华东师范大学数据学院上机实践报告

课程名称：信息检索　　年级：2018　　上机实践成绩:

指导教师：张蓉　　姓名：孙秋实

上机实践名称：基于 Naive Bayes 的文本分类　　学号：10185501402　　上机实践日期：2021/12/9

Part 1
实验目的

(1) 朴素贝叶斯文本分类

Part 2
实验任务

(1) 朴素贝叶斯法简介与推导

(2) 实现朴素贝叶斯算法

(3) 模型评估

(4) 使用交叉验证辅助参数选取

Part 3
使用环境

(1) Google Colab

(2) Python 3.7

Part 4
实验过程

Section 1
Naive Bayes

朴素贝叶斯算法是一种基于贝叶斯定理的机器学习算法，它在文本分类任务和垃圾邮件（信息）检测任务，其最基本的假设即每个样本特征与其他的特征不相关，表示如下

$$P(x \mid y) = P\left(x_1, x_2, \ldots \mid y\right) = P\left(x_1 \mid y\right) P\left(x_2 \mid y\right) \ldots = \prod_{i=1}^{n} P\left(x_i \mid y\right)$$

朴素贝叶斯算法实际上学习到生成数据的机制，所以朴素贝叶斯算法属于**生成模型**。

在使用朴素贝叶斯法分类时，对给定的输入 x，通过学习到的模型计算后验概率分布 $P\left(Y = c_k \mid X = x\right)$，将后验概率最大的类作为 x 的类输出. 后验概率计算根据贝叶斯定理进行，后验概率如下所示

$$P\left(Y = c_k \mid X = x\right) = \frac{P\left(X = x \mid Y = c_k\right) P\left(Y = c_k\right)}{\sum_k P\left(X = x \mid Y = c_k\right) P\left(Y = c_k\right)}$$

Section 2
算法流程

我们把独立性假设扩展到条件独立性假设，条件概率分布的参数数量为指数级增长，这在应用中是行不通的，我们需要以下假设：

$$P\left(X=x \mid Y=c_k\right)=P\left(X^{(1)}, \ldots, X^{(n)} \mid Y=c_k\right)=\prod_{j=1}^{n} P\left(X^{(j)}=x^{(j)} \mid Y=c_k\right)$$

算法的核心是最大化后验概率，朴素贝叶斯法将当前的样本分类到后验概率最大的类中，这步等价于期望风险最小化。

$$P(X|Y)posterior=\frac{\overbrace{P(Y|X)}^{likelihood}\overbrace{P(X)}^{prior}}{P(Y)evidence}=\frac{\overbrace{P(Y|X)}^{likelihood}\overbrace{P(X)}^{prior}}{\underbrace{\sum_{x} P(Y|X)P(X)}_{evidence}}$$

平滑贝叶斯估计如下，当 $\lambda=1$ 时，这个平滑方案叫做 Laplace Smoothing。拉普拉斯平滑可视作给未知变量给定了先验概率，也防止了出现分母接近于 0 的情况。

$$P_{\lambda}\left(X^{(j)}=a_{jl} \mid Y=c_k\right)=\frac{\sum_{i=1}^{N} I\left(x_i^j=a_{jl}, y_j=c_k\right)+\lambda}{\sum_{i=1}^{N} I\left(y_j=c_k\right)+S_j\lambda}$$

值得注意的是，平滑参数 λ 的选取因任务的不同而不同，在本次实验中，我们将使用交叉验证方法确定对文本分类最佳的超参数值。

Section 3
算法实现

该实验任务的概览和主要算法如下

首先读入训练集数据

```
path = './train'
label = list(os.listdir(path))
```

```
['talk.politics.mideast',
 'rec.autos',
 'comp.sys.mac.hardware',
 'alt.atheism',
 'rec.sport.baseball',
 'comp.os.ms-windows.misc',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.med',
 'talk.politics.misc',
 'rec.motorcycles',
 'comp.windows.x',
 'comp.graphics',
 'comp.sys.ibm.pc.hardware',
 'sci.electronics',
```

```
 'talk.politics.guns',
 'sci.space',
 'soc.religion.christian',
 'misc.forsale',
 'talk.religion.misc']
```

可见有 10 个新闻类别，随后读入数据

```
def get_filelist(dir):
    Filelist = []
    for home, dirs, files in os.walk(dir):
        for filename in files:
            Filelist.append(os.path.join(home, filename))
    return Filelist
```

将这些新闻文本建立两组字典，并进行词频统计

(1) 对单独一类新闻的字典 + 词频统计

(2) 总字典 + 词频统计

随后再统计文档，得到下列结果

```
{'talk.politics.mideast': 191464,
 'rec.autos': 80347,
 'comp.sys.mac.hardware': 62676,
 'alt.atheism': 104488,
 'rec.sport.baseball': 89986,
 'comp.os.ms-windows.misc': 94431,
 'rec.sport.hockey': 133506,
 'sci.crypt': 124474,
 'sci.med': 105481,
 'talk.politics.misc': 149480,
 'rec.motorcycles': 75803,
 'comp.windows.x': 104955,
 'comp.graphics': 108093,
 'comp.sys.ibm.pc.hardware': 74731,
 'sci.electronics': 75584,
 'talk.politics.guns': 125506,
 'sci.space': 114462,
 'soc.religion.christian': 133781,
 'misc.forsale': 68681,
 'talk.religion.misc': 118364}
```

实现后验概率的计算

```
# cal post perior prob
postperior = {}
for labelname in label:
    temp = {}
```

```
    for word in v[labelname]:
        word_low = word #w.lower()
        temp[word_low] = (dic_doc[labelname][word_low] + lambda)/(num_doc[labelname]+len(dict))
    postperior[lb] = temp
```

朴素贝叶斯预测

```
for lb in label:
        predict_score = np.log(len(docu[lb])/sum_doc)
#         print(predict_score)
        for w in test.split():
            if w in p_lb[lb]:
                predict_score += np.log(p_lb[lb][w])
            else:
                predict_score += np.log(1/(num_doc[lb]+len(dict)))
        score_list[lb] = predict_score
```

最后对单个测试文件的预测取最大值所对应的标签，并输出为以下形式

```
cbd055b607cdb0cd0d38fe46a2d8a2fe:talk.politics.mideast
38bb90cfef3662a464eac334754173a2:sci.med
62d491c8061e4f5daf87e69243416798:talk.politics.guns
8c7a62967cd1871feb7718b73adac71a:sci.crypt
5f02d5cc239f10e90c0e4b0fcb1e346f:talk.politics.guns
6fc4ef8b520cf5e5982926139fdae27d:talk.politics.misc
ba57244016cb711ac530424dbc014830:talk.politics.guns
6e0cf31ab8869301906f8b378a5b19b4:alt.atheism
d9197d33e4f4863a54f7628cab1099ed:talk.politics.misc
72c1ef79f08caeab93debe9f2922b49a:comp.sys.mac.hardware
...
```

Section 4
交叉验证

在实现朴素贝叶斯算法的过程中涉及超参数的选取，在此使用交叉验证法辅助选取超参数，并绘制误差曲线。

*此次实验中，为了防止频繁词项对判别产生干扰，对于每个标签所对应的词典去除了频次 >800 的词项。

在以下超参数空间进行搜索：$lambda\_list = [0.0001, 0.001, 0.01, 0.05, 0.125, 0.25, 0.5, 0.75, 1, 2]$

交叉验证的搜索结果如下所示（纵坐标数值为 Accuracy）

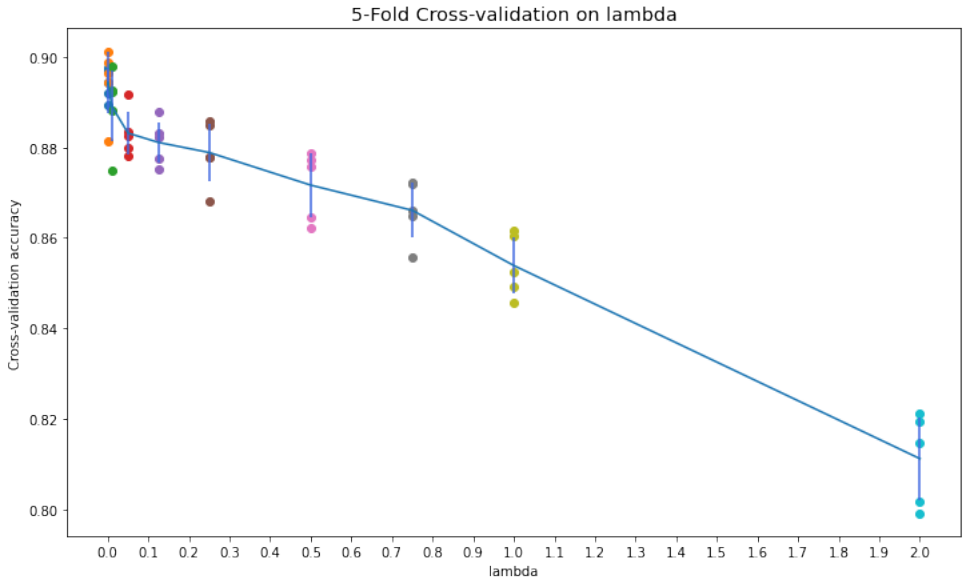

图 1: Caption

随后比较各超参数所对应的组均值，选择 $\lambda = 0.001$ 为平滑参数，验证集准确率为 0.86（后在测试集准确度为 0.83），文本分类均值比较如图 2 所示。

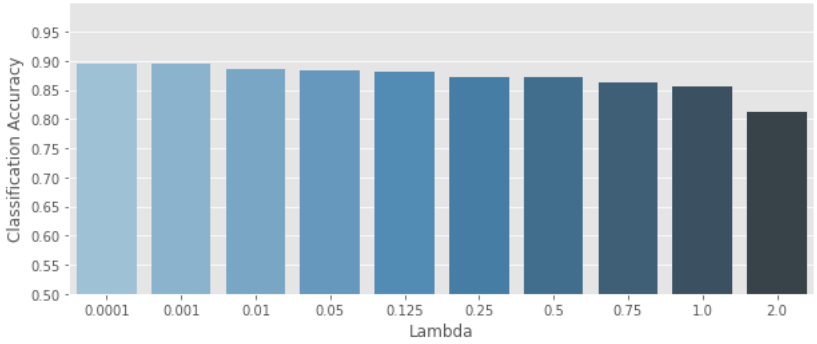

图 2: Caption

个人对实验结果的理解：如图 1 所示，Laplacian Smoothing 越小，文本分类模型准确率越高，可能的原因是 $\lambda = 0$ 时的极大似然估计是理论上的最优解，Laplacian Smoothing 只是为了排除概率接近或等于 0 的情况下导致计算出错，所以理论情况下，只要使概率不为 0，超参数 λ 取值越小越逼近理论最优值。

Part 5

实验总结

本次实验完成了基于朴素贝叶斯分类器的文本分类任务。通过本次实验，我推理了朴素贝叶斯算法的流程，对该算法有了更加深入的理解。此外，在获得最终的分类结果之前，我进行了交叉验证并可视化其结果，辅助了朴素贝叶斯分类器超参数的选取。在该数据集中，20 分类问题的验证集准确率接近了 0.9，这体现出了朴素贝叶斯模型的有效性。而对 Laplacian Smoothing 的分析中可见平化程度越小，文本分类准确率会相应越高（理论上越逼近最优）。

Part 6

参考资料

(1)《统计学习方法（第二版）》李航. 清华大学出版社

(2) Stanford CS229 Maching Learning: Deep Learning Cheatsheet

(3) Stanford CS221 Artificial Intelligence