朴素贝叶斯：

rom numpy import \*  
  
# 过滤网站的恶意留言 侮辱性：1 非侮辱性：0  
# 创建一个实验样本  
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]  
 return postingList, classVec  
  
# 创建一个包含在所有文档中出现的不重复词的列表  
def createVocabList(dataSet):  
 vocabSet = set([]) # 创建一个空集  
 for document in dataSet:  
 vocabSet = vocabSet | set(document) # 创建两个集合的并集  
 return list(vocabSet)  
  
# 将文档词条转换成词向量  
def setOfWords2Vec(vocabList, inputSet):  
 returnVec = [0]\*len(vocabList) # 创建一个其中所含元素都为0的向量  
 for word in inputSet:  
 if word in vocabList:  
 # returnVec[vocabList.index(word)] = 1 # index函数在字符串里找到字符第一次出现的位置 词集模型  
 returnVec[vocabList.index(word)] += 1 # 文档的词袋模型 每个单词可以出现多次  
 else: print("the word: %s is not in my Vocabulary!" % word)  
 return returnVec  
  
# 朴素贝叶斯分类器训练函数 从词向量计算概率  
def trainNB0(trainMatrix, trainCategory):  
 numTrainDocs = len(trainMatrix)  
 numWords = len(trainMatrix[0])  
 pAbusive = sum(trainCategory)/float(numTrainDocs)  
 # p0Num = zeros(numWords); p1Num = zeros(numWords)  
 # p0Denom = 0.0; p1Denom = 0.0  
 p0Num = ones(numWords); # 避免一个概率值为0,最后的乘积也为0  
 p1Num = ones(numWords); # 用来统计两类数据中，各词的词频  
 p0Denom = 2.0 # 用于统计0类中的总数  
 p1Denom = 2.0 # 用于统计1类中的总数  
 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 = p1Num / p1Denom  
 # p0Vect = p0Num / p0Denom  
 p1Vect = log(p1Num / p1Denom) # 在类1中，每个次的发生概率  
 p0Vect = log(p0Num / p0Denom) # 避免下溢出或者浮点数舍入导致的错误 下溢出是由太多很小的数相乘得到的  
 return p0Vect, p1Vect, pAbusive  
  
# 朴素贝叶斯分类器  
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  
  
def testingNB():  
 listOPosts, listClasses = loadDataSet()  
 myVocabList = createVocabList(listOPosts)  
 trainMat = []  
 for postinDoc in listOPosts:  
 trainMat.append(setOfWords2Vec(myVocabList, postinDoc))  
 p0V, p1V, pAb = trainNB0(array(trainMat), array(listClasses))  
 testEntry = ['love','my','dalmation']  
 thisDoc = array(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))  
  
# 调用测试方法  
testingNB()

运行结果：

