DataSplitter

2019年3月21日

20:10

* 1. DataSpliter接口：
     1. 声明枚举类型 SplitterType；
        1. 该枚举类型定义了四个变量 GENERIC, GIEVNN, RATIO, VALIDATION
     2. 声明void型无参方法 splitterData() ，可抛出 LibrecException (引用包自定义异常)异常
        1. 实现：拆分数据
     3. 声明void型 （DataConvertor 参数）方法setDataConvertor(DataConvertor dataConvertor)
        1. 实现：设置数据转换器
     4. 引用包 顺序访问稀疏矩阵 （SequentialAccessSparseMatrix ）方法

|  |  |  |
| --- | --- | --- |
| getTrainData();、  获得训练数据 | getTestData();  获得测试数据 | getValidData()；  获得验证数据 |

* 1. 声明布尔型无参方法 nextFold()；

AbstractDataSplitter

2019年3月21日

20:11

* 1. 声明了抽象类 AbstractDataSplitter （抽象数据拆分器）：该抽象类继承于类Configured 并实现了接口DataSplitter
     1. 设置了常量 LOG 用于日志打印
     2. 声明了DataConvertor 型 变量 dataConvertor
     3. 声明了SequentialAccessSparseMatrix （顺序访问稀疏矩阵） 型变量 trainMatrix （训练矩阵）
     4. 声明了SequentialAccessSparseMatrix （顺序访问稀疏矩阵） 型变量 testMatrix （测试矩阵）
     5. 声明了SequentialAccessSparseMatrix （顺序访问稀疏矩阵） 型变量 validationMatrix （验证矩阵）
     6. 声明了SequentialAccessSparseMatrix （顺序访问稀疏矩阵）型链表 assignMatrixList
     7. 定义了SequentialAccessSparseMatrix （顺序访问稀疏矩阵） 型变量preferenceMatrix（偏好矩阵），datetimeMatrix（日期矩阵）为null
     8. 声明了 DataConvertor 型变量dataConvertor的 set 方法 setDataConvertor(DataConvertor dataConvertor):为变量dataConvertor赋值

* 1. 声明了 SequentialAccessSparseMatrix 型无参方法 getTrainData（）；getTestData()；getValidData()；分别返回了在本抽象类AbstractDataSplitter 的 受保护变量 trainMatrix；testMatrix; validationMatrix;

* 1. 声明了 void 型 含参（参数为 SequentialAccessSparseMatrix型参数）方法 setPreferenceMatrix(SequentialAccessSparseMatrix preferenceMatrix)为变量 preferenceMatrix 设值

* 1. 重写了布尔型方法 nextFold()
     1. 判断 若 SequentialAccessSparseMatrix 型Linkedlist 变量 assignMatrixList为空值
        1. 则为 assignMatrixList 初始化一个新的Linkedlist;返回true
     2. 若 assignMatrix 长度为<=0
        1. 返回false
     3. 若 assignMatrix 长度大于0
        1. 定义了一个SequentialAccessSparseMatrix 型变量assign，调用assignMatrix的poll方法为其赋值
           1. 分别为变量testMatrix和trainMatrix赋值为 preferenceMatrix.clone()

设置MatrixEntry 型变量 matrixEntry对变量preferenceMatrix内容进行遍历

若 assignMatrix的poll方法获取当前变量的 matrixEntry.row(), matrixEntry.column() 为1

则调用 trainMatrix 变量的setAtColumnPosition方法，并传入值

matrixEntry.row(), matrixEntry.columnPosition(), 0.0D

否则

调用 testMatrix 变量的setAtColumnPosition方法，并传入值

matrixEntry.row(), matrixEntry.columnPosition(), 0.0D

* + - * 1. 并分别调用trainMatrix 和 testMatrix 的 reshape()方法
        2. 返回true

GivenNDataSplitter

2019年3月18日

20:06

* 1. 定义了类 GivenNDataSplitter继承自 AbstractDataSplitter 类
     1. 定义了 SequentialAccessSparseMatrix 型变量 preferenceMatrix(偏好矩阵)
     2. 定义了 SequentialAccessSparseMatrix 型变量 datetimeMatrix(日期矩阵)
     3. 声明了无参构造GivenNDataSplitter()
     4. 声明了有参构造GivenNDataSplitter( DataConvertor dataConvertor, Configuration conf)
        1. 变量分别为为 dataConvertor； conf 并赋值
     5. 定义了void型方法splitData() 可抛出LibrecExceptio 异常（//该方法对数据进行拆分）
        1. 若preferenceMatrix 变量为空
           1. 则 调用 dataConvertor 的 getPreferenceMatrix(conf) 为其赋值（//conf为Configuration 型变量）
           2. 若dataConvertor 变量不是ArffDataConvertor 类的实例化 且StringUtils 与conf.get("data.column.format"), "UIRT") 方法相等

则调用dataConvertor的getDatetimeMatrix()方法为datetimeMatrix赋值

* + - 1. 并调用conf变量的get(String)方法传值("data.splitter.givenn")为String 字符串 Splitter赋值，判断Splitter值
         1. 根据Splitter值是否为“user” “item”"userdate" " itemdate" (//若不是这四个值则break返回)
         2. 则分别尝试调用相应的
         3. getGivenNByUser()
         4. getGivenNByItem()
         5. getGivenNByUserDate()
         6. getGivenNByItemDate()
         7. 方法，并传入Integer调用parseInt()方法，其值为(conf.get("data.splitter.givenn.n")) 参数(//conf.get("data.splitter.givenn.n")
    1. 然后分别实现了
       1. getGivenNByUser(int numGiven);
       2. getGivenNByItem(int numGiven);
       3. getGivenNByUserDate(int numGiven);
       4. getGivenNByItemDate(int numGiven);
       5. 四个方法
    2. 其中getGivenNByUser(int numGiven);
       1. 若nunGiven>0
          1. 对trainMatrix和 testMatrix变量 实例化为（偏好矩阵 preferenceMatrix）
          2. 从nunSize=0开始对preferenceMatrix.rowSize进行循环

设置int变量 numRated调用 preferenceMatrix.row(rowIndex).getNumEntries()获得数字比率为其赋值

若 numRated大于传入参数值 numGiven

定义int 数组 变量 givenPositions ，并产生（numRated numGiven）之间的随机数为其赋值

从0 开始循环 数字比率numRated 变量

若（当前数字比率变量小于数组givenPositions长度） 且（数组givenPositions的从0开始递增的一个变量值（testColumnPosition）位置的数值的大小为当前数字比率）

则为testColumnPosition递增+1，并调用testMatrix变量的setAtColumnPosition（int，int，double ）方法（//传入值为rowIndex, columnPosition, 0.0D）

否则调用 变量trainMatrix的setAtColumnPosition（）方法

* + - 1. 对trainMatrix和 testMatrix变量分别调用SequentialAccessSparseMatrix的resgape()方法
    1. getGivenNByUserDate(int numGiven)；方法
       1. 若nunGiven>0
          1. 对trainMatrix和 testMatrix变量 实例化为（偏好矩阵 preferenceMatrix）
          2. 从nunSize=0开始对preferenceMatrix.rowSize进行循环

创建SequentialSparseVector型变量 itemRatingVector调用preferenceMatrix.row(int)方法为其赋值，传入参数为当前所循环的numSize值

若变量itemRatingVector的getNumEntries()方法值小于1，进行continue上层for循环

设置List<RatingContext> 变量 itemRatingList 实例化为ArrayList<>（） 传入值为变量itemRatingVecto的getNumEntries()值（//itemRatingVector.getNumEntries()）

设置Vector.VectorEntry 型变量 vectorEntry遍历itemRatingVector{循环调用itemRatingList变量的add（）方法，add（）方法中新建实例RatingContext(rowIndex, vectorEntry.position(), (long) vectorEntry.get())}

使用Collections.sort(itemRatingList)（传入itemRatingList链表，整理其值）;

从index=0开始对 itemRatingList.size()进行循环

若index小于numGiven调用testMatrix变量的setAtColumnPosition（int，int，double ）方法

否则调用 trainMatrix变量的setAtColumnPosition（int，int，double ）方法

* + - 1. 对trainMatrix和 testMatrix变量分别调用SequentialAccessSparseMatrix的resgape()方法
    1. getGivenNByItem(int numGiven)；方法
       1. 若nunGiven>0
          1. 对trainMatrix和 testMatrix变量 实例化为（偏好矩阵 preferenceMatrix）
          2. 从columnIndex=0开始对preferenceMatrix.columnSize()进行循环

设置int变量 numRated调用 preferenceMatrix.row(columnIndex)).getNumEntries()获得数字比率为其赋值

若 numRated大于传入参数值 numGiven

定义int 数组 变量 givenPositions ，并产生（numRated numGiven）之间的随机数为其赋值

从0 开始循环 数字比率numRated 变量

若（当前数字比率变量小于数组givenPositions长度） 且（数组givenPositions的从0开始递增的一个变量值（testRowPosition）位置的数值的大小为当前数字比率）

则为testRowPosition递增+1，并调用testMatrix变量的setAtRowPosition（int，int，double ）方法（//传入值为rowIndex, rowPosition, 0.0D）

否则调用 变量trainMatrix的setAtRowPosition（）方法

否则从0循环变量numRated{调用 testMatrix对象的setAtRowPosition(rowPosition, columnIndex, 0.0D)}

* + - 1. 对trainMatrix和 testMatrix变量分别调用SequentialAccessSparseMatrix的resgape()方法
    1. getGivenNByItemDate(int numGiven)；方法

同getGivenNByUserDate方法，拆分值为变行为列

GivenTestSetDataSplitter

2019年3月22日

16:22

* 1. 定义类GivenTestSetDataSplitter（//指定测试数据拆分器）继承自AbstractDataSplitter
     1. 声明了SequentialAccessSparseMatrix型私有变量preferenceMatrix（//偏好矩阵）
     2. 声明了无参构造GivenTestSetDataSplitter()
     3. 声明了有参构造GivenTestSetDataSplitter(DataConvertor , Configuration )
        1. 为变量 convertor和变量 conf（//配置信息）赋值
     4. 定义了 void 型方法splitData() 抛出 LiberException 异常
        1. 声明了DataConvertor 型变量 testConvertor（）（//测试变换器赋值为null）
        2. 定义 字符串变量 dataFormat 调用 conf 的get（）方法（//传入参数为"data.model.format"） 为其赋值
        3. 定义字符串数组 inputDataPath 调用 conf 的get(String).trim().split(":")（//去掉空格并以“:”分割字符数据，留下多个分割完成后的小字符串）方法为其赋值；
        4. 循环数组 inputDataPath 为每个位置上的字符调用 conf.get(Configured.CONF\_DFS\_DATA\_DIR) + "/" + inputDataPath[i];为其赋值 （//Configured包静态常量CONF\_DFS\_DATA\_DIR值为 "dfs.data.dir"//指向数据目录）
        5. 判断变量dataFormat 小写值
           1. 若为“text”：创建 String 变量dataColumnFormat ，调用conf.get(Configured.CONF\_DATA\_COLUMN\_FORMAT, "UIR")为其赋值，初始化testConverto 为文本数据变换器 TextDataConvertor，传入值为(dataColumnFormat,inputDataPath,"[\t;, ]")
           2. 若为“arff”：则将初始化 testConverto 为数据变换器 ArffDataConvertor，传入值为inputDataPath变量
           3. 否则打印提示信息
        6. 尝试调用testConvertor变量的processData()方法（返回稀疏矩阵（SparseMatrix）型对象 （//表示存储速率数据））
        7. 分别为testMatrix 和trainMatrix 变量调用testConvertor（测试变换器）和dataConvertor（数据变换器）的getPreferenceMatrix()方法 传入配置信息变量 conf
        8. 创建MatrixEntry 变量 遍历变量 testMatrix（//循环调用）trainMatrix.set(int，int，double)（列和行数据 ，0.0）方法，
        9. 调用trainMatrix对象reshape()方法

KCVDataSplitter

2019年3月22日

17:51

* 1. 定义了类KCVDataSplitter 继承自AbstractDataSplitter
     1. 声明了SequentialAccessSparseMatrix型变量assignMatrix（标记矩阵）
     2. 声明了 int 变量cvNumber 和cvIndex；
     3. 声明了无参构造KCVDataSplitter()
     4. 定义了含参构造KCVDataSplitter((DataConvertor , Configuration ) //为变量 convertor和变量 conf（//配置信息）赋值
     5. 定义了 void型 含参splitData(int kFold) （//拆分数据放在传入参数文件夹下面）
        1. 若偏好矩阵为空，实例化偏好矩阵preferenceMatrix，（调用 dataConvertor类的getPreferenceMatrix(conf)方法）
           1. 若变量dataConvertor 不是类ArffDataConvertor 的实例化 并且conf.get("data.column.format")数据与"UIRT" 不相等则为变量datetimeMatrix实例化（//getDatetimeMatrix()）
        2. 若参数kFold>0
           1. 实例化标记矩阵为偏好矩阵的克隆（调用偏好矩阵的clone()方法）
           2. 调用偏好矩阵的getNumEntries()方法为变量numRates 赋值
           3. 判断kFold ， numRates 大小赋值 给 变量numFold
           4. 创建List<Map.Entry<Integer, Double>型 变量 rdm 实例化为ArrayList，初始长度为 numRates
           5. 创建变量indvCount 值大小为(numRates + 0.0) / numFold
           6. 从 0 开始循环 numRates，调用rdm.add()方法为链表赋值
           7. 创建int 数组 长度为 numRates
           8. 为 rdm 链表 排序
           9. 将链表值赋值给数组
           10. 创建MatrixEntry型变量matrixEntry 遍历assignMatrix（标记矩阵）（//循环执行标记矩阵的setAtColumnPosition()//传入参数为矩阵当前行，列，以及当前位置的fold 数组值）
        3. 若编辑矩阵为空
           1. 创建 List<Table<Integer, Integer, Integer>> 型链表 tableList （ArrayList）初始长度为 kFold+1 调用HashBasedTable.create() 为链表赋值
           2. 创建MatrixEntry型变量me遍历assignMatrix（标记矩阵）（//循环执行判断：若(me.get() != 0)则执行tableList.get（）方法//传入值为矩阵 行，列，1）
           3. 为变量assignMatrixList实例化为LinkedList<> 型链表
           4. 循环为变量assignMatrixLis调用add()方法添加值，（//传入参数为SequentialAccessSparseMatrix型实例（//其参数为标记矩阵行数，列数，以及当前位置的tableList值））

* 1. 定义了 void型 无参splitData()可抛出 LibrecException 异常
     1. 调用了Configuration类的getint（String，int）（//传入值为"data.splitter.cv.number", 5 ）为变量cvNumber赋值
     2. 若标记矩阵（assignMatrix）为空，调用本类 方法splitData(this.cvNumber)进行数据拆分
  2. 定义了List<SequentialAccessSparseMatrix> 型 无参方法（获得标记矩阵链表）getAssignMatrixList()（//返回变量assignMatrixList）

LOOCVDataSplitter

2019年3月23日

23:22

* 1. 定义了类LOOCVDataSplitter 继承自类AbstractDataSplitter，
     1. 声明了 SequentialAccessSparseMatrix 型变量 preferenceMatrix（偏好矩阵），datetimeMatrix（日期矩阵）
     2. 声明了KCVDataSplitter 型变量 kcv
     3. 定义了无参构造 LOOCVDataSplitter()
     4. 定义了含参构造 LOOCVDataSplitter(DataConvertor ，Configuration )分别为属性dataConvertor， conf 赋值
     5. 定义了枚举类 LOOCVType ，包含两个变量 LOOByUser, LOOByItem
     6. 定义了void型 无参方法splitData()，可抛出LibrecException 异常
        1. 在该方法中判断变量 preferenceMatrix 若为空，
           1. 则调用dataConvertor对象的getPreferenceMatrix(Configuration)方法,传入值为conf。

并判断dataConvertor属性若为类 ArffDataConvertor的实例且属性conf的get(String）方法所返回的值与“UIRT”的两个String 变量是否相等，符合条件则调用dataConvertor对象的getDatetimeMatrix()方法为datetimeMatrix变量赋值

调用 对象conf 的get(String)（//传入值为”data.splitter.loocv“（//传入入数据））方法，并根据其值分别调用 （//若非这四个值则打印出提示信息）

getLOOByUser(); getLOOByItems(); getLOOByUserDate(); getLooByItemsDate();

四个void型无参方法，

* 1. 定义了getLOOByUser();
     1. 为 trainMatrix 调用 SequentialAccessSparseMatrix类的SequentialAccessSparseMatrix(SequentialAccessSparseMatrix) 方法传入值为preferenceMatrix
     2. 定义了 Table 型变量 dataTable ,调用类 HashBasedTable.create()方法实例化，其参数类型为<Integer,Integer,Double>
     3. preferenceMatrix对象调用rowSize()方法，从0开始遍历其值

若preferenceMatrix对象的row(int)(//getNumEntries)方法为空，continue 上层循环；

设置随机行位置变量randomRowPosition

循环为trainMatrix对象调用setAtColumnPosition（int，int，double）方法；（//传入当前行值，随即行数值，0.0）

为dataTable对象调用put方法（//传入当前行值，偏好矩阵的的随即行数位置的数值，偏好矩阵的列位置的方法（//getAtColumnPosition(rowIndex（当前行数）, randomRowPosition（当前随即行数）)））

对对象trainMatrix（训练矩阵）调用SequentialAccessSparseMatrix的resgape()方法；

为testMatrix调用SequentialAccessSparseMatrix(int , int，TABLE<>(//即变量dataTable))方法（重新实例化）（传入的Int值为偏好矩阵的行，列的数目值）

* 1. 定义了getLOOByUserDate();
     1. 为 trainMatrix 调用 SequentialAccessSparseMatrix类的SequentialAccessSparseMatrix(SequentialAccessSparseMatrix) 方法传入值为preferenceMatrix
     2. 定义了 Table 型变量 dataTable ,调用类 HashBasedTable.create()方法实例化，其参数类型为<Integer,Integer,Double>
     3. preferenceMatrix对象调用rowSize()方法，从0开始遍历其值
        1. 定义顺序稀疏容器 itemRatingVector调用preferenceMatrix对象的row(）传入当前值（rowIndex)

为SequentialSparseVector对象 itemRatingVector调用getNumEntries()，（//为空则继续上层循环）

非空则实例化List<RatingContext>型变量ratingContexts

对容器VectorEntry遍历，

为变量变量ratingContexts添加值，并为其内容排序

对trainMatrix对象调用setAtColumnPosition（int，int，double）方法；dataTable对象调用put方法

对对象trainMatrix（训练矩阵）调用SequentialAccessSparseMatrix的resgape()方法；

为testMatrix调用SequentialAccessSparseMatrix(int , int，TABLE<>(//即变量dataTable))方法（重新实例化）（传入的Int值为偏好矩阵的行，列的数目值）

* 1. 定义了getLOOByItems()
     1. 为 trainMatrix 调用 SequentialAccessSparseMatrix类的SequentialAccessSparseMatrix(SequentialAccessSparseMatrix) 方法传入值为preferenceMatrix
     2. 定义了 Table 型变量 dataTable ,调用类 HashBasedTable.create()方法实例化，其参数类型为<Integer,Integer,Double>

使preferenceMatrix对象调用columnsize()方法，从0开始遍历其值

若preferenceMatrix对象的column(int)方法非空（//为空则进行上层循环）

设置随机行位置变量randomRowPosition

循环为trainMatrix对象调用setAtRowPosition（int，int，double）方法；（//传入当前行值，随即行数值，0.0）

为dataTable对象调用put方法（//传入当前行值，偏好矩阵的的随即行数位置的数值，偏好矩阵的列位置的方法（//getAtRowPosition(rowIndex（当前列数）, randomColumnPosition（当前随即行列数）)））

对对象trainMatrix（训练矩阵）调用SequentialAccessSparseMatrix的resgape()方法；

为testMatrix调用SequentialAccessSparseMatrix(int , int，TABLE<>(//即变量dataTable))方法（重新实例化）（传入的Int值为偏好矩阵的行，列的数目值）

* 1. 定义了getLOOByItemsDate()
     1. 为 trainMatrix 调用 SequentialAccessSparseMatrix类的SequentialAccessSparseMatrix(SequentialAccessSparseMatrix) 方法传入值为preferenceMatrix
     2. 定义了 Table 型变量 dataTable ,调用类 HashBasedTable.create()方法实例化，其参数类型为<Integer,Integer,Double>
     3. preferenceMatrix对象调用columnSize()方法，从0开始遍历其值
        1. 定义顺序稀疏容器userRatingVector调用preferenceMatrix对象的column（）传入当前值（columnIndex)

为SequentialSparseVector对象 userRatingVector调用getNumEntries()，（//为空则继续上层循环）

非空则实例化List<RatingContext>型变量ratingContexts

对容器VectorEntry遍历，

为变量变量ratingContexts添加值（实例化RatingContext型类（传入容器位置，列数，日期矩阵行位置方法（//参数vectorEntry.position(), columnIndex））），并为其内容排序

对trainMatrix对象调用setAtRowPosition（int，int，double）方法；dataTable对象调用put方法（//传入偏好矩阵列位置当前位置的值，列数目，行位置值）

对对象trainMatrix（训练矩阵）调用SequentialAccessSparseMatrix的resgape()方法；

为testMatrix调用SequentialAccessSparseMatrix(int , int，TABLE<>(//即变量dataTable))方法（重新实例化）（传入的Int值为偏好矩阵的行，列的数目值）

RatioDataSplitter

2019年3月23日

23:22

* 1. 定义了类RationDataSplitter （//比率数据拆分）继承自类AbstractDataSplitter，
     1. 定义了 SequentialAccessSparseMatrix 类私有变量datetimeMatrix
     2. 定义了无参构造RatioDataSplitter()
     3. 定义了含参构造RatioDataSplitter(DataConvertor ，Configuration )分别为属性dataConvertor， conf 赋值
     4. 定义了void型 无参方法splitData()，可抛出LibrecException 异常
        1. 在该方法中判断变量 preferenceMatrix 若为空
           1. ，则调用dataConvertor对象的getPreferenceMatrix(Configuration)方法,传入值为conf。

并判断dataConvertor属性若为类 ArffDataConvertor的实例且属性conf的get(String）方法返回的值与“UIRT”的两个String 变量相等则调用dataConvertor对象的getDatetimeMatrix()方法为datetimeMatrix变量赋值

* 1. 调用 对象conf 的get(String)方法，并根据其值分别调用不同的 double 型 比率参数调用方法；分别为：（//不包含则打印提示）
     1. void型方法 getRatioByRating(double )：
        1. 若比率介于0~1，分别为

testMatrix 和 trainMatrix对象实例化为 SequentialAccessSparseMatrix类传入(preferenceMatrix);

使用MatrixEntry 类对象matrixEntry遍历preferenceMatrix 对象

获得随机数rdm

判断随机数与传入比率调用 testMatrix或trainMatrix的etAtColumnPosition（int，int）方法，传入参数为（matrixEntry的行，列，0.0）

遍历结束后分别调用 testMatrix和trainMatrix对象的reshape()方法

* 1. void型方法 getRatioByRatingDate(double )：
     1. 若double参数介于0~1，则 testMatrix和trainMatrix对象重新实例化SequentialAccessSparseMatrix（SequentialAccessSparseMatrix）

并重新为 List<RatingContext> 型 对象ratingContexts实例化并

添加值，（调用add（）方法），并排序

定义 变量trainSize 为ratingContexts变量的长度与比率的乘积

从0开始循环其长度

为小于ratingContexts长度比率的部分调用testMatrix对象的setAtColumnPosition(int,int,double),否则调用trainMatrix对象的setAtColumnPosition(int,int,double)方法（//传入值为ratingContexts.get(index//当前循环的所在值)的getUser（），与getItem，0.0）

最后调用了 对象testMatrix和trainMatrix的reshape方法

* 1. void型方法 getFixedRatioByUser(double )：
     1. 若double参数介于0~1，则 testMatrix和trainMatrix对象重新实例化SequentialAccessSparseMatrix（SequentialAccessSparseMatrix）

从0开始遍历preferenceMatrix对象的rowSize()方法

声明VectorEntry 型变量vectorEntry 遍历偏好矩阵preferenceMatrix.row(rowIndex)

为小于double参数的（产生的随机数）部分调用testMatrix对象的setAtColumnPosition(int,int,double),否则调用trainMatrix对象的setAtColumnPosition(int,int,double)方法

最后调用了 对象testMatrix和trainMatrix的reshape方法

* 1. void型方法getFixedRatioByUser（double）
     1. 若double参数介于0~1，则 testMatrix和trainMatrix对象重新实例化SequentialAccessSparseMatrix（SequentialAccessSparseMatrix）

从0开始遍历preferenceMatrix对象的rowSize()方法

定义numRated 为 preferenceMatrix变量的行数目（//.row(rowIndex).getNumEntries()）;

对preferenceMatrix对象的row()比率进行Math方法取整操作，筛选大于1的值对其进行

获得随机数，组成位置数组，对小于numRated变量的数组值从0开始循环对列位置进行筛选，分别调用testMatrix（//需使testColumnPosition值++）或， trainMatrix的setAtColumnPosition(rowIndex, testColumnPosition, 0.0D)方法

trainMatrix对象的setAtColumnPosition(int,int,double)操作

最后调用了 对象testMatrix和trainMatrix的reshape方法

* 1. void型方法 getRatioByUserDate(double )：
     1. 若double参数介于0~1，则 testMatrix和trainMatrix对象重新实例化

对preferenceMatrix对象rowSize()遍历

定义numRated 为 preferenceMatrix变量的行数目（从0循环调用）（//.row(rowIndex).getNumEntries()）;

定义SequentialSparseVector 型 itemRatingVector调用偏好矩阵的row方法，其值小于1 则继续上层循环

对List<RatingContext>型对象itemRatingList，调用add（） 添加值并排序，

从0调用temRatingList.size()判断其大小，为其按比率调用testMatrix和trainMatrix对象的setAtColumnPosition(int,int,double)方法

最后调用了 对象testMatrix和trainMatrix的reshape方法

* 1. void型方法 getRatioByItem(double )：
     1. 若double参数介于0~1，则 testMatrix和trainMatrix对象重新实例化

遍历preferenceMatrix对象列数目，内层遍历（VectorEntry 型变量vectorEntry遍历偏好矩阵的列）对象按比率调用testMatrix和trainMatrix对象的setAtColumnPosition(int,int,double)方法

最后调用了 对象testMatrix和trainMatrix的reshape方法

* 1. void型方法 getRatioByItemDate(double )：//拆分值与 getRatioByUserDate(double )为不同值
     1. 若double参数介于0~1，则 testMatrix和trainMatrix对象重新实例化SequentialAccessSparseMatrix（SequentialAccessSparseMatrix）循环偏好矩阵的列值（小于一则继续循环不进行其他操作）

并重新遍历userRatingVector容器为 List<RatingContext> 型 对象ratingContexts添加值，排序完成后

按比率调用testMatrix和trainMatrix对象的setAtRownPosition(int,int,double)方法

最后调用了 对象testMatrix和trainMatrix的reshape方法

* 1. 定义了void型方法 getRatio(double , double )
     1. 筛选比率实例化 三个 SequentialAccessSparseMatrix 对象（训练矩阵，验证矩阵，测试矩阵）；定义MatrixEntry类matrixEntry变量遍历对象preferenceMatrix

随机数判断（比率的值）调用三个equentialAccessSparseMatrix型矩阵的setAtColumnPosition(int,int,double)方法

最后调用了 三个值d reshape（）方法，完成测试、训练和验证效果

2019年3月21日

19:23