Lab7_homework

2018 年 11 月 3 日

1 实验目的

python 实现朴素贝叶斯分类器及其应用

2 实验环境

- windows 10 64 位
- anaconda3
- jupyter notebook

3 实验步骤

3.1 准备数据, 从文本中构建词向量

3.1.1 分词: 切分文本成词

```
In [1]: mySent= 'This book is the best book on Python or M.L. I have ever laid eyes upon.'

In [2]: mySent.split() # 使用空格切分

Out[2]: ['This',
         'book',
         'is',
         'the',
         'best',
         'book',
         'on',
         'Python',
         'or',
         'M.L.',
```

```
 'I',
 'have',
 'ever',
 'laid',
 'eyes',
 'upon.']
```

```
In [3]: import re
        regEx=re.compile('\\W+') # 使用正则表达式切分
        listOfTokens= regEx.split(mySent)
        listOfTokens
```

```
Out[3]: ['This',
 'book',
 'is',
 'the',
 'best',
 'book',
 'on',
 'Python',
 'or',
 'M',
 'L',
 'I',
 'have',
 'ever',
 'laid',
 'eyes',
 'upon',
 '']
```

```
In [4]: a1 = [tok for tok in listOfTokens if len(tok)>0]
        a1
```

```
Out[4]: ['This',
 'book',
 'is',
 'the',
 'best',
```

```
          'book',
          'on',
          'Python',
          'or',
          'M',
          'L',
          'I',
          'have',
          'ever',
          'laid',
          'eyes',
          'upon']
```

```
In [5]: [tok.lower() for tok in listOfTokens if len(tok)>0]
```

```
Out[5]: ['this',
          'book',
          'is',
          'the',
          'best',
          'book',
          'on',
          'python',
          'or',
          'm',
          'l',
          'i',
          'have',
          'ever',
          'laid',
          'eyes',
          'upon']
```

```
In [6]: emailText=open('email/ham/6.txt').read()
```

```
In [7]: listOfTokens= regEx.split(emailText)
```

3.1.2 生成词汇表

```
In [8]: import bayes
        listOPost, listClasses = bayes.loadDataSet()
        myVocabList = bayes.createVocabList(listOPost) # 生成词汇表
        myVocabList
```

```
Out[8]: ['garbage',
         'quit',
         'dog',
         'help',
         'to',
         'take',
         'my',
         'so',
         'I',
         'how',
         'not',
         'licks',
         'stupid',
         'flea',
         'is',
         'worthless',
         'him',
         'food',
         'posting',
         'stop',
         'buying',
         'cute',
         'park',
         'dalmation',
         'mr',
         'has',
         'problems',
         'maybe',
         'steak',
         'love',
         'ate',
```

```
 'please']
```

3.1.3 生成词向量

```
In [9]: bayes.bagOfWords2Vec(myVocabList, listOPost[0])
        bayes.bagOfWords2Vec(myVocabList, listOPost[3])
```

```
Out[9]: [1,
         0,
         0,
         0,
         0,
         0,
         0,
         0,
         0,
         0,
         0,
         0,
         1,
         0,
         0,
         1,
         0,
         0,
         1,
         1,
         0,
         0,
         0,
         0,
         0,
         0,
         0,
         0,
         0,
         0,
         0,
```

```
0]
```

3.2 训练算法-从词向量计算概率

```
In [10]: from numpy import *
         listOPost, listClasses = bayes.loadDataSet()
         trainMat = []
         for postinDoc in listOPost:
             trainMat.append(bayes.bagOfWords2Vec(myVocabList, postinDoc))
         p0V,p1V,pAb = bayes.train(trainMat,listClasses)
```

```
In [11]: pAb
```

```
Out[11]: 0.5
```

```
In [12]: p0V
```

```
Out[12]: array([-3.25809654, -3.25809654, -2.56494936, -2.56494936, -2.56494936,
                -3.25809654, -1.87180218, -2.56494936, -2.56494936, -2.56494936,
                -3.25809654, -2.56494936, -3.25809654, -2.56494936, -2.56494936,
                -3.25809654, -2.15948425, -3.25809654, -3.25809654, -2.56494936,
                -3.25809654, -2.56494936, -3.25809654, -2.56494936, -2.56494936,
                -2.56494936, -2.56494936, -3.25809654, -2.56494936, -2.56494936,
                -2.56494936, -2.56494936])
```

```
In [13]: p1V
```

```
Out[13]: array([-2.35137526, -2.35137526, -1.94591015, -3.04452244, -2.35137526,
                -2.35137526, -3.04452244, -3.04452244, -3.04452244, -3.04452244,
                -2.35137526, -3.04452244, -1.65822808, -3.04452244, -3.04452244,
                -1.94591015, -2.35137526, -2.35137526, -2.35137526, -2.35137526,
                -2.35137526, -3.04452244, -2.35137526, -3.04452244, -3.04452244,
                -3.04452244, -3.04452244, -2.35137526, -3.04452244, -3.04452244,
                -3.04452244, -3.04452244])
```

3.3 测试过程

```
In [14]: bayes.spamTest()
```

```
classfication error ['yeah', 'ready', 'may', 'not', 'here', 'because', 'jar', 'jar', 'has', 'pl
the error rate is  0.1
```

4 操作练习

实验中如何解决零概率问题?

在 `train()` 函数中添加拉普拉斯平滑, 在默认情况下每个词在两类中都有 1/2 的先验出现概率. 具体如下:

```
p0Num=ones(numWords);p1Num=ones(numWords) # 词出现次数默认为 1
p0Denom=2.0;p1Denom=2.0 # 总词数默认为 2, 拉普拉斯平滑
```

如何解决概率值太小会产生溢出的问题?

修改 `train()` 函数, 使 p0V 以及 p1V 以对数形式返回

```
p1Vec=log(p1Num/p1Denom) # 类 1 中每个单词的概率取对数
p0Vec=log(p0Num/p0Denom)# 类 0 中每个单词的概率取对数
```

修改 `classify()` 函数, 将 pClass 取对数, 并将乘积改为和

```
p1=sum(vec2classfy*p1Vec)+log(pClass1) # 取对数后的条件概率计算
p0=sum(vec2classfy*p0Vec)+log(1-pClass1)
```

利用 sklearn 中不同的 NB 分类器分类该数据集

使用先验分布为多项式分布的贝叶斯模型进行分类, 将 `spamTest()` 函数修改如下:

```
def spamTest():
    fullTest=[];docList=[];classList=[]
    for i in range(1,26):
        wordList=textParse(open('email/spam/%d.txt' % i,encoding="ISO-8859-1").read())
        docList.append(wordList)
        fullTest.extend(wordList)
        classList.append(1)
        wordList=textParse(open('email/ham/%d.txt' % i,encoding="ISO-8859-1").read())
        docList.append(wordList)
        fullTest.extend(wordList)
        classList.append(0)
    vocabList=createVocabList(docList)
    trainSet=list(range(50));testSet=[]
    for i in range(10):
        randIndex=int(random.uniform(0,len(trainSet)))
```

```
        testSet.append(trainSet[randIndex])
        del(trainSet[randIndex])
    trainMat=[];trainClass=[]
    testMat=[];testClass=[]
    for docIndex in trainSet: # 构建训练集和训练标签
        trainMat.append(bagOfWords2Vec(vocabList,docList[docIndex]))
        trainClass.append(classList[docIndex])
    p0,p1,pSpam=train(array(trainMat),array(trainClass))
    for docIndex in testSet: # 构建测试集和测试标签
        testMat.append(bagOfWords2Vec(vocabList,docList[docIndex]))
        testClass.append(classList[docIndex])
        # 先验分布为多项式分布的贝叶斯
    classifier = MultinomialNB(alpha=1.0,fit_prior=True).fit(trainMat, trainClass)
    test_accuracy = classifier.score(testMat, testClass) # 测试集准确率
    return test_accuracy
```

```
In [15]: import bayes_sklearn as b
         score = b.spamTest()
```

```
In [16]: score # 测试集得分
```

```
Out[16]: 1.0
```