華東理工大學

模式识别大作业

题　　目　垃圾短信分类

学　　院　信息科学与工程

专　　业　控制科学与控制工程

组　　员　吕奕

指导教师　赵海涛

完成日期：　**2018** 年 **10** 月 **25** 日

模式识别作业报告——垃圾短信分类

本次大作业针对一个简单的数据集对朴素贝叶斯的基本原理做一个简单的理解，通过实验来巩固所学内容。

一、题目描述

本题提供一个文本格式的数据集,它包括 5572 条英文短信，每条短信内容由几个长短不一的句子组成。其中，每条短信后面都标注好了是否为垃圾短信，垃圾短信用 1 表示，非垃圾短信用 0 表示。通过该训练集训练出一个分类器，预测短信内容是否为垃圾短信。

二、分类问题

根据题目要求我们可以知道，本题是一个分类问题。从数学角度来说，分类问题可做如下定义：

已知集合：$C = \{y_1, y_2, \dots, y_n\}$和$I = \{x_1, x_2, \dots, x_m, \dots\}$，确定映射规则$y = f(x)$，使得任意$x_i \in I$有且仅有一个$y_i \in C$使得$y_i = f(x_i)$成立。

其中 C 叫做类别集合，其中每一个元素是一个类别，而 I 叫做项集合，其中每一个元素是一个待分类项,f 叫做分类器。分类算法的任务就是构造分类器 f。

分类与回归很容易混淆，要注意分类与回归问题的区别在于输出类型和目的的不同，表 2-1 比较清楚地说明了分类与回归的差异。

表 2-1

特性	分类（监督学习）	回归
输出类型	离散数据	连续数据
目的	寻找决策边界	寻找最优拟合
评价方法	精度、混淆矩阵	SSE（sum of square errors）、拟合优度

三、贝叶斯分类

3.1 贝叶斯定理

通常，事件 A 在事件 B 的条件下的概率，与事件 B 在事件 A 的条件下的概率是不一样的；然而，这两者是有确定的关系，贝叶斯法则就是这种关系的陈述。

贝叶斯法则又被称为贝叶斯定理、贝叶斯规则，是指概率统计中的应用所观察到的现象对有关概率分布的主观判断（即先验概率）进行修正的标准方法。当分析样本大到接近总体数时，样本中事件发生的概率将接近于总体中事件发生的概率。

贝叶斯定理的具体表现形式为：$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$

换个表达形式就是$P(\text{类别}|\text{特征}) = \frac{P(\text{特征}|\text{类别})P(\text{类别})}{P(\text{特征})}$

此公式表示两个互换的条件概率之间的关系，他们通过联合概率关联起来，这样使得我们知道$P(B|A)$的情况下去计算$P(A|B)$成为了可能，而我们的贝叶斯模型便是通过贝叶斯准则去计算某个样本在不同类别条件下的条件概率并取具有最大条件概率的那个类型作为分类的预测结果。

3.2 两个基本概念

贝叶斯统计中的两个基本概念是先验分布和后验分布：

1、先验分布。总体分布参数 θ 的一个概率分布。贝叶斯学派的根本观点，是认为在关于总体分布参数 θ 的任何统计推断问题中，除了使用样本所提供的信息外，还必须规定一个先验分布，它是在进行统计推断时不可缺少的一个要素。他们认为先验分布不必有客观的依据，可以部分地或完全地基于主观信念。

2、后验分布。根据样本分布和未知参数的先验分布，用概率论中求条件概率分布的方法，求出的在样本已知下，未知参数的条件分布。因为这个分布是在抽样以后才得到的，故称为后验分布。贝叶斯推断方法的关键是任何推断都必须且只须根据后验分布，而不能再涉及样本分布。

3.3 朴素贝叶斯分类原理及流程

朴素贝叶斯的思想基础是这样的：对于给出的待分类项，求解在此项出现的条件下各个类别出现的概率，哪个最大，就认为此待分类项属于哪个类别。

如果给定训练数据集（x,y），其中每个样本 x 都包括 n 维特征，即$x = (x_1, x_2, x_3, \cdots, x_n)$，类标记集合含有 k 种类别，即 y=$(y_1, y_2, y_3, \cdots, y_n)$。

如果现在来了一个新样本 x，我们要怎么判断它的类别？从概率的角度来看，这个问题就是给定 x，它属于哪个类别的概率最大。那么问题就转化为求解 $P(y_1|x), P(y_2|x), \ldots, P(y_k|x)$ 中最大的那个，即求后验概率最大的输出：$\arg\max_k P(y_k|x)$。

由贝叶斯定理可以求得$P(y_k|x)$：

$$P(y_k|x) = \frac{P(x|y_k)P(y_k)}{P(x)}$$

根据全概率公式，可以进一步地分解上式中的分母：

$$P(y_k|x) = \frac{P(x|y_k)P(y_k)}{\sum_k P(x|y_k)P(y_k)}$$

分子中的$P(y_k)$是先验概率，根据训练集就可以简单地计算出来。而条件概率$P(x|y_k) = P(x_1, x_2, x_3, \cdots, x_n|y_k)$，它的参数规模是指数数量级别的，假设第 i 维特征x_i可取值的个数有S_i个，类别取值个数为 k 个，那么参数个数为：$k\prod_{i=1}^{n} S_i$。这显然不可行。针对这个问题，朴素贝叶斯算法对条件概率分布做出了独立性的假设，通俗地讲就是说假设各个维度的特征$x_1, x_2, x_3, \cdots, x_n$互相独立，在这个假设的前提上，条件概率可以转化为：

$$P(x|y_k) = P(x_1, x_2, x_3, \cdots, x_n|y_k) = \Pi_{i=1}^{n} P(x_i|y_k)$$

这样，参数规模就降到$\Sigma_{i=1}^{n} S_i k$。

以上就是针对条件概率所作出的特征条件独立性假设，至此，先验概率$P(y_k)$和条件概率$P(x|y_k)$的求解问题就都解决了，我们就可以求解我们所要的后验概率$P(y_k|x)$了。

继续上面关于$P(y_k|x)$的推导

$$P(y_k|x) = \frac{P(y_k)\Pi_{i=1}^{n} P(x_i|y_k)}{\Sigma_k P(y_k)\Pi_{i=1}^{n} P(x_i|y_k)}$$

于是朴素贝叶斯分类器可表示为：

$$f(x) = \arg\max_{y_k} P(y_k|x) = \arg\max_{y_k} \frac{P(y_k)\Pi_{i=1}^{n} P(x_i|y_k)}{\Sigma_k P(y_k)\Pi_{i=1}^{n} P(x_i|y_k)}$$

因为对所有的y_k，上式中的分母的值都是一样的，所以可以忽略分母部分，朴素贝叶斯分类器最终表示为：

$$f(x) = \arg\max_{y_k} P(y_k)\, \Pi_{i=1}^{n} P(x_i|y_k)$$

其正式定义如下：

（1） 设$x = \{a_1, a_2, \ldots, a_m\}$为一个待分类项，而每个 a 为 x 的一个特征属性

（2） 有类别集合$C = \{y_1, y_2, \cdots, y_n\}$。

（3） 计算 $P(y_1|x), P(y_2|x), \cdots, P(y_n|x)$。

（4） 如果 $P(y_k|x) = max\{P(y_1|x), P(y_2|x), \ldots, P(y_n|x)\}$，则$x \in y_k$。

那么现在的关键就是如何计算第三步的各个条件概率：

（1） 找到一个已知分类的待分类项集合，这个集合叫做训练样本集。

（2） 统计得到在各个类别下各个特征属性的条件概率估计，即

（3） $P(a_1|y_1), P(a_2|y_1), \ldots, P(a_m|y_1); P(a_1|y_2), P(a_2|y_2), \ldots, P(a_m|y_2); \ldots;$

$$P(a_1|y_n), P(a_2|y_n), \ldots, P(a_m|y_n).$$

（4） 如果各个特征属性是条件独立的，则根据贝叶斯定理有如下推导：

$$P(y_i|x) = \frac{P(x|y_i)P(y_i)}{P(x)}$$

因为分母对于所有类别为常数，因为我们只要将分子最大化皆可。又因为各特征属性是条件独立的，所以有：

$$P(x|y_i)P(y_i) = P(a_1|y_i)P(a_2|y_i) \ldots P(a_m|y_i)P(y_i) = P(y_i)\prod_{j=1}^{m} P(a_j|y_i)$$

可以看到，整个朴素贝叶斯分类分为三个阶段：

第一阶段——准备工作阶段，这个阶段的任务是为朴素贝叶斯分类做必要的准备，主要工作是根据具体情况确定特征属性，并对每个特征属性进行适当划分，然后由人工对一部分待分类项进行分类，形成训练样本集合。这一阶段的输入是所有待分类数据，输出是特征属性和训练样本。这一阶段是整个朴素贝叶斯分类中唯一需要人工完成的阶段，其质量对整个过程将有重要影响，分类器的质量很大程度上由特征属性、特征属性划分及训练样本质量决定。

第二阶段——分类器训练阶段，这个阶段的任务就是生成分类器，主要工作是计算每个类别在训练样本中的出现频率及每个特征属性划分对每个类别的条件概率估计，并将结果记录。其输入是特征属性和训练样本，输出是分类器。这一阶段是机械性阶段，根据前面讨论的公式可以由程序自动计算完成。

第三阶段——应用阶段。这个阶段的任务是使用分类器对待分类项进行分类，其输入是分类器和待分类项，输出是待分类项与类别的映射关系。这一阶段也是机械性阶段，由程序完成。

可以用流程图 3-1 来表示

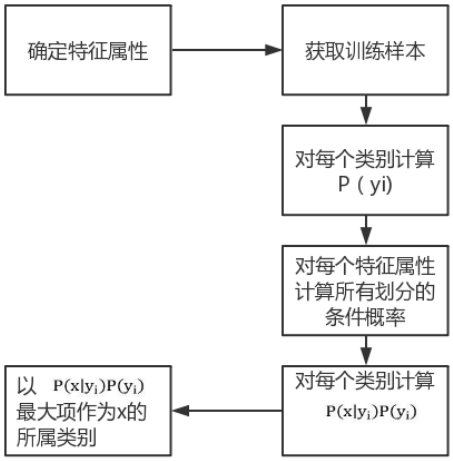

图 3-1　朴素贝叶斯分类流程图

四、实验过程分析

4.1 算法思想

我们想判断一个短信是不是垃圾短信，需要知道这个短信中词的分布，同时，我们还需要知道垃圾短信中某些词的出现频率是多少，这就可以利用朴素贝叶斯定理得到。根据概率大小判断是否符合垃圾短信特点区分垃圾与非垃圾短信。

在朴素贝叶斯分类器中有一个假设就是每个特征同等重要。虽然在现实中这种想法有些不太合理，毕竟语言都是要讲究上下文内容的，朴素贝叶斯的朴素就体现在简单看待问题，将复杂问题简单化。

4.2 算法内容分析

4.2.1 读取短信内容 readTxt()

算法中采用定义一个单独的程序对短信内容进行读取，并将文本和分类标签区分开来，具体代码如下：

```
#读取短信数据
def readTxt():
    #arraylines = csv.reader(open('D:/研一课程/模式识别/lintcode数据/mess
    text=open(r'D:\\message.txt',encoding='gb18030',errors='ignore')
    arraylines=text.readlines()
    docList=[]
    classList=[]
    for line in arraylines:
        docList.append(textParse(line[:-3]))
        try:
            classList.append(int(line[-2]))
        except:
            classList.append(1)
    return docList,classList
```

4.2.2 分词 textParse(bigString)

把邮件进行分词，就是将邮件内容划分成一个个单词的形式，算法中采用正则表达式对长的字符串进行分词操作。正则表达式是对字符串(包括普通字符(例如，a 到 z 之间的字母)和特殊字符(称为"元字符"))操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个"规则字符串"，这个"规则字符串"用来表达对字符串的一种过滤逻辑。正则表达式是一种文本模式，模式描述在搜索文本时要匹配的一个或多个字符串。具体实现代码如下：

```
#文档分词函数
#将单词长度小于等于2的过滤掉，并且将其变成小写字母。
def textParse(bigString):
    listOfTokens = re.split(r'\W*' , bigString)  # re.split，支持正则及多个字符切割
    return [tok.lower() for tok in listOfTokens if len(tok) > 2]
```

其中，re.split(r'\W*' , bigString),表示以除了数字，字母和下划线的符合进行划分，return 的是一个列表推到式生成的列表，其中将单词长度小于等于 2 的过滤掉，并且将其变成小写字母。

4.2.3 生成词汇表函数 createVocabList(dataSet)

将所有的邮件进行分词后生成一个 dataSet，然后生成一个词汇表，这个词汇表是一个集合，即每个单词只出现一次，词汇表是一个列表形式如：

[“cute”,” love”,” help”,” garbage”,” quit”…]

具体实现代码如下：

```
#创建一个带有所有单词的词汇列表
def createVocabList(dataSet):
    vocabSet = set([])
    for document in dataSet:
        vocabSet = vocabSet | set(document)
    return list(vocabSet)
```

4.2.4 生成词向量函数 bagOfWords2VecMN(vocabList, inputSet)

每一封邮件的词汇都存在了词汇表中，因此可以将每一封邮件生成一个词向量，存在几个则为几，不存在为 0，例如：[“love”，“garbage”],则他的词向量为 [0,1,0,1,0,…],其位置是与词汇表所对应的，因此词向量的维度与词汇表相同。

生成词向量函数中运用了文档词袋模型，使用数组代替集合数据结构，可以保存词汇频率信息。词袋模型是在自然语言处理和信息检索中的一种简单假设。在这种模型中，文本（段落或者文档）被看作是无序的词汇集合，忽略语法甚至是单词的顺序。

一条短信可以看作是无序的词汇集合，这些词汇从两种概率分布中被选出。一个代表垃圾短信，一个代表非垃圾短信。这里假设有两个装满词汇的袋子。一个袋子里面装的是在垃圾短信中发现的词汇。另一个袋子装的是非垃圾短信中的词汇。尽管给定的一个词可能出现在两个袋子中，这时就要看到底哪个袋子的可能性更高。

贝叶斯分类器假设邮件来自于两个词袋中的一个，使用贝叶斯概率来决定那个袋子更可能产生这样的一条短信。具体实现代码如下：

```
#文档词袋模型
#vocablist为词汇表，inputSet为输入的邮件
def bagOfWords2VecMN(vocabList, inputSet):
    returnVec = [0]*len(vocabList)#大小与词向量相同
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] += 1#查找单词的索引
    return returnVec
```

4.2.5 训练函数 trainNB0(trainMatrix,trainCatergory)

训练函数是算法的核心，其中要计算两部分内容：

1. 先验概率

2. $P(\omega_i|c_1)$，$P(\omega_i|c_0)$，这里 0 表示正常短信，1 表示垃圾短信。

这里需要重点的理解是如何计算第二步的概率，例如，$P(\omega_i|c_1)$表示在垃圾短信的条件下第 i 个特征的概率，首先将所有的类别为 1 的词向量相加，可以得到每个特征的个数，因此在除以在类别 1 的单词总数就是在垃圾邮件中每个单词的概率了。注意，这里所说的特征即词汇的每一个单词。具体代码如下：

```
#训练函数
#trainMat是训练样本的词向量，可以看做一个矩阵，他的每一行为一个短信的词向量
#trainGategory为与trainMat对应的类别，值为0表示正常，1表示垃圾短信
def trainNB0(trainMatrix,trainCatergory):
    numTrainDoc = len(trainMatrix)
    numWords = len(trainMatrix[0])#词汇表长度
    pAbusive = sum(trainCatergory)/float(numTrainDoc)
    #防止多个概率的成绩当中的一个为0
    p0Num = ones(numWords)
    p1Num = ones(numWords)
    p0Denom = 2.0
    p1Denom = 2.0
    for i in range(numTrainDoc):
        if trainCatergory[i] == 1:
            p1Num +=trainMatrix[i]#统计垃圾短信类中每个单词的个数
            p1Denom += sum(trainMatrix[i])#计算垃圾短信中的单词总数
        else:
            p0Num +=trainMatrix[i]
            p0Denom += sum(trainMatrix[i])
    p1Vect = log(p1Num/p1Denom)#计算垃圾短信中每个单词概率
                                #出于精度的考虑，否则很可能到限归零
    p0Vect = log(p0Num/p0Denom)
    return p0Vect, p1Vect, pAbusive
```

4.2.6 分类函数 classifyNB(vec2Classify, p0Vec, p1Vec, pClass1)

分别计算这个向量属于垃圾短信和正常短信的后验概率，并判断两个集合中哪个的概率高。具体实现代码如下：

```
#分类函数
def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):
    p1 = sum(vec2Classify * p1Vec) + log(pClass1)
    p0 = sum(vec2Classify * p0Vec) + log(1.0 - pClass1)
    if p1 > p0:
        return 1
    else:
        return 0
```

4.2.7 测试函数 spamTest()

这里首先将 5572 封邮件读进 docList 列表中，然后生成一个词汇表包含所有的单词，接下来使用交叉验证，随机的选择 1000 个样本进行测试，剩余样本进行训练。具体代码如下：

```
def spamTest():
    docList, classList = readTxt()
    vocabList = createVocabList(docList)
    #lendoc = len(docList)
    trainingSet = range(5572)
    testSet = []
    for i in range(1000):
        randindex = int(random.uniform(0,5572))
        testSet.append(trainingSet[randindex])
    trainMat = []
    for j in range(5572):
      #trainMat.append(Wordtovector(vocabList, docList[j]))
      #trainMat.append(setOfWords2Vec(vocabList, docList[j]))
      trainMat.append(bagOfWords2VecMN(vocabList, docList[j]))

    p0V,p1V,pSpam = trainNB0(array(trainMat),array(classList))
    errorCount = 0
    for docIndex in testSet:        #classify the remaining items
        wordVector = bagOfWords2VecMN(vocabList, docList[docIndex])
        #wordVector = setOfWords2Vec(vocabList, docList[docIndex])
        #wordVector = Wordtovector(vocabList, docList[i])
        if classifyNB(array(wordVector),p0V,p1V,pSpam) != classList[docIndex]:
            errorCount += 1
           # print("classification error",docList[i])
    print('the error rate is: ',float(errorCount)/len(testSet))
    #return vocabList,fullText
```

由于采取交叉验证的方式，随机过程会导致每次的结果不尽相同，因此每次的错误率都有所差别，进行了十次实验结果如下表所示：

试验次数	1	2	3	4	5	6	7	8	9	10
错误率	0.014	0.018	0.019	0.010	0.013	0.02	0.017	0.019	0.016	0.012

错误率基本保持在 0.01-0.02 之间，证明分类算法有效。

五、作业总结

本文主要是用朴素贝叶斯模型处理垃圾短信分类问题。

不同于其他分类器，朴素贝叶斯是一种基于概率理论的分类算法。

朴素贝叶斯特征之间的相互独立，虽然这种假设不符合实际，显得有些粗鲁，然而事实证明这种分类算法有一定的正确性。

在具体算法实施过程中，要考虑很多实际问题：比如因为“下溢”问题，需要对概率乘积取对数，否则很有可能到限归零；对数据的预处理，剔除各种干扰等等。

第一次接触 Python 的程序编程，在作业的完成过程中遇到不少难题，可以说是相当煎熬的一个星期了。我认为老师布置作业的主要目的是为了让大家对上课所讲述的方法做一个更为深刻的理解，应该把重点放在算法本身，而我花费了大量的时间用来学习 Python 的使用，程序的编写，没有把握好重心。网络上的大部分例程都是直接调用已经调好的包，程序虽然简单清晰，但是却不太容易理解，在选择了几个比较复杂的程序但是没能调试成功后，最终确定了用朴素贝叶

斯模型来做垃圾短信的分类。

通过此次大作业再一次意识到自己编程能力的不足，在今后的学习过程中应该格外注意编程能力的提高。虽然过程是煎熬的，但是在学习的过程中积累了宝贵的学习资源，对 Python 编程有了初步认识，对贝叶斯定理、朴素贝叶斯模型有了进一步的理解；意识到自己能力的不足，明确了今后努力的方向；对模式识别的兴趣更为浓郁，这些也算是此次大作业的收获吧。

最后感谢赵教练的督促指导。

附：文件说明

本次附件一共包含有：

1 模式识别大作业报告；

2 最终的 Python 实现程序源码：naivebayes.py

3 程序所用数据集 messages.txt