("text1")
            .setMetric("SIMHASH_HAMMING_SIM").setTopN(3);
        pipeline.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

id	text1	text2
0	{"ID":"[0,1,4]","METRIC":"[1.0,0.96875,0.921875]"}	a a b c e
1	{"ID":"[1,0,4]","METRIC":"[1.0,0.96875,0.921875]"}	a a b b e d
2	{"ID":"[2,4,1]","METRIC":"[1.0,0.9375,0.890625]"}	b b c e f a
3	{"ID":"[3,4,2]","METRIC":"[1.0,0.828125,0.828125]"}	d d e a c
4	{"ID":"[4,2,1]","METRIC":"[1.0,0.9375,0.921875]"}	a e e f b c

文本最近邻 (TextNearestNeighbor)

Java 类名：com.alibaba.alink.pipeline.similarity.TextNearestNeighbor

Python 类名：TextNearestNeighbor

功能介绍

文本相似度是在字符串相似度的基础上，基于词，计算两两文章或者句子之间的相似度，文章或者句子需要以空格分割的文本，计算方式和字符串相似度类似：支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine三种精确相似度计算方式，通过选择metric参数可计算不同的相似度。

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

Levenshtein（Levenshtein Distance）相似度=(1-距离)/length，length为两个字符长度的最大值离，应选metric的参数为LEVENSHTEIN_SIM。

LCS（Longest Common SubString）相似度=(1-距离)/length，length为两个字符长度的最大值，应选择metric的参数为LCS_SIM。

SSK（String Subsequence Kernel）支持相似度计算，应选择metric的参数为SSK。

Cosine（Cosine）支持相似度计算，应选择metric的参数为COSINE。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范
idCol	id列名	id列名	String	✓	
selectedCol	选中的列名	计算列对应的列名	String	✓	
lambda	匹配字符权重	匹配字符权重，SSK中使用	Double		
metric	距离类型	用于计算的距离类型	String		"LEVENSHTE "LEVENSHTE "LCS_SIM", " "SSK", "COSI

modelFilePath	模型的文件路径	模型的文件路径	String		
outputCol	输出结果列	输出结果列列名，可选，默认null	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
radius	radius值	radius值	Double		
reservedCols	算法保留列名	算法保留列	String[]		
topN	TopN的值	TopN的值	Integer		[1, +inf)
windowSize	窗口大小	窗口大小	Integer		[1, +inf)
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "a b c d e", "a a b c e"],
    [1, "a a c e d w", "a a b b e d"],
    [2, "c d e f a", "b b c e f a"],
    [3, "b d e f h", "d d e a c"],
    [4, "a c e d m", "a e e f b c"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2 string')

pipeline = TextNearestNeighbor().setIdCol("id").setSelectedCol("text1").setMetric("LEVENSHTEIN_SIM").setTopN(3)

pipeline.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.similarity.TextNearestNeighbor;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class TextNearestNeighborTest {
    @Test
    public void testTextNearestNeighbor() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "a b c d e", "a a b c e"),
            Row.of(1, "a a c e d w", "a a b b e d"),
            Row.of(2, "c d e f a", "b b c e f a"),
            Row.of(3, "b d e f h", "d d e a c"),
            Row.of(4, "a c e d m", "a e e f b c")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        TextNearestNeighbor pipeline = new
TextNearestNeighbor().setIdCol("id").setSelectedCol("text1").setMetric(
            "LEVENSHTEIN_SIM").setTopN(3);
        pipeline.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

id	text1	text2
0	{"ID":"[0,1,4]","METRIC":"[1.0,0.5,0.4]"}	a a b c e
1	{"ID":"[1,4,0]","METRIC":"[1.0,0.6666666666666667,0.5]"}	a a b b e d
2	{"ID":"[2,3,4]","METRIC":"[1.0,0.6,0.19999999999999996]"}	b b c e f a
3	{"ID":"[3,2,4]","METRIC":"[1.0,0.6,0.19999999999999996]"}	d d e a c
4	{"ID":"[4,1,0]","METRIC":"[1.0,0.6666666666666667,0.4]"}	a e e f b c

文本两两相似度计算 (TextSimilarityPairwise)

Java 类名：com.alibaba.alink.pipeline.similarity.TextSimilarityPairwise

Python 类名：TextSimilarityPairwise

功能介绍

文章相似度是在字符串相似度的基础上，基于词，计算两两文章或者句子之间的相似度，文章或者句子需要以空格分割的文本，计算方式和字符串相似度类似： 支持Levenshtein Distance，Longest Common SubString，String Subsequence Kernel，Cosine，SimHashHamming，MinHash和Jaccard七种相似度计算方式，通过选择metric参数可计算不同的相似度。

Levenshtein（Levenshtein Distance）支持距离和相似度两种方式，相似度=(1-距离)/length，length为两个字符长度的最大值，距离应选择metric的参数为LEVENSHTEIN，相似度应选metric的参数为LEVENSHTEIN_SIM。

LCS（Longest Common SubString）支持距离和相似度两种参数，相似度=(1-距离)/length，length为两个字符长度的最大值，距离应选择metric的参数为LCS，相似度应选择metric的参数为LCS_SIM。

SSK（String Subsequence Kernel）支持相似度计算，应选择metric的参数为SSK。

Cosine（Cosine）支持相似度计算，应选择metric的参数为COSINE。

SimhashHamming（SimHash_Hamming_Distance)，支持距离和相似度两种方式，相似度=1-距离/64.0，距离应选择metric的参数为SIMHASH_HAMMING，相似度应选择metric的参数为SIMHASH_HAMMING_SIM。

MinHash 支持相似度计算，应选择metric的参数为MINHASH_SIM。

Jaccard 支持相似度计算，应选择metric的参数为JACCARD_SIM。

Alink上文本相似度算法包括Batch组件和Stream组件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		

selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
lambda	匹配字符权重	匹配字符权重，SSK中使用	Double			0.5
metric	度量类型	计算距离时，可以取不同的度量	String		"LEVENSHTEIN", "LEVENSHTEIN_SIM", "LCS", "LCS_SIM", "SSK", "COSINE", "SIMHASH_HAMMING", "SIMHASH_HAMMING_SIM", "JACCARD_SIM"	"LEVENSHTEI
numBucket	分桶个数	分桶个数	Integer			10
numHashTables	哈希表个数	哈希表的数目	Integer			10
reservedCols	算法保留列名	算法保留列	String[]			null
seed	采样种子	采样种子	Long			0
windowSize	窗口大小	窗口大小	Integer		[1, +inf)	2

numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "a b c d e", "a a b c e"],
    [1, "a a c e d w", "a a b b e d"],
    [2, "c d e f a", "b b c e f a"],
    [3, "b d e f h", "d d e a c"],
    [4, "a c e d m", "a e e f b c"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id long, text1 string, text2
string')
similarity = TextSimilarityPairwise().setSelectedCols(["text1",
"text2"]).setMetric("LEVENSHTEIN").setOutputCol("output")
similarity.transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.similarity.TextSimilarityPairwise;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;
```

```
public class TextSimilarityPairwiseTest {
    @Test
    public void testTextSimilarityPairwise() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "a b c d e", "a a b c e"),
            Row.of(1, "a a c e d w", "a a b b e d"),
            Row.of(2, "c d e f a", "b b c e f a"),
            Row.of(3, "b d e f h", "d d e a c"),
            Row.of(4, "a c e d m", "a e e f b c")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, text1
string, text2 string");
        TextSimilarityPairwise similarity = new
TextSimilarityPairwise().setSelectedCols("text1", "text2").setMetric(
            "LEVENSHTEIN").setOutputCol("output");
        similarity.transform(inOp).print();
    }
}
```

运行结果

id	text1	text2	output
0	a b c d e	a a b c e	2.0
1	a a c e d w	a a b b e d	3.0
2	c d e f a	b b c e f a	3.0
3	b d e f h	d d e a c	3.0
4	a c e d m	a e e f b c	4.0

向量近似最近邻 (VectorApproxNearestNeighbor)

Java 类名：com.alibaba.alink.pipeline.similarity.VectorApproxNearestNeighbor

Python 类名：VectorApproxNearestNeighbor

功能介绍

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
idCol	id列名	id列名	String	✓	
selectedCol	选中的列名	计算列对应的列名	String	✓	
metric	距离度量方式	距离类型	String		"EUCLIDEAN "JACCARD"
modelFilePath	模型的文件路径	模型的文件路径	String		
numHashTables	哈希表的数目	哈希表的数目	Integer		
numProjectionsPerTable	每个哈希表中的哈希函数个数	每个哈希表中的哈希函数个数	Integer		

outputCol	输出结果列	输出结果列列名，可选，默认null	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
projectionWidth	桶的宽度	桶的宽度	Double		
radius	radius值	radius值	Double		
reservedCols	算法保留列名	算法保留列	String[]		
seed	采样种子	采样种子	Long		
solver	近似方法	近似方法，包括KDTREE和LSH	String		"KDTREE", "LSH"
topN	TopN的值	TopN的值	Integer		[1, +inf)
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		

modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "1 1 1"],
    [2, "2 2 2"]
])

inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
pipeline =
VectorApproxNearestNeighbor().setIdCol("id").setSelectedCol("vec").setTopN(3)
pipeline.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.similarity.VectorApproxNearestNeighbor;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorApproxNearestNeighborTest {
    @Test
    public void testVectorApproxNearestNeighbor() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
```

```
            Row.of(1, "1 1 1"),
            Row.of(2, "2 2 2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, vec
string");
        VectorApproxNearestNeighbor pipeline = new
VectorApproxNearestNeighbor().setIdCol("id").setSelectedCol("vec")
            .setTopN(3);
        pipeline.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

id	vec
0	{"ID":"[0,1,2]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}
1	{"ID":"[1,2,0]","METRIC":"[0.0,1.7320508075688772,1.7320508075688772]"}
2	{"ID":"[2,1,0]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}

向量最近邻 (VectorNearestNeighbor)

Java 类名：com.alibaba.alink.pipeline.similarity.VectorNearestNeighbor

Python 类名：VectorNearestNeighbor

功能介绍

该功能由训练和预测组成，支持计算1. 求最近邻topN 2. 求radius范围内的邻居。该功能由预测时候的topN和radius参数控制, 如果填写了topN，则输出最近邻，如果填写了radius，则输出radius范围内的邻居。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
idCol	id列名	id列名	String	✓	
selectedCol	选中的列名	计算列对应的列名	String	✓	
metric	距离度量方式	聚类使用的距离类型	String		"EUCLIDEAN "COSINE", "INNERPROD "CITYBLOCK "JACCARD", "PEARSON"
modelFilePath	模型的文件路径	模型的文件路径	String		
outputCol	输出结果列	输出结果列列名，可选，默认null	String		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
radius	radius值	radius值	Double		

reservedCols	算法保留列名	算法保留列	String[]		
topN	TopN的值	TopN的值	Integer		[1, +inf)
numThreads	组件多线程线程个数	组件多线程线程个数	Integer		
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String		
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer		
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用yyyy-mm-dd hh:mm:ss.fffffffff格式，详见Timestamp.valueOf(String s)	String		

代码示例

Python 代码

```
from pyalink.alink import *

import pandas as pd

useLocalEnv(1)

df = pd.DataFrame([
    [0, "0 0 0"],
    [1, "1 1 1"],
    [2, "2 2 2"]
])
```

```
inOp = BatchOperator.fromDataframe(df, schemaStr='id int, vec string')
pipeline =
VectorNearestNeighbor().setIdCol("id").setSelectedCol("vec").setTopN(3)

pipeline.fit(inOp).transform(inOp).print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.similarity.VectorNearestNeighbor;
import org.junit.Test;

import java.util.Arrays;
import java.util.List;

public class VectorNearestNeighborTest {
    @Test
    public void testVectorNearestNeighbor() throws Exception {
        List <Row> df = Arrays.asList(
            Row.of(0, "0 0 0"),
            Row.of(1, "1 1 1"),
            Row.of(2, "2 2 2")
        );
        BatchOperator <?> inOp = new MemSourceBatchOp(df, "id int, vec
string");
        VectorNearestNeighbor pipeline = new
VectorNearestNeighbor().setIdCol("id").setSelectedCol("vec").setTopN(3);
        pipeline.fit(inOp).transform(inOp).print();
    }
}
```

运行结果

id	vec
0	{"ID":"[0,1,2]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}
1	{"ID":"[1,2,0]","METRIC":"[0.0,1.7320508075688772,1.7320508075688772]"}
2	{"ID":"[2,1,0]","METRIC":"[0.0,1.7320508075688772,3.4641016151377544]"}

MFCC特征提取 (ExtractMfccFeature)

Java 类名：com.alibaba.alink.pipeline.audio.ExtractMfccFeature

Python 类名：ExtractMfccFeature

功能介绍

从 Alink Tensor 格式的音频数据中提取 MFCC 特征。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
sampleRate	采样率	采样率	Integer	✓		
selectedCol	选中的列名	计算列对应的列名	String	✓		
hopTime	相邻窗口时间间隔	相邻窗口时间间隔	Double			0.032
numMfcc	mfcc参数	mfcc参数	Integer			128
outputCol	输出结果列	输出结果列列名，可选，默认null	String			null
reservedCols	算法保留列名	算法保留列	String[]			null
windowTime	一个窗口的时间	一个窗口的时间	Double			0.128
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

```
DATA_DIR = "https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/audio"
SAMPLE_RATE = 16000

df = pd.DataFrame([
    "246.wav",
    "247.wav"
])
```

```
source = BatchOperator.fromDataframe(df, "wav_file_path string") \
        .link(ReadAudioToTensorBatchOp()
                .setRootFilePath(DATA_DIR)
                .setSampleRate(SAMPLE_RATE)
                .setRelativeFilePathCol("wav_file_path")
                .setDuration(2.)
                .setOutputCol("tensor")
        )

mfcc = ExtractMfccFeature() \
        .setSelectedCol("tensor") \
        .setSampleRate(SAMPLE_RATE) \
        .setWindowTime(0.128) \
        .setHopTime(0.032) \
        .setNumMfcc(26) \
        .setOutputCol("mfcc")

mfcc.transform(source).print()
```

Java 代码

```
public class ExtractMfccFeatureTest {
    @Test
    public void testExtractMfccFeature() throws Exception {
        String DATA_DIR = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/audio";
        String[] allFiles = {"246.wav", "247.wav"};
        int SAMPLE_RATE = 16000;
        BatchOperator source = new MemSourceBatchOp(allFiles, "wav_file_path")
                .link(new ReadAudioToTensorBatchOp()
                        .setRootFilePath(DATA_DIR)
                        .setSampleRate(SAMPLE_RATE)
                        .setRelativeFilePathCol("wav_file_path")
                        .setDuration(2)
                        .setOutputCol("tensor")
                );

        ExtractMfccFeature mfcc = new ExtractMfccFeature()
                .setSelectedCol("tensor")
                .setSampleRate(SAMPLE_RATE)
                .setWindowTime(0.128)
                .setHopTime(0.032)
                .setNumMfcc(26)
                .setOutputCol("mfcc");

        mfcc.transform(source).print();
```

```
    }
}
```

音频转张量 (ReadAudioToTensor)

Java 类名：com.alibaba.alink.pipeline.audio.ReadAudioToTensor

Python 类名：ReadAudioToTensor

功能介绍

读取音频文件，并转换为 Alink FloatTensor 格式。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
relativeFilePathCol	文件路径列	文件路径列	String	✓		
rootFilePath	文件路径	文件路径	String	✓		
sampleRate	采样率	采样率	Integer	✓		
duration	采样持续时间	采样持续时间	Double			
offset	采样开始时刻	采样开始时刻	Double			0.0
reservedCols	算法保留列名	算法保留列	String[]			null
numThreads	组件多线程线程个数	组件多线程线程个数	Integer			1

代码示例

Python 代码

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

```
dataDir = "https://alink-test-data.oss-cn-hangzhou.aliyuncs.com/audio";

df = pd.DataFrame([
    ["246.wav"],
    ["247.wav"]
])
```

```
allFiles = BatchOperator.fromDataframe(df, schemaStr='wav_file_path string')
SAMPLE_RATE = 16000

readOp = ReadAudioToTensor().setRootFilePath(dataDir) \
.setSampleRate(SAMPLE_RATE) \
.setRelativeFilePathCol("wav_file_path") \
.setOutputCol("tensor") \
.transform(allFiles)

readOp.print()
```

Java 代码

```
package example;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.audio.ReadAudioToTensor;
import org.junit.Test;

public class ReadAudioToTensorTest {

    @Test
    public void testReadAudioToTensor() throws Exception {
        String dataDir = "https://alink-test-data.oss-cn-
hangzhou.aliyuncs.com/audio";
        String[] allFiles = {"246.wav", "247.wav"};
        int sampleRate = 16000;
        BatchOperator source = new MemSourceBatchOp(allFiles, "wav_file_path");
        ReadAudioToTensor pipe = new ReadAudioToTensor()
                .setRootFilePath(dataDir)
                .setSampleRate(sampleRate)
                .setRelativeFilePathCol("wav_file_path")
                .setDuration(2)
                .setOutputCol("tensor");
        pipe.transform(source).print();
    }

}
```

运行结果

wav_file_path	tensor
246.wav	FLOAT#32000,1#-7.324219E-4 -0.0010986328 -9.460449E-4 ...
247.wav	FLOAT#32000,1#-0.0057678223 -0.0051574707 -0.0036315918 ...

图片转张量 (ReadImageToTensor)

Java 类名：com.alibaba.alink.pipeline.image.ReadImageToTensor

Python 类名：ReadImageToTensor

功能介绍

将图片列转换为张量。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
outputCol	输出结果列列名	输出结果列列名，必选	String	✓		
relativeFilePathCol	文件路径列	文件路径列	String	✓		
rootFilePath	文件路径	文件路径	String	✓		
imageHeight	图片高度	图片高度	Integer			
imageWidth	图片宽度	图片宽度	Integer			
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
df_data = pd.DataFrame([
    'sphx_glr_plot_scripted_tensor_transforms_001.png'
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr = 'path string')

ReadImageToTensor()\
    .setRootFilePath("https://pytorch.org/vision/stable/_images/")\
    .setRelativeFilePathCol("path")\
    .setOutputCol("tensor")\
    .transform(batch_data)\
    .print()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.pipeline.image.ReadImageToTensor;
import org.junit.Test;

import java.util.Collections;
import java.util.List;

public class ReadImageToTensorTest {

    @Test
    public void testReadImageToTensor() throws Exception {

        List <Row> data = Collections.singletonList(
            Row.of("sphx_glr_plot_scripted_tensor_transforms_001.png")
        );

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "path
string");

        new ReadImageToTensor()
            .setRootFilePath("https://pytorch.org/vision/stable/_images/")
            .setRelativeFilePathCol("path")
            .setOutputCol("tensor")
            .transform(memSourceBatchOp)
            .print();
    }

}
```

运行结果

| path | tensor | |----------+----------| | sphx_glr_plot_scripted_tensor_transforms_001.png | FLOAT#250,520,4#1.0 1.0 1.0... |

张量转图片 (WriteTensorToImage)

Java 类名：com.alibaba.alink.pipeline.image.WriteTensorToImage

Python 类名：WriteTensorToImage

功能介绍

将张量列转换为图片，并写入根目录对应的相对路径列中，然后原样输出结果。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
relativeFilePathCol	文件路径列	文件路径列	String	✓		
rootFilePath	文件路径	文件路径	String	✓		
tensorCol	tensor列	tensor列	String	✓		
imageType	图片类型	图片类型	String		"PNG", "JPEG"	"PNG"
reservedCols	算法保留列名	算法保留列	String[]			null

代码示例

Python 代码

```
df_data = pd.DataFrame([
    'sphx_glr_plot_scripted_tensor_transforms_001.png'
])

batch_data = BatchOperator.fromDataframe(df_data, schemaStr = 'path string')

readImageToTensorBatchOp = ReadImageToTensorBatchOp()\
    .setRootFilePath("https://pytorch.org/vision/stable/_images/")\
    .setRelativeFilePathCol("path")\
    .setOutputCol("tensor")

writeTensorToImageBatchOp = WriteTensorToImageBatchOp()\
            .setRootFilePath("/tmp/write_tensor_to_image")\
```

```
            .setTensorCol("tensor")\
            .setImageType("png")\
            .setRelativeFilePathCol("path")

batch_data.link(readImageToTensorBatchOp).link(writeTensorToImageBatchOp).print
()
```

Java 代码

```
import org.apache.flink.types.Row;

import com.alibaba.alink.operator.batch.source.MemSourceBatchOp;
import com.alibaba.alink.params.image.HasImageType.ImageType;
import com.alibaba.alink.pipeline.image.WriteTensorToImage;
import org.junit.Test;

import java.util.Collections;
import java.util.List;

public class WriteTensorToImageTest {

    @Test
    public void testWriteTensorToImage() throws Exception {

        List <Row> data = Collections.singletonList(
            Row.of("sphx_glr_plot_scripted_tensor_transforms_001.png")
        );

        MemSourceBatchOp memSourceBatchOp = new MemSourceBatchOp(data, "path
string");

        ReadImageToTensorBatchOp readImageToTensorBatchOp = new
ReadImageToTensorBatchOp()
            .setRootFilePath("https://pytorch.org/vision/stable/_images/")
            .setRelativeFilePathCol("path")
            .setOutputCol("tensor");

        WriteTensorToImage writeTensorToImageBatchOp = new WriteTensorToImage()
            .setRootFilePath("/tmp/write_tensor_to_image")
            .setTensorCol("tensor")
            .setImageType(ImageType.PNG)
            .setRelativeFilePathCol("path");


writeTensorToImageBatchOp.transform(memSourceBatchOp.link(readImageToTensorBatc
hOp)).print();
    }
```

```
}
```

运行结果

可以在 /tmp/write_tensor_to_image/sphx_glr_plot_scripted_tensor_transforms_001.png 中找到 https://pytorch.org/vision/stable/_images/sphx_glr_plot_scripted_tensor_transforms_001.png

同时组件的输出结果为：

| path | tensor | |----------+----------| | sphx_glr_plot_scripted_tensor_transforms_001.png | FLOAT#250,520,4#255.0 255.0... |

ONNX 模型预测 (OnnxModelPredictor)

Java 类名：com.alibaba.alink.pipeline.onnx.OnnxModelPredictor

Python 类名：OnnxModelPredictor

功能介绍

加载 ONNX 模型进行预测。

使用方式

模型路径 `modelPath` 需要是 ONNX 模型。

参与模型预测的数据通过参数 `selectedCols` 设置，需要注意以下几点：

- ONNX 模型使用 input name 来标识模型输入桩的，因此需要设置 `inputNames`，与 `selectedCols` 一一对应，表明某列对应某输入桩。`inputNames` 不填写时，默认与列名一致。
- 仅支持输入桩为 `Tensor` 类型，不支持 `Sequences` 和 `Maps` 类型。
- 所选择的列的类型需要是 `float, double, int, long, byte, string` 类型及其对应的 Alink `Tensor` 类型。

模型输出信息通过参数 `outputSchemaStr` 指定，包括输出列名以及名称，需要注意以下几点：

- ONNX 模型使用 output name 来标识模型输出桩的，因此需要设置 `outputNames`，与 `outputSchemaStr` 一一对应，表明某列对应某输入桩。`outputNames` 不填写时，默认与列名一致。
- 仅支持输出桩为 `Tensor` 类型，不支持 `Sequences` 和 `Maps` 类型。
- `outputSchemaStr` 填写的输出类型需要是对应的输出桩类型，例如 输出桩类型 为 Float 类型的 Tensor 时，对应的 Alink 类型可以是 `TENSOR` 或者 `FLOAT_TENSOR`，当输出仅包含一个元素时，还可以是 `FLOAT`。

组件使用的是 ONNX 1.11.0 版本，当有 GPU 时，自动使用 GPU 进行推理，否则使用 CPU 进行推理。

在 Windows 下运行时，如果遇到 `UnsatisfiedLinkError`，请下载 Visual C++ 2019 Redistributable Packages 并重启，然后重新运行。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelPath	模型的URL路径	模型的URL路径	String	✓		
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如 "f0 string, f1 bigint, f2 double"	String	✓		

inputNames	ONNX 模型输入名	ONNX 模型输入名，用逗号分隔，需要与输入列一一对应，默认与选择列相同	String[]			null
outputNames	ONNX 模型输出名	ONNX 模型输出名，用逗号分隔，并且与输出 Schema 一一对应，默认与输出 Schema 中的列名相同	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null

模型路径说明

模型路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
test = AkSourceStreamOp()\
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

test = VectorToTensorStreamOp()\
    .setTensorDataType("float")\
    .setTensorShape([1, 1, 28, 28])\
    .setSelectedCol("vec")\
    .setOutputCol("tensor")\
    .setReservedCols(["label"])\
    .linkFrom(test)

predictor = OnnxModelPredictStreamOp() \
    .setModelPath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/cnn_mnist_pytorch.onnx") \
```

```
    .setSelectedCols(["tensor"]) \
    .setInputNames(["0"]) \
    .setOutputNames(["21"]) \
    .setOutputSchemaStr("probabilities FLOAT_TENSOR")

test = predictor.linkFrom(test).select("label, probabilities")
test.print()
StreamOperator.execute()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.VectorToTensorBatchOp;
import com.alibaba.alink.operator.batch.source.AkSourceBatchOp;
import com.alibaba.alink.pipeline.onnx.OnnxModelPredictor;
import org.junit.Test;

public class OnnxModelPredictorTest {
    @Test
    public void testOnnxModelPredictor() throws Exception {
        BatchOperator.setParallelism(1);
        BatchOperator <?> test = new AkSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

        test = new VectorToTensorBatchOp()
            .setTensorDataType("float")
            .setTensorShape(1, 1, 28, 28)
            .setSelectedCol("vec")
            .setOutputCol("tensor")
            .setReservedCols("label")
            .linkFrom(test);

        OnnxModelPredictor predictor = new OnnxModelPredictor()
            .setModelPath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/cnn_mnist_pytorch.onnx")
            .setSelectedCols("tensor")
            .setInputNames("0")
            .setOutputNames("21")
            .setOutputSchemaStr("probabilities FLOAT_TENSOR");

        test = predictor.transform(test).select("label, probabilities");
        test.print();
    }
}
```

TorchScript 模型预测 (TorchModelPredictor)

Java 类名：com.alibaba.alink.pipeline.pytorch.TorchModelPredictor

Python 类名：TorchModelPredictor

功能介绍

加载 TorchScript 模型进行预测。

使用方式

模型路径 `modelPath` 需要是一个通过 `torch.jit` 导出的模型文件路径。

参与模型预测的数据通过参数 `selectedCols` 设置，需要注意以下几点：

- TorchScript 模型调用 `forward` 方法时是通过位置来传入参数的，所以 `selectedCols` 中各列的顺序是有意义的。
- 所选择的列的类型需要是 Alink `Tensor` 类型或者 4 种基本数据类型（`Long, Double, Boolean, String` 及其兼容类型），不接受其他类型。

模型输出信息通过参数 `outputSchemaStr` 指定，包括输出列名以及名称，需要注意以下几点：

- 输出列的数量需要与模型输出结果匹配。
- 输出类型可以是 Alink `Tensor` 类型或者 Alink 支持的类型，如果从模型预测输出的结果转换到指定类型失败那么将报错；暂不支持列表或字典类型。

组件使用的是 PyTorch 1.8.1 CPU 版本，如果需要使用 GPU 功能，可以自行替换插件文件。

在 Windows 下运行时，如果遇到 `UnsatisfiedLinkError`，请下载 Visual C++ 2015 Redistributable Packages 并重启，然后重新运行。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelPath	模型的URL路径	模型的URL路径	String	✓		
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如 "f0 string, f1 bigint, f2 double"	String	✓		

selectedCols	选择的列名	计算列对应的列名列表	String[]	✓		
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4
reservedCols	算法保留列名	算法保留列	String[]			null

模型路径说明

模型路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
test = AkSourceBatchOp()\
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

test = VectorToTensorBatchOp()\
    .setTensorDataType("float")\
    .setTensorShape([1, 1, 28, 28])\
    .setSelectedCol("vec")\
    .setOutputCol("tensor")\
    .setReservedCols(["label"])\
    .linkFrom(test)

predictor = TorchModelPredictor()\
    .setModelPath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/mnist_model_pytorch.pt")\
    .setSelectedCols(["tensor"])\
    .setOutputSchemaStr("probabilities FLOAT_TENSOR")

test = predictor.transform(test).select("label, probabilities")
test.print()
```

Java 代码

```
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.VectorToTensorBatchOp;
import com.alibaba.alink.operator.batch.source.AkSourceBatchOp;
import com.alibaba.alink.pipeline.pytorch.TorchModelPredictor;
import org.junit.Test;

public class TorchModelPredictorTest {

    @Test
    public void testTorchModelPredictor() throws Exception {
        BatchOperator.setParallelism(1);
        BatchOperator <?> test = new AkSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

        test = new VectorToTensorBatchOp()
            .setTensorDataType("float")
            .setTensorShape(1, 1, 28, 28)
            .setSelectedCol("vec")
            .setOutputCol("tensor")
            .setReservedCols("label")
            .linkFrom(test);

        TorchModelPredictor predictor = new TorchModelPredictor()
            .setModelPath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/mnist_model_pytorch.pt")
            .setSelectedCols("tensor")
            .setOutputSchemaStr("probabilities FLOAT_TENSOR");

        test = predictor.transform(test).select("label, probabilities");
        test.print();
    }
}
```

TF2表模型 (TF2TableModelTrainer)

Java 类名：com.alibaba.alink.pipeline.tensorflow.TF2TableModelTrainer

Python 类名：TF2TableModelTrainer

功能介绍

该组件支持用户传入 TensorFlow2 脚本，进行模型训练。

用户需要提供自己编写的 TensorFlow2 脚本文件。 脚本的编写需要依赖 akdl 库，可以参考 `alink_dl_predictors/predictor-tf/src/test/resources/tf_dnn_train.py` 。

脚本中必须要将模型保存为 SavedModel 格式，并导出到指定的目录下 (`TrainTaskConfig#saved_model_dir`)。

调用这个组件的 fit 方法可以得到一个 `TFTableModelPredictor` 进行预测。 需要注意的是：参与预测的列名一般与参与训练的列名不同（预测没有 label 列），需要通过参数 `inferSelectedCols` 来指定参与预测的列名。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
mainScriptFile	主脚本文件路径	主脚本文件路径，需要是参数 userFiles 中的一项，并且包含 main 函数	String	✓	
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如 "f0 string, f1 bigint, f2 double"	String	✓	
userFiles	所有自定义脚本文件的路径	所有自定义脚本文件的路径	String	✓	
graphDefTag	graph标签	graph标签	String		
inferSelectedCols	用于推理的列名数组	用于推理的列名列表	String[]		
inputSignatureDefs	输入SignatureDef	SavedModel 模型的输入 SignatureDef 名，用逗号分隔，需要与输入列一一对应，默认与选择列相同	String[]		

intraOpParallelism	Op 间并发度	Op 间并发度	Integer			
modelFilePath	模型的文件路径	模型的文件路径	String			
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer			
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer			
outputSignatureDefs	TF 输出 SignatureDef 名	模型的输出 SignatureDef 名，多个输出时用逗号分隔，并且与输出 Schema 一一对应，默认与输出 Schema 中的列名相同	String[]			
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			
reservedCols	算法保留列名	算法保留列	String[]			
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			
signatureDefKey	signature标签	signature标签	String			
userParams	自定义参数	用户自定义参数，JSON 字典格式的字符串	String			

modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			

脚本路径说明

脚本路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
import json

source = RandomTableSourceBatchOp() \
    .setNumRows(100) \
    .setNumCols(10)

colNames = source.getColNames()
source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as
label")
label = "label"

userParams = {
    'featureCols': json.dumps(colNames),
    'labelCol': label,
    'batch_size': 16,
    'num_epochs': 1
```

```
}

trainer = TF2TableModelTrainer() \
    .setUserFiles(["https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/tf_dnn_train.py"]) \
    .setMainScriptFile("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/tf_dnn_train.py") \
    .setUserParams(json.dumps(userParams)) \
    .setOutputSchemaStr("logits double") \
    .setOutputSignatureDefs(["logits"]) \
    .setSignatureDefKey("predict") \
    .setInferSelectedCols(colNames)
model = trainer.fit(source)
model.transform(source).print()
```

Java 代码

```
import com.alibaba.alink.common.utils.JsonConverter;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.RandomTableSourceBatchOp;
import com.alibaba.alink.pipeline.tensorflow.TF2TableModelTrainer;
import com.alibaba.alink.pipeline.tensorflow.TFTableModelPredictor;
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

public class TF2TableModelTrainerTest {

    @Test
    public void testTF2TableModelTrainer() throws Exception {
        BatchOperator.setParallelism(3);

        BatchOperator<?> source = new RandomTableSourceBatchOp()
            .setNumRows(100L)
            .setNumCols(10);

        String[] colNames = source.getColNames();
        source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as label");
        String label = "label";

        Map <String, Object> userParams = new HashMap <>();
        userParams.put("featureCols", JsonConverter.toJson(colNames));
        userParams.put("labelCol", label);
        userParams.put("batch_size", 16);
        userParams.put("num_epochs", 1);
```

```
        TF2TableModelTrainer trainer = new TF2TableModelTrainer()
            .setUserFiles(new String[] {"res:///tf_dnn_train.py"})
            .setMainScriptFile("res:///tf_dnn_train.py")
            .setUserParams(JsonConverter.toJson(userParams))
            .setNumWorkers(2)
            .setNumPSs(1)
            .setOutputSchemaStr("logits double")
            .setOutputSignatureDefs(new String[]{"logits"})
            .setSignatureDefKey("predict")
            .setInferSelectedCols(colNames);

        TFTableModelPredictor model = trainer.fit(source);
        model.transform(source).print();
    }
}
```

TF SavedModel 模型预测 (TFSavedModelPredictor)

Java 类名：com.alibaba.alink.pipeline.tensorflow.TFSavedModelPredictor

Python 类名：TFSavedModelPredictor

功能介绍

该组件支持直接使用 SavedModel 进行预测。

模型路径需要时一个压缩文件，解压后能得到一个目录，目录内包含 SavedModel 的文件。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围	默认值
modelPath	模型的URL路径	模型的URL路径	String	✓		
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如"f0 string, f1 bigint, f2 double"	String	✓		
graphDefTag	graph标签	graph标签	String			"serve"
inputSignatureDefs	输入SignatureDef	SavedModel模型的输入SignatureDef名，用逗号分隔，需要与输入列一一对应，默认与选择列相同	String[]			null
intraOpParallelism	Op 间并发度	Op 间并发度	Integer			4

outputSignatureDefs	TF 输出 SignatureDef 名	模型的输出 SignatureDef 名，多个输出时用逗号分隔，并且与输出 Schema 一一对应，默认与输出 Schema 中的列名相同	String[]			null
reservedCols	算法保留列名	算法保留列	String[]			null
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			null
signatureDefKey	signature标签	signature标签	String			"serving_default"

模型路径说明

模型路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
test = AkSourceBatchOp()\
    .setFilePath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

test = VectorToTensorBatchOp()\
    .setTensorDataType("float")\
    .setTensorShape([1, 28, 28, 1])\
    .setSelectedCol("vec")\
    .setOutputCol("tensor")\
    .setReservedCols(["label"])\
```

```
    .linkFrom(test)

predictor = TFSavedModelPredictor()\
    .setModelPath("https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/mnist_model_tf.zip")\
    .setSelectedCols(["tensor"])\
    .setInputSignatureDefs(["input_1"])\
    .setOutputSignatureDefs(["output_1"])\
    .setOutputSchemaStr("probabilities FLOAT_TENSOR")

test = predictor.transform(test).select("label, probabilities")
test.print()
```

Java 代码

```
package examples;

import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.dataproc.VectorToTensorBatchOp;
import com.alibaba.alink.operator.batch.source.AkSourceBatchOp;
import com.alibaba.alink.pipeline.tensorflow.TFSavedModelPredictor;
import org.junit.Test;

public class TFSavedModelPredictorTest {

    @Test
    public void testTFSavedModelPredictor() throws Exception {
        BatchOperator.setParallelism(1);
        BatchOperator <?> test = new AkSourceBatchOp()
            .setFilePath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/mnist_test_vector.ak");

        test = new VectorToTensorBatchOp()
            .setTensorDataType("float")
            .setTensorShape(1, 28, 28, 1)
            .setSelectedCol("vec")
            .setOutputCol("tensor")
            .setReservedCols("label")
            .linkFrom(test);

        TFSavedModelPredictor predictor = new TFSavedModelPredictor()
            .setModelPath("https://alink-release.oss-cn-
beijing.aliyuncs.com/data-files/mnist_model_tf.zip")
            .setSelectedCols("tensor")
            .setInputSignatureDefs(new String[] {"input_1"})
            .setOutputSignatureDefs(new String[] {"output_1"})
            .setOutputSchemaStr("probabilities FLOAT_TENSOR");
```

```
        test = predictor.transform(test).select("label, probabilities");
        test.print();
    }
}
```

TF 表模型预测 (TFTableModelPredictor)

Java 类名：com.alibaba.alink.pipeline.tensorflow.TFTableModelPredictor

Python 类名：TFTableModelPredictor

功能介绍

由 `TFTableModelTrainer` 或者 `TF2TableModelTrainer` 调用 fit 方法产生的模型，可以进行预测。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如 "f0 string, f1 bigint, f2 double"	String	✓	
graphDefTag	graph标签	graph标签	String		
inputSignatureDefs	输入SignatureDef	SavedModel 模型的输入 SignatureDef 名，用逗号分隔，需要与输入列一一对应，默认与选择列相同	String[]		
intraOpParallelism	Op 间并发度	Op 间并发度	Integer		
modelFilePath	模型的文件路径	模型的文件路径	String		
outputSignatureDefs	TF 输出SignatureDef名	模型的输出 SignatureDef 名，多个输出时用逗号分隔，并且与输出 Schema 一一对应，默认与输出 Schema 中的列名相同	String[]		
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean		
reservedCols	算法保留列名	算法保留列	String[]		

selectedCols	选中的列名数组	计算列对应的列名列表	String[]			
signatureDefKey	signature标签	signature标签	String			
modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			

脚本路径说明

脚本路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
import json

source = RandomTableSourceBatchOp() \
    .setNumRows(100) \
    .setNumCols(10)

colNames = source.getColNames()
source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as label")
label = "label"
```

```
userParams = {
    'featureCols': json.dumps(colNames),
    'labelCol': label,
    'batch_size': 16,
    'num_epochs': 1
}

trainer = TF2TableModelTrainer() \
    .setUserFiles(["https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/tf_dnn_train.py"]) \
    .setMainScriptFile("https://alink-release.oss-cn-beijing.aliyuncs.com/data-files/tf_dnn_train.py") \
    .setUserParams(json.dumps(userParams)) \
    .setOutputSchemaStr("logits double") \
    .setOutputSignatureDefs(["logits"]) \
    .setSignatureDefKey("predict") \
    .setInferSelectedCols(colNames)
model = trainer.fit(source)
model.transform(source).print()
```

Java 代码

```
import com.alibaba.alink.common.utils.JsonConverter;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.RandomTableSourceBatchOp;
import com.alibaba.alink.pipeline.tensorflow.TF2TableModelTrainer;
import com.alibaba.alink.pipeline.tensorflow.TFTableModelPredictor;
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

public class TF2TableModelTrainerTest {

    @Test
    public void testTF2TableModelTrainer() throws Exception {
        BatchOperator.setParallelism(3);

        BatchOperator<?> source = new RandomTableSourceBatchOp()
            .setNumRows(100L)
            .setNumCols(10);

        String[] colNames = source.getColNames();
        source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as label");
        String label = "label";
```

```
        Map <String, Object> userParams = new HashMap <>();
        userParams.put("featureCols", JsonConverter.toJson(colNames));
        userParams.put("labelCol", label);
        userParams.put("batch_size", 16);
        userParams.put("num_epochs", 1);

        TF2TableModelTrainer trainer = new TF2TableModelTrainer()
            .setUserFiles(new String[] {"res:///tf_dnn_train.py"})
            .setMainScriptFile("res:///tf_dnn_train.py")
            .setUserParams(JsonConverter.toJson(userParams))
            .setNumWorkers(2)
            .setNumPSs(1)
            .setOutputSchemaStr("logits double")
            .setOutputSignatureDefs(new String[]{"logits"})
            .setSignatureDefKey("predict")
            .setInferSelectedCols(colNames);

        TFTableModelPredictor model = trainer.fit(source);
        model.transform(source).print();
    }
}
```

TF 表模型 (TFTableModelTrainer)

Java 类名：com.alibaba.alink.pipeline.tensorflow.TFTableModelTrainer

Python 类名：TFTableModelTrainer

功能介绍

该组件支持用户传入 TensorFlow 脚本，进行模型训练。

用户需要提供自己编写的 TensorFlow 脚本文件。 脚本的编写需要依赖 akdl 库，可以参考 `alink_dl_predictors/predictor-tf/src/test/resources/tf_dnn_train.py` 。

脚本中必须要将模型保存为 SavedModel 格式，并导出到指定的目录下 (`TrainTaskConfig#saved_model_dir`)。

调用这个组件的 fit 方法可以得到一个 `TFTableModelPredictor` 进行预测。 需要注意的是：参与预测的列名一般与参与训练的列名不同（预测没有 label 列），需要通过参数 `inferSelectedCols` 来指定参与预测的列名。

参数说明

名称	中文名称	描述	类型	是否必须?	取值范围
mainScriptFile	主脚本文件路径	主脚本文件路径，需要是参数 userFiles 中的一项，并且包含 main 函数	String	✓	
outputSchemaStr	Schema	Schema。格式为"colname coltype[, colname2, coltype2[, ...]]"，例如 "f0 string, f1 bigint, f2 double"	String	✓	
userFiles	所有自定义脚本文件的路径	所有自定义脚本文件的路径	String	✓	
graphDefTag	graph标签	graph标签	String		
inferSelectedCols	用于推理的列名数组	用于推理的列名列表	String[]		
inputSignatureDefs	输入SignatureDef	SavedModel 模型的输入 SignatureDef 名，用逗号分隔，需要与输入列一一对应，默认与选择列相同	String[]		

intraOpParallelism	Op 间并发度	Op 间并发度	Integer			
modelFilePath	模型的文件路径	模型的文件路径	String			
numPSs	PS 角色数	PS 角色的数量。值未设置时，如果 Worker 角色数也未设置，则为作业总并发度的 1/4（需要取整），否则为总并发度减去 Worker 角色数。	Integer			
numWorkers	Worker 角色数	Worker 角色的数量。值未设置时，如果 PS 角色数也未设置，则为作业总并发度的 3/4（需要取整），否则为总并发度减去 PS 角色数。	Integer			
outputSignatureDefs	TF 输出 SignatureDef 名	模型的输出 SignatureDef 名，多个输出时用逗号分隔，并且与输出 Schema 一一对应，默认与输出 Schema 中的列名相同	String[]			
overwriteSink	是否覆写已有数据	是否覆写已有数据	Boolean			
pythonEnv	Python 环境路径	Python 环境路径，一般情况下不需要填写。如果是压缩文件，需要解压后得到一个目录，且目录名与压缩文件主文件名一致，可以使用 http://, https://, oss://, hdfs:// 等路径；如果是目录，那么只能使用本地路径，即 file://。	String			
reservedCols	算法保留列名	算法保留列	String[]			
selectedCols	选中的列名数组	计算列对应的列名列表	String[]			
signatureDefKey	signature标签	signature标签	String			
userParams	自定义参数	用户自定义参数，JSON 字典格式的字符串	String			

modelStreamFilePath	模型流的文件路径	模型流的文件路径	String			
modelStreamScanInterval	扫描模型路径的时间间隔	描模型路径的时间间隔，单位秒	Integer			
modelStreamStartTime	模型流的起始时间	模型流的起始时间。默认从当前时刻开始读。使用 yyyy-mm-dd hh:mm:ss.fffffffff格式，详见 Timestamp.valueOf(String s)	String			

脚本路径说明

脚本路径可以是以下形式：

- 本地文件：`file://` 加绝对路径，例如 `file:///tmp/dnn.py`；
- Java 包中的资源文件：`res://` 加路径，例如 `res:///dnn.py`；
- http/https 文件：`http://` 或 `https://` 路径；
- OSS 文件：`oss://` 加路径和 Endpoint 和 access key 等信息，例如 `oss://bucket/xxx/xxx/xxx.py?host=xxx&access_key_id=xxx&access_key_secret=xxx`；
- HDFS 文件：`hdfs://` 加路径；

代码示例

以下代码仅用于示意，可能需要修改部分代码或者配置环境后才能正常运行！

Python 代码

```
import json

source = RandomTableSourceBatchOp() \
    .setNumRows(100) \
    .setNumCols(10)

colNames = source.getColNames()
source = source.select("*, case when RAND() > 0.5 then 1. else 0. end as
label")
label = "label"

userParams = {
    'featureCols': json.dumps(colNames),
    'labelCol': label,
    'batch_size': 16,
    'num_epochs': 1
```

```
}

trainer = TFTableModelTrainer() \
    .setUserFiles(["https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/tf_dnn_train.py"]) \
    .setMainScriptFile("https://alink-release.oss-cn-beijing.aliyuncs.com/data-
files/tf_dnn_train.py") \
    .setUserParams(json.dumps(userParams)) \
    .setOutputSchemaStr("logits double") \
    .setOutputSignatureDefs(["logits"]) \
    .setSignatureDefKey("predict") \
    .setInferSelectedCols(colNames)
model = trainer.fit(source)
model.transform(source).print()
```

Java 代码

```
import com.alibaba.alink.common.utils.JsonConverter;
import com.alibaba.alink.operator.batch.BatchOperator;
import com.alibaba.alink.operator.batch.source.RandomTableSourceBatchOp;
import com.alibaba.alink.pipeline.tensorflow.TFTableModelPredictor;
import com.alibaba.alink.pipeline.tensorflow.TFTableModelTrainer;
import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

public class TFTableModelTrainerTest {

    @Test
    public void testTFTableModelTrainer() throws Exception {
        BatchOperator.setParallelism(3);

        BatchOperator<?> source = new RandomTableSourceBatchOp()
            .setNumRows(100L)
            .setNumCols(10);

        String[] colNames = source.getColNames();
        source = source.select("*, case when RAND() > 0.5 then 1. else 0. end
as label");
        String label = "label";

        Map <String, Object> userParams = new HashMap <>();
        userParams.put("featureCols", JsonConverter.toJson(colNames));
        userParams.put("labelCol", label);
        userParams.put("batch_size", 16);
        userParams.put("num_epochs", 1);
```

```
        TFTableModelTrainer trainer = new TFTableModelTrainer()
            .setUserFiles(new String[] {"res:///tf_dnn_train.py"})
            .setMainScriptFile("res:///tf_dnn_train.py")
            .setUserParams(JsonConverter.toJson(userParams))
            .setNumWorkers(2)
            .setNumPSs(1)
            .setOutputSchemaStr("logits double")
            .setOutputSignatureDefs(new String[]{"logits"})
            .setSignatureDefKey("predict")
            .setInferSelectedCols(colNames);

        TFTableModelPredictor model = trainer.fit(source);
        model.transform(source).print();
    }
}
```