基于决策树等算法的数据分析与预测

隋唯一
清华大学
2017011430
swy17@mails.tsinghua.edu.cn

摘要
本报告介绍一次有关数据分类预测以及聚类分析的实验，实验基于 python sklearn 库，并提供了实验者自己对于朴素贝叶斯算法以及 K-Means 聚类算法的实现。之后基于 graphviz 可视化决策树，基于 t-sne 和 matplotlib 对聚类结果进行降维与可视化。最后使用 10 折交叉验证对分类预测结果进行评价；使用 Fowlkes-Mallows 指数（以下简称 f-m 指数）和调整后兰德指数（以下简称兰德指数）对于聚类结构进行评价

类别及主题描述
【核心数据结构】：矩阵/字典

【编程语言】：python

【核心算法】：决策树、朴素贝叶斯、岭回归算法、支持向量机、K-Means 聚类算法、DBSCAN 算法

大类
算法, 性能, 实验

关键词
人工智能，可视化，数据分析，决策树，聚类.

.

1. 第一部分：银行精准营销解决方案

1.1 数据集选择
本次实验构建了两个数据集：分别为 age & balance 属性构成的数据集以及 duration, campaign, pdays, previous 四个属性构成的数据集。数据集所包含的属性是这样被选择的：所选择的属性在概率上应当尽可能独立。比如，balance 与 loan 属性相关性较强（由常识可知），所以二者不应当出现在同一数据集中。

1.2 实验环境
本次实验基于集成了 anaconda 的 python，打开随报告提交的项目文件夹即可。如果没有集成 anaconda，请手动导入 sklearn，pandas 等第三方库

1.3 实验结果
1.3.1 决策树可视化
数据集 1，最大深度为 4：

请查看：../output/TreeForAgeAndBalanceD4.pdf

数据集 1，最大深度为 8：

请查看：../output/TreeForAgeAndBalanceD8.pdf

数据集 2，最大深度为 3：

请查看：../output/TreeForDrtCpnPdsPrevD3.pdf

数据集 2，最大深度为 4：

请查看：../output/TreeForDrtCpnPdsPrevD4.pdf

数据集 2，最大深度为 8：

请查看：../output/TreeForDrtCpnPdsPrevD8.pdf

1.3.2 实验评价
决策树算法、朴素贝叶斯算法、支持向量机、岭回归算法的 10 折交叉验证得分分别见../output/scoresOfTrees.txt, ../output/scoresOfBayes.txt, ../output/scoresOfSvc.txt, ../output/scoresOfReg.txt

得分计算方法即为准确率。

整体上看，采用第二个数据集，且最大深度为 4 的决策树模型的表现是最好的，其余的决策树模型或者过拟合，或者欠拟合。而其他的模型表现不佳的原因有多种。对于朴素贝叶斯模型，由于实验给定的数据集并不是均匀分布（前面的数据均为标签为 0，后面的数据均为标签为 1），所以虽然表现尚可，但有侥幸之嫌（即使无论如何都给出标签为 0 的预测结果，也能达到约 80 的准确率）。在改进数据集的划分后，得分会突飞猛进（见下文实验者自行实现的朴素贝叶斯算法）

而其他的两个算法本质相同（岭回归是支持向量机的改进）。事实上，对于这类值较为松散的数据，决策树模型具有先天优势，这两个算法表现不佳也是意料中的。

1.4 实验者自行实现的朴素贝叶斯算法

我所实现的数据分析与预测算法是朴素贝叶斯算法。需要注意的是，我对数据进行了预处理：即将标签为 0 与标签为 1 的数据分别分为十份，每次从两者各取其中九份为训练集，取另一份作为测试集，得到了较高（甚至是极高）的评分。（参见../output/scoresOfMyBayes.txt）。得分的计算方法为计算预测的准确率。

需要注意的是，得分的方差较大，最高可达 1.0（也就是全部命中），最低只有不到 0.2。但如果删去最高得分和最低得分，总体得分还是稳定的分布在 0.7 至 1.0 的范围内。

算法的具体实现请查看代码../codes/mtBayes.py，思路完全遵循朴素贝叶斯算法，这里不再赘述。

2. 第二部分：青蛙叫声聚类分析

2.1 数据集选择

本次实验选择了两种数据集。一个是 1、5、9、13、17、21 六个维度组成的数据集，另一个是 3、7、11、15、19 五个维度组成的数据集。

根据 MFCC 算法，帧的选取具有重叠性。所以在选取数据集时，应当避免选取连续的维度。

2.2 实验环境

与第一部分相同，为集成了 anaconda 的 python 环境，编程 IDE 为 pycharm。

2.3 实验结果

2.3.1 聚类可视化结果（采用 t-sne 降维）

数据集 1，K-Means 算法：

请查看：../output/kMeansSet_1.png

数据集 1，DBSCAN 算法：

请查看：..\output\dbscanSet_1.png

数据集 1，实验者自行实现的 K-Means 算法：

请查看：..\output\myKMeansSet_1.png

数据集 2，K-Means 算法：

请查看：..\output\kMeansSet_2.png

数据集 2，DBSCAN 算法：

请查看：..\output\dbscanSet_2.png

数据集 2，实验者自行实现的 K-Means 算法：

请查看：..\output\myKMeansSet_2.png

2.3.2 聚类原始结果（xlsx 文件）

数据集 1，K-Means 算法：

请查看：..\output\kMeansSet_1.xlsx

数据集 1，DBSCAN 算法：

请查看：..\output\dbscanSet_1.xlsx

数据集 1，实验者自行实现的 K-Means 算法：

请查看：..\output\myKMeansSet1.xlsx

数据集 2，K-Means 算法：

请查看：..\output\kMeansSet_2.xlsx

数据集 2，DBSCAN 算法：

请查看：..\output\dbscanSet_2.xlsx

数据集 2，实验者自行实现的 K-Means 算法：

请查看：..\output\myKMeansSet2.xlsx

2.4 实验过程

在调整 DBSCAN 模型的参数（eps, min_samples）时遇到了一点麻烦。最后配合 m-f 分数，先找到一个较为理想的参数对（有极大运气成分），然后在这个参数对附近进行参数微调，寻找局部极大值。

由于数据集 1 与数据集 2 的稀疏程度不同,故而最后得到的最佳参数不同。具体调参记录参见..\records\dbscan调参记录.md

2.5 实验评价

本次实验使用 Fowlkes-Mallows 指数和调整后兰德指数进行评分（见..\output\scoreOfClustering.txt）。总的来说，数据集 1 的结果优于数据集 2，因为数据集 1 比数据集 2 多一个维度。K-Means 与 DBSCAN 算法的表现不相上下，因为给定的数据集中聚类簇的形状是凸的，DBSCAN 的优势不明显。实验者自己实现的 K-Means 算法（评分见..\output\scoreOfMyKMeans.txt）相对于 sklearn 库中的 K-Means 算法，性能不够稳定：最佳评分与库中的实现差不多，但是最低评分远劣于库中实现的 K-Means 算法。其中原因很可能是库中的 K-Means 算法对初始质心的选择进行了优化，而实验者自行实现的 K-Means 算法初始质心是完全随机选取的。

有趣的是，在对于同一个算法，同一个数据集得到的结果进行评分时，调整后兰德指数显著低于 m-f 指数。这是二者的计算方式不同导致的。调整后兰德指数考虑的是整体的相似度，而 m-f 指数仅考虑准确率和召回率。

3. 实验心得

本次实验让我走入了机器学习的大门，虽然仅仅是简单的调库，自己实现的算法也是相对简单的朴素贝叶斯和 K-Means 算法，但我还是很兴奋。毕竟自己训练的模型能够在大多数情况下做出正确的预测，也有了一定的“智能”。

我似乎也找到了将来深造的方向，因为我发现我对于这方面非常感兴趣。

感谢老师和助教带我领略机器学习之美。

4. 鸣谢

感谢李春平教授的讲解以及何涛助教提供的数据资料。

感谢 sklearn 开发者们。

感谢 sklearn 官方文档的汉化者们

5. 参考

[1] Sklearn 官方文档中文版 https://sklearn.apachecn.org/

[2] 周志华《机器学习》

[3] 博客https://blog.csdn.net/swy_swy_swy/article/list

[4] 博客https://swy20190.github.io