Project2:文本分类

姓名：王溢阳	学号：10204602470	学院：数据科学与工程学院
指导老师：董启文	上机实践时间：2023年12月22日	上机实践成绩：

实验要求

- 数据集：http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-beyes.html
- 任务：20000个文档分成20类，五重交叉验证结果，不要使用网站上的代码
- 源码+实验报告
- 交给助教
- Deadline:学期末考试前

项目结构

```
|- 20_newsgroups                数据集解压版
|- code                         实验代码
    |- cross_validation.py      交叉验证
    |- model.py                 贝叶斯算法
    |- predict.py               预测
    |- show.py                  可视化
    |- train.py                 训练
|- 20new_sbydate.tar.gz         数据集
|- Project2 文本分类.md          实验报告
```

实验步骤

导入基本库：代码见code/model.py

```
import numpy as np
import re
import random
import pandas as pd
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report

import matplotlib.pyplot as plt
import seaborn as sns
```

设置绘图尺寸

```
plt.rcParams['figure.figsize'] = (10.0, 8.0)
```

朴素贝叶斯算法

对于给定的输入向量 x = (x_1, x_2, ..., x_n)，其中每个 x_k 代表一个特征，通过贝叶斯定理，模型可以计算每个类别 y 的后验概率 p(y | x)，并选择具有最高后验概率的类别作为预测结果。

$$p(y|x) = (p(x|y) * p(y))/p(x)$$

在朴素贝叶斯分类器中，使用了"朴素"的假设，即所有特征之间相互独立。

基于此假设，可以将条件概率 p(x | y) 进行分解：

$$p(x|y) = p(x_1|y) * p(x_2|y) * ... * p(x_n|y)$$

根据训练数据，可以通过统计计算以下概率：

- p(y)：先验概率，表示类别 y 在训练集中的出现概率。
- $p(x_k | y)$：条件概率，表示在类别 y 下特征 x_k 出现的概率。

通过对训练数据进行学习，计算上述概率，应用到测试数据，对测试样本进行分类。

> 函数解析

- **createVocabList**
  - vocabSet：存储词汇表
  - 遍历 dataSet中的每个文档，添加新词

```
def createVocabList(dataSet):
    vocabSet = set([])
    for document in dataSet:
        vocabSet = vocabSet | set(document)
    return list(vocabSet)
```

- **setOfWords2Vec**
  - 遍历inputSet中的每个单词
    - 如果单词存在，将对应的 returnVec元素置为1
    - 打印该单词不在词汇表中

```
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
```

- **bagOfWords2VecMN**
  - 遍历inputSet中的每个单词
    - 单词在 vocabList中存在，对应returnVec元素+1
  - 返回最终的词袋模型特征向量 returnVec

```
def bagOfWords2VecMN(vocabList, inputSet):
    returnVec = [0]*len(vocabList)
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] += 1
    return returnVec
```

- **NaiveBayes_Train**
  - 获取训练文档的数量和单词的数量
  - 计算正类别文档的先验概率 pAbusive
  - 初始化条件概率估计的分子项
  - 初始化条件概率估计的分母项
  - 遍历训练文档：
    - 文档属于正类别，更新 p1Num和 p1Denom
    - 更新 p0Num和 p0Denom
    - 计算条件概率估计项，使用对数避免概率累乘导致下溢

```
def NaiveBayes_Train(trainMatrix, trainCategory):
    numTrainDocs = len(trainMatrix)
    numWords = len(trainMatrix[0])
    pAbusive = sum(trainCategory)/float(numTrainDocs)
    p0Num = np.ones(numWords); p1Num = np.ones(numWords)
    p0Denom = 2.0; p1Denom = 2.0
    for i in range(numTrainDocs):
        if trainCategory[i] == 1:
            p1Num += trainMatrix[i]
            p1Denom += sum(trainMatrix[i])
        else:
            p0Num += trainMatrix[i]
            p0Denom += sum(trainMatrix[i])
    p1Vect = np.log(p1Num/p1Denom)
    p0Vect = np.log(p0Num/p0Denom)
    return p0Vect, p1Vect, pAbusive
```

- **NaiveBayes_Classify**
  - 计算待分类文档属于正类别的概率 p1
  - 计算待分类文档属于负类别的概率 p0
  - 比较 p1和p0
    - p1大于 p0，预测为正类别
    - 预测为负类别

```
def NaiveBayes_Classify(vec2Classify, p0Vec, p1Vec, 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
```

- **textParse**
  - 将 bigString拆分成单词的列表 listOfTokens
  - 将每个单词转换为小写，只保留长度大于2的单词

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

- **获取新闻数据集**

```
news = fetch_20newsgroups(subset="all")
print(news.target)
```

alt.atheism	0.83	0.88	0.85	194
comp.graphics	0.66	0.88	0.76	245
comp.os.ms-windows.misc	0.84	0.16	0.27	253
comp.sys.ibm.pc.hardware	0.65	0.83	0.73	245
comp.sys.mac.hardware	0.87	0.84	0.86	225
comp.windows.x	0.72	0.90	0.80	255
misc.forsale	0.94	0.75	0.84	258
rec.autos	0.90	0.90	0.90	230
rec.motorcycles	0.96	0.96	0.96	243
rec.sport.baseball	0.97	0.95	0.96	242
rec.sport.hockey	0.97	0.97	0.97	251
sci.crypt	0.87	0.97	0.92	271
sci.electronics	0.89	0.85	0.87	249
sci.med	0.93	0.93	0.93	232
sci.space	0.94	0.95	0.95	256
soc.religion.christian	0.83	0.95	0.88	241
talk.politics.guns	0.90	0.94	0.92	236
talk.politics.mideast	0.93	0.99	0.96	252
talk.politics.misc	0.79	0.87	0.83	188
talk.religion.misc	0.94	0.52	0.67	146
accuracy			0.86	4712
macro avg	0.87	0.85	0.84	4712
weighted avg	0.87	0.86	0.84	4712

预测:代码见code/predict.py

将文本数据转换为特征向量表示，利用朴素贝叶斯分类器进行训练和预测，输出分类准确率和性能

```
x_train, x_test, y_train, y_test =
vec = CountVectorizer()
x_train = vec.fit_transform(x_train)
x_test = vec.transform(x_test)
mnb = MultinomialNB()
mnb.fit(x_train, y_train)
y_predict = mnb.predict(x_test)

print("Accuracy:", mnb.score(x_test, y_test))
print("Indicators: \n",classification_report(y_test, y_predict,
target_names=news.target_names))
```

	precision	recall	f1-score	support
alt.atheism	0.90	0.93	0.92	122
comp.graphics	0.73	0.84	0.78	144
comp.os.ms-windows.misc	0.95	0.48	0.64	152
comp.sys.ibm.pc.hardware	0.69	0.90	0.78	158
comp.sys.mac.hardware	0.85	0.85	0.85	144
comp.windows.x	0.85	0.86	0.85	146
misc.forsale	0.86	0.75	0.80	146
rec.autos	0.92	0.92	0.92	153
rec.motorcycles	0.98	0.98	0.98	155
rec.sport.baseball	0.98	0.96	0.97	148
rec.sport.hockey	0.98	0.99	0.98	175
sci.crypt	0.90	0.98	0.94	153
sci.electronics	0.80	0.86	0.83	127
sci.med	0.92	0.93	0.92	135
sci.space	0.94	0.94	0.94	147
soc.religion.christian	0.97	0.93	0.95	164
talk.politics.guns	0.90	0.93	0.91	138
talk.politics.mideast	0.99	0.99	0.99	140
talk.politics.misc	0.88	0.89	0.89	113
talk.religion.misc	0.83	0.78	0.80	67
accuracy			0.89	2827
macro avg	0.89	0.88	0.88	2827
weighted avg	0.89	0.89	0.89	2827

交叉验证:代码见code/cross_validation.py

- 循环不同的超参数值和多次交叉验证
- 计算每个超参数值下的平均准确率
- 将每折的准确率存储在列表中

```
cross_validation_list= []
lambda_list = [0.0001,0.001,0.01,0.05,0.125,0.25,0.5,0.75,1,2]
for hyper_param in lambda_list:
    temp_acc = 0
    temp_acc_list = []
    for fold in range(0,5):
        rand_state = random.randrange(0,100)
        x_train, x_test, y_train, y_test =
train_test_split(news.data,news.target,test_size=0.2,random_state=rand_state)
        vec = CountVectorizer()
        x_train = vec.fit_transform(x_train)
        x_test = vec.transform(x_test)
        multinomial_naive_bayes = MultinomialNB(alpha=hyper_param)
        multinomial_naive_bayes.fit(x_train, y_train)
        y_predict = multinomial_naive_bayes.predict(x_test)
        fold_accuracy = multinomial_naive_bayes.score(x_test, y_test)
        print("validation accuracy:", fold_accuracy
,"fold",fold+1,"lambda=",hyper_param)
        temp_acc+=fold_accuracy
        temp_acc_list.append(fold_accuracy)
    print("lambda:",hyper_param,"accuracy:",temp_acc/5)
    cross_validation_list.append(temp_acc_list)
```

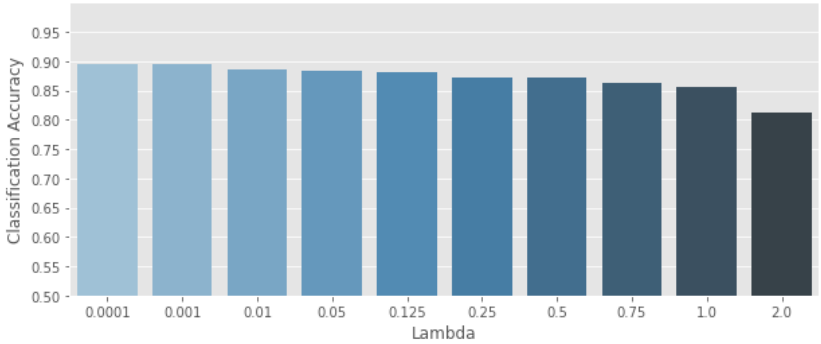
```
validation accuracy: 0.8872679045092838 fold 1 lambda= 0.0001
validation accuracy: 0.8968169761273209 fold 2 lambda= 0.0001
validation accuracy: 0.8978779840848806 fold 3 lambda= 0.0001
validation accuracy: 0.8917771883289125 fold 4 lambda= 0.0001
validation accuracy: 0.9013262599469496 fold 5 lambda= 0.0001
lambda: 0.0001 accuracy: 0.8950132625994695
validation accuracy: 0.8970822281167109 fold 1 lambda= 0.001
validation accuracy: 0.8938992042440318 fold 2 lambda= 0.001
validation accuracy: 0.8909814323607427 fold 3 lambda= 0.001
validation accuracy: 0.8986737400530505 fold 4 lambda= 0.001
validation accuracy: 0.8856763925729443 fold 5 lambda= 0.001
lambda: 0.001 accuracy: 0.893262599469496
validation accuracy: 0.8809018567639257 fold 1 lambda= 0.01
validation accuracy: 0.9021220159151193 fold 2 lambda= 0.01
validation accuracy: 0.8896551724137931 fold 3 lambda= 0.01
validation accuracy: 0.8827586206896552 fold 4 lambda= 0.01
validation accuracy: 0.8827586206896552 fold 5 lambda= 0.01
lambda: 0.01 accuracy: 0.8876392572944297

validation accuracy: 0.8917771883289125 fold 1 lambda= 0.05
validation accuracy: 0.879840848806366 fold 2 lambda= 0.05
validation accuracy: 0.8883289124668435 fold 3 lambda= 0.05
validation accuracy: 0.8838196286472149 fold 4 lambda= 0.05
validation accuracy: 0.8771883289124669 fold 5 lambda= 0.05
lambda: 0.05 accuracy: 0.8841909814323607
validation accuracy: 0.8824933687002653 fold 1 lambda= 0.125
validation accuracy: 0.8713527851458885 fold 2 lambda= 0.125
validation accuracy: 0.8822281167108753 fold 3 lambda= 0.125
validation accuracy: 0.8907161803713528 fold 4 lambda= 0.125
validation accuracy: 0.8925729442970822 fold 5 lambda= 0.125
lambda: 0.125 accuracy: 0.8838726790450927
```

结果可视化:代码见code/show.py

- 可视化，将不同超参数值下的交叉验证平均准确率进行比较，选择最佳的超参数值

```
cv_mean=[] # 交叉验证得到的均值
for lst in cross_validation_list:
    cv_mean.append(np.mean(lst))
df = pd.DataFrame()
df['Lambda'] = lambda_list
df['Classification Accuracy'] =  cv_mean
plt.figure(figsize=(10, 4))
plt.ylim((0.5,1))
my_y_ticks = np.arange(0.5,1,0.05)
plt.yticks(my_y_ticks)
p1=sns.barplot( data=df, x='Lambda', y='Classification
Accuracy',palette="Blues_d")
plt.show()
```



选择参数：代码见code/train.py

- 朴素贝叶斯分类器对数据进行训练和测试
- 输出模型的准确率
- 打印精确度、召回率、F1值等指标

```
x_train, x_test, y_train, y_test =
train_test_split(news.data,news.target,test_size=0.15,random_state=1)
vec = CountVectorizer()
x_train = vec.fit_transform(x_train)
x_test = vec.transform(x_test)
mnb = MultinomialNB(alpha=0.0001)
mnb.fit(x_train, y_train)
y_predict = mnb.predict(x_test)

print("Accuracy:", mnb.score(x_test, y_test))
print("Indicators: \n",classification_report(y_test, y_predict,
target_names=news.target_names))
```

遇到的问题及解决方法

- HTTPError: HTTP Error 403 : Forbidden
  - 使用news = fetch_20newsgroups(subset="all")获取新闻数据集时出现的报错，下载数据集并按照图示方式放在对应位置，再运行就可以了