哈尔滨工业大学

<<大数据分析>>

实验报告之二

(2021 年度春季学期)

姓名:	卢兑玑
学号:	L170300901
学院:	计算机学院
教师:	

实验二 聚类和分类

一、实验目的

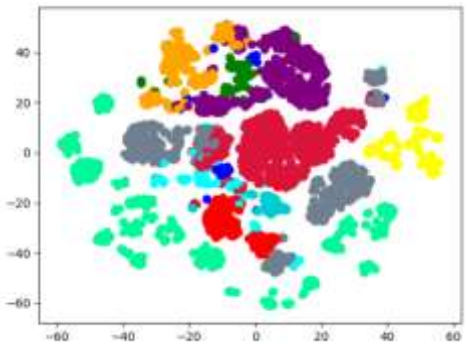
掌握对数据使用聚类分析和分类分析，并理解其在大数据环境下的实现方式。

二、实验环境

Ubuntu 16.04
Hadoop 2.7.1

三、实验过程及结果

3.1 聚类分析
3.1.1 KMeans 聚类分析
主要思想：利用两类 Mapper 和 Reducer，其中第一对 Mapper-Reducer 主要用于中心点的选择，即初始化等工作；第二对 Mapper-Reducer 主要用于中心点的选择，即初始化等工作；第二对 Mapper-Reducer 主要用作迭代过程。
第一类 Mapper
•输入：原始数据
•输出：(1,原始数据中的一条)，共 K 个。
•随机选择 K 个元素作为初始化的聚簇中心点，利用 run 函数实现。
由于此处仅仅需要 K 个元素作为初始化的聚簇中心点，所以只能使用 1 个第一类 Mapper 处理原始数据。
第一类 Reducer
•输入：(1,[c_0,c_1,...,c_{k-1}])，其中 c_i,i∈{0,1,k−1}为原始数据中的一条。
•输出：(i,c_i+\t+'-1')，其中 i 为聚簇编号，\t 为制表符，加法为定义在 String 上的加法，即字符串的连接。
对于第一类 Reducer 而言，其输入的元组 Key 均为 1，所以仅有 1 个第一类 Reducer。
第二类 Mapper
•输入：原始数据
•输出：(clusterCenterID,v;minDis)，其中 Key 为 clusterCenterID，即该元组距离最近的聚类中心的编号；Value 为 v;minDis，其中 v 为该条原始数据，minDis 为该原始数据与最近的聚类中心的欧式距离，二者以英文分号“;”分割。
第二类 Reducer
•输入：(clusterCenterID,[v_0;$minDis_0$,v_1;$minDis_1$,...])
•输出：(clusterCenterID,new_c+\t+disSum)，其中 newc 为属于该聚簇的计算出的新的聚类中心，disSum 为所有属于该聚簇的元素到该中心的距离和，用于判断 Kmeans 迭代收敛。
最终的 Kmeans 实现步骤如下，相关结果如下图所示。



1.使用 1 个第一类 Mapper 随机取 K 个聚类中心，利用 Reducer 将结果存入 HDFS。
2.读入上一轮（或者随机取的 K 个元素）中心点，并利用 Configuration 保存中心点。
3.利用第二类 Mapper 计算每个元素所属的聚簇。
4.利用第二类 Reducer 重新计算聚簇中心。
5.如果收敛，算法结束；否则重新返回第 2 步。

3.1.2 GMM（混合高斯模型）聚类分析

原理主要使用了两类 Mapper 和 Reducer，其中第一类 Mapper-Reducer 负责读取 Kmeans 的结果作为初始化的均值并初始化协方差阵；第二类 Mapper-Reducer 利用 EM 算法更新每个高斯分量的均值、协方差和混合系数。

第一类 Mapper

•输入：KMeans 聚类中心

•输出：(1,value)，其中 Key 为 1，保证所有的值都在同一个 Reducer 中处理，value 为聚类中心。

第一类 Reducer

•输入：(1,[$value_0$,...,$value_{k-1}$])

•输出：(i,π_i,μ_i,σ_i)，其中 Key 为 i，即第 i 个高斯分量；π_i,μ_i,σ_i 分别为第 i 个高斯分量的混合系数、均值和协方差阵，均以字符串进行存储。

第二类 Mapper 负责处理 EM 算法中的 E 步。

•输入：原始数据

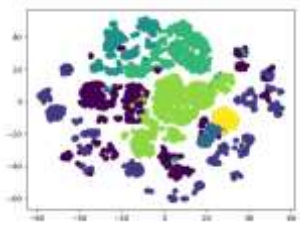
•输出：(i,γ_{ij})，KEY 为高斯分量的标号 i，γij 给出样本 xj 由第 i 个高斯分量生成的后验概率。

第二类 Reducer 负责处理 EM 算法中的 M 步。

•输入：($i,[\gamma_{i0},\gamma_{i1},...,\gamma_{iN}]$)，其中 N 为样本数量。

•输出：($i,\pi'_i,\mu'_i,\sigma'_i$)，其中 Key 为 i，即第 i 个高斯分量；$\pi'_i,\mu'_i,\sigma'_i$ 分别为第 i 个高斯分量新的混合系数、均值和协方差阵。

最终 GMM 实现的步骤如下，实验结果如下图所示，迭代固定轮数为 10 轮（由于计算的复杂度过高）：



1.读入 KMeans 结果，利用第一对 Mapper 和 Reducer 初始化 K 个高斯分量的混合系数、均值和协方差阵。
2.利用第二类 Mapper 实现 EM 算法的 E 步，即计算每个样本由各个高斯分量生成的后验概

率。

3.利用第二类 Reducer 实现 EM 算法的 M 步，即更新每个高斯分量的混合系数、均值和协方差阵。

3.2 分类分析

3.2.1 朴素贝叶斯

原理：由于使用的数据每一维特征都是连续型的数据，所以其处理与离散型的朴素贝叶斯处理有所不同。因此，假设数据的每一维都符合高斯分布，而高斯分布的均值和方差均通过训练数据中的均值和方差来代替。数据第 i 维取值为 xi 的类条件概率为：

$$P(X_i = x_i|Y = y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

第一类 Mapper

•输入：训练数据

•输出：(label_k,v_k)，其中 label 为该样本中标记的类别编号，k 为属性的第 k 维，vk 为该样本第 k 维属性的取值。

第一类 Reducer

•输入：(label_k,[v_{k0},v_{k1},...])

•输出：(label_k,$mean_k$+\t+var_k)，即计算出属于 label 类的第 k 维训练数据的均值和方差，另加法为字符串的连接。

第二类 Mapper

•输入：测试数据

•输出：(compute_label,label)，其中 compute_label 为朴素贝叶斯得到的类别编号，而 label 为数据中原本标注的类别编号。

第二类 Reducer

•输入：(compute_label,[$label_0$,$label_1$,...])

•输出：(compute_label,correct+\t+wrong)，其中 correct 为正确分类样本数目，wrong 为错误分类数目。

最终的 NaiveBayes 在 MapReduce 上实现的步骤为：

1.读取训练数据，利用第一类 Mapper 按照标记的类别和维度进行划分。

2.利用第一类 Reducer 将每一类对应的每一维的均值和方差，并输出到 HDFS 进行保存。

3.读取 HDFS 中的每一类对应的每一维的均值和方差保存在 Configuration 中。

4.读取测试数据，利用第二类 Mapper 计算每一个样本最大后验对应的标签，按照计算出的标签输出至相应的 Reducer。

5.利用第二类 Reducer 统计每一类，正确分类个数和错误分类个数，输出到 HDFS 保存记为最终结果。最后得到的利用朴素贝叶斯分类器得到的分类结果，在训练数据上如表 1，在测试数据上结果如表 2。

类别	正确分类树	错误分类数
0	2411175	1024025
1	1263802	300998

表 1

类别	正确分类树	错误分类数
0	478517	202793
1	251201	59939

表 2

3.2.2 逻辑回归

原理：在给定数据集上最大化对数似然，即

$$\ell(\mathbf{w}, b) \quad \sum_{i=1}^{m} \ln p(y_i|\mathbf{x}_i; \mathbf{w}, b)$$

其中 x_i 为第 i 个样本数据，yi 为第 i 个数据的标签，m 为数据维度。

可以利用梯度下降法来求解该问题。为了方便，将更新的参数记为 β=(w;b)。用于逻辑回归的同样有两类 Mapper 和 Reducer，其中第一类 Mapper 和 Reducer 负责进行梯度下降，而第二类 Mapper 和 Reducer 负责进行结果的统计。

第一类 Mapper

•输入：训练数据

•输出：$\left(i,\ \alpha\left(\frac{\partial \ell(\beta))}{\partial \beta}\right)_i\right)$，其中 Key 为 i，即第 i 维标号，α 为学习率，$\left(\frac{\partial \ell(\beta))}{\partial \beta}\right)_i$ 为对 β 导数的第 i 维的值。

第一类 Reducer

•输入：$\left(i,\ [\alpha\left(\frac{\partial \ell(\beta))}{\partial \beta}\right)_{i0},\ \alpha\left(\frac{\partial \ell(\beta))}{\partial \beta}\right)_{i1},\ \ldots,\ \alpha\left(\frac{\partial \ell(\beta))}{\partial \beta}\right)_{iN}]\right)$，N 为训练样本数量。

•输出：(i,β′i)，Key 仍然为 i，即第 i 维数据，value 为 β′，即更新后参数 β 的第 i 维数据。更新的公式为 $\beta_i' = \beta_i - \frac{1}{N}(\lambda\beta_i + \sum_{j=0}^{N}\alpha\left(\frac{\partial \ell(\beta))}{\partial \beta}\right)_{ij})$，其中 λ 为正则化参数。

第二类 Mapper

•输入：测试数据

•输出：(compute_label,correct_label)，其中 Key 为 compute_label，即利用 sigmoid 函数计算得到的类别标签，correct_label 为数据中正确的标签。

第二类 Reducer

•输入：(compute_label,[$label_0$,$label_1$,...])

•输出：(compute_label,correct+\t+wrong)，其中 correct 为正确分类样本数目，wrong 为错误分类数目，与朴素贝叶斯的第二类 Reducer 相同。

最终实现逻辑回归的步骤如下，其中学习率 $\alpha = 0.1$，正则化参数 $\lambda = 0.01$，测试结果如 表 3 所示，正确率为 77.1770%。

类别	正确分类树	错误分类数
0	474846	162897
1	291097	63610

表 3

1.初始化 β=0。

2.读入训练数据，利用第一类 Mapper 对 β 求导。

3.利用第一类 Reducer 更新 β，如果两次更新 β 的差距超过 1×10−4，返回第 2 步，否则继续下一步。

4.读入测试数据，利用第二类 Mapper 计算相应的标签。

5.利用第二类 Reducer 统计正确分类数和错误分类数目。

四、实验心得

•关于 MR 编程框架中确定 Mapper 的数量，其根据的是输入数据的大小按照 HDFS 的固定分开 128MB 计算分块数量，进而确定 Mapper 数量。因此可以通过修改 HDFS 的默认分块大小或者使用 Mapper 读入时的分块大小来固定 Mapper 的数量。

•关于如何更好的利用伪分布式的多核 CPU，利用 YARN 进行资源管理，然后确定 Mapper 的数量为逻辑核心数量，就可以充分使用本地的计算资源。

•关于 GMM 迭代时出现奇异矩阵，可以在协方差阵加上一个较小的对角阵避免迭代终止。