实验五　朴素贝叶斯算法和垃圾邮件分类

【实验目的】

1. 了解 Anaconda 和 python 的使用与设置。
2. 掌握朴素贝叶斯算法的基本原理和实现方法。
3. 掌握垃圾邮件分类的基本原理和方法

【实验学时】

建议 2 学时

【实验环境配置】

1、 Windows 环境
2、 Anaconda
3、 Pandas
4、 Sklearn
5、 spacy

【实验原理】

算法的实现和垃圾邮件分类：

1、Pandas 数据导入
2、朴素贝叶斯算法
3、垃圾邮件分类

【实验步骤】

1. 打开 Anaconda 的 jupyter notebook，创建实验五，明确标题和步骤。
2. 导入 pandas 库和 os 库，从数据文件中，导入实验数据。
   （注：可能涉及的函数 os.path.join, os.walk, pandas.read_csv）
3. 对导入的实验数据，根据《机器学习实战》教程第四章，完成朴素贝叶斯算法，并完成对于垃圾邮件数据的分类
4. 对导入的实验数据，根据 sklearn 的朴素贝叶斯算法模块 sklearn.naive_bayes，完成对于垃圾邮件数据的分类

导入 Spacy 分词，统计分析词性

```
[56]: # spacy分词
import spacy
import re
nlp = spacy.load('en_core_web_sm')
```

```
[6]: doc = nlp('He went to play basketball sawfwag')
```

```
[10]: print(len(doc))
```

```
6
```

```
[59]: def textParse(filename):
    with open(filename,'r', encoding='gb18030', errors='ignore') as f:
        doc = nlp(f.read())
        for token in doc:
            print(token.text, "-->", token.pos_)# 统计词性
```

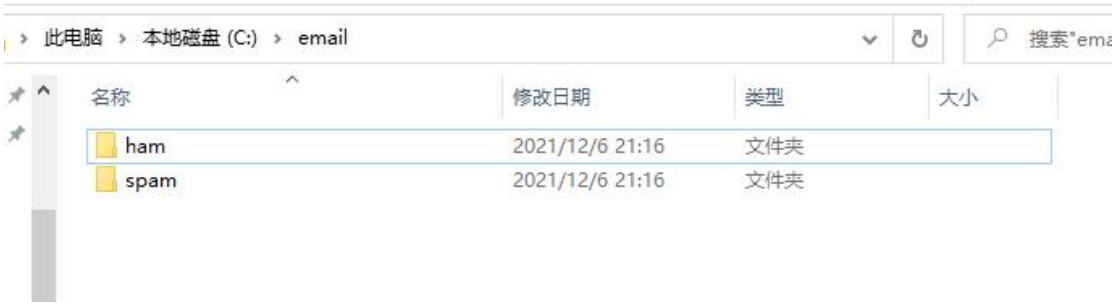
```
[60]: # 遍历文件
import os
for filepath, dirnames, filenames in os.walk(r'C:\email'):
    for filename in filenames:
        print(os.path.join(filepath, filename))
        textParse(os.path.join(filepath, filename))
```

```
C:\email\ham\1.txt
Hi --> INTJ
Peter --> PROPN
, --> PUNCT


 --> SPACE
With --> ADP
Jose --> PROPN
out --> ADP
of --> ADP
town --> NOUN
, --> PUNCT
do --> AUX
you --> PRON
want --> VERB
to --> PART

 --> SPACE
meet --> VERB
```

spacy统计词性



准备好相关的数据集，如图，ham 文件夹下的文件为正常邮件，spam 文件夹下的文件为垃圾邮件。

一、朴素贝叶斯算法

1、导入实验数据

textParse 函数接收一个大字符串(即文本中的英语词句)，以特殊字符作为切分标志进行字符串切分，返回字符串列表。

createVocabList 函数接收一个词条列表，将其整理成不重复的词条列表。

导入实验数据

[56]:
```
# spacy分词
import spacy
import re
nlp = spacy.load('en_core_web_sm')
```

[6]:
```
doc = nlp('He went to play basketball sawfwag')
```

[10]:
```
print(len(doc))
```

```
6
```

114]:
```
"""
函数说明:接收一个大字符串并将其解析为字符串列表
"""
def textParse(bigString):  # 将字符串转换为字符列表
    listOfTokens = re.split(r'\W', bigString)  # 将特殊符号作为切分标志进行字符串切分，即非字母、非数字
    return [tok.lower() for tok in listOfTokens if len(tok) > 2]  # 除了单个字母，例如大写的I，其它单词变成小写
```

115]:
```
"""
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
"""
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)
```

通过 os.walk 遍历文件夹，os.path.join 获取文件绝对路径。

```
# 遍历文件
import random
docList = []
fullText = []
classList = []
for i in range(1,26):
    classList.append(0) # 标记正常邮件，0表示正常邮件
for j in range(1,26):
    classList.append(1) # 标记垃圾邮件，1表示垃圾邮件
for filepath, dirnames, filenames in os.walk(r'C:\email'):
    for filename in filenames:
#         print(os.path.join(filepath, filename))
        wordList = textParse(open(os.path.join(filepath, filename), 'r', errors='ignore').read()) # 读取每个垃圾邮件，并字符串转换成字符
        docList.append(wordList)
        fullText.append(wordList)
print(docList)
```

```
[['peter', 'with', 'jose', 'out', 'town', 'you', 'want', 'meet', 'once', 'while', 'keep', 'things', 'going', 'and', 'some', 'interesti
ng', 'stuff', 'let', 'know', 'eugene'], ['ryan', 'whybrew', 'commented', 'your', 'status', 'ryan', 'wrote', 'turd', 'ferguson', 'but
t', 'horn'], ['arvind', 'thirumalai', 'commented', 'your', 'status', 'arvind', 'wrote', 'you', 'know', 'reply', 'this', 'email', 'comm
ent', 'this', 'status'], ['thanks', 'peter', 'definitely', 'check', 'this', 'how', 'your', 'book', 'going', 'heard', 'chapter', 'cam
e', 'and', 'was', 'good', 'shape', 'hope', 'you', 'are', 'doing', 'well', 'cheers', 'troy'], ['jay', 'stepp', 'commented', 'your', 'st
atus', 'jay', 'wrote', 'the', 'reply', 'this', 'email', 'comment', 'this', 'status', 'see', 'the', 'comment', 'thread', 'follow', 'th
e', 'link', 'below'], ['linkedin', 'kerry', 'haloney', 'requested', 'add', 'you', 'connection', 'linkedin', 'peter', 'like', 'add', 'y
ou', 'professional', 'network', 'linkedin', 'kerry', 'haloney'], ['peter', 'the', 'hotels', 'are', 'the', 'ones', 'that', 'rent', 'ou
t', 'the', 'tent', 'they', 'are', 'all', 'lined', 'the', 'hotel', 'grounds', 'much', 'for', 'being', 'one', 'with', 'nature', 'more',
'like', 'being', 'one', 'with', 'couple', 'dozen', 'tour', 'groups', 'and', 'nature', 'have', 'about', '100m', 'pictures', 'from', 'th
at', 'trip', 'can', 'through', 'them', 'and', 'get', 'you', 'jpgs', 'favorite', 'scenic', 'pictures', 'where', 'are', 'you', 'and', 'j
ocelyn', 'now', 'new', 'york', 'will', 'you', 'come', 'tokyo', 'for', 'chinese', 'new', 'year', 'perhaps', 'see', 'the', 'two', 'you',
'then', 'will', 'thailand', 'for', 'winter', 'holiday', 'see', 'mom', 'take', 'care'], ['yeah', 'ready', 'may', 'not', 'here', 'becaus
e', 'jar', 'jar', 'has', 'plane', 'tickets', 'germany', 'for'], ['benoit', 'mandelbrot', '1924', '2010', 'benoit', 'mandelbrot', '192
4', '2010', 'wilmott', 'team', 'benoit', 'mandelbrot', 'the', 'mathematician', 'the', 'father', 'fractal', 'mathematics', 'and', 'advo
cate', 'more', 'sophisticated', 'modelling', 'quantitative', 'finance', 'died', '14th', 'october', '2010', 'aged', 'wilmott', 'magazin
e', 'has', 'often', 'featured', 'mandelbrot', 'his', 'ideas', 'and', 'the', 'work', 'others', 'inspired', 'his', 'fundamental', 'insig
hts', 'you', 'must', 'logged', 'view', 'these', 'articles', 'from', 'past', 'issues', 'wilmott', 'magazine'], ['peter', 'sure', 'thin
g', 'sounds', 'good', 'let', 'know', 'what', 'time', 'would', 'good', 'for', 'you', 'will', 'come', 'prepared', 'with', 'some', 'idea
```

初始化数据集，classList 标记正常邮件和垃圾邮件，正常邮件用 0 标记，垃圾邮件用 1 标记，各 25 个。Errors='ignore'忽略特殊字符。

2、文本解析向量函数

根据词汇表将输入的文本向量化，每个元素为 0 或 1，遍历每个词条，如果词条存在于词汇表中，则置 1，默认为 0。

将文本解析成词条向量

```
5]:
"""
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
"""
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
```

3、朴素贝叶斯算法模型

trinNB0 函数根据训练文档矩阵、训练类别标签向量(文本解析向量函数 setOfWords2Vec 返回的文档矩阵)计算正常邮件和垃圾邮件的条件概率数组，并计算文档属于垃圾邮件类的概率，即计算训练类别标签向量中 1 的个数(求和)占总训练文档的比例。

训练朴素贝叶斯模型函数

```
"""
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 - 正常邮件类的条件概率数组
    p1Vect - 垃圾邮件类的条件概率数组
    pAbusive - 文档属于垃圾邮件类的概率
"""
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)  # 文档属于垃圾邮件类的概率
    p0Num = np.ones(numWords)
    p1Num = np.ones(numWords)  # 创建numpy.ones数组,词条出现数初始化为1,拉普拉斯平滑
    p0Denom = 2.0
    p1Denom = 2.0  # 分母初始化为2 ,拉普拉斯平滑
    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 = np.log(p1Num / p1Denom)
    p0Vect = np.log(p0Num / p0Denom)   #取对数，防止下溢出
    return p0Vect, p1Vect, pAbusive  # 返回属于正常邮件类的条件概率数组，属于侮辱垃圾邮件类的条件概率数组，文档属于垃圾邮件类的概率
```

classifyNB 分类器分类函数根据 trainNB0 返回的三个列表，再根据输入的待分类的词条数组估计该词条数组是属于正常邮件还是垃圾邮件，返回 1 表示垃圾邮件，返回 0 表示正常邮件。

```
"""
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 - 正常邮件类的条件概率数组
    p1Vec - 垃圾邮件类的条件概率数组
    pClass1 - 文档属于垃圾邮件的概率
Returns:
    0 - 属于正常邮件类
    1 - 属于垃圾邮件类
"""
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)
    p1=sum(vec2Classify*p1Vec)+np.log(pClass1)
    p0=sum(vec2Classify*p0Vec)+np.log(1.0-pClass1)
    if p1 > p0:
        return 1
    else:
        return 0
```

4、训练测试集

从 50 个邮件中，随机选出 40 个作为训练集，10 个做测试集，并通过朴素贝叶斯算法返回正常邮件类的条件概率数组、垃圾邮件类的条件概率数组、文档属于垃圾邮件类的概率；再通过朴素贝叶斯分类器分类结果来比较分类是否正确。

训练测试集

```
[120]: vocabList = createVocabList(docList) # 创建词汇表，不重复
       trainingSet = list(range(50))
       testSet = [] # 创建存储训练集的索引值的列表和测试集的索引值的列表
       for i in range(10):
           randIndex = int(random.uniform(0,len(trainingSet))) # 随机选取索引值
           testSet.append(trainingSet[randIndex])
           del (trainingSet[randIndex])  # 在训练集列表中删除添加到测试集的索引值

       trainMat = []
       trainClasses = []  # 创建训练集矩阵和训练集类别标签系向量
       for docIndex in trainingSet:  # 遍历训练集
           trainMat.append(setOfWords2Vec(vocabList, docList[docIndex]))  # 将生成的词集模型添加到训练矩阵中
           trainClasses.append(classList[docIndex])  # 将类别添加到训练集类别标签系向量中
```

5、计算错误率

如果经过 classiyNB 分类器分类的结果和原始的标记邮件列表 classList 不同，说明分类错误。

计算错误率

```
1]: p0V, p1V, pSpam = trainNB0(np.array(trainMat), np.array(trainClasses))  # 训练朴素贝叶斯模型
    errorCount = 0  # 错误分类计数
    for docIndex in testSet:  # 遍历测试集
        wordVector = setOfWords2Vec(vocabList, docList[docIndex])  # 测试集的词集模型
        if classifyNB(np.array(wordVector), p0V, p1V, pSpam) != classList[docIndex]:  # 如果分类错误
            errorCount += 1  # 错误计数加1
            print("分类错误的测试集：", docList[docIndex])
    print('错误率：%.2f%%' % (float(errorCount) / len(testSet) * 100))
```

```
分类错误的测试集： ['scifinance', 'now', 'automatically', 'generates', 'gpu', 'enabled', 'pricing', 'risk', 'model', 'source', 'code',
'that', 'runs', '300x', 'faster', 'than', 'serial', 'code', 'using', 'new', 'nvidia', 'fermi', 'class', 'tesla', 'series', 'gpu', 'scifi
nance', 'derivatives', 'pricing', 'and', 'risk', 'model', 'development', 'tool', 'that', 'automatically', 'generates', 'and', 'gpu', 'en
abled', 'source', 'code', 'from', 'concise', 'high', 'level', 'model', 'specifications', 'parallel', 'computing', 'cuda', 'programming',
'expertise', 'required', 'scifinance', 'automatic', 'gpu', 'enabled', 'monte', 'carlo', 'pricing', 'model', 'source', 'code', 'generatio
n', 'capabilities', 'have', 'been', 'significantly', 'extended', 'the', 'latest', 'release', 'this', 'includes']
错误率：10.00%
```

可以看到实验结果平均错误率为 10%左右。

二、sklearn 的朴素贝叶斯算法模块 sklearn.naive_bayes

1、数据读取

1.数据读取

```
]: # 1. 数据集的读取
import os
import numpy as np
dataset_x = []
dataset_y = []
for filename in list(os.listdir("spam")):
    file_content = None
    with open("spam/" + filename, mode="r", encoding="utf-8",errors='ignore') as f:
        file_content = f.readlines()
    content = ""
    for c in file_content:
        if len(content) !=0 :
            content += " "
        content += c
    dataset_x.append(content)
    dataset_y.append(1)

for filename in list(os.listdir("ham")):
    file_content = None
    with open("ham/" + filename, mode="r", encoding="utf-8", errors='ignore') as f:
        file_content = f.readlines()
    content = ""
    for c in file_content:
        if len(content) !=0 :
            content += " "
        content += c
    dataset_x.append(content)
    dataset_y.append(0)

dataset_x = np.array(dataset_x)
dataset_y = np.array(dataset_y)
```

2、打乱顺序

这里，因为要划分下训练集和测试集，所需需要先打乱顺序。

2.shuffle打乱顺序

```
]: # 2. 数据集打乱顺序
import numpy as np
np.random.seed(116)
np.random.shuffle(dataset_x)
np.random.seed(116)
np.random.shuffle(dataset_y)
```

3、向量化表示

3.向量化表示

```
# 3. 使用sklearn的tf-idf来得到词向量表示
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=2, ngram_range=(1,2), stop_words='english', strip_accents='unicode',norm='l2')
dataset_x_vec = vectorizer.fit_transform(dataset_x)
```

4、数据集划分

为了在后面更加正规的计算准确率等，这里划分数据集。

```
from sklearn.model_selection import train_test_split
train_x_vec,test_x_vec,train_y,test_y = train_test_split(dataset_x_vec, dataset_y,test_size=0.2)
```

5、建立模型

直接使用 `sklearn` 中的朴素贝叶斯，我们这里假设特征的先验概率为多项式分布，即 `MultinomialNB`。
当然，还有二元伯努利分布、正态分布，分别对应 `BernoulliNB`、`GaussianNB`。

```
# 5. 分类器
from sklearn.naive_bayes import MultinomialNB
clf=MultinomialNB().fit(train_x_vec,train_y)
```

6、查看结果

训练向量的维度

结果：精确率、召回率、F1 score(精确率和召回率的调和平均值)、support （各分类样本的数量或测试集样本的总数量）。

【思考题】

1. 查阅文献，思考还有哪些分类算法比较适合垃圾邮件分类，为什么？。

答：朴素贝叶斯算法的优点和缺点：优点是在数据较少的情况下依然有效，可以处理多类别问题；缺点是对于输入数据的准备方式较为敏感，由于朴素贝叶斯的“朴素”特点，即要求事件之间相互独立，所以会带来一些准确率上的损失；可以使用拉普拉斯平滑解决零概率问题。

基于线性核函数的 svm 算法、KNN、LR 和机器学习算法等。

机器学习算法可以学习不同特征之间的相互作用，而朴素贝叶斯有一个限制，那就是需要特征变量之间需要是相互独立的，所以如果样本属性有关联时其效果不太好。机器学习算法可以对各个算法一个个地进行测试，进行比较，然后调整参数确保每个算法达到最优解，最后选择最好的一个。