第二次作业报告

沈冠霖

软 73

2017013569

第一部分

1. 模型

我选用了三个分类器：朴素贝叶斯分类器，SVM 分类器和全连接神经网络分类器。其中，朴素贝叶斯分类器既调用了 sklearn 的库（bayes.py）又自己手动实现了(self_model.py),SVM 分类器调用了 sklearn 的库（svm.py），神经网络调用了 pytorch 的库，由三个使用 ReLU 激活函数的全连接层构成（mlp.py 和 mlp_model.py）。

2. 数据

2.1 选取情况

我选取了 6 组特征：

1."balance", "duration", "pdays"

2."balance", "duration", "previous"

3."balance", "duration", "poutcome"

4."duration", "pdays"

5."duration", "previous"

6."duration", "poutcome"

2.2 选取原因

首先，我将所有 string 数据编码成 int 数据，然后求得了各个数据之间的相关性（correlation.py），参考 correlation_task1.png。

可以发现，结果 y 和以下变量相关性较高：duration(0.39), pdays(-0.18), previous(0.088), poutcome(0.2), contact(0.14)。因此，从相关性看，我们优先选取以上的特征。

同时，根据常识，pdays, previous, poutcome, duration 代表客户和银行的交流效果，和客户是否会购买服务是有关的，contact 只是交流方式，个人觉得这个关系不大，就没有选取。同时，balance 代表客户的财务情况，也和客户是否会购买服务逻辑上有联系，我也考虑了这个特征，但是也比较了有这个和没这个的情况。

同时，因为贝叶斯分类器要求选取的特征之间尽量无关，而特征相关性太高也的确会导致算力的浪费，因此我也考虑了这些变量之间的相关性。首先，balance，duration 之间，以及他们和其他三个变量的相关性都不大（绝对值小于 0.05），因此把这两个当做独立出现的变量。其次，pdays，previous，poutcome 三者间互相相关性比较大（绝对值至少为 0.19），因此不作为独立出现的变量，三者只能同时出现一个。

最终，因为不同分类器的特性，也要考虑选取的变量是离散还是连续的。Balance，duration，pdays 的变化范围较大，可以近似看作连续变量，poutcome 和 previous 的取值和变化范围就很小，可以看做离散变量。我们这样取值也能比较出不同分类器的特点。

3. 评价

3.1 评价准则

我选取了 Accuracy，Precision, Recall, F1 这四个评价准则来看。Precision 代表被预测会购买的人里有多少实际会购买，是预测的效率；Recall 代表实际会购买的人有多少被推荐算法预测到了，是预测精准程度。F1 则是这两个准则的结合。因为无论是预测效率还是精准程度都很重要，因此觉得 F1 最能代表模型的效果。

我使用 k 折交叉验证，然后取平均值的方法进行评价。

3.2 结果与对比

Table 1. 不同分类器和特征组合的结果对比

注：朴素贝叶斯为调用的标准库，四个结果从上到下分别是 Accuracy，Precision，Recall 和 F1

特征组合/分类器	朴素贝叶斯	SVM	神经网络
1	88.26% 49.72% 29.62% **37.12%**	88.41% 64.77% 1.93% **3.74%**	88.88% 58.58% 16.95% **26.30%**
2	88.12% 48.61% 27.79% **35.37%**	88.33% 57.45% 0.91% **1.80%**	88.85% 56.13% 21.18% **30.75%**
3	89.19% 54.97% 41.68% **47.41%**	88.36% 62.07% 1.22% **2.38%**	88.92% 59.26% 16.75% **26.12%**
4	88.92% 54.88% 29.45% **38.33%**	89.02% 61.00% 16.95% **26.53%**	89.08% 57.39% 25.84% **35.63%**

5	88.62% 52.64% 27.69% **35.94%**	88.98% 59.56% 18.10% **27.76%**	88.98% 57.87% 21.24% **31.08%**
6	89.69% 58.37% 41.44% **48.47%**	88.96% 59.22% 17.90% **27.49%**	89.48% 62.00% 25.90% **36.54%**

3.3 分析

3.3.1 模型

无论是哪组特征，效果都是贝叶斯>神经网络>SVM。因为待预测的 y 只有是和否两种情况，编码成 1 和 0，是典型的离散数据，编码的差值较小，而贝叶斯最适合预测这样的数据，其结果和数据编码无关（编码成 1 和 0，还是 100 和 0 都没有影响）。而神经网络和 SVM 的预测结果则和数据编码有密切关系，表现就比贝叶斯差。而 SVM 的分类效果和数据间的距离密切相关，受到影响就比较大，而神经网络的拟合和表达能力都很强，受到影响较小。

而对比特征组合 1 和 3，还有 4 和 6（仅仅是 pdays 和 poutcome 的区别）的效果，也可以看出三种分类器的性质。Pdays 和 poutcome 和 y 的相关性差异不大（-0.18 和 0.2）,但是前者类似连续数据，后者更像离散数据。在朴素贝叶斯网络里，使用 poutcome 的效果比使用 pdays 好很多(第三组 47.41% > 第一组 37.12%, 第六组 48.47% > 第四组 38.33%)，而另外两组网络里，两者效果差异不大。这可以看出，朴素贝叶斯网络更适合分类取值情况更少的离散数据，而不是取值情况更多的连续数据。

3.3.2 特征

首先，通过对比数据组 1 和 4,2 和 5,3 和 6 的情况，可以看出，balance 这个变量的确是一个干扰项，它和 y 的关系真的不大，选取它会干扰分类器，降低模型效果，这在 SVM 中体现尤其明显，使用 balance 变量让 SVM 的 F1 下降到个位数，因为 y 为 1 和 0 的不同数据，其 balance 可能很接近，这让 SVM 难以分类。

其次，使用 previous 的效果普遍差于使用 pdays 和 poutcome，因为其和 y 相关性（0.088）低于另外两者(-0.18,0.2)。

最终，在适合处理离散数据的朴素贝叶斯网络中，使用 poutcome 效果更好。而其他网络则是 poutcome 和 pdays 怎么选取都可以。

3.3.3 评价指标

无论是选用哪个神经网络，哪个特征，分类准确率都接近 90%，但是这个数据集的绝大多数都是拒绝，分类准确率高并不意味着分类效果好---预测所有数据都是被拒绝也能得到很高的准确率。从结果也可以看出，无论是 precision 还是 recall，还是 F1，这些网络的效果都并不是特别好。因此，综合参考各个评价指标，尤其是 F1 这种综合指标，比单纯看准确率要好。

3.3.4 总结

模型上看，这个问题最适合朴素贝叶斯分类器。

特征上看，最适合特征 6（duration + poutcome）

评价上看，最好选择 F1，而不是 accuracy 作为衡量模型效果的准则

3.4 自己实现的网络

Table 2. 自己实现的朴素贝叶斯网络和标准库对比

注：四个结果从上到下分别是 Accuracy，Precision，Recall 和 F1

特征组合/分类器	标准库	自己实现
1	88.26% 49.72% 29.62% **37.12%**	79.17% 25.84% 41.78% **31.93%**
2	88.12% 48.61% 27.79% **35.37%**	79.05% 24.42% 37.76% **29.66%**
3	89.19% 54.97% 41.68% **47.41%**	79.36% 26.43% 42.89% **32.71%**
4	88.92% 54.88% 29.45% **38.33%**	88.54% 51.75% 29.42% **37.51%**
5	88.62% 52.64% 27.69% **35.94%**	88.35% 50.49% 20.91% **29.57%**
6	89.69% 58.37% 41.44% **48.47%**	89.20% 57.26% 30.36% **39.68%**

可以看出，在各方面效果上，自己实现的贝叶斯分类器还是比标准分类器要差的。因为我的贝叶斯分类器严格按照定义来做，没有对连续数据做一些特殊处理，因此在处理连续数据上比较吃亏。

但是，我实现的贝叶斯分类器的效果并不算差，仍旧比使用 SVM 和神经网络要好一些。而且，我实现的贝叶斯分类器在不同特征下的性质也吻合调用标准库的贝叶斯分类器的性质，也可以证明我实现的正确性。

第二部分

1. 模型

1.1 模型实现

我使用了 k-means 算法和凝聚层次聚类算法。K-means 算法既调用了 sklearn 库(kmeans.py)，又自己实现了二分 k-means 算法（self_model.py）。凝聚层次聚类算法调用了 sklearn 库（hierarchy.py），以组平均距离。

1.2 距离度量选择

我选择的是余弦距离度量。因为数据集 readme.md 里提到，MFCC 提取的每一维度实际上大小都是不同的，经过 normalize，MFCCs_i := MFCCs_i/(max(abs(MFCCs_i)))之后才变成数据集的情况的，数据集想保留的是各个分量占最大值的比例，改变了向量绝对长度。余弦度量考虑的是相似性，对向量长度不敏感，这和欧氏距离和曼哈顿距离不同[1]。而且欧氏距离和曼哈顿距离一般采用原始数据，不会进行这种 normalize 处理[2]。

因为 sklearn 自带的 k-means 只支持欧氏距离，因此这个没法改。但是我自己实现的 k-means 和凝聚层次聚类使用的是余弦距离。

1.3 K 值选择

我选择 k 值为 4。

我用自己实现的 k-means 模型，使用之后提到的特征组 3（6,11,19），测试了 k 为 2,3,4,5,6,7,8 的各种情况，每组情况测试五次取平均值，结果如下：

Table 3. 不同 k 值的结果对比

K 值/指标	Purity	SSE	BSS
2	76%	1190.73	763.50
3	76%	1197.23	755.49
4	79%	1157.13	799.35
5	78%	1171.16	782.32
6	76%	1191.42	760.80
7	78%	1171.71	784.16
8	76%	1199.85	754.95

可以看出，无论选取哪个值，三个指标都相差不大。因此，我选择 4，因为一共 4 个科，而我们的聚类正是以动物的科作为基准的，这样选取比较合理。

2. 数据

2.1 选取情况

我选取了 4 组特征：

1. (4,11,19)
2. (4,13,19)
3. (6,11,19)
4. (6,13,19)

2.2 选取原因

首先，我将所有 string 数据编码成 int 数据，然后求得了各个数据之间的相关性（correlation.py），参考 correlation_task2.png。

可以看出，特征 4,6,11,13,15,17,19,20,22 和青蛙的科相关性绝对值较高，说明这些特征有利于更好的分类。

之后，因为选取相关性非常高的两个特征进行分类是比较浪费的，因此我选择相对彼此间相关性较小的以上 4 组特征进行分类。同时，因为特征太少会导致信息太少，容易出现偏差，特征太多又很费算力，我选择了一组 3 个特征。

3. 评价

3.1 评价准则

基于数据外在度量，也就是以青蛙的科作为基准，可以用熵和纯度来度量。基于数据内在度量，可以用 SSE 和 BSS。因为使用熵，经常出现 P=0 的情况，因此我用了纯度，SSE,BSS 三个准则来评价。

3.2 结果与对比

Table 4. 不同算法和特征组合的结果对比

注：K-means 为调用的标准库，三个结果从上到下分别是 Purity，SSE,BSS。每组数据都是五次实验取平均。

特征组合/分类器	K-means	层次聚类
1	82% 1121.92 832.68	62% 1790.59 85.39
2	78% 1118.16 839.00	62% 1797.09 79.40
3	83% 1083.95 872.77	81% 1141.39 814.71
4	78% 1083.03 877.94	62% 1544.18 379.34

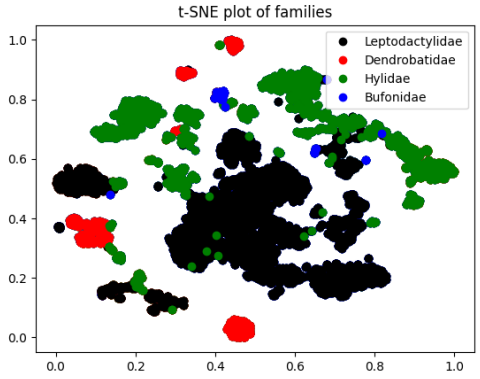
3.3 分析

3.3.1 模型

可以看出，K-means 算法的效果是比层次聚类要好的，包括更高的纯度，更低的 SSE 和更高的 BSS。首先，看 t-SNE 可视化结果可以看出，数据按照科的分类簇是大小相差很大，

分配不均衡的（黑色簇很大，红色和绿色都分布不均）。用单一的组平均度量方法很处理这种复杂的分布，因此层次聚类效果普遍不好。

图 1：按照科分类的可视化结果



3.3.2 特征

第三组特征（6,11,19）无论是在 k-means 还是层次聚类里都是表现较好的。6,11,19 这三个特征相互间相关性不高，都只有不到 0.35，而且 19 和科的相关性最高，达到 0.62，是很好的特征。而数据 4 和数据 11,13 的相关性就较高，绝对值超过 0.5，已经超过 4 和科本身的相关性了，4 和 11,4 和 13 组合对样本的代表性不如 6 和 11 的组合。

3.3.3 评价指标

整体上看，使用这三个评价指标都是相对合理的。一方面，这三个评价指标能全面反应分类结果的内在和外在度量。另一方面，这三个评价指标是正相关的---模型越好，purity 就越大，SSE 就越小，BSS 就越大，代表结果和科的关联更高，更加高内聚低耦合。

3.3.4 总结

模型上看，K-means 算法表现更好。

数据上看，特征 6,11,19 这种互相之间相关性低，而且都和科相关性高的数据最有代表性。

评价上看，综合使用三个指标很有效果。

3.4 自己实现的网络

Table 5. 标准库和自己实现的二分 k-means 算法对比

注：三个结果从上到下分别是 Purity，SSE,BSS，每组数据都是五次实验取平均。

特征组合/分类器	标准库	自己实现
1	82% 1121.92 832.68	72% 1434.17 496.29
2	78% 1118.16 839.00	73% 1292.27 659.94
3	83% 1083.95 872.77	76% 1188.27 766.78
4	78% 1083.03 877.94	77% 1219.09 735.58

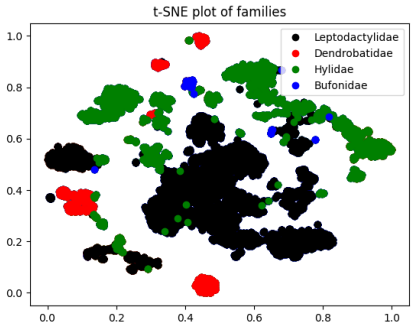
可以看出，在各方面效果上，自己实现的分类器还是比标准库要差一些的。而且，在一些边界条件上，自己实现的算法缺乏有效处理，鲁棒性没有那么强。

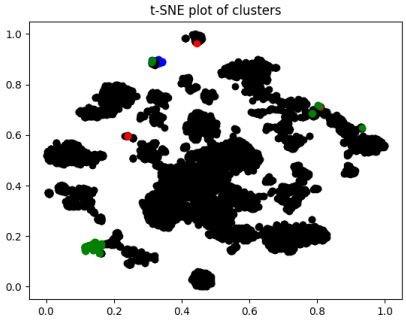
但是，自己实现的算法的结果仍然符合标准库的规律，而且整体上仍然比层次聚类的效果好，也可以说明自己实现算法的正确性。

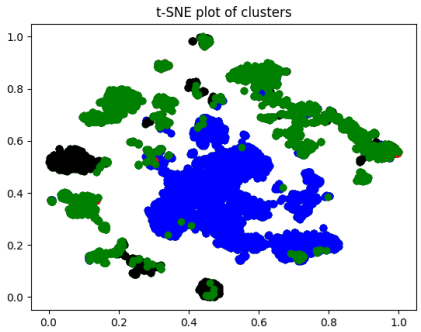
3.5 T-SNE 可视化结果

我将使用不同 feature 分类的结果，和以科作为标签的结果按照同样的 t-SNE 属性进行可视化，有助于对比。具体结果在 task2 的 result 文件夹里。这里仅举层次聚类的可视化结果的例子。

图 2：层次聚类按照科/特征 1 聚类/特征 3 聚类的结果







对比这三个结果，可以发现，如果使用特征组合 1，那么几乎所有数据都被分类到一个簇里，效果不好。而使用特征组合 3，中心的数据簇和边缘的数据簇和按照科分类的结果基本吻合，效果较好。
可以看出，可视化结果和上面用到的纯度，SSE,BSS 评价标准是较为吻合的。

参考资料

[1] 聚类（一）：相似性度量
https://zhuanlan.zhihu.com/p/71610113

[2] 聚类分析中距离度量方法比较
https://blog.csdn.net/jbfsdzpp/article/details/48497347