机器学习——朴素贝叶斯算法

贝叶斯算法相对还是比较容易理解的，同时它的学习效率高，对于一些分类问题是一个很好的解决方案。这里涉及到的数学知识主要是概率论的东西，再仔细一点其实就是贝叶斯公式：

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

其中$P(A)$是先验概率，指明了A事件概率的一个判断，$P(A|B)$是后验概率，指出了B事件发生之后，对A事件的评估，$\frac{P(B|A)}{P(B)}$是可能性函数，使得预估概率更加接近真实概率。而朴素贝叶斯则对条件概率做了条件独立性的假设，例如这样的条件概率可以由以下式子进行拆分：

$$P(X|a)P(a)=P(x_1,x_2,x_3,\dots x_n|a)P(a)$$

在本次的实验中，通过使用朴素贝叶斯进行过滤垃圾邮件，这是个很经典的例子，我们首先需要收集数据，也就是对文本文件进行提取，然后对这些文本文件进行解析，把它解析成词条向量，通过建奶茶词条确保解析的正确性，然后使用我们建立的训练函数进行训练，之后用一个测试函数来计算文档的错误率。

代码：

```
import numpy as np
import random
import re

def creatVocList(dataSet):
    vocaSet=set([])
    for document in dataSet:
        #不断取并集
        vocaSet=vocaSet | set(document)
    return list(vocaSet)

def set2Vec(vocablist,inputSet):
    returnVec=[0]*len(vocablist)
    for word in inputSet:
        if word in vocablist:
            returnVec[vocablist.index(word)]=1
        else:print("the word: %s is not in my vocablist!" %word)
    return returnVec


def trainNB0(trainMatrix,trainCategory):
    numTrainDocs=len(trainMatrix)
    numWords=len(trainMatrix[0])
    pAbusive=sum(trainCategory)/float(numTrainDocs)
    p0Num=np.ones(numWords)
    p1Num=np.ones(numWords)
    p0Denom=2.0
    p1Denom=2.0
    for i in range(numTrainDocs):
        if trainCategory[i]==1:
            p1Num+=trainMatrix[i]
            p1Denom+=sum(trainMatrix[i])
```

```
        else:
            p0Num+=trainMatrix[i]
            p0Denom+=sum(trainMatrix[i])
    p1Vect=np.log(p1Num/p1Denom)
    p0Vect=np.log(p0Num/p0Denom)
    return p0Vect,p1Vect,pAbusive

def classifyNB(vec2Classify,p0Vec,p1Vec,pClass1):
    p1=sum(vec2Classify*p1Vec)+np.log(pClass1)
    p2=sum(vec2Classify*p0Vec)+np.log(1.0-pClass1)
    if p1>p2:
        return 1
    else:
        return 0

def textParse(bigString):
    listOfTokens=re.split(r'\W+',bigString)
    return [tok.lower() for tok in listOfTokens if len(tok)>2]

def spamTest():
    doclist=[]
    classlist=[]
    fullText=[]
    for i in range(1,26):
        wordlist=textParse(open('email/spam/%d.txt' %i ,'r').read())
        doclist.append(wordlist)
        fullText.append(wordlist)
        classlist.append(1)
        wordlist=textParse(open('email/ham/%d.txt' %i ,'r').read())
        doclist.append(wordlist)
        fullText.append(wordlist)
        classlist.append(0)
    vocablist=creatVocList(doclist)
    trainingSet=list(range(50))
    testSet=[]
    for i in range(10):
        randIndex=int(random.uniform(0,len(trainingSet)))
        testSet.append(trainingSet[randIndex])
        del(trainingSet[randIndex])
    trainMat=[]
    trainClass=[]
    for docIndex in trainingSet:
        trainMat.append(set2Vec(vocablist,doclist[docIndex]))
        trainClass.append(classlist[docIndex])
    p0V,p1V,pSpam=trainNB0(np.array(trainMat),np.array(trainClass))
    errorCount=0
    for docIndex in testSet:
        wordVec=set2Vec(vocablist,doclist[docIndex])
        if classifyNB(np.array(wordVec),p0V,p1V,pSpam)!=classlist[docIndex]:
            errorCount+=1
            print("分类错误的测试集：",doclist[docIndex])
    print('错误率：%.2f%%' %(float(errorCount)/len(testSet)*100))

spamTest()
```

> 错误率：0.00%

> 分类错误的测试集： ['yay', 'you', 'both', 'doing', 'fine', 'working', 'mba', 'design', 'strategy', 'cca', 'top', 'art', 'school', 'new', 'program', 'focusing', 'more', 'right', 'brained', 'creative', 'and', 'strategic', 'approach', 'management', 'the', 'way', 'done', 'today']
> 错误率：10.00%

总结：

贝叶斯分类适合多分类的任务，同时适合增量式训练，算法原理简单，但是就是由于这个简单使得它准确率会有一定程度的损失。当然贝叶斯的概率也需要一定的改正，例如拉普拉斯平滑。