Lab 8 Naïve Bayes

学 号	17363011	学 院	专 业
姓 名	陈政培	智能工程学院	智能科学与技术

1、实验目的： **利用 python 的文本处理能力将文档切分成词，通过集合元素的唯一性生成词列表（不包含重复词汇），进而构建词向量（词集向量或词袋向量），从词向量计算概率，然后构建分类器对邮件文档进行垃圾邮件分类**

2、实验环境：python3.6.8、vs code

3、实验步骤：

① 切分文本词

② 生成词汇表

提供的 bayes.py 有少量的语法错误和格式错误，需要修正后正常使用

③ 生成词向量

在 bayes.py 文件中，setOfWords2Vec 函数中

```
returnVec[vocabList.index(word)] = 1
```

仅实现了统计这个词语是否出现，为了让词向量能够统计量我们将代码更改为

```
returnVec[vocabList.index(word)] += 1
```

④ 训练算法，计算概率

计算出概率 P(c1)、P(w|c1)、P(w|c0)

⑤ 利用 50 封邮件进行测试和训练

因为每一次训练集和测试集都是随机生成的，所以每一次运行结果可能不太相同

4、实验结果与分析：

① 切分词——textParse()函数内容实现

```
['This', 'book', 'is', 'the', 'test', 'book', 'on', 'Python', 'on', 'M.L.', 'I',
 'have', 'ever', 'laid', 'eyes', 'upon.']
['This', 'book', 'is', 'the', 'test', 'book', 'on', 'Python', 'on', 'M', 'L', 'I
', 'have', 'ever', 'laid', 'eyes', 'upon', '']
['This', 'book', 'is', 'the', 'test', 'book', 'on', 'Python', 'on', 'M', 'L', 'I
', 'have', 'ever', 'laid', 'eyes', 'upon']
```

② 生成词汇表

```
['ate', 'licks', 'dalmation', 'take', 'problems', 'park', 'worthless', 'how', 'n
ot', 'garbage', 'please', 'stop', 'cute', 'dog', 'has', 'I', 'buying', 'my', 'po
sting', 'steak', 'to', 'stupid', 'mr', 'him', 'flea', 'maybe', 'so', 'love', 'is
', 'quit', 'food', 'help']
```

③ 生成词向量

```
the word: T is not in my vocabulary
the word: h is not in my vocabulary
the word: i is not in my vocabulary
the word: s is not in my vocabulary
This
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0]
the word: t is not in my vocabulary
the word: h is not in my vocabulary
the word: e is not in my vocabulary
the
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0]
```

由于单词 This 和单词 the 并没有在词汇表中出现所以向量全部为 0

④ 计算概率

```
 0.5
 [0.03846154 0.03846154 0.07692308 0.03846154 0.07692308 0.07692308
  0.07692308 0.07692308 0.07692308 0.07692308 0.03846154 0.07692308
  0.03846154 0.07692308 0.03846154 0.07692308 0.07692308 0.07692308
  0.07692308 0.03846154 0.03846154 0.07692308 0.11538462 0.07692308
  0.07692308 0.03846154 0.07692308 0.07692308 0.03846154 0.15384615
  0.07692308 0.03846154]
 [0.0952381  0.14285714 0.04761905 0.0952381  0.04761905 0.04761905
  0.0952381  0.04761905 0.04761905 0.04761905 0.19047619 0.04761905
  0.0952381  0.04761905 0.0952381  0.04761905 0.04761905 0.04761905
  0.04761905 0.0952381  0.0952381  0.04761905 0.0952381  0.14285714
  0.0952381  0.0952381  0.04761905 0.04761905 0.0952381  0.04761905
  0.04761905 0.0952381 ]
```

⑤ 测试与训练

```
classfication error ['bargains', 'here', 'buy', 'phentermin', 'buy', 'genuine',
'phentermin', 'low', 'cost', 'visa', 'accepted', '130', '219', '292', '120', '36
6', '180', '513']
classfication error ['home', 'based', 'business', 'opportunity', 'knocking', 'yo
ur', 'door', 'don', 'rude', 'and', 'let', 'this', 'chance', 'you', 'can', 'earn'
, 'great', 'income', 'and', 'find', 'your', 'financial', 'life', 'transformed',
'learn', 'more', 'here', 'your', 'success', 'work', 'from', 'home', 'finder', 'e
xperts']
classfication error ['codeine', 'the', 'most', 'competitive', 'price', 'net', 'c
odeine', 'wilson', '30mg', '156', 'codeine', 'wilson', '30mg', '291', 'freeviagr
a', 'pills', 'codeine', 'wilson', '30mg', '396', 'freeviagra', 'pills', 'codeine
', 'wilson', '30mg', '120', '492', 'freeviagra', 'pills']
the error rate is  0.3
```

本次运行 error rate 是 0.3，也出现过 0.1 或者 0.5

5、作业：

① 利用 sklearn 中 BernoulliNB 分类该邮件数据集

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score
```

主要使用的就是 sklearn 库中的这两个，一个是 BernoulliNB 模型，一个是用于测试正确率的 accuracy_score

其他数据预处理，训练集测试集随机分配都是沿用的 bayes.py 中的函数

测试结果:

```
PS C:\Users\93744\Desktop\新建文件
sktop/新建文件夹/BernoulliNB.py
the accuracy is  8
PS C:\Users\93744\Desktop\新建文件
sktop/新建文件夹/BernoulliNB.py
the accuracy is  9
PS C:\Users\93744\Desktop\新建文件
sktop/新建文件夹/BernoulliNB.py
the accuracy is  10
```

训练正确率相对比 naïve bayes 稳定高一些，稳定一些

② Bayes.py 中语句更换

详见代码文件 bayes.py

③ 将向量集用 TF-IDF 词向量替代，测试分析结果

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

采用了 sklearn 中的几个转换器，其中 CountVectorizer 进行词频统计，TfidfTransformer 将矩阵转化为 TF-IDF 矩阵

但是无论如何调试数据类型，生成的 TF 矩阵总是无法和调用的分类器相互匹配，出现数据类型的报错

```
1109, in transform
    self._check_vocabulary()
  File "D:\Python36\lib\site-packages\sklearn\feature_extraction\text.py", line
389, in _check_vocabulary
    check_is_fitted(self, 'vocabulary_', msg=msg),
  File "D:\Python36\lib\site-packages\sklearn\utils\validation.py", line 914, in
 check_is_fitted
    raise NotFittedError(msg % {'name': type(estimator).__name__})
sklearn.exceptions.NotFittedError: TfidfVectorizer - Vocabulary wasn't fitted.
```

④ 编程实现

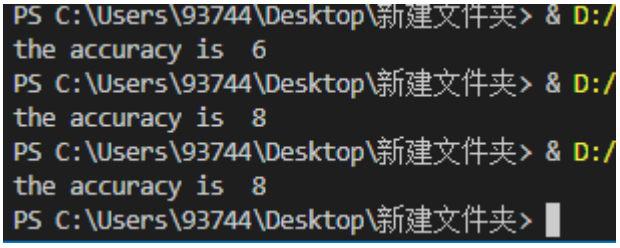
⑤ 基于 tensorflow 实现朴素贝叶斯分类器

由于是基于 tensorflow1 的代码基础进行更改的，所以在 tensorflow2 中无法正常运行

```
tf.distributions.Normal(loc=mean, scale=tf.sqrt(var))
```

此函数 distributions 属性只在 1.0 版本有定义，2.0 版本取消了这个属性并且没有制作新的同类型的参数

在 1.0 环境下运行 accuracy 结果



相比 BernoulliNB 性能会差一些

6、实验总结：

- TF-IDF 和普通的词袋模型还是有一定差异的，尤其是在 sklearn 库中提供的 TF-IDF 转换生成的是一个根据输入数据变化维度的矩阵
- 先用词袋模型筛选出一些高热度词汇，再用 TF-IDF 计算其权值，得到词袋模型中词汇的 TF-IDF 值，值越高说明该词区分每条语句的效果越好。两者联合使用的效果会更好
- Sklearn.naive bayes 库中除了 BernoulliNB 还有 GaussianNB 等不同的分类器，使用方法基本类似，都是先生成对应 xxxNB()的类，然后调用相关函数方法

7、参考文献

1. https://yq.aliyun.com/articles/408869
2. https://www.jianshu.com/p/0422853b57a8
3. https://stackoverflow.com/questions/55010025/tensorflow-probability-error-attributeerror-module-tensorflow-probability-ha
4. https://www.cnblogs.com/to-creat/p/6888295.html
5. https://scikit-learn.org/dev/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB