《人工智能导论》Project-2

吴佳龙
软件 83
2018013418
wu-jl18@mails.tsinghua.edu.cn

ABSTRACT
本次实验分别使用有监督学习中的分类算法，通过银行营销数据，预测用户是否会购买银行产品；使用无监督学习中的聚类算法，根据 MFCC 特征，对不同科属的青蛙叫声进行聚类分析。

Keywords
有监督学习, 分类, 无监督学习, 聚类.

1. 银行精准营销解决方案

1.1 算法实现

1.1.1 特征工程
以下描述对数据集的预处理，具体细节详见 classification/data.py 文件。
将存在序关系的非数值型字段 'education', 'poutcome', 'month' 分别转化为区间 [-1,1], [-1,1], [0,11] 内的整数；将布尔类型的字段'default', 'housing', 'loan' 转化为 0 或 1；将 'pdays' 字段中的 -1 替换为 999。将没有明显序关系的非数值型字段 'job', 'marital', 'contact' 转化为 one-hot 编码。最后，将 'age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous', 'month' 字段进行标准化，即将其变换到均值为 0，标准差为 1。

1.1.2 特征选择
本次实验选择了两种特征组合，具体的选择字段见表 1。
第一组特征组合采用手工选择，选择的理由是：工作、受教育程度、违约、余额、贷款等比较能体现用户的信用程度；而最后一次联系的时间、上次活动的结果、交流次数等则可以体现用户对该银行产品的兴趣程度。
第二组特征组合采用前向选择算法选出，依据 SVM 分类器在 5 折交叉验证上的 AUC 指标来衡量特征子集的优劣程度。可以看出，手动选择的特征与自动选择的特征有相交的部分，同时也有较大的差异。

1.1.3 过采样
为了缓解类别不平衡的训练数据，在将数据传进分类器进行拟合之前，先对数据进行一次 oversampling，使得正例和负例所占的比例基本一致。
该过程调用 imblearn.over_sampling. RandomOverSampler 实现。

1.1.4 分类器
共使用了**五种**分类器完成分类任务。分别是 k 近邻，决策树，随机森林，支持向量机，多层感知机，分别调用：
- sklearn.neighbors.KNeighborsClassifier
- sklearn.tree.DecisionTreeClassifier
- sklearn.ensemble.RandomForestClassifier
- sklearn.svm.SVC
- sklearn.neural_network.MLPClassifier

另外还自己实现了具有一个隐藏层的多层感知机类 MyMLP。该分类器的隐藏层具有 100 个神经元，激活函数是 ReLU，输出层的激活函数是 sigmoid，采用交叉熵损失函数。使用带动量的 SGD 训练，默认的超参是：batch_size = 512, learning_rate=1e-2, weight_decay=1e-4, max_iter = 50000。

1.2 实验结果

1.2.1 评价指标
本次实验对于分类器的分类效果采用了 5 种评价指标，指标及选择理由如下：
- accuracy：准确率，分类问题的常用评价指标，但是对于数据分布不平衡的问题可能是不合适的，更全面的衡量需要用到混淆矩阵。
- precision：精度，预测为正的样本中预测正确的频率。
- recall：召回率，正样本中预测正确的频率。以上两者都与混淆矩阵相关，能够在数据不平衡的情况下对准确率进行补充，更全面地评估。
- F1：precision 和 recall 经常是数值高低经常是互斥的，F1 将两者组合起来，将两个指标进行平衡，越高越好。
- AUC：ROC 曲线下的面积，与 F1 类似，能够综合评价不同阈值下的精度和召回率，越高越好。

以上指标通过在训练集上进行 5 折交叉验证得出，调用了 sklearn 的 API：sklearn.model_selection.cross_validate。
具体实现见 classification/main.py 文件。

表 1 分类算法的两种特征组合

特征组合	选择字段
特征组合 1 (manual)	['job', 'education', 'default', 'balance', 'housing', 'loan', 'month', 'pdays', 'previous', 'poutcome']
特征组合 2 (forward select)	['age', 'education', 'housing', 'loan', 'contact', 'day', 'month', 'duration', 'pdays', 'poutcome']

1.2.2 超参选择

K 近邻算法的主要超参 k 的选取采用特征组合 3，并枚举 k 的取值在 1~10 之间，比较交叉验证的 F1 指标，如图 1。可以看到，在 k=5 时，F1 达到极值，并在 k 继续增大后开始下降，因此在下一小节的实验中，对 k 近邻的超参选取都选择 k=5。

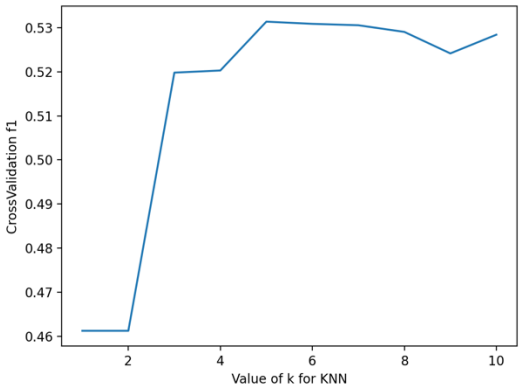

图 1 K 近邻的超参选择

其余算法的超参选择，以默认参数为主，详见 classification/models.py。

1.2.3 结果分析

使用不同特征组合的不同分类器的不同指标结果汇总如表 2。

不同特征的比较：通过比较表 2 的上半部分与下半部分，能够看出两者具有较大差异：手动选择的特征组合的结果显著地差于前向选择算法给出的特征组合。这说明，自动的特征选择算法能够找到更潜在更合理的特征组合。

不同分类器的比较：分析表 2 数据得出以下结论：

- 在两种特征组合中，决策树都取得了最差的结果，猜测的可能原因是，决策树的分类边界是与坐标轴平行的超平面的组合，在高维特征空间上出现了一定程度的过拟合。
- KNN 是第二差的，可能的原因是，欧氏距离可能并不是最适合的距离度量，且 KNN 模型简单，表达能力较差。
- 剩下三者（随机森林、SVM、MLP）的结果相近，他们都是较常用的分类器，具有较好的分类能力和鲁棒性。一个差别是随机森林的 precision 和 recall 相当，而 SVM 和 MLP 的 recall 要显著高于 precision。另外，随机森林的结果要显著好于决策树，这说明集成学习中的 bagging 方法能有效地提升模型的鲁棒性。

库与自己实现的 MLP 的比较：在分类结果上，MyMLP 在特征组合 1 上的结果要略优于 sklearn 的 MLP，在特征组合 2 上则略差于 MLP。这说明自己对多层感知机的实现是正确有效的。

在运行时间效率上，对比两者对整个数据集进行一次拟合的时间，分别为：MyMLP 103.21s；MLP 136.77s。可以看到两者的时间效率也是相当的。

表 2 使用不同特征组合的不同分类器的各种评价指标

特征组合	分类器	Accuracy	Precision	Recall	F1	AUC
特征组合 1 (manual)	K Neighbors	0.7410	0.2209	0.4802	0.3026	0.6533
	Decision Tree	0.8262	0.2619	0.2678	0.2647	0.5809
	Random Forest	**0.8405**	**0.3116**	0.2998	0.3053	0.6847
	SVM	0.7800	0.2750	**0.5369**	**0.3632**	**0.7200**
	MLP	0.7532	0.2450	0.5322	0.3355	0.6981
	MyMLP (自己实现)	0.7541	0.2482	0.5362	0.3382	0.7012
特征组合 2 (forward select)	K Neighbors	0.8515	0.4215	0.7190	0.5313	0.8390
	Decision Tree	0.8790	0.4821	0.4572	0.4690	0.6960
	Random Forest	**0.9002**	**0.5748**	0.5673	**0.5709**	**0.9244**
	SVM	0.8210	0.3832	**0.8672**	0.5314	0.9105
	MLP	0.8573	0.4406	0.8112	**0.5709**	0.9131
	MyMLP (自己实现)	0.8265	0.3886	0.8389	0.5310	0.9048

2. 青蛙叫声聚类分析

2.1 算法实现

2.1.1 特征选择

本次实验选择了三种特征组合，具体的选择字段见表格 3。第一组特征采用全部特征。在了解了 MFCC 特征的提取原理之后，我认识到，这是一种符合人体听觉，具有良好识别性能，广泛应用于语音识别的特征，因此有理由相信每一维特征都是重要且有用的。

第二组特征采用手工选取。选择的方式为计算每一维特征的标准差，去除标准差 < 0.1 的特征，剩余 17 维。

第三组特征采用采用前向选择算法选出，依据 KMeans 算法在聚类数目为 4 的情况下的轮廓系数来衡量一组特征子集的优劣，要求最终也选出 17 维特征。关于 KMeans 的聚类数目这一超参以及聚类评价指标轮廓系数等，详见下文。

表 3 聚类算法的三种特征组合

特征组合	选择字段
特征组合 1 (all)	'MFCCs_ 1', 'MFCCs_ 2', 'MFCCs_ 3', 'MFCCs_ 4', 'MFCCs_ 5', 'MFCCs_ 6', 'MFCCs_ 7', 'MFCCs_ 8', 'MFCCs_ 9', 'MFCCs_10', 'MFCCs_11', 'MFCCs_12', 'MFCCs_13', 'MFCCs_14', 'MFCCs_15', 'MFCCs_16', 'MFCCs_17', 'MFCCs_18', 'MFCCs_19', 'MFCCs_20', 'MFCCs_21', 'MFCCs_22'
特征组合 2 (std top)	'MFCCs_ 8', 'MFCCs_16', 'MFCCs_ 6', 'MFCCs_22', 'MFCCs_10', 'MFCCs_17', 'MFCCs_14', 'MFCCs_12', 'MFCCs_ 4', 'MFCCs_ 5', 'MFCCs_ 7', 'MFCCs_ 9', 'MFCCs_11', 'MFCCs_15', 'MFCCs_13', 'MFCCs_ 2', 'MFCCs_ 3'
特征组合 3 (forward select)	'MFCCs_ 1', 'MFCCs_22', 'MFCCs_10', 'MFCCs_ 7', 'MFCCs_15', 'MFCCs_12', 'MFCCs_ 6', 'MFCCs_ 5', 'MFCCs_11', 'MFCCs_17', 'MFCCs_ 9', 'MFCCs_20', 'MFCCs_21', 'MFCCs_18', 'MFCCs_19', 'MFCCs_13', 'MFCCs_ 2'

2.1.2 距离度量

本次实验采用的 MFCC 特征之间的距离度量为**欧几里得距离**。理由如下：

- 在经典的度量时序数据相似性的算法 DTW [1] 中，对于 MFCC 特征序列中**两帧** MFCC 特征的距离度量，常用的就是欧氏距离以及标准化欧氏距离 [2] [3]，而根据数据集说明，特征已经进行过归一化处理，使用欧氏距离即可。
- KMeans 算法是为欧氏距离设计的算法，在非欧氏距离上，不一定能保证收敛，可能需要满足额外的条件。

2.1.3 聚类算法

共使用了两种聚类算法，分别是 KMeans 和凝聚式层次聚类，分别调用：

- sklearn.cluster.KMeans
- sklearn.cluster.AgglomerativeClustering

另外，还自己实现了 KMeans 算法。实现细节如下：初始随机选择数据集中的 k 个点作为初始的簇中心点，然后执行 KMeans 算法；每次 fit_predict 共要执行 T 次如上过程，默认 T=10，并依据收敛后的类内凝聚度 SSE 选择最优的聚类（SSE 也为自己实现）。详见 clustering/model.py 文件。

2.2 实验结果

2.2.1 评价指标

本次实验对聚类算法的聚类效果采用了两类评价指标，分别是未知真实标签的评价指标和已知真实标签的评价指标：

- 未知真实标签：CH 分数 calinski harabasz score, 轮廓系数 silhouette score, SSE。前两者都综合评价了类内的凝聚度和类间的分离度，数值越大越好；SSE 仅衡量了类内凝聚度，数值越小越好。
- 已知真实标签：熵，纯度。在这种情况下，我们固定 k 为真实的聚类数目，即 k=4。

各种结果见图 2、表 4。

注：由于 MyKMeans 没有设置随机种子，重复实验，结果可能会有一些微小的变化。

图 2 不同特征组合的不同算法的未知真实标签的聚类指标

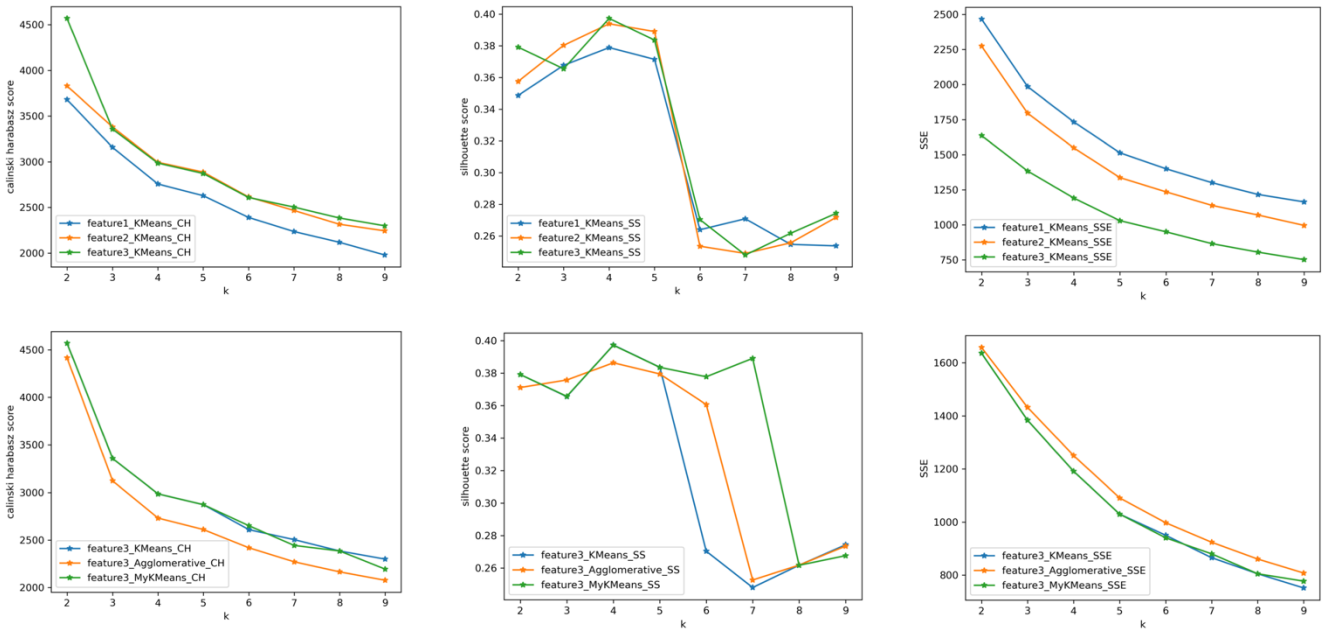

表 4 不同特征组合不同算法的熵和纯度(k=4)

特征组合	聚类算法	熵	纯度
特征组合 1 (all)	KMeans	0.4616	0.8068
	MyKMeans	0.4631	0.8063
	Agglomerative	0.5424	0.7816
特征组合 2 (std top)	KMeans	**0.4597**	0.8077
	MyKMeans	**0.4597**	0.8077
	Agglomerative	0.4655	**0.8130**
特征组合 3 (forward select)	KMeans	0.5383	0.7767
	MyKMeans	0.5379	0.7770
	Agglomerative	0.4985	0.7861

2.2.2 超参选择

KMeans：算法的主要超参是聚类数目 k。根据图 2 第一行，不论取用何种特征组合，随着 k 的增大，CH 分数和 SSE 都呈现下降趋势，且无明显的斜率变化的拐点，因此无法通过 elbow method[4] 来决定 k。

但是可以观察轮廓系数，发现对于三种特征组合，KMeans 都在 k=4 达到最高，因此可选择超参为 k=4。

AgglomerativeClustering：算法的 k 选择与 KMeans 类似。当选择特征组合 1 或 2 时，k=5 轮廓系数达到极值，而特征组合 3 时，轮廓系数的极值为 k=4。详见图片 clustering/figure/Agglomerative_SS.png。因此可以选择 k=4 或 5，依据主观感受或实际需求而定。

该算法另一个更加重要的超参是簇之间的距离度量方式，即 linkage 参数，有四种取值 'ward', 'average', 'complete', 'single'。在选用特征组合 3，k=4 时，使用不同的 linkage，得到的聚类结果用 tSNE 展示如下图 3。可以看到，average 和 single 的聚类效果明显差于 ward 和 complete，而后两者中，ward 的聚类效果主观上更好。因此选择 linkage='ward'。

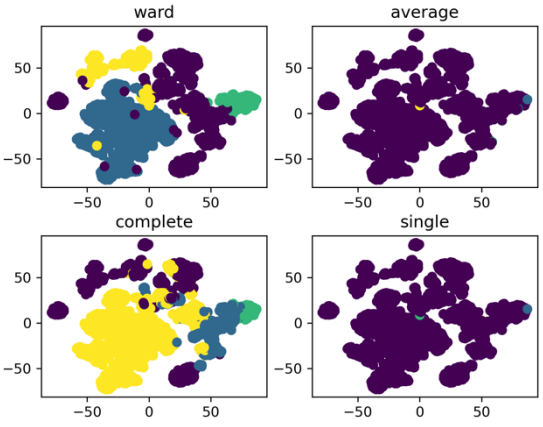

图 3 AgglomerativeClustering 中超参数 linkage 的选择

2.2.3 结果分析

不同特征的比较：根据图 2 第一行，对于 KMeans，特征组合 2 和 3 都得到了比特征组合 1（简单地选择全部特征）更好的数值结果。前两者在 CH 和 SS 上表现相当，而在 SSE 上，则是前向选择的特征组合 3 要优于依据标准差选择的特征组合 2。

以上是在未知真实标签的指标上的结果，在已知真实标签的指标上则有不一样的结果。根据表 4，通过熵和纯度分别比较，都有组合 2 优于组合 1 优于组合 3。

这说明，我们很难找到在各个方面都优于其他组合的特征组合，很多时候需要依据实际需求选择。

不同分类器的比较：通过特征组合 3 上的 CH 和 SSE 来看，KMeans 要优于 Agglomerative。从熵和纯度看，组合 1 上 KMeans 更优，组合 2 上两者相当，组合 3 上 Agglomerative 更优。

这也说明了，没有各方面都优于其他算法的聚类算法，需要依据实际需求。

库与自己实现的 KMeans 的比较：在聚类结果上，MyKMeans 与 sklearn 实现的 KMeans，无论在 CH、SS、SSE，还是在熵和纯度上，都得到了很接近的结果。下一小节的可视化结果也很类似。这说明自己实现的 KMeans 是正确的有效的。

在时间效率上，比较两者对于整个数据集进行一次聚类的时间：KMeans 0.3557s；MyKMeans 18.62s。可以看到两者有数量级上的差异，比较遗憾地，MyKMeans 的效率远不如 KMeans。

2.2.4 可视化结果

使用特征组合 3，k=4，将不同的算法的聚类结果使用 tSNE 可视化到二维或三维如图 4。

可以看到，KMeans 与 Agglomerative 得到了相近的聚类结果，且都比较自然，符合直观。而 KMeans 与 MyKMeans 在某次运行中得到了几乎一样的结果（注意 MyKMeans 没有设置随机种子，重复可能会得到不同但相近的结果），这说明自己对于 KMeans 的实现是成功的。

3. 参考资料

[1] Dynamic time warping – Wikipedia, https://en.wikipedia.org/wiki/Dynamic_time_warping.

[2] matlab - How to perform DTW on an array of MFCC coefficients? - Stack Overflow, https://stackoverflow.com/questions/39752462/how-to-perform-dtw-on-an-array-of-mfcc-coefficients.

[3] pierre-rouanet/dtw: DTW (Dynamic Time Warping) python module, https://github.com/pierre-rouanet/dtw.

[4] Elbow method (clustering) – Wikipedia, https://en.wikipedia.org/wiki/Elbow_method_(clustering).

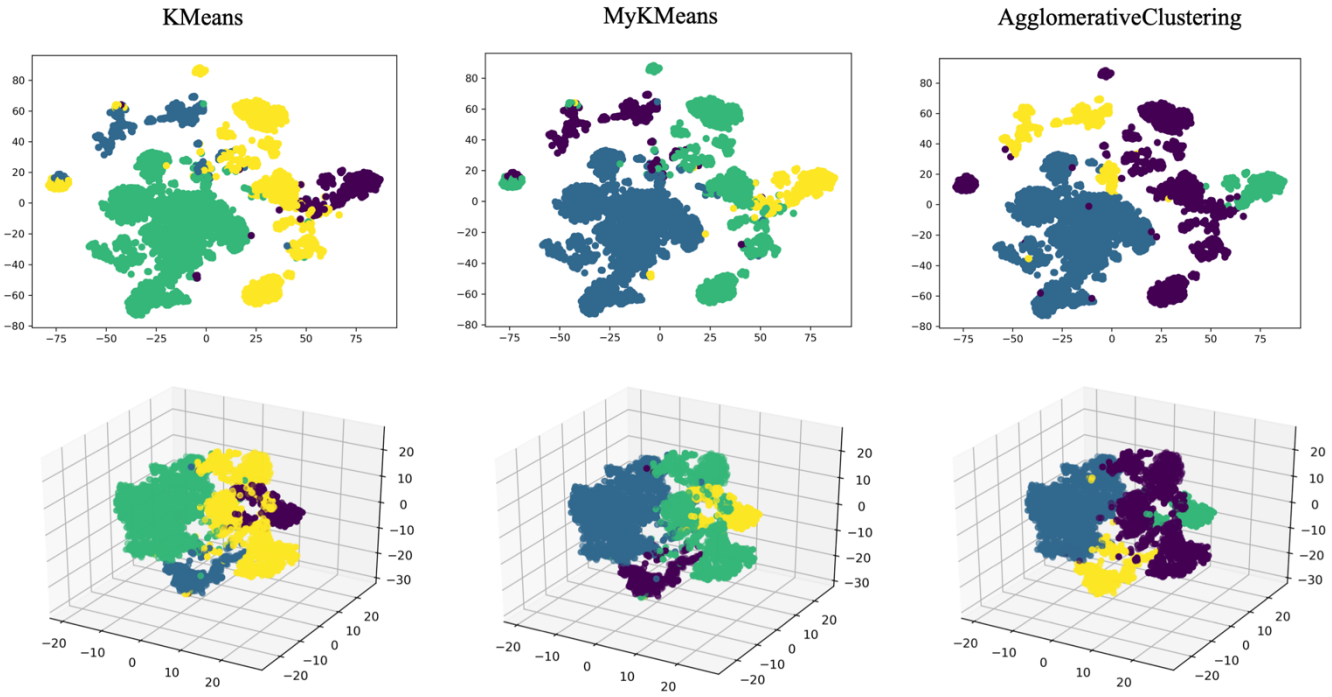

图 4 通过 tSNE 可视化聚类结果