《人工智能导论》 Project-2 实验报告

黄翔

2017013570

清华大学 软件学院

wish142857@163.com

1. 实验概述

1.1 实验选题

- 根据提供的银行营销数据，构建合适的分类模型， 预测客户是否会购买该银行的产品；
- 根据青蛙的叫声所提取的 MFCC 特征，对不同科属的青蛙进行聚类分析。

1.2 实验环境

- 操作系统：Windows 10
- 编程语言：Python 3.7
- 集成开发环境：PyCharm Professional

1.3 实验内容

1.3.1 数据处理

- ✓ 利用 NumPy、pandas 程序库进行数据的读取与处理
- ✓ 依据客观经验手工选择了两种的特征组合
- ✓ 对离散性特征使用了 one-hot 编码，以消去其顺序差别

1.3.2 模型构建

- ✓ 整合了 sklearn 机器学习库中朴素贝叶斯、决策树、支持向量机、K-Means、谱聚类的相关接口
- ✓ 实现了基于高斯分布的朴素贝叶斯算法、基于欧几里得距离与余弦相似度的 K-Mean 算法（K-Means++）

1.3.3 模型评价

- ✓ 对于分类任务，使用了 K-折交叉验证对分类器进行评估；测量指标包括准确度（Accuracy）、精度（Precision）、召回率（Recall）、F-值度量（F-measure）
- ✓ 对于聚类任务，测量指标包括熵值（Entropy）和纯度（Purity）；同时，对聚类结果使用 TSNE 降维后，利用 pyplot 绘图库进行了可视化绘图

2. 程序架构

2.1 文件结构

表 1. 分类项目文件结构

classification（分类项目）	
algorithm.py	内含分类器生成、预测函数与高斯分布朴素贝叶斯算法实现
evaluator.py	内含分类器评估函数
loader.py	内含数据加载与处理函数
main.py	内含主函数，可在此修改调用内容
timer.py	内含计时类，用于调用计时

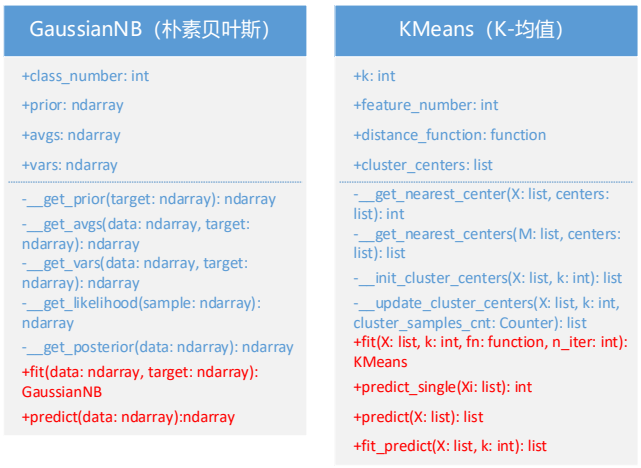
表 2. 聚合项目文件结构

clustering（聚合项目）	
algorithm.py	内含聚类器生成、预测函数与 K-Means 算法实现
evaluator.py	内含聚类器评估函数
loader.py	内含数据加载与处理函数
main.py	内含主函数，可在此修改调用内容
painter.py	内含聚类结果绘制函数
timer.py	内含计时类，用于调用计时

2.2 类结构

以下为本次实验所实现的两个算法（高斯朴素贝叶斯与 K-均值）的类结构：

图 1. 自实现算法类结构



注意：更详细的变量与函数描述请参见源代码，均有详细注释。

3. 算法实现

3.1 数据处理

3.1.1 输入特征选择

【分类任务】输入特征根据经验通过手工选择而出，共计两组。

第一组包含字段：['age', 'job', 'education', 'default', 'balance', 'housing', 'loan']，侧重于其账户存款、贷款、违约情况。

第二组包含字段：['job', 'contact', 'duration', 'pdays', 'previous', 'poutcome']，侧重于其沟通与联络的方式、频次、时长、距离时间与交流结果。

【聚类任务】输入特征根据经验通过手工选择而出，共计两组。由于较后维度数据偏差较大，利于聚类分析，输入特征优先选择较后维度。

第一组包含 MFCC 22 维特征向量的后 6 维度。

第二组包含 MFCC 22 维特征向量的后 12 维度。

3.1.2 数据预处理

【分类任务】在分类任务中，银行营销数据集中有较多的类别特征（即，数据为离散标称型，而非数值连续型）。在进行模型构建前，需要对这些数据进行类别特征编码，将标称型特征（categorical features）转换为整数编码（integer codes）。

为防止将类别简单地按序编码（0 到 n_categories - 1）而引入的类别潜在有序性，数据预处理时，使用了独热码（dummy encoding）进行编码，将每一个具有 n_categories 个可能取值的 categorical 特征变换为长度为 n_categories 的二进制特征向量，里面只有一个地方是 1，其余位置都是 0。从而保证了类别特征数据的无序性。

3.2 模型构建

【分类任务】在分类任务中，程序实践了朴素贝叶斯（Naive Bayes）、决策树（Decision tree）、支持向量机（SVM）等分类算法。其中，实现了基于高斯分布的朴素贝叶斯算法（GaussianNB）。

【聚类任务】在聚类任务中，程序实践了 K-均值（K-Means）、谱聚类（Spectral clustering）等聚类算法。其中，实现了基于欧几里得距离与余弦相似度的 K-Means 算法（KMeans）。该算法采用 K-Means++ 方法，对初始簇中心进行选择。

【聚类任务 距离度量方式】在此次度量方式选择过程中，尝试了欧几里得距离与余弦相似度。由于 MFCC 特征数据的度量标准一致，且数据较为标准化、中心化，因而在模型评估时，采用了欧几里得距离度量方法，较为简单与直观。

【聚类任务 超参数选择】在聚类超参数选择上，例如 K 值（簇数量）的选择上，可采用肘部法则、轮廓系数等方法选择出最为合适的 K 值。但在此次实验中，我们尝试对青蛙所属的科进行聚类结果比较。由于数据集中青蛙共有 4 个科，因而为方便结果比较与评估，选择 K=4 进行模型评估。

3.3 模型评价

【分类任务】在分类任务中，使用 K-折交叉验证（K=10）的方法对分类器进行评估；测量指标包括准确度（Accuracy）、精度（Precision）、召回率（Recall）、F-值度量（F-measure）。通过综合比较这些数据，对模型进行评价。

【聚类任务】在聚类任务中，测量指标包括熵值（Entropy）和纯度（Purity）。同时，对聚类进行了可视化绘图。可以通过绘制结果直观地观察聚类结果的合理性。

3.4 图形绘制

【聚类任务】对于聚类任务，程序实现了对聚类结果进行可视化绘图的功能。

具体地，对于高维聚类结果，使用了 TSNE 进行降维。之后对于二维数据，利用 pyplot 绘图库进行了可视化绘图。

4. 结果评价

4.1 分类任务

4.1.1 测试结果

对于分类任务，程序使用了 K-折交叉验证（K=10）进行评估，测量指标包括准确度、精度、召回率、F-值度量，经过多次测试取平均，整理如下（耗时为整个 K-折交叉验证过程）：

表 3. 分类任务测试结果

模型	数据	准确度	精度	召回率	F-值度量	耗时
朴素贝叶斯	第一组	0.775	0.358	0.218	0.271	333.7 ms
	第二组	0.815	0.516	0.319	0.394	320.7 ms
决策树	第一组	0.824	0.282	0.263	0.272	1.3s
	第二组	0.872	0.374	0.446	0.407	937.4 ms
支持向量机	第一组	0.883	0	0	0	67.9 s
	第二组	0.890	0.186	0.599	0.283	119.4 s
朴素贝叶斯 (自实现)	第一组	0.751	0.402	0.208	0.274	827.5 ms
	第二组	0.814	0.513	0.318	0.392	796.6 ms

4.1.2 结果分析

【输入特征比较】比较两组输入特征，第一组输入特征关注用户账户存款、贷款、违约情况等账面信息；而第二组输入特征关注与用户沟通与联络的方式、频次、时长、距离时间与交流结果。两组输入特征，显然第二组有着更佳的预测效果。

特别地，对于支持向量机，第一组特征训练出的模型出现了全数预测结果为 0 的情况，足以证明第一组特征选择得不恰当。

【分类算法比较】比较三种分类算法，由于样本数据中 0 值数据远大于 1 值数据，所以值得参考的评测指标并非准确度（全预测 0 亦有较高准确度），而是精度、召回率与二者的结合——F-值度量。

就分类效果而言，三种算法中，支持向量机分类效果最差。决策树与朴素贝叶斯效果相当，而较为稳定。分析其原因，原理上支持向量机在高维空间中应当较为高效，但或因为对输入数据并未进行良好的正则化处理，其实际效果较差。对于决策树与朴素贝叶斯，或受特征选择的限制，以及数据预处理的缺乏，其分类正确率实际较低。

就运行耗时而言，支持向量机耗时远大于决策树与朴素贝叶斯，决策树耗时略大于朴素贝叶斯。这与输入数据的结构与算法本身的原理相对应。

【库算法与自实现算法比较】 对比 sklearn 库中提供的高斯朴素贝叶斯（GaussianNB）与自己实现的，可以看出二者在分类性能上差异不大，而在执行效率上有较大差异。具体地，尽管利用了 NumPy 进行辅助运算，自己实现的算法执行耗时为库中实现的两倍以上。显然，在常规算法之外，sklearn 库中进行了特殊的优化。

4.2 聚类任务

4.2.1 测试结果

对于聚类任务，测量指标包括熵值和纯度，整理如下：

表 4. 聚类任务测试结果

模型	数据	熵值	纯度	耗时
K-均值	第一组	0.810	0.772	5.4 s
	第二组	0.593	0.849	2.8 s
谱聚类	第一组	0.742	0.790	16.0 s
	第二组	0.675	0.836	13.9 s
K-均值 (自实现)	第一组	0.809	0.772	7.7 s
	第二组	0.632	0.845	5.2 s

4.2.2 绘图结果

经过 TSNE 降至二维后，聚类结果由 pyplot 绘图如下：

图 2. K-均值 第一组数据-6 维

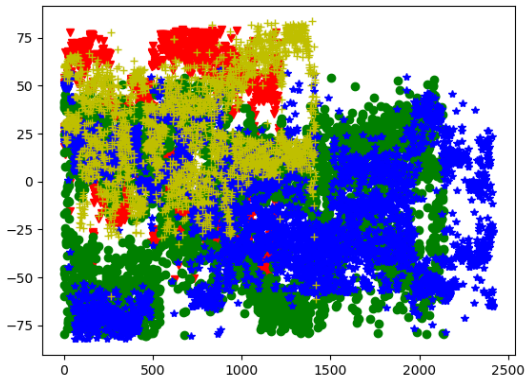


图 3. K-均值 第二组数据-12 维

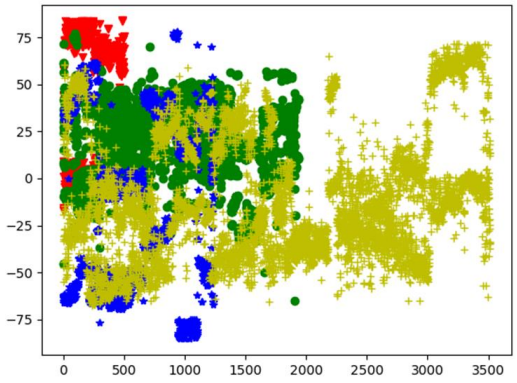


图 4. 谱聚类 第一组数据-6 维

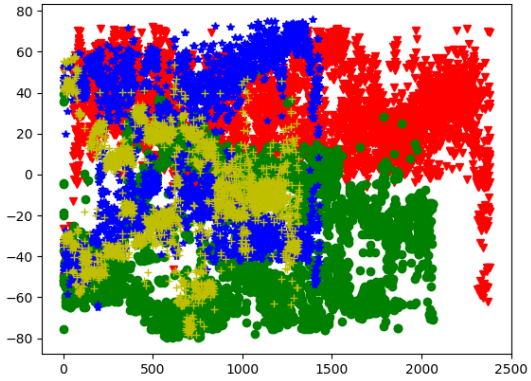


图 5. 谱聚类 第二组数据-12 维

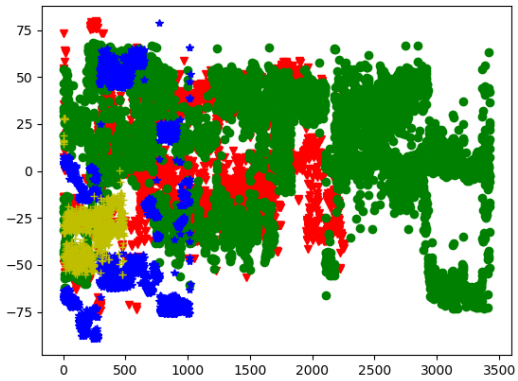


图 6. 自实现 K-均值 第一组数据-6 维

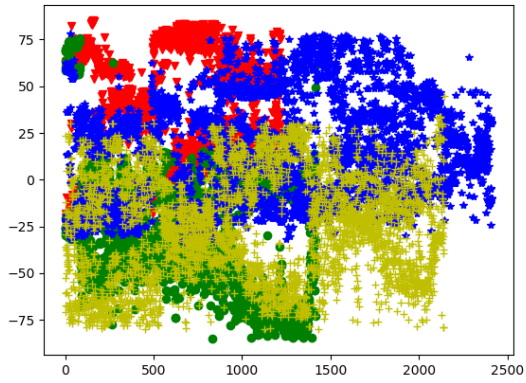
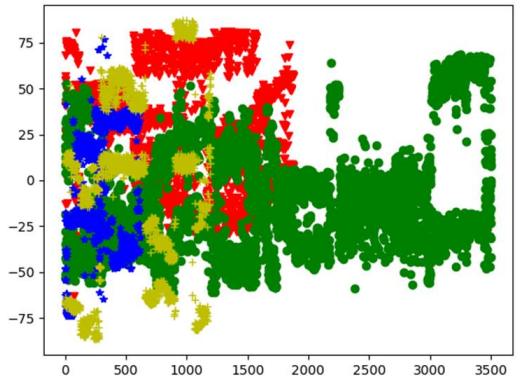


图 7. 自实现 K-均值 第二组数据-12 维



4.2.3 结果分析

【输入特征比较】比较两组输入特征，第一组为 MFCC 特征向量后 6 维数据，第二组为 MFCC 特征向量后 12 维向量。显然，对于三种算法，第二组数据效果更佳。更高维数据的选择，提高了数据的差异度，事实上不仅利于结果熵值的降低，纯度的提升，还能使 K-Means 过程更快收敛，这从第二组平均运行耗时低于第一组可以看出。

【聚类算法比较】比较两种聚类算法，本次实验中使用了熵值与纯度进行效果评估。

就聚类效果而言，尽管谱聚类算法的聚类效果常优于传统聚类算法，在本次实验中两种算法并无显著差异。这可能受限于问题背景、输入数据与度量方式。

就运行耗时而言，由于谱聚类涉及较多的矩阵操作（相似度矩阵、拉普拉斯矩阵等），因而计算量通常大于 K-均值，耗时较多也在预料之中。

【库算法与自实现算法比较】对比 sklearn 库中提供的 K-均值与自己实现的，可以看出二者在分类性能上差异不大，而在执行效率上略有差异，这可能是因为在 sklearn 库中，对数值计算进行了特殊的优化。

5. 使用说明

- **【项目结构】**本次实验的两个任务分别在不同的项目中：分类任务在项目<classification>中，聚类任务在项目<clustering>中。
- **【项目打开】**请使用 Pycharm 打开项目，并运行程序入口 main.py 文件（请注意，工作路径需为"classification\src" 或 “clustering\src”）。
- **【运行修改】**在 main.py 文件，根据 TODO 标识的提示，对代码进行简单修改，可进行不同的接口调用（评测不同的模型算法，绘制聚类图等）。

6. 实验感想

在本次实验中，综合运用了 numpy、pandas 数据处理库，sklearn 机器学习库，pyplot 图形绘制库，接触并部分实现了三种分类算法、两种聚类算法，以及机器学习数据预处理、模型评估的相关算法。

numpy、pandas 等数据处理库的使用，让我对数据处理有了更深的体会。例如，在数据读取时，pandas 的自动类型识别对数据结构的组织起到极大地帮助作用；在数据处理时，pandas 的 get_dummies 函数辅助了 one-hot 编码的生成；在数据计算时，numpy 提供的相关函数提高了计算效率。sklearn 机器学习库与 pyplot 图形绘制库的使用，让我理解了机器学习任务的一般过程与基础方法。对不同算法的原理与优劣有了一个初步而较为全面的了解。

总而言之，此次基础分类任务与聚类任务的实践，虽然内容较为基础，但极好地为我们理解了机器学习的流程与方法提供了平台，为我们之后的深层探索打下了基础。

7. 参考资料

[1] scikit-learn 官方文档：https://sklearn.apachecn.org/

[2] NumPy 用户手册：https://numpy.org/devdocs/user/quickstart.html

[3] pandas 用户手册：https://pandas.pydata.org/pandas-docs/stable/getting_started/index.html

[4] 分类型变量编码处理介绍博客：https://www.cnblogs.com/wyy1480/p/10295084.html

[5] MFCC 介绍博客：https://www.jianshu.com/p/24044f4c3531

[6] K-Means++算法介绍博客：https://www.cnblogs.com/wang2825/articles/8696830.html

[7] 谱聚类算法介绍博客：https://blog.csdn.net/qq_24519677/article/details/82291867

[8] TSNE 降维方法介绍博客：https://www.cnblogs.com/bonelee/p/7849867.html

[9] 算法实现参考：https://github.com/tushushu/imylu