**实验二 朴素贝叶斯**

贝叶斯定理如下式：

其中，x，y表示特征变量，表示分类，即表示在特征为x，y的情况下分入类别的概率，因此，结合条件概率和贝叶斯定理，有：

* 如果>，那么分类应当属于类；
* 如果<，那么分类应当属于类；

**当仅仅是为了比较大小的时候，由于分母相同，因此只需要比较分子即可。**

朴素贝叶斯最常见的分类应用是对文档进行分类，因此最常见的特征条件是文档中出现词汇的情况。通常将词汇出现的特征条件用**词向量**表示，由多个数值组成，数值的个数和训练样本集中的词汇表个数一样。因此，上述贝叶斯条件概率可以表示为：

朴素贝叶斯是基于特征条件独立这个假设的。也就是说，我们假设词汇表中各个单词独立出现，不会相互影响，因此可以将展开成独立事件概率相乘的形式，即：

这个假设极大地降低了计算的复杂度，这也是朴素贝叶斯中“朴素”一词的由来。

**实验要求：**以在线社区的留言板为例，运用朴素贝叶斯算法，对文本自动分类。

**实验过程：**

创建一个名为bayes.py的文件，将下面出现的代码及自己编写的代码均放入该文件中。

①准备数据。使用提供的函数：

*def loadDataSet():  
 postingList=[['my', 'dog', 'has', 'flea', 'problems', 'help', 'please'],  
 ['maybe', 'not', 'take', 'him', 'to', 'dog', 'park', 'stupid'],  
 ['my', 'dalmation', 'is', 'so', 'cute', 'I', 'love', 'him'],  
 ['stop', 'posting', 'stupid', 'worthless', 'garbage'],  
 ['mr', 'licks', 'ate', 'my', 'steak', 'how', 'to', 'stop', 'him'],  
 ['quit', 'buying', 'worthless', 'dog', 'food', 'stupid']]  
 classVec=[0,1,0,1,0,1] #1表示侮辱性言论，0表示正常言论  
 return postingList,classVec*

②构建词汇表生成函数：  
 *def createVocabList(dataSet):  
 vocabSet=set([])  
 for document in dataSet:  
 vocabSet=vocabSet|set(document) #取两个集合的并集  
 return list(vocabSet)*

③构建词向量。这里采用的是词集模型，即只需记录每个词是否出现，而不考虑其出现的次数。需要记录词出现的次数的叫词袋模型。

*def setOfWords2Vec(vocabList,inputSet):  
 returnVec=zeros(len(vocabList)) #生成零向量的array  
 for word in inputSet:  
 if word in vocabList:  
 returnVec[vocabList.index(word)]=1 #单词出现则记为1  
 else: print('the word:%s is not in my Vocabulary!'% word)  
 return returnVec #返回全为0和1的向量*

**④根据训练集计算概率**。根据上面的公式：

正如前面分析的那样，我们只需考虑分子即可。

用训练集中，属于类别的样本数量除以总的样本数量即可；

可以根据前面的独立性假设，先分别计算，，等项，再将其结果相乘即可，而的计算公式为：

在实现算法时，需要考虑两个问题：

a.当使用连乘计算时，若某一个词的概率为0，那么最终的结果也会为0，这是不正确的。为防止这种情况，需要将所有词项出现的次数都初始化为1，每一类所有词项数量初始化为2；

b.在连乘时，为防止单项概率过小导致连乘结果下溢，需要对结果求自然对数将其转化为加法，因为。

⑤根据上一步计算出来的概率编写分类器函数。

*def classifyNB(vec2Classify,p0Vec,p1Vec,pClass1):  
 p1=sum(vec2Classify\*p1Vec)+log(pClass1)  
 p0=sum(vec2Classify\*p0Vec)+log(1-pClass1)  
 if p1>p0:  
 return 1  
 else:  
 return 0*

其中，p0Vec，p1Vec，pClass1均为上一步函数的返回值，分别代表公式中的，以及。

⑥编写测试函数。

def testingNB():  
 listPosts,listClasses=loadDataSet()  
 myVocabList=createVocabList(listPosts)  
 trainMat=[]  
 for postinDoc in listPosts:  
 trainMat.append(setOfWords2Vec(myVocabList,postinDoc))  
 p0V,p1V,pAb=trainNB1(trainMat,listClasses)  
 testEntry=['love','my','dalmation']  
 thisDoc=setOfWords2Vec(myVocabList,testEntry)  
 print(testEntry,'classified as:',classifyNB(thisDoc,p0V,p1V,pAb))  
 testEntry=['stupid','garbage']  
 thisDoc=array(setOfWords2Vec(myVocabList,testEntry))  
 print(testEntry,'classified as:',classifyNB(thisDoc,p0V,p1V,pAb))

在python命令提示符中输入以下语句进行测试：

*import bayes*

*bayes.testingNB()*

**\*\*\*延伸\*\*\***

可参考《机器学习实战》一书**4.6**节使用朴素贝叶斯过滤垃圾邮件。注意这一示例使用的是词袋模型。