demo_email-spam

September 25, 2023

[1]:
```
import numpy as np
import re
import random
```

0.1 1. Bayes

[6]:
```
"""
  :    :
Parameters:
    trainMatrix -     setOfWords2Vec  returnVec
    trainCategory -       loadDataSet  classVec
Returns:
    p0Vect -
    p1Vect -
    pAbusive -
"""
def trainNB0(trainMatrix, trainCategory):
    numTrainDocs = len(trainMatrix)  #
    numWords = len(trainMatrix[0])  #
    pAbusive = sum(trainCategory) / float(numTrainDocs)  #
    p0Num = np.ones(numWords)
    p1Num = np.ones(numWords)  #  numpy.ones ,     1,
    p0Denom = 2.0
    p1Denom = 2.0  #    2 ,
    for i in range(numTrainDocs):
        if trainCategory[i] == 1:  #           P(w0|1),P(w1|1),P(w2|1)···
            p1Num += trainMatrix[i]
            p1Denom += sum(trainMatrix[i])
        else:  #           P(w0|0),P(w1|0),P(w2|0)···
            p0Num += trainMatrix[i]
            p0Denom += sum(trainMatrix[i])
    p1Vect = np.log(p1Num / p1Denom)
    p0Vect = np.log(p0Num / p0Denom)   #
    return p0Vect, p1Vect, pAbusive  #
```

[7]:
```
"""
  :    :
```

```
Parameters:
        vec2Classify -
        p0Vec -
        p1Vec -
        pClass1 -
Returns:
        0 -
        1 -
"""
def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):
    #p1 = reduce(lambda x, y: x * y, vec2Classify * p1Vec) * pClass1  #
    #p0 = reduce(lambda x, y: x * y, vec2Classify * p0Vec) * (1.0 - pClass1)
    p1=sum(vec2Classify*p1Vec)+np.log(pClass1)
    p0=sum(vec2Classify*p0Vec)+np.log(1.0-pClass1)
    if p1 > p0:
        return 1
    else:
        return 0
```

0.2 2.

[8]:
```
"""
  :
Parameters:
    dataSet -
Returns:
    vocabSet -
"""
def createVocabList(dataSet):
    vocabSet = set([])  #
    for document in dataSet:
        vocabSet = vocabSet | set(document)  #
    return list(vocabSet)
```

[9]:
```
"""
  : vocabList   inputSet      1 0
Parameters:
    vocabList - createVocabList
    inputSet -
Returns:
    returnVec -   ,
"""
def setOfWords2Vec(vocabList, inputSet):
    returnVec = [0] * len(vocabList)                #       0
    for word in inputSet:                           #
        if word in vocabList:                       #       1
            returnVec[vocabList.index(word)] = 1
```

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

[10]:
```
"""
  : vocabList
Parameters:
    vocabList - createVocabList
    inputSet -
Returns:
    returnVec -   ,
"""
def bagOfWords2VecMN(vocabList, inputSet):
    returnVec = [0] * len(vocabList)  #        0
    for word in inputSet:             #
        if word in vocabList:         #
            returnVec[vocabList.index(word)] += 1
    return returnVec  #
```

[11]:
```
"""
  :
"""
def textParse(bigString):  #
    listOfTokens = re.split(r'\W*', bigString)  #
    return [tok.lower() for tok in listOfTokens if len(tok) > 2]  #       I
```

0.3 3. Email classification

[12]:
```
docList = []
fullText = []
classList = []
for i in range(1, 26):  #  25 txt
    wordList = textParse(open('./email/spam/%d.txt' % i, 'r').read())  #␣
 ↪
    docList.append(wordList)
    fullText.append(wordList)
    classList.append(1)  #     1

    wordList = textParse(open('./email/ham/%d.txt' % i, 'r').read())  #␣
 ↪
    docList.append(wordList)
    fullText.append(wordList)
    classList.append(0)  #     0
```

[13]:
```
vocabList = createVocabList(docList)  #
```

[14]:
```
trainingSet = list(range(50))
testSet = []  #
for i in range(10):  # 50     40   ,10
    randIndex = int(random.uniform(0, len(trainingSet)))  #
    testSet.append(trainingSet[randIndex])  #
    del (trainingSet[randIndex])  #
```

[15]:
```
trainMat = []
trainClasses = []  #
for docIndex in trainingSet:  #
    trainMat.append(setOfWords2Vec(vocabList, docList[docIndex]))  #
    trainClasses.append(classList[docIndex])  #
```

[ ]:
```
```

[16]:
```
p0V, p1V, pSpam = trainNB0(np.array(trainMat), np.array(trainClasses))  #
```

[ ]:
```
```

[17]:
```
errorCount = 0  #
for docIndex in testSet:  #
    wordVector = setOfWords2Vec(vocabList, docList[docIndex])  #
    if classifyNB(np.array(wordVector), p0V, p1V, pSpam) != classList[docIndex]:
 ↪  #
        errorCount += 1  #    1
        print("     ", docList[docIndex])
print('  %.2f%%' % (float(errorCount) / len(testSet) * 100))
```

```
     []
     []
     []
     []
     []
     []
 60.00%
```