哈尔滨工业大学

**<<大数据分析>>**

**实验报告二**

**(2022年度春季学期)**

|  |  |
| --- | --- |
| **姓名：** | **李金宣** |
| **学号：** | **1181910201** |
| **学院：** | **大数据** |
| **教师：** | **张开旗 杨东华** |

实验二 聚类分析和分类分析

## 实验目的

掌握对数据进行聚类分析和分类分析的方法；

理解其在大数据环境下的实现方式。

## 实验环境

操作系统：Windows、Linux虚拟机(vmware ubuntu 20.04)

框架：伪分布式Hadoop环境

编程语言：Java

## 实验过程及结果

公用类FileOperator.java，与HDFS进行文件交互，包含rmdir删除文件，touch创建文件，rename重命名文件，appendContentToFile为HDFS上文件增加内容，getContentFromHDFS从HDFS文件读取内容等一系列方法，具体详见源代码，有详细注释。



1. 聚类分析-KMeans算法

Mapreduce类1：

KMeans.java

run方法：

输入文件路径/project2/data/cluster.txt（聚类数据），输出文件路径/project2/result/k-means\_tmp（存储k个中心点的值）



Mapper类：

启动前调用已写好的FileOperator类中的getContentFromHDFS方法，读取k-means.txt中k个中心值的数据，然后在map中计算cluster.txt中传来的每个点与k个中心点的距离（使用了自定义的util类，类中的方法很简单，用于计算欧氏距离，可参看源码），找到每个点距离最近的那个中心点，作为nearestPoint，将nearestPoint作为k值，中心点的数据作为value传递给reducer



Reducer类：

遍历每个key值（本类型的中心点）所聚集的点，计算其均值即新的k个中心点，将其写入/project2/result/k-means\_tmp



MapReduce类2：

PrintCluster.java：

Run方法：

输入文件路径/project2/data/cluster.txt（聚类数据），输出文件路径project2/result/cluster\_result（聚类结果）



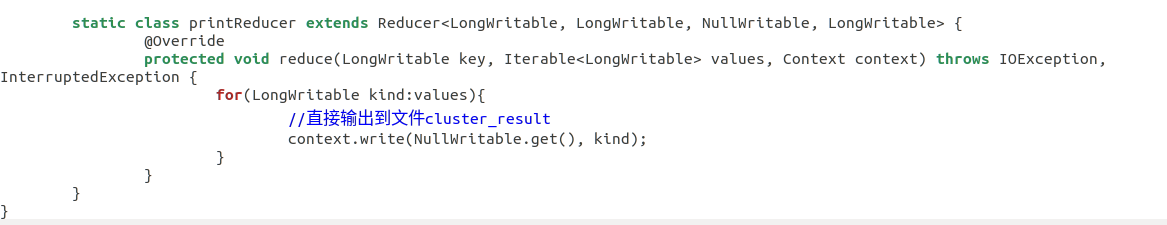
Mapper类：

启动前都k-means.txt中k个中心值的数据，计算每个点与k个中心点距离，选择最近的中心点据类，输出的key-value，key为行号，value为所属的聚类中心点的序号，将k-v传递到reducer



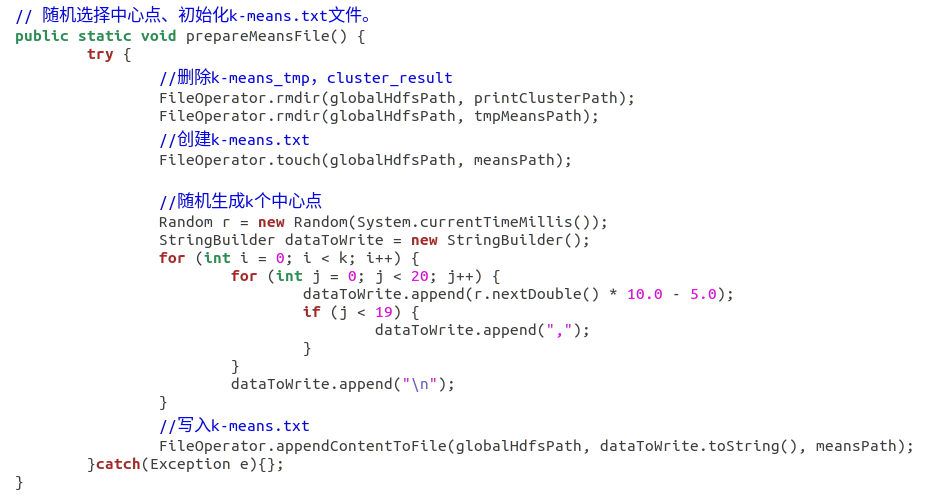
Reducer类：

直接将Mapper传递给Reducer的内容写到输出文件路径project2/result/cluster\_result（聚类结果）中，即每行表都写有聚类中心点序号的形式



KMeansMain类：

prepareMeansFile方法，创建k-means.txt，随机生成k（k=20）个中心点写入k-means.txt

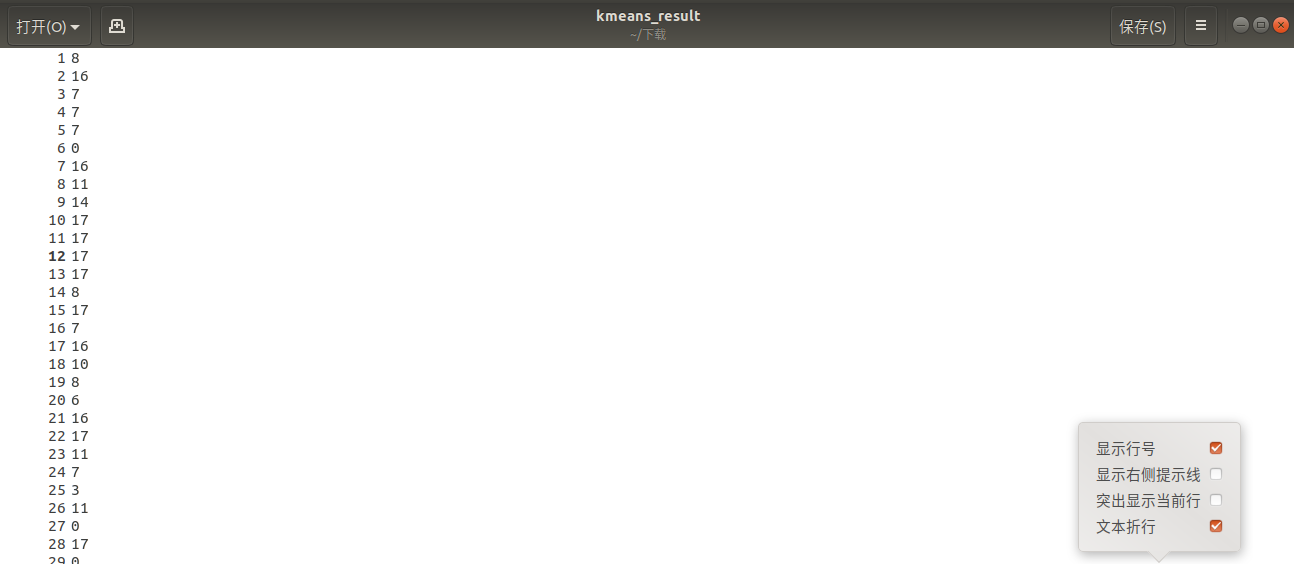


主类中，设置中心点个数k为20，算法迭代轮数20，首先使用prepareMeansFile方法，生成k-means.txt，然后进入迭代循环，调用KMeans.run的Mapreduce过程，生成/project2/result/k-means\_tmp中间文件，文件中记录了新的k个中心点的值，然后删除写有旧中心点的k-means.txt，将写有新中心点的k-means\_tmp重命名为k-means.txt，这样就可以使用写有新中心点的k-means.txt进入下一轮迭代，一直到循环结束，然后调用PrintCluster.run方法进行聚类，得到最终结果project2/result/cluster\_result



从HDFS下载的最终结果文件效果：

行号-对应的聚类中心点序号



1. 聚类分析-KMeans++算法

KMeans算法的时间复杂度是O（n），KMeans聚类在本人电脑上的运行时间大概为4-5个小时左右，而其他聚类算法时间复杂度极高，CLARANS高达O（nlogn）PAM算法，层次聚类等能高达O（n^2），在本机可能要运行几天甚至数周才能得到结果，故采用了仍在KMeans算法基础上进行延伸的算法，即KMeans++，时间复杂度相差不大，运行效率仍在本机性能范围之内

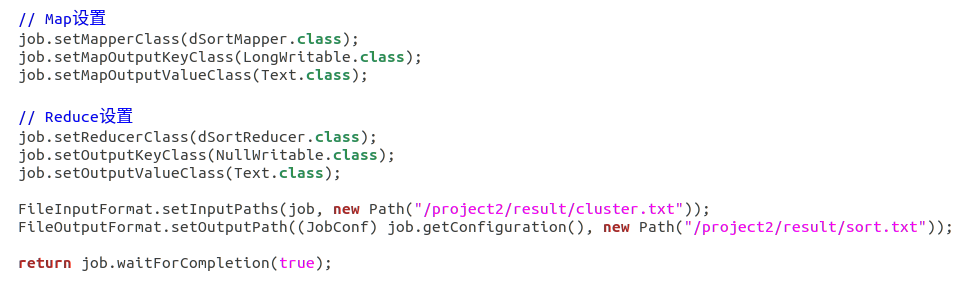
与传统的KMeans算法不同的是，传统的KMeans算法选择初始中心点时是随机的，其初始中心点分布并不均匀，聚类效果可能不好，需要更多轮迭代才能收敛，而KMeans++算法，首先选择一个零点，然后按照各个点到零点的距离，均匀地在点空间中选取初始中心点，使聚类效果更好，收敛更快。

而初始中心点的选择，在MapReduce中表现为先按照各点到初始零点的距离排序，然后按一定距离均匀采样得到中心点，作为初始中心点的KMeans.txt文件

相比KMeans多了几轮排序选择初始中心点的MapReduce过程，之后的迭代等过程与KMeans一致。

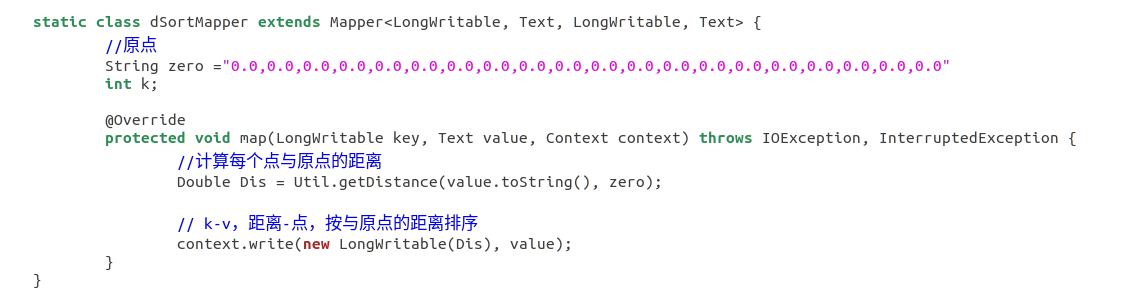
dSort.java

本轮MapReduce的run方法：输入文件路径/project2/result/cluster.txt，输出文件路径/project2/result/sort.txt

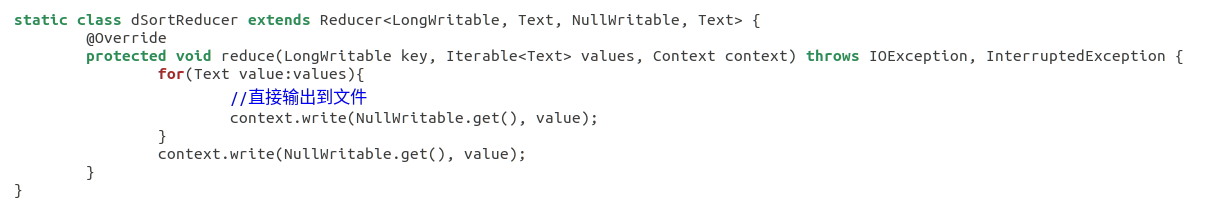


Mapper类：

计算每个点到原点的距离，按距离-点数据的key-value键值对传递给Reducer，注意传递过去的数据是按key即到远点的距离排好序的



Reducer类：

直接将按key排好序的value，即各个点的数据写入

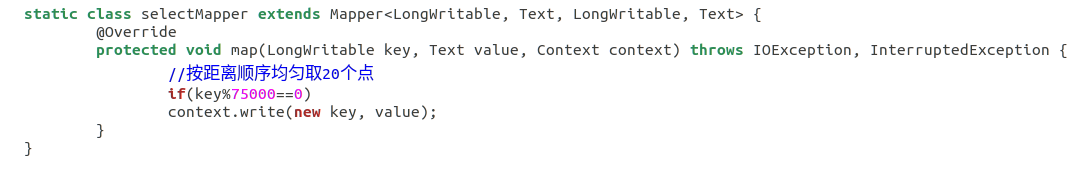
Select.java类：

Run方法：输入文件路径project2/result/sort.txt，输出文件路径/project2/result/k-means.txt



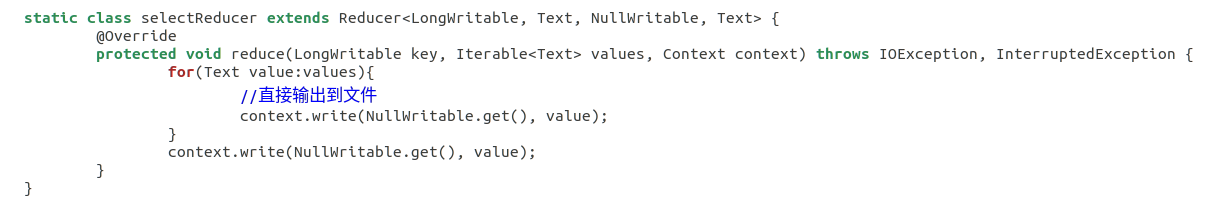
Mapper类：

150w个点均匀取20个作为初始中心点，于是每隔75000取一个点，按行号-点数据的key-value形式传递给Reducer，此时传递依然是有序的

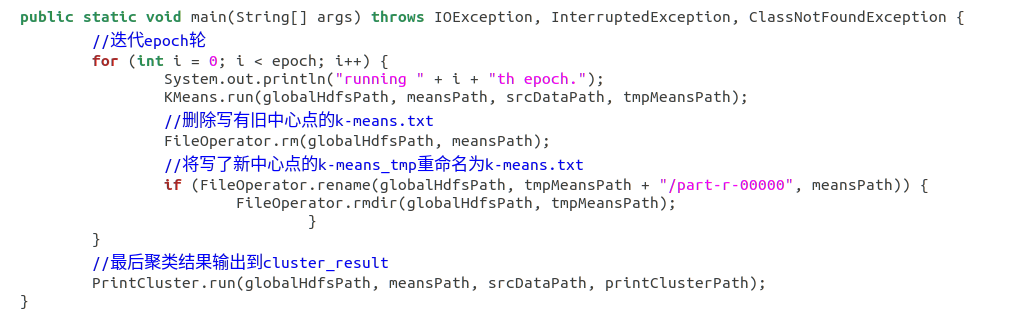


Reducer类：

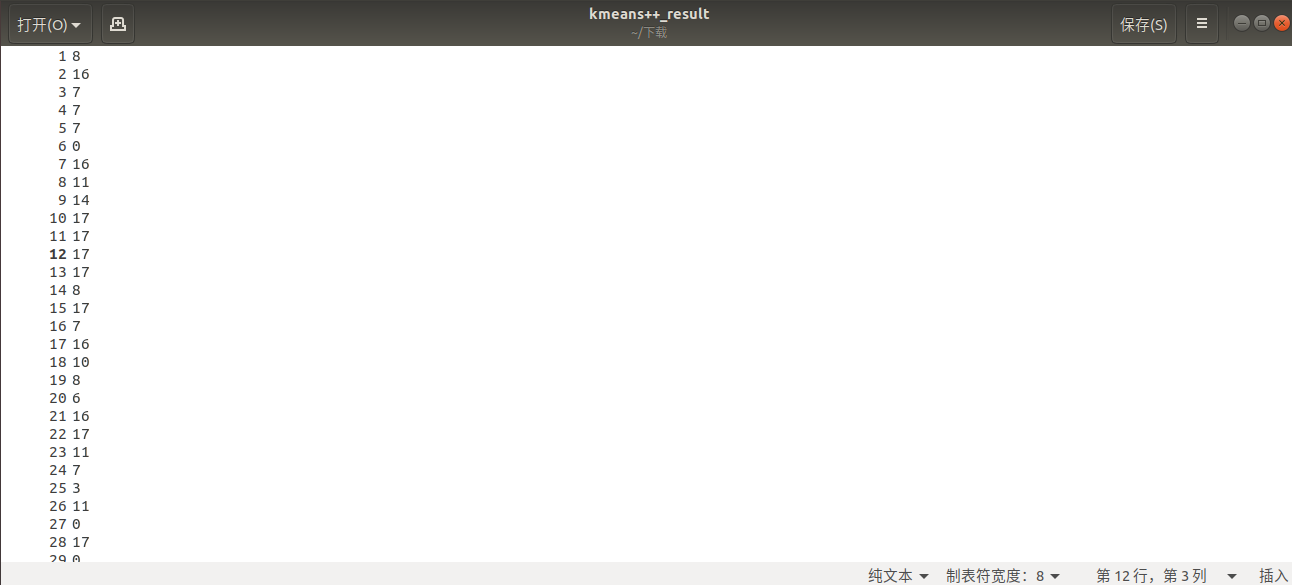
将得到的20个初始中心点直接写入KMeans.txt



得到按距离均匀选取的20个中心点文件kmeans.txt后，其余的迭代和聚类过程与KMeans算法一致，只不过main方法有些变化，无需初始化kmeans.txt文件，直接使用进行迭代即可



从HDFS下载的最终文件结果



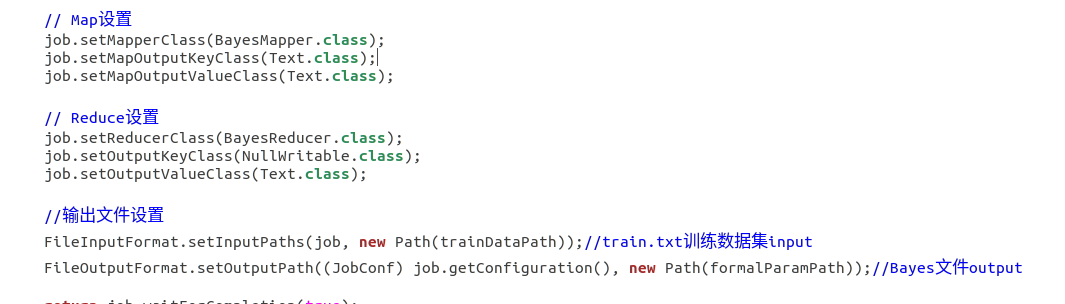
1. 分类分析-朴素贝叶斯算法

MapReduce类1：

NaiveBayes.java

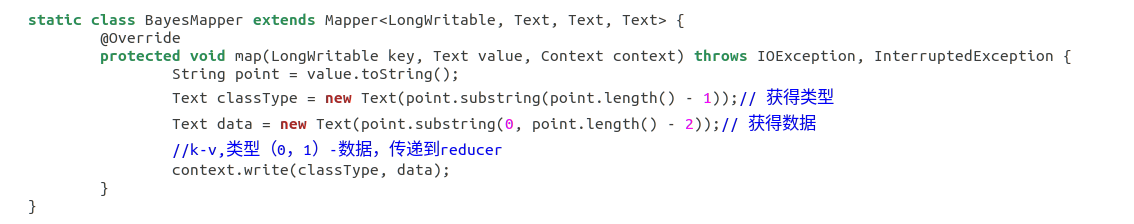
Run方法

输入文件路径/project2/data/train.txt，输出文件路径/project2/result/Bayes



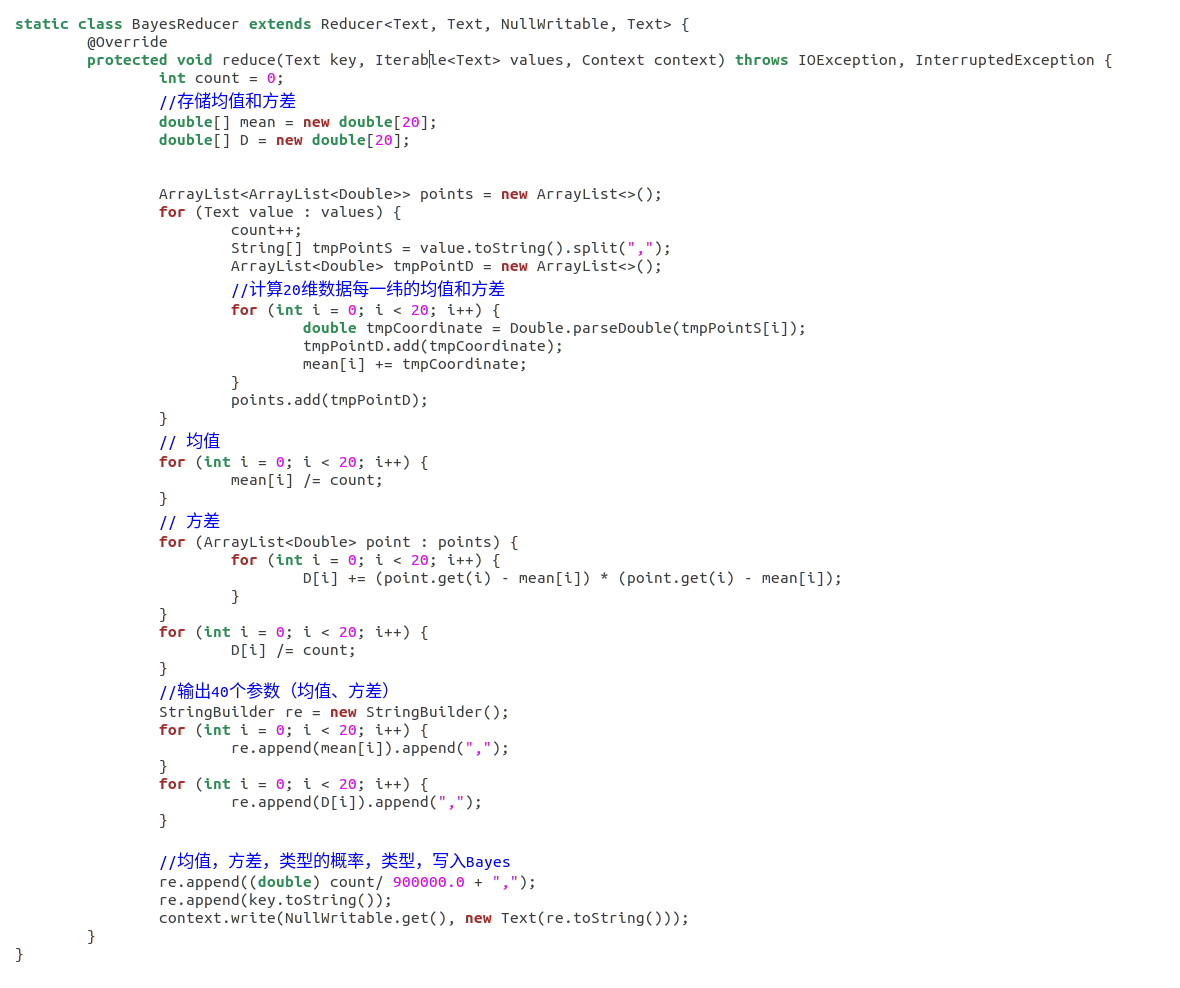
Mapper类：

将train.txt文件的21维数据（20维数据加1维类型）,转化为类型-20维数据的key-value键值对传递给Reducer



Reducer类：

计算出20维数据每一维对应的均值与方差，写入到结果文件/project2/result/Bayes



MapReduce类2：

PrintBayesClassify.java

Run方法

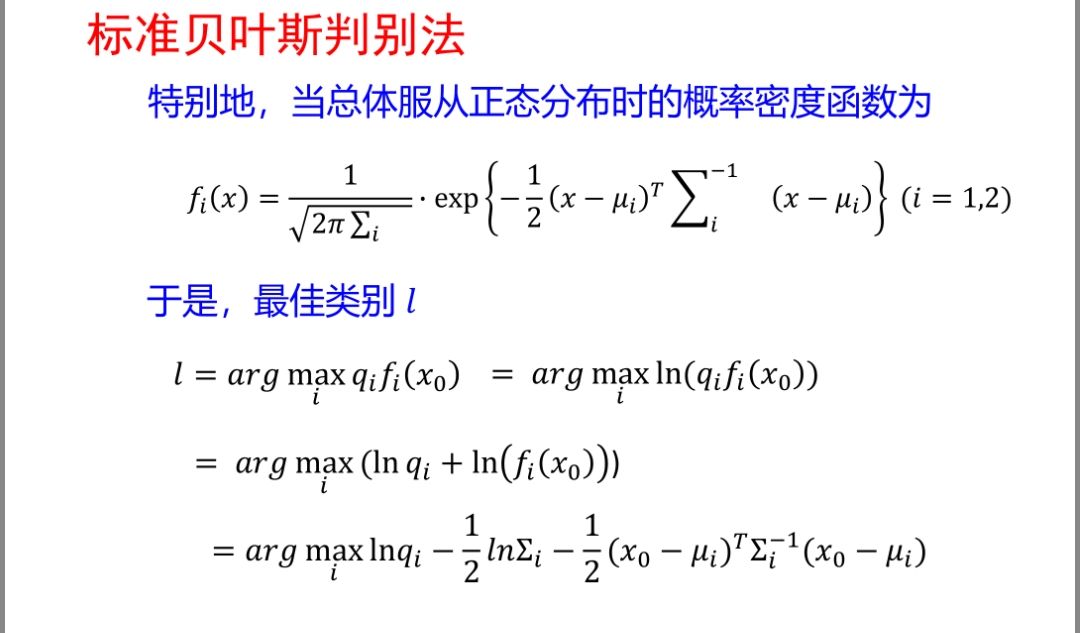
输入文件路径/project2/data/val.txt（测试集），输出文件路径/project2/result/Classify\_result



Mapper类：

首先从上一次MapReduce产生的结果文件加载各个类型的均值与方差，

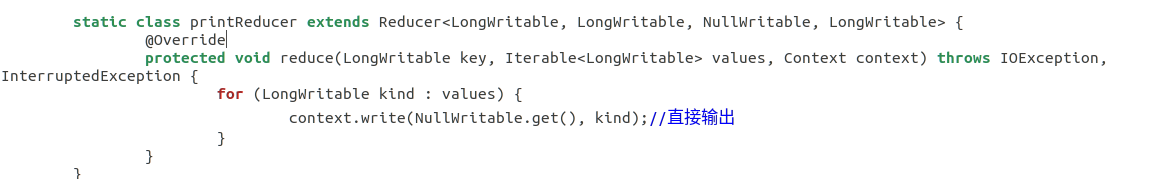
依据连续正态分布的贝叶斯概率分布函数，计算各行数据对应各个类别的后验概率，比较大小确定数据所属类别，按行号-所属类别的key-value键值对传递给reducer





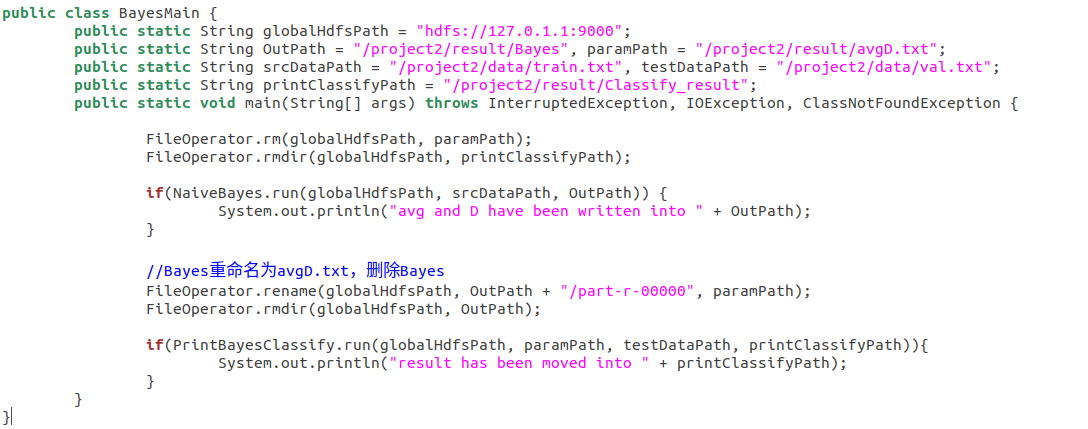
Reducer类：

直接输出到/project2/result/Classify\_result

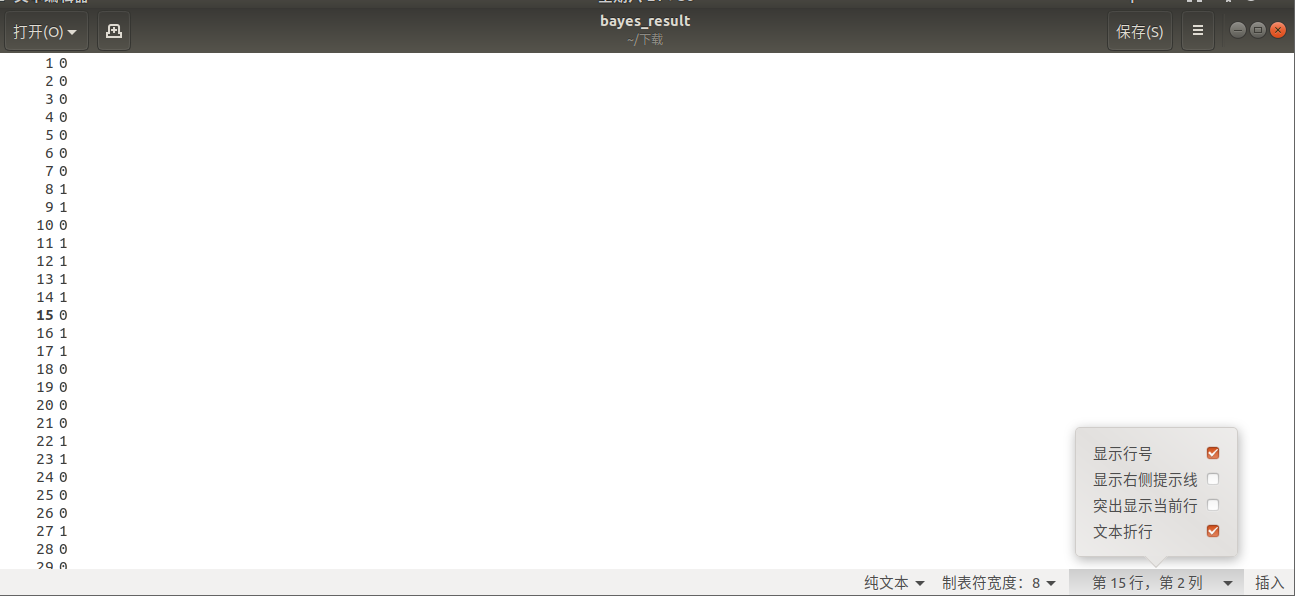


BayesMain主类：

先后调用NaiveBayes和PrintBayesClassify，得到结果文件Classify\_result



从HDFS下载的训练后的结果



写了一个小程序比较测试数据集训练后的分类结果与提供的数据结果，30w个数据，大概有24w+类型匹配正确，准确率在80%以上

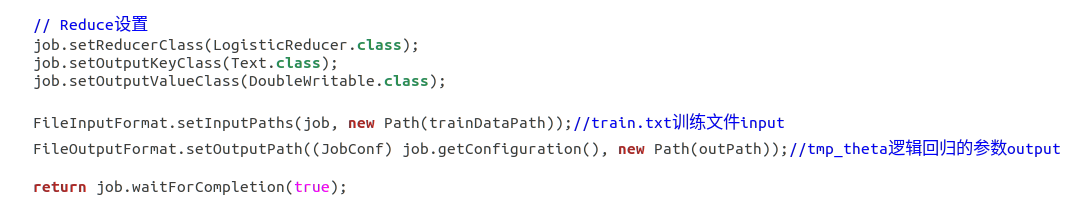
1. 分类分析-逻辑回归算法

MapReduce类1：

LogisticRegeression.java

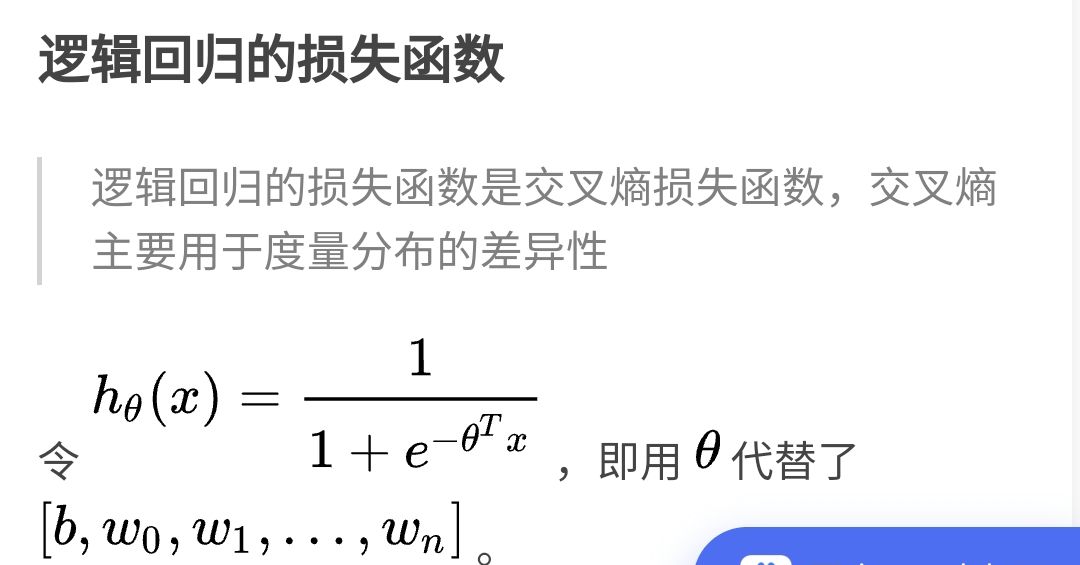
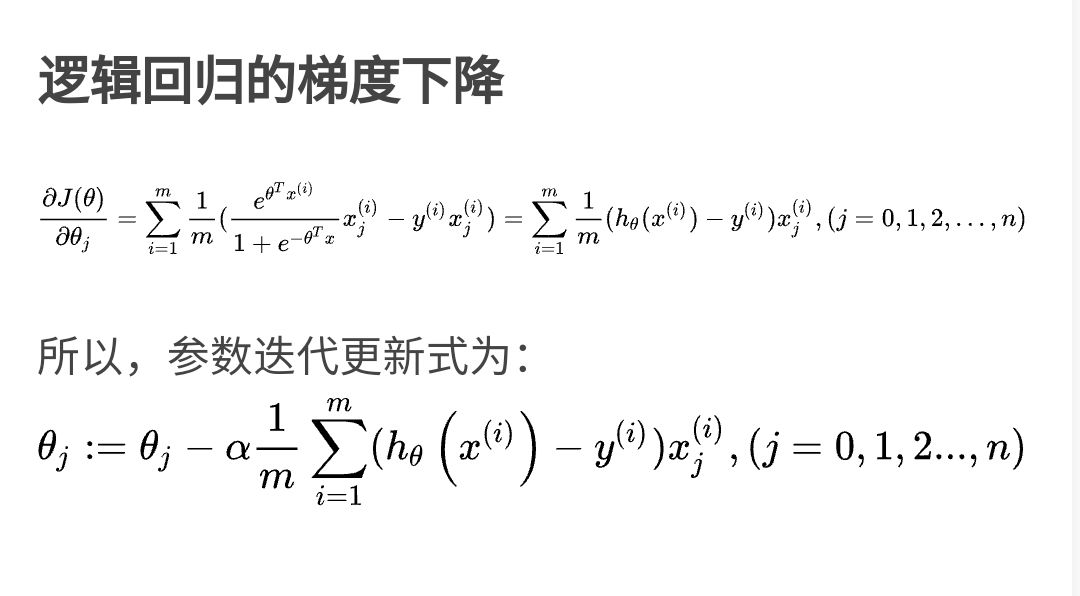
Run方法：

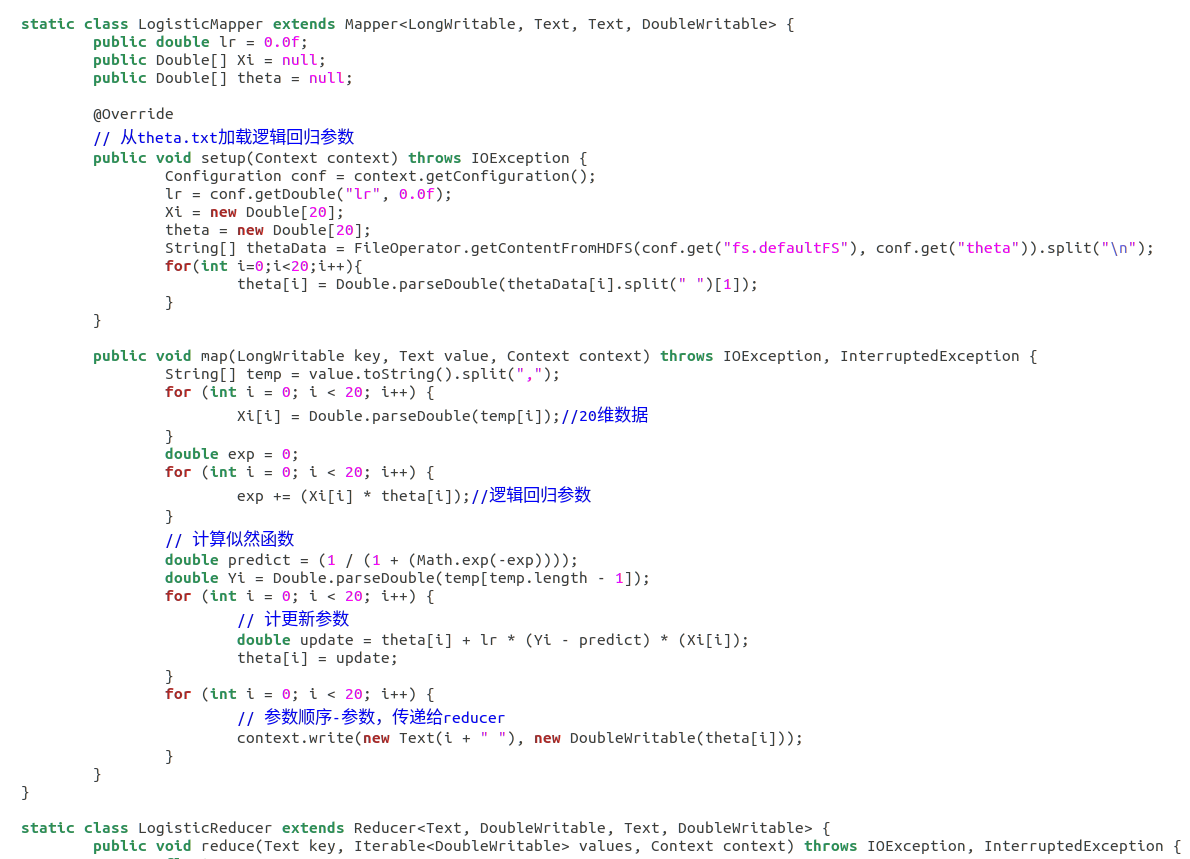
输入文件路径/project2/data/train.txt，输出文件路径/project2/result/tmp\_theta



Mapper方法：

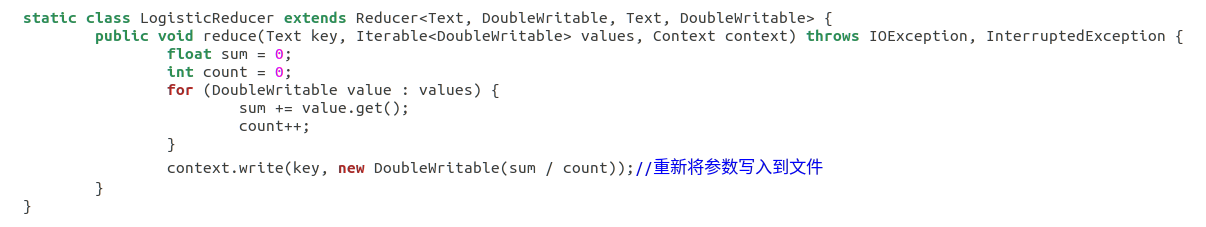
从theta.txt加载20个逻辑回归参数，利用下图的公式计算似然函数并进行梯度下降的参数迭代，每次的参数迭代系数为0.05，将参数顺序-参数的key-value键值对传递给reducer



Reducer类：

将参数按顺序写入文件/project2/result/tmp\_theta



MapReduce类2：

PrintLogClassify.java

Run方法：

输入文件路径/project2/data/val.txt，输出文件路径/project2/result/Logistic\_result



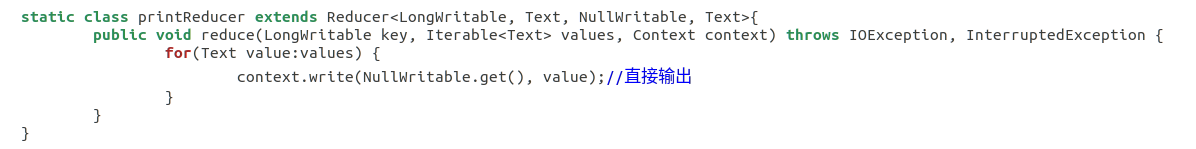
Mapper类：

从theta.txt加载参数，计算似然函数，似然概率，大于0.5为1类型，小于0.5为0类型，按行号-类型的key-value键值对传递给reducer



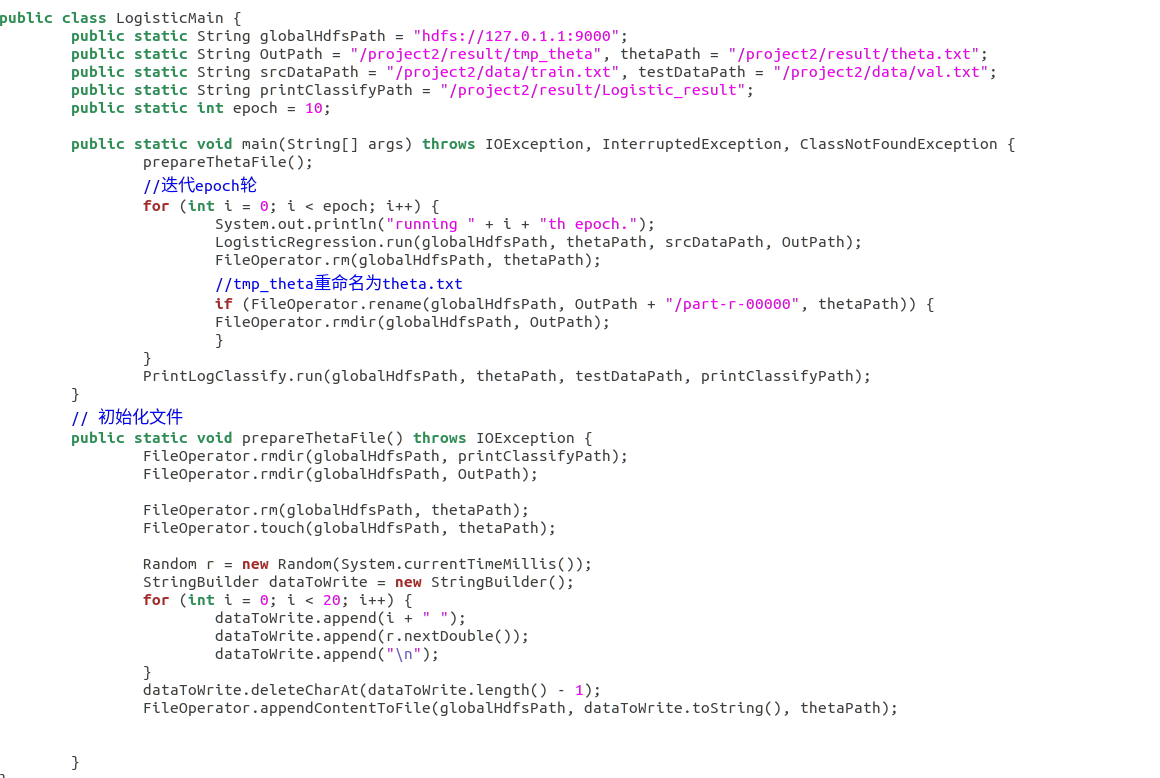
Reducer类：

直接输出即可

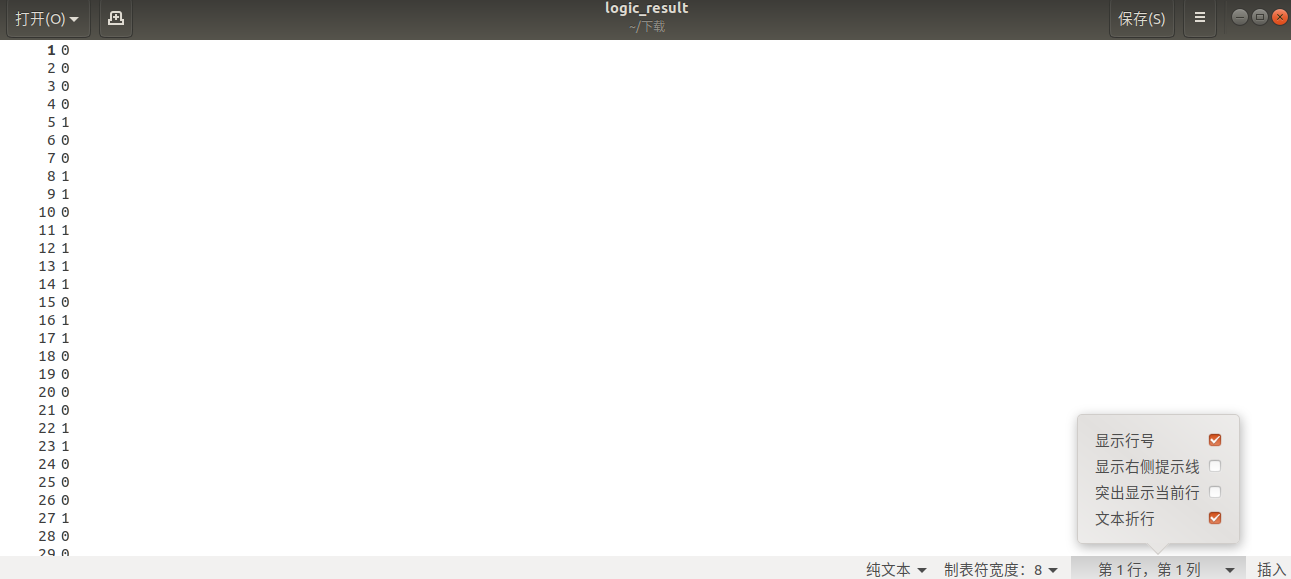


LosisticMain主类:

首先初始化文件，随机生成生成20个参数theta.txt，然后在循环迭代中调用LogisticReression类，生成临时文件tmp\_theta存储迭代后的参数，然后将临时文件重命名为theta.txt计入下一轮迭代，循环10次迭代，最后用PrintLogClassify类进行分类。



从HDFS上下载的文件结果



使用之前的小程序测试，准确率也在80%左右

## 实验心得

各种分类聚类算法的MapReduce实现很困难，参考了许多csdn等网上的代码思路最终才得以实现