模型-机器学习-分类-朴素贝叶斯【hxy】

1. 模型名称

朴素贝叶斯（Naive Bayes，NB）

2. 模型评价

2.1 优点

- 在数据较少的情况下仍然有效
- 可以处理多类别问题

2.2 缺点

- 对于输入数据的准备方式较为敏感

3. 基本算法

设有**样本数据集**$D=d_1,d_2,\dots,d_n$，对应样本数据的**特征属性集**为$X=x_1,x_2,\dots,x_d$，**类变量**为$Y=y_1,y_2,\dots,y_m$，即D可以分为y_m类别，其中$x_1,x_2,..,x_d$相互独立且随机，且Y的**先验概率**$P_{prior}=P(Y)$，Y的**后验概率**$P_{post}=P(Y|X)$

由**朴素贝叶斯算法**可得

$$P(Y|X)=\frac{p(Y)P(X|Y)}{P(X)}$$

由于朴素贝叶斯假设各特征之间**相互独立**，在给定类别为y的情况下，上式可进一步表示为

$$P(X|Y=y)=\prod_{i=1}^{d}P(x_i|Y=y)$$

$$\Rightarrow P_{post}=P(Y|X)=\frac{P(Y)\prod_{i=1}^{d}P(x_i|Y)}{P(X)}$$

由于$P(X)$的大小是固定不变的，因此在比较后验概率时，只比较上式的分子部分即可，因此可以得到一个样本数据集数语类别y_i的朴素贝叶斯计算

$$P(y_i|x_1,x_2,\ldots,x_d)=\frac{P(y_i)\prod_{j=1}^{d}P(x_j|y_i)}{\prod_{j=1}^{d}P(x_j)}$$

4. 实例

4.1 数据介绍

该数据集放在**email**文件夹中，该文件夹又包含两个字文件夹，分别是**spam**与**ham**，每封邮件以**序号.txt**命名，示例路径为`'email/sham/1.txt'`，示例文件内容为

```
--- Codeine 15mg -- 30 for $203.70 -- VISA Only!!! --

-- Codeine (Methylmorphine) is a narcotic (opioid) pain reliever
-- We have 15mg & 30mg pills -- 30/15mg for $203.70 - 60/15mg for $385.80 - 90/15mg for
$562.50 -- VISA Only!!! ---
```

4.2 实验目的

电子邮件垃圾过滤

4.3 代码实现

NB.py

代码：

```
# 导入numpy库
from numpy import *
# 导入正则表达式库
import re

# 创建一个包含在所有文档中出现的不重复次的列表
def createVocabList(dataSet):
    vocabSet = set([])  #create empty set
    for document in dataSet:
        vocabSet = vocabSet | set(document) #union of the two sets
    return list(vocabSet)

# 输入：vacabList是包含所有词的列表，inputSet是待建立文档向量的文档
# 输出：inputSet对应的文档向量
# 与bagOfWords2Vec不同的是，该向量记录的是每个单词在该文档中是否出现
def setOfWords2Vec(vocabList, inputSet):
    returnVec = [0]*len(vocabList)
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] = 1
        else: print("the word: %s is not in my Vocabulary!" % word)
    return returnVec

# 输入：vacabList是包含所有词的列表，inputSet是待建立文档向量的文档
```

```
# 输出：inputSet对应的文档向量
# 与setOfWords2Vec不同的是，该向量记录的是每个单词在该文档中的出现次数
def bagOfWords2VecMN(vocabList, inputSet):
    returnVec = [0]*len(vocabList)
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] += 1
    return returnVec

# 训练分类器
# 输入：trainMatrix是所有文档向量构成的文档矩阵，trainCategory是所有文档类别构成的文档类别向量
# 输出：p0Vect是分类为class0的各单词的概率矩阵，p1Vect是分类为class1的各单词的概率矩阵
# 输出：pAbusive是任意文档属于class0的概率
def trainNB0(trainMatrix,trainCategory):
    numTrainDocs = len(trainMatrix)
    numWords = len(trainMatrix[0])
    pAbusive = sum(trainCategory)/float(numTrainDocs)
    # 为了避免其中一个概率值为0，最后乘积也为0
    # 将所有词的出现数初始化为1，并将分母初始化为2
    p0Num = ones(numWords); p1Num = ones(numWords)
    p0Denom = 2.0; p1Denom = 2.0
    for i in range(numTrainDocs):
        if trainCategory[i] == 1:
            p1Num += trainMatrix[i] # 向量相加
            p1Denom += sum(trainMatrix[i])
        else:
            p0Num += trainMatrix[i] # 向量相加
            p0Denom += sum(trainMatrix[i])
    # 为了解决*下溢出*（即很多很小的因子相乘被近似为0），对概率取对数
    p1Vect = log(p1Num/p1Denom)
    p0Vect = log(p0Num/p0Denom)
    return p0Vect,p1Vect,pAbusive

# 朴素贝叶斯分类
# 输入：vec2Classify待分类的向量
# 输出：分类类别
def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):
    # 概率相乘（由于取了对数，变成相加）
    p1 = sum(vec2Classify * p1Vec) + log(pClass1)
    p0 = sum(vec2Classify * p0Vec) + log(1.0 - pClass1)
    if p1 > p0:
        return 1
    else:
        return 0

# 输入：一封邮件内容
# 输出：小写的以除单词、数字外的任意字符串分割的长于2字符的单词列表
def textParse(bigString):
    import re
```

```
    #匹配非英文字母和数字
    regEx = re.compile('\\W+')
    listOfTokens = regEx.split(bigString)
    # 小写，并过滤掉小于2的单词
    return [tok.lower() for tok in listOfTokens if len(tok) > 2]

# 测试主代码
def spamTest():
    docList=[]; classList = []; fullText =[]
    # 读入数据集
    for i in range(1,26):
        # 用‘gb18030’的编码方式打开txt文件，并忽视无法编码的特殊字符
        # wordList暂存当前文档的单词列表
        wordList = textParse(open('email/spam/%d.txt' % i,encoding='gb18030',
errors='ignore').read())
        # 检查分割邮件的效果
        if i==1:
            print('Division result for email/spam/1.txt is:')
            print(wordList)
        # docList每个元素都是一个单词列表
        docList.append(wordList)
        # fullText每个元素都是一个单词
        fullText.extend(wordList)
        classList.append(1)
        wordList = textParse(open('email/ham/%d.txt' % i,encoding='gb18030',
errors='ignore').read())
        docList.append(wordList)
        fullText.extend(wordList)
        classList.append(0)
    # 创建一个包含在所有文档中出现的不重复次的单词列表
    vocabList = createVocabList(docList)
    # 创建测试集
    trainingSet = list(range(50)); testSet=[]
    # 随机选择10封邮件作为训练集，训练分类器，并从测试集中删去
    for i in range(10):
        randIndex = int(random.uniform(0,len(trainingSet)))
        testSet.append(trainingSet[randIndex])
        del(trainingSet[randIndex])
    trainMat=[]; trainClasses = []
    for docIndex in trainingSet:
        trainMat.append(bagOfWords2VecMN(vocabList, docList[docIndex]))
        trainClasses.append(classList[docIndex])
    p0V,p1V,pSpam = trainNB0(array(trainMat),array(trainClasses))
    errorCount = 0
    # 将剩下测试集中的40封邮件分类
    for docIndex in testSet:
        wordVector = bagOfWords2VecMN(vocabList, docList[docIndex])
        if classifyNB(array(wordVector),p0V,p1V,pSpam) != classList[docIndex]:
            errorCount += 1
```

```
            # 打印错误分类的单词，可以根据这些找错误的邮件
            print("Classification error:")
            print(docList[docIndex])
    # 打印错误的比例
    print('The error rate is: ',float(errorCount)/len(testSet))

spamTest()
```

结果：

```
==================== RESTART: /Users/xinyuanhe/Desktop/NB.py ====================
Division result for email/spam/1.txt is:
['codeine', '15mg', 'for', '203', 'visa', 'only', 'codeine', 'methylmorphine', '
narcotic', 'opioid', 'pain', 'reliever', 'have', '15mg', '30mg', 'pills', '15mg'
, 'for', '203', '15mg', 'for', '385', '15mg', 'for', '562', 'visa', 'only']
Classification error:
['yay', 'you', 'both', 'doing', 'fine', 'working', 'mba', 'design', 'strategy',
'cca', 'top', 'art', 'school', 'new', 'program', 'focusing', 'more', 'right', 'b
rained', 'creative', 'and', 'strategic', 'approach', 'management', 'the', 'way',
 'done', 'today']
The error rate is:  0.1
```

5. 参考资料

1. 《机器学习实战》P72-P91 [图灵程序设计丛书].机器学习实战.pdf
2. 百度百科-朴素贝叶斯（算法原理）