Bayes Naif(message abusif)

October 30, 2023

MST IASD 2023-2024 (Département Génie Informatique)

Module "Apprentissage automatique" (M. AIT KBIR)

Problème : Classification de texte, utilisant Bayes Naif, pour un problème à deux classes (Texte abusif ou non) :

(Exemple du livre "Machine learning in action")

Désactiver les commentaires pour voir les résultats intermédiaires

Etape 1 : construire la base des exemples (Chaque ligne est un exemple/Message)

[1]:
```
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 is abusive, 0 not
    return postingList,classVec
a,b=loadDataSet()
print(a[2],b[2])
```

```
['my', 'dalmation', 'is', 'so', 'cute', 'I', 'love', 'him'] 0
```

Etape 2 : construire la liste des mots depuis la base des exemples

[2]:
```
def createVocabList(dataSet):
    vocabSet = set([])  # ensemble vide
    for document in dataSet:
        doc=[tok for tok in document if len(tok)>5 ] #Garder les mots de plus␣
 ↪de 3  caractères
        vocabSet = vocabSet | set(doc) #union d'ensembles
    return list(vocabSet)
a,b =loadDataSet() # Juste pour visualiser les résultats intermédiares
c=createVocabList(a)
print(len(c),c)
```

```
8 ['dalmation', 'worthless', 'problems', 'please', 'buying', 'posting',
'garbage', 'stupid']
```

Etape 3 : Calcul du vecteur des caractéristiques à partir d'un message (vecteur de chaines de caractères)

[3]:
```
def setOfWords2Vec(vocabList, inputSet):
    returnVec = [0]*len(vocabList)
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] = 1

    return returnVec
# Vecteur calculé par rapport au premier document
v0=setOfWords2Vec(c,a[3])
print(c)
print(a[3])
print(b[3])
print(v0)
```

```
['dalmation', 'worthless', 'problems', 'please', 'buying', 'posting', 'garbage',
'stupid']
['stop', 'posting', 'stupid', 'worthless', 'garbage']
1
[0, 1, 0, 0, 0, 1, 1, 1]
```

Etape 4 : Apprentissage

[4]:
```
from numpy import *
def trainNB0(trainMatrix,trainCategory):
    numTrainDocs = len(trainMatrix)
    numWords = len(trainMatrix[0])
    pAbusive = sum(trainCategory)/float(numTrainDocs) # Probabilté à priori des␣
 ↪messages abusifs
    p0Num    = zeros(numWords); p1Num = zeros(numWords)  # Nombre des documents␣
 ↪où apparait le mot
    p0Denom  = 0.0; p1Denom = 0.0     # Nombre des mots du vocabulaire présents␣
 ↪dans tous les documents
    for i in range(numTrainDocs):
        if trainCategory[i] == 1:
            p1Num   += trainMatrix[i]
            p1Denom += sum(trainMatrix[i])
        else:
            p0Num   += trainMatrix[i]
            p0Denom += sum(trainMatrix[i])
    # Lissage
    p1Vect = (1+p1Num)/(2+p1Denom)         # (...+1) /(2+...) Lessage pour un␣
 ↪problème à deux classes
    p0Vect = (1+p0Num)/(2+p0Denom)
```

```
    return p0Vect, p1Vect, pAbusive
```

Etape 5 : généralisation

[5]:
```
def classifyNB(vec2Classify, p0Vect, p1Vect, pC1):

    # Calcul de probabilité selon la loi de Bernoulli
    p1 = sum(vec2Classify*log(p1Vect)+(1-vec2Classify)*log(1-p1Vect))+log(pC1)
    p0 = sum(vec2Classify*log(p0Vect)+(1-vec2Classify)*log(1-p0Vect))+log(1-pC1)

    if p1 > p0:
        return 1
    else:
        return 0
```

Etape 6 : entrainement

[6]:
```
listOPosts,listClasses = loadDataSet()
myVocabList = createVocabList(listOPosts)
trainMat=[]
for postinDoc in listOPosts:
    trainMat.append(setOfWords2Vec(myVocabList, postinDoc))

# Pour afficher le nombre des exemples d'apprentissage
print(trainMat)

p0V,p1V,pAb = trainNB0(array(trainMat),array(listClasses))
print(p0V,p1V,pAb)
```

```
[[0, 0, 1, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 1], [1, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 1, 1, 1], [0, 0, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 1, 0, 0, 1]]
[0.4 0.2 0.4 0.4 0.2 0.2 0.2 0.2] [0.1 0.3 0.1 0.1 0.2 0.2 0.2 0.4] 0.5
```

Généralisation

[7]:
```
 # Utiliser les paramètres calculés pour classer des nouveaux messages
testEntry = ['love', 'my', 'dalmation']
thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
print(thisDoc)
print(testEntry,'classified as: ',classifyNB(thisDoc,p0V,p1V,pAb))

testEntry = ['stupid', 'garbage']
thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
print(thisDoc)
print(testEntry,'classified as: ',classifyNB(thisDoc,p0V,p1V,pAb))
```

```
[1 0 0 0 0 0 0 0]
['love', 'my', 'dalmation'] classified as:  0
[0 0 0 0 0 0 1 1]
['stupid', 'garbage'] classified as:  1
```