《数据挖掘技术概论》实验报告

实验二　多项式朴素贝叶斯文本分类

学号 2017141463226

姓名 刘雨欣

一、 实验目的

通过实现多项式分布 Naïve Bayes 的文本分类算法，理解文本分类及朴素贝叶斯分类、生成模型的基本原理和基本的程序实现方法，同时学习文本预处理（包括中文分词）的基本实现方法，自己选择数据集进行实验验证。

程序设计语言：C、C++、Java 或 Python

二、 算法描述和分析

朴素贝叶斯法是基于贝叶斯定理与特征条件独立假设的分类方法。对于给定的训练数据集，首先基于特征条件独立假设学习输入/输出的联合概率分布，然后基于此模型，对给定的输入 x，利用贝叶斯定理求出后验概率最大的输出 y。[1]

算法步骤 1：计算先验概率以及条件概率

$$P(Y=c_k)=\frac{\sum_{i=1}^{N} I(y_i=c_k)}{N},\quad k=1,2,\cdots,K$$

$$P(X^{(j)}=a_{jl}\mid Y=c_k)=\frac{\sum_{i=1}^{N} I(x_i^{(j)}=a_{jl},y_i=c_k)}{\sum_{i=1}^{N} I(y_i=c_k)}$$

$$j=1,2,\cdots,n;\quad l=1,2,\cdots,S_j;\quad k=1,2,\cdots,K$$

算法步骤 2：对于给定的实例$x=(x^{(1)},x^{(2)},\cdots,x^{(n)})^T$,计算后验概率：

$$P(Y=c_k)\prod_{j=1}^{n} P(X^{(j)}=x^{(j)}\mid Y=c_k),\quad k=1,2,\cdots,K$$

算法步骤 3：确定实例 x 的类

$$y=\arg\max_{c_k} P(Y=c_k)\prod_{j=1}^{n} P(X^{(j)}=x^{(j)}\mid Y=c_k)$$

考虑到可能会出现概率值为 0 的情况，会影响到后验概率的计算，于是采用一些平滑措施，条件概率的计算为：

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

其中S_j为类别的数目。当$\lambda = 1$时，即为本次实验所使用的拉普拉斯平滑。同样，先验概率的计算为：

$$P_\lambda(Y=c_k)=\frac{\sum_{i=1}^{N} I(y_i=c_k)+\lambda}{N+K\lambda}$$

当面临着文本处理任务时，考虑到使用朴素贝叶斯算法，词的概率作为我们训练分类器的特征，于是我们选择使用词袋模型来进行表达。词袋模型不考虑语料的语法和上下文关系，每个词都是独立的。

三、 程序实现技术技巧的介绍和分析

本次实验将在 Jupyter 环境下使用 python 3 编码实现多项式朴素贝叶斯文本分类器的构建，并在所选的数据集上进行训练和验证。程序实现将分为四部分，分别是数据预处理、词袋模型生成、训练和测试。

在数据预处理阶段，我们将把数据集 txt 文档中所有非中文字符去除，包括空格、换行符、标点符号等。然后使用 jieba 分词，提取出每一条数据中所有中文词。数据处理的同时，我们将进行词袋模型的生成，整理出一个包含所有中文词的词表，并且生成每条数据的词向量，以供后续的文本分类实现。数据初始化函数 initial_data 将返回词袋模型的词表、所有词向量组成的训练矩阵和对应的标签列表。该部分的主要代码如下：

```
"""
数据集预处理并根据数据集生成词库
jieba 分词
"""
#去掉数据集中的非中文字符
def clean_str(string):
    string = re.sub(r"[^\u4e00-\u9fff]", " ", string)
    string = re.sub(r"\s{2,}", " ", string)
    return string.strip()

#深入数据集文件夹，进行数据清洗同时生成词库和训练集词数组
def initial_data(vocabBase0,comments0,labels0,pathroot):
    files = os.listdir(pathroot)
    vocabBase = vocabBase0
    comments = comments0
    labels = labels0
    for file in files:
        if os.path.isdir(pathroot+"/"+file):
            vocabBase,comments,labels=
initial_data(vocabBase,comments,labels,pathroot+"/"+file)
        else:
            print('\r',"                                                    ",end='')
            print('\r',"initial data:"+str(len(comments))+"/3800",end='')
```

```
            if file[0] == 'n':
                labels.append(0)
            else:
                labels.append(1)
            comment = ''
            f = codecs.open(pathroot+'/'+file,'r','utf-8',errors='ignore')
            for line in f:
                line = clean_str(line)
                comment += line
            f.close()
            comment=[word for word in jieba.cut(comment) if word.strip() != '']
            comments.append(comment) #使用原始的数据集词数组，未去重
             vocabBase = vocabBase | set(comment) #set 有去重功能，且排序
    return vocabBase,comments,labels
```

在数据处理完后，将进行朴素贝叶斯算法的训练过程，操作初始化后的训练矩阵计算每个词的类条件概率和正面、负面两个类别的先验概率。此处我们为避免 0 概率的出现，将进行拉普拉斯平滑，类条件概率的分子初始化为 1，分母初始化为类别数 2，最后返回整个数据集的一个类先验概率（另一个先验概率即为 1 减此概率）和包含所有词的类条件概率的两个向量（一个类对应一个条件概率向量）。主要实现代码如下：

```
#训练函数
def trainNB(commentMatrix,labelList):
    numComments = len(commentMatrix)
    sizeVocabBase = len(commentMatrix[0])
    #求正面评论出现的总概率
    pPos = sum(labelList)/float(numComments)

    #初始化词向量，统计各词在两类评论中出现的总次数
    ####使用拉普拉斯平滑，分子初始化为1，分母初始化为分类数
    p0Num = ones(sizeVocabBase)
    p1Num = ones(sizeVocabBase)
    p0Denom = 2.0
    p1Denom = 2.0
    for i in range(numComments):
        print('\r','                                                    ',end='')
        print('\r',"NB: "+str(i)+"/"+str(numComments)+" ",end='')
        if label_list[i] == 1:
            p1Num += commentMatrix[i]          #出现的词次数+1
            p1Denom += sum(commentMatrix[i])#分母加此邮件总词数
        else:
            p0Num += commentMatrix[i]
            p0Denom += sum(commentMatrix[i])
```

```
    p1Vec = p1Num / p1Denom
    p0Vec = p0Num / p0Denom
    return p0Vec,p1Vec,pPos
```

在进行数据的分类测试时，我们将计算每个输入词向量属于每一类的后验概率，并将其归为概率大的那一类。在计算后验概率时，需要将测试数据中每个词属于某类的类条件概率进行连续相乘，为了避免浮点数太小导致下溢，将乘法转为 log 的加法。主要代码实现如下：

```
def classifyNB(inputVec,p0Vec,p1Vec,pPos):
    L0 = inputVec*p0Vec
    L1 = inputVec*p1Vec
    #手动避免 0 概率
    i = 0
    for label in L0:
        if label == 0:
            L0[i] = 1
        i += 1
    i = 0
    for label in L1:
        if label == 0:
            L1[i] = 1
        i += 1
    p1 = sum(log(L1)) + log(pPos)
    p0 = sum(log(L0)) + log(1.0-pPos)
    if p1 > p0:
        return 1
    else:
        return 0
```

最后对于测试集中的每一条测试数据，我们都使用模型进行分类，并比较预测分类和预期分类是否相等，并以此计算模型的分类准确度。

四、 实验数据和实验方法

实验数据集 ChnSentiCorp 是一个中文酒店评论数据集，共有 4000 条酒店评论，评论只有正面和负面两个类别，因此我们将解决的是二分类问题。其中正面评论 2000 条，负面评论 2000 条，不会出现数据不平衡的问题。

在实验中，我们随机选取 3800 数据作为训练集，其中正面数据 1900 条，负面数据 1900 条，选取 200 条数据作为测试集，正面 100 条，负面 100 条。训练集与测试集的比例为 19：1。我们首先利用词袋模型对训练集进行训练，获得算法必要参数如先验概率和类条件概率，再利用多项式朴素贝叶斯算法对测试数据的每一类后验概率进行计算，将其归为概率更大的一类。最后比较预测分类与预期分类是否相同，以此来计算分类模型的准确度。

五、 实验结果、结果分析和实验结论

在完成训练过程和测试过程后，我们得到本实验的多项式朴素贝叶斯文本分类模型的准确度为 85.5%，准确度较高。

在检查了分类错误的测试数据后，发现这些评论大多为正面和负面词汇参杂的语料，分类模型在这些噪声较大的评论上的表现就较差。此外，中文文本分类对于分词的效果要求很高，虽然直接使用了已经很成熟的 jieba 分词来对数据集进行处理，但在某些词汇上仍出现了错误。同时，本次实验在文本预处理的阶段，没有去除停用词，对实验结果也产生了一定影响。总之，整个实验流程细节还有很多值得改进的地方，所得的文本分类模型也有很大的优化空间。

六、 小结

在完成了本次多项式朴素贝叶斯文本分类模型后，我学习到了文本分类以及朴素贝叶斯分类的基本原理和基本程序实现方法，同时理解了语料数据集预处理的基本方法，接触了词袋模型的理论。在动手编程实现朴素贝叶斯文本分类器的过程中，我的实践能力提高了，当看见自己的分类器有了一定准确度的时候，我也获得了极大成就感，并也在机器学习和数据挖掘的探索道路上获得了新的动力。

参考文献

[1] 李航.统计学习方法[M].北京：清华大学出版社，2012

[2] 蒋盛益, 李霞, 郑琪.数据挖掘原理与实践[M].北京：电子工业出版社，2013