# Homework2 实验报告

| 姓名： | 程倩楠 | 学号： | 201834858 |
|---|---|---|---|
| 实验课程： | Data Mining | 实验名称： | NBC |
| 指导老师： | 尹建华 | 完成时间： | 2018.11.16 |

## 一、实验目的

理解朴素贝叶斯的基本原理，能够编程实现朴素贝叶斯分类器。

## 二、实验环境

python3.6

## 三、实验内容

实现朴素贝叶斯分类器，测试在 20 Newsgroups 数据集上的效果。

## 四、实验过程

### （一）数据预处理

在 Homework1 中借助 Lucene 工具用 java 程序实现新闻文档的预处理，这里借助 nltk 模块用 python 程序实现预处理，处理步骤如下：

1. 词条化（利用非字母的字符对文本进行分割）
2. 去除停用词
3. 小写化
4. 词干化（Porter Stemmer）
5. 统计出每个单词的集合频率（Collection Frequency, CF），即单词在所有新闻文档中出现的次数。去除文档中 CF<5 的所有低频单词。

### （二）划分数据

划分数据依靠函数 dataSeg(fold,rightCateFile,trainDataProp=0.8)来实现，函数的主要功能为划分出一定比例的测试数据和训练数据，并将标注的测试数据记录在特定文件中。

| 参数 | 解释 |
|---|---|
| fold | 将第几份数据作为测试数据<br>(用于 n-Fold Cross Validation) |
| rightCateFile | 记录标注测试数据的文件 |

| | <newsName_newsClass　newsClass> |
|---|---|
| trainDataProp | 划分为训练数据的比例 |

```
def dataSeg(fold,rightCateFile,trainDataProp=0.8):
    fw=open(rightCateFile,'w')
    srcDir='used_news'
    srcClassList=os.listdir(srcDir)
    for i in range(len(srcClassList)):
        srcClassDir=srcDir+'/'+srcClassList[i]
        srcNewsList=os.listdir(srcClassDir)
        m=len(srcNewsList) #新闻类中所包含的新闻文档的数目
        testBeginIndex=fold*(m*(1-trainDataProp)) #测试数据的起始索引
        testEndIndex=(fold+1)*(m*(1-trainDataProp)) #测试数据的结束索引
        for j in range(m): #遍历新闻类下的所有新闻文档
            if (j>=testBeginIndex) and (j<testEndIndex):
                #记录标注：文档id(文档名_所属类) 所属类
                fw.write('%s %s\n' % (srcNewsList[j]+'_'+srcClassList[i], \
                                      srcClassList[i]))
                targetClassDir='TestData'+str(fold)+'/'+srcClassList[i]
            else:
                targetClassDir='TrainData'+str(fold)+'/'+srcClassList[i]
            if os.path.exists(targetClassDir)==False:
                os.makedirs(targetClassDir)
            nf=open(targetClassDir+'/'+srcNewsList[j],'w')
            lineList=open(srcClassDir+'/'+srcNewsList[j],'rb').readlines()
            for line in lineList:
                line=line.decode('utf-8','ignore')
                nf.write('%s\n' % line.strip('\n'))
            nf.close()
    fw.close()
```

由于20news-18828中含有重名文档，这里利用newsName_newsClass作为文档标识符

## （三）朴素贝叶斯分类器

朴素贝叶斯的实现主要包括以下几个函数：

| 1. 函数 trainNB |
|---|
| 训练朴素贝叶斯分类器：对训练样本数据进行统计，得到类 cate 下单词 word 出现的次数、每个新闻类中单词总数、训练数据中不重复的单词总数。 |

```
def trainNB(trainDir):
    cateWordCount={}
    cateWordNum={}
    vocab=set()
    classList=os.listdir(trainDir)
    for i in range(len(classList)):
        count=0 #记录每个新闻类中单词总数
        classDir=trainDir+'/'+classList[i] #类目录
        newsList=os.listdir(classDir)
        for j in range(len(newsList)):
            newsDir=classDir+'/'+newsList[j]
            lines=open(newsDir,'rb').readlines()
            for line in lines:
```

```
                line=line.decode('utf-8','ignore')
                count+=1
                word=line.strip('\n')
                vocab.add(word) #记录训练数据中不重复词汇
                key=classList[i]+'_'+word
                cateWordCount[key]=cateWordCount.get(key,0)+1 #记录每个类中每个单词
        cateWordNum[classList[i]]=count
    vocabNum=len(vocab)
    return cateWordCount,cateWordNum,vocabNum
```

## 2. 函数 calCateProb

计算测试文档属于某个类的概率：

p(cate|doc)=p(w1,w2,...|cate)*p(cate)

多项式模型+平滑技术+取对数（防止下溢）：

p(word|cate)=(类 cate 下单词 word 出现的次数+1)/(类 cate 下单词总数+训练数据中不重复的单词总数)

p(cate)=类 cate 下单词总数/训练数据中单词总数

$$p(cate \mid doc) = \sum_{i=1}^{n} \log(p(w_i \mid cate)) + \log(p(cate))$$

```
def calCateProb(k,testNewsWords,cateWordCount,cateWordNum,totalNum,vocabNum):
    prob=0
    wordNumInCate=cateWordNum[k] #新闻类k中单词总数
    for i in range(len(testNewsWords)):
        key=k+'_'+testNewsWords[i]
        if key in cateWordCount:
            wordCountInCate=cateWordCount[key]
        else:
            wordCountInCate=0.0
        xcProb=np.log((wordCountInCate+1)/(wordNumInCate+vocabNum))
        prob=prob+xcProb
    res=prob+np.log(wordNumInCate)-np.log(totalNum)
    return res
```

## 3. 函数 classifyNB

朴素贝叶斯分类器通过比较 $p(cate_i \mid doc)$，$i = 1,2,...,20$ 对每个测试文档进行分类，将分类结果记录在特定文件中。

```
#朴素贝叶斯对测试文档进行分类
def classifyNB(trainDir,testDir,resultCateFile):
    fw=open(resultCateFile,'w')
    #训练分类器
    cateWordCount,cateWordNum,vocabNum=trainNB(trainDir)
    #得到训练数据单词总数
    totalNum=sum(cateWordNum.values())
    #对测试文档做分类
    testClassList=os.listdir(testDir)
    for i in range(len(testClassList)):
        testClassDir=testDir+'/'+testClassList[i]
        testNewsList=os.listdir(testClassDir)
```

```
        for j in range(len(testNewsList)):
            testNewsWords=[] #测试文档的单词列表
            testNewsDir=testClassDir+'/'+testNewsList[j]
            lines=open(testNewsDir,'rb').readlines()
            for line in lines:
                line=line.decode('utf-8','ignore')
                word=line.strip('\n')
                testNewsWords.append(word)
            maxP=0.0
            trainClassList=os.listdir(trainDir)
            for k in range(len(trainClassList)):
                p=calCateProb(trainClassList[k],testNewsWords,cateWordCount, \
                              cateWordNum,totalNum,vocabNum)
                if k==0:
                    maxP=p
                    bestCate=trainClassList[k]
                    continue
                if p>maxP:
                    maxP=p
                    bestCate=trainClassList[k]
            fw.write('%s %s\n' % (testNewsList[j]+'_'+testClassList[i], \
                                  bestCate))
    fw.close()
```

4. 函数 errorRate

依据标注测试样本文件与朴素贝叶斯分类结果文件，计算机朴素贝叶斯分类器的错误率。

```
def errorRate(rightCateFile,resultCateFile):
    rightCateDict={}
    resultCateDict={}
    errorCount=0.0
    for line in open(rightCateFile,'rb').readlines():
        line=line.decode('utf-8','ignore')
        (newsID,cate)=line.strip('\n').split()
        rightCateDict[newsID]=cate
    for line in open(resultCateFile,'rb').readlines():
        line=line.decode('utf-8','ignore')
        (newsID,cate)=line.strip('\n').split()
        resultCateDict[newsID]=cate
    for key in rightCateDict.keys():
        #输出分类结果
        print('新闻ID：'+key)
        print('朴素贝叶斯分类：'+resultCateDict[key])
        print('新闻真实所属类：'+rightCateDict[key])
        if rightCateDict[key]!=resultCateDict[key]:
            errorCount+=1.0
    errorRate=errorCount/len(rightCateDict)
    print('error rate: %f' % (errorRate))
    return errorRate
```

**（四）5 折交叉验证**

将数据分为五折，选择其中一折作为测试数据，其余作为训练数据计算错误率，取五次错误率的平均值，作为最终朴素贝叶斯分类器的错误率。包括以下几个过程：

1. 生成五次实验的测试数据、训练数据、标注文件；

2. 朴素贝斯分类器对五次实验的测试数据进行分类；
3. 计算并记录五次实验的错误率；
4. 绘制条形图：可视化每次实验的错误率；
5. 计算五次实验的平均错误率；

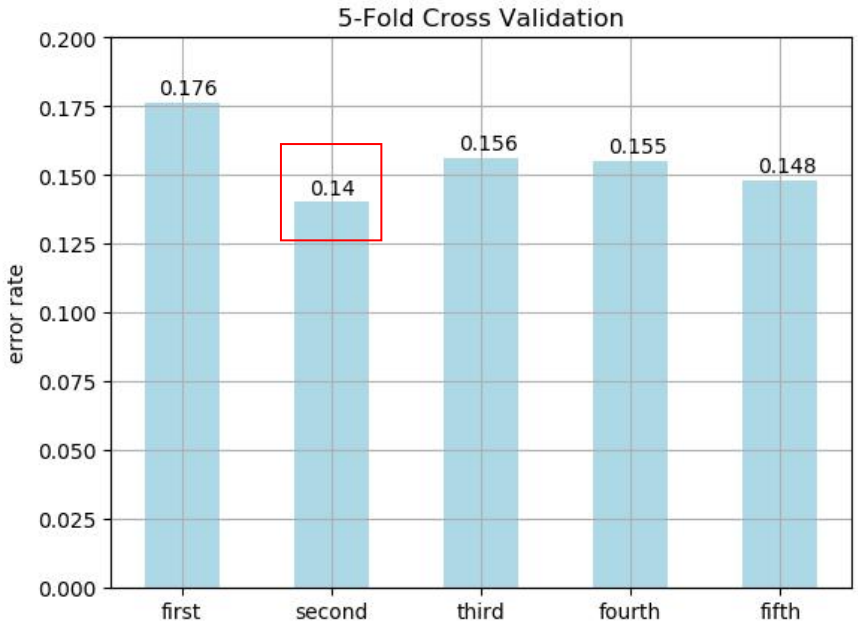
## 五、实验结果

### （一）随机划分数据（20%测试数据、80%训练数据）

```
新闻ID：83528_talk.religion.misc
朴素贝叶斯分类：talk.religion.misc
新闻真实所属类：talk.religion.misc
新闻ID：83529_talk.religion.misc
朴素贝叶斯分类：soc.religion.christian
新闻真实所属类：talk.religion.misc
新闻ID：83535_talk.religion.misc
朴素贝叶斯分类：talk.religion.misc
新闻真实所属类：talk.religion.misc
新闻ID：83544_talk.religion.misc
朴素贝叶斯分类：talk.religion.misc
新闻真实所属类：talk.religion.misc
新闻ID：83547_talk.religion.misc
朴素贝叶斯分类：soc.religion.christian
新闻真实所属类：talk.religion.misc
新闻ID：83558_talk.religion.misc
朴素贝叶斯分类：talk.politics.mideast
新闻真实所属类：talk.religion.misc
error rate: 0.175504
Naive Bayes Finished!
```

### （二）五折交叉验证

1. 条形图可视化五次实验的错误率



2. 输出五次实验的错误率和平均错误率