人工智能第八次实验报告

朴素贝叶斯朴素贝叶斯

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  。最为广泛的两种分类模型是决策树模型(Decision Tree Model)和朴素贝叶斯模型（Naive Bayesian Model，NBM），本文主要讨论朴素贝叶斯模型。我们称之为“朴素”，是因为整个形式化过程只做最原始、最简单的假设。和决策树模型相比，朴素贝叶斯分类器(Naive Bayes Classifier,或 NBC)发源于古典数学理论，有着坚实的数学基础，以及稳定的分类效率。同时，NBC模型所需估计的参数很少，对缺失数据不太敏感，算法也比较简单。理论上，NBC模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为NBC模型假设属性之间相互独立，这个假设在实际应用中往往是不成立的，这给NBC模型的正确分类带来了一定影响。

源程序:# -\*- coding: UTF-8 -\*-

import numpy as np

from functools import reduce

"""

函数说明:创建实验样本

Parameters:

无

Returns:

postingList - 实验样本切分的词条

classVec - 类别标签向量

Modify:

2018-10-12

"""

def loadDataSet():

postingList=[['my', 'dog', 'has', 'flea', 'problems', 'help', 'please'], #切分的词条

['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] #类别标签向量，1代表侮辱性词汇，0代表不是

return postingList,classVec #返回实验样本切分的词条和类别标签向量

"""

函数说明:将切分的实验样本词条整理成不重复的词条列表，也就是词汇表

Parameters:

dataSet - 整理的样本数据集

Returns:

vocabSet - 返回不重复的词条列表，也就是词汇表

Modify:

2018-10-12

"""

def createVocabList(dataSet):

vocabSet = set([]) #创建一个空的不重复列表

for document in dataSet:

vocabSet = vocabSet | set(document) #取并集

return list(vocabSet)

"""

函数说明:根据vocabList词汇表，将inputSet向量化，向量的每个元素为1或0

Parameters:

vocabList - createVocabList返回的列表

inputSet - 切分的词条列表

Returns:

returnVec - 文档向量,词集模型

Modify:

2018-10-12

"""

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: #遍历每个词条

if word in vocabList: #如果词条存在于词汇表中，则置1

returnVec[vocabList.index(word)] = 1

else: print("the word: %s is not in my Vocabulary!" % word)

return returnVec #返回文档向量

"""

函数说明:朴素贝叶斯分类器训练函数

Parameters:

trainMatrix - 训练文档矩阵，即setOfWords2Vec返回的returnVec构成的矩阵

trainCategory - 训练类别标签向量，即loadDataSet返回的classVec

Returns:

p0Vect - 侮辱类的条件概率数组

p1Vect - 非侮辱类的条件概率数组

pAbusive - 文档属于侮辱类的概率

Modify:

2018-10-12

"""

def trainNB0(trainMatrix,trainCategory):

numTrainDocs = len(trainMatrix) #计算训练的文档数目

numWords = len(trainMatrix[0]) #计算每篇文档的词条数

pAbusive = sum(trainCategory)/float(numTrainDocs) #文档属于侮辱类的概率

p0Num = np.zeros(numWords); p1Num = np.zeros(numWords) #创建numpy.zeros数组,

p0Denom = 0.0; p1Denom = 0.0 #分母初始化为0.0

for i in range(numTrainDocs):

if trainCategory[i] == 1: #统计属于侮辱类的条件概率所需的数据，即P(w0|1),P(w1|1),P(w2|1)···

p1Num += trainMatrix[i]

p1Denom += sum(trainMatrix[i])

else: #统计属于非侮辱类的条件概率所需的数据，即P(w0|0),P(w1|0),P(w2|0)···

p0Num += trainMatrix[i]

p0Denom += sum(trainMatrix[i])

p1Vect = p1Num/p1Denom #相除

p0Vect = p0Num/p0Denom

return p0Vect,p1Vect,pAbusive #返回属于侮辱类的条件概率数组，属于非侮辱类的条件概率数组，文档属于侮辱类的概率

"""

函数说明:朴素贝叶斯分类器分类函数

Parameters:

vec2Classify - 待分类的词条数组

p0Vec - 侮辱类的条件概率数组

p1Vec -非侮辱类的条件概率数组

pClass1 - 文档属于侮辱类的概率

Returns:

0 - 属于非侮辱类

1 - 属于侮辱类

Modify:

2018-10-12

"""

def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):

p1 = reduce(lambda x,y:x\*y, vec2Classify \* p1Vec) \* pClass1 #对应元素相乘

p0 = reduce(lambda x,y:x\*y, vec2Classify \* p0Vec) \* (1.0 - pClass1)

print('p0:',p0)

print('p1:',p1)

if p1 > p0:

return 1

else:

return 0

"""

函数说明:测试朴素贝叶斯分类器

Parameters:

无

Returns:

无

Modify:

2018-10-12

"""

def testingNB():

listOPosts,listClasses = loadDataSet() #创建实验样本

myVocabList = createVocabList(listOPosts) #创建词汇表

trainMat=[]

for postinDoc in listOPosts:

trainMat.append(setOfWords2Vec(myVocabList, postinDoc)) #将实验样本向量化

p0V,p1V,pAb = trainNB0(np.array(trainMat),np.array(listClasses)) #训练朴素贝叶斯分类器

testEntry = ['love', 'my', 'dalmation'] #测试样本1

thisDoc = np.array(setOfWords2Vec(myVocabList, testEntry)) #测试样本向量化

if classifyNB(thisDoc,p0V,p1V,pAb):

print(testEntry,'wuru') #执行分类并打印分类结果

else:

print(testEntry,'not belong insult') #执行分类并打印分类结果

testEntry = ['stupid', 'garbage'] #测试样本2

thisDoc = np.array(setOfWords2Vec(myVocabList, testEntry)) #测试样本向量化

if classifyNB(thisDoc,p0V,p1V,pAb):

print(testEntry,'is belong insult') #执行分类并打印分类结果

else:

print(testEntry,'not belong insult') #执行分类并打印分类结果

if \_\_name\_\_ == '\_\_main\_\_':

testingNB()

运行截图：

