人工智能实验报告

姓名：孟衍璋 学号：16337183

一、请描述一下本次实验的主要内容:

（1）包括实验做的是什么任务？

使用朴素贝叶斯分类器对邮件进行分类，判断其是否为垃圾邮件。用的数据集分为训练集和测试集，其中训练集包括 700 份邮件，测试集包括 260 份邮件，垃圾邮件的比例均为 50%。邮件内容经过一定处理之后，按每三个数一组，表明在第几份邮件中的编号第几的单词出现的次数。之后利用训练集算出 phi_(k|y=1)和 phi_(k|y=0)。在测试集中计算 log p (x | y = 1) + log p(y = 1) 与 log p(x | y = 0) + log p(y = 0)哪个更大，从而判断测试邮件是否为垃圾邮件。

（2）用的哪个数据集？

ex3DataPrepared

（3）采用什么分类模型？

朴素贝叶斯分类器

二、Python 实现代码

```
from scipy import sparse
from numpy import *
import numpy

# 700 * 2500 的矩阵
numTrainDocs = 700
numTokens = 2500
```

```
with open('train-features.txt','r') as f:
    # 将文件中的数全部存入一个 list 中，其中每三个数第一个代表矩阵的行、第二个代表矩阵的列、第三个代表矩阵对应位置的数据
    ele = f.read().split()
    # print(list)
    # 将行单独列出来，从第一个数开始每隔三个数取一个。因为 python 中数组下标从 0 开始，所以需要做减 1 的操作
    row = [int(x) - 1 for x in ele[::3]]
    # 将列单独列出来，从第一个数开始每隔三个数取一个。因为 python 中数组下标从 0 开始，所以需要做减 1 的操作
    col = [int(x) - 1 for x in ele[1::3]]
    # 将数据项单独列出来，即每个单词出现的次数
    data = [int(x) for x in ele[2::3]]
    # 将数据项存入稀疏矩阵中
    spmatrix = sparse.coo_matrix((data, (row, col)), shape = (numTrainDocs, numTokens))
    train_matrix = spmatrix.toarray()
    # print(spmatrix)
    # print(train_matrix)

train_labels = list()
# 每份邮件是否为垃圾邮件
M = numpy.loadtxt('train-labels.txt',dtype = int)
for i in M:
    train_labels.append(i)
# print(train_labels)

# 垃圾邮件与非垃圾邮件的份数
spam_indices = nonspam_indices = 0
for i in train_labels:
    if(i == 1):
        spam_indices += 1
    else:
        nonspam_indices += 1

# 垃圾邮件出现的概率
prob_spam = spam_indices / numTrainDocs

# 每份邮件的单词数
email_lengths = numpy.sum(train_matrix, axis = 1)
# print(email_lengths)

# 垃圾邮件与非垃圾邮件出现的单词的个数
spam_wc = numpy.sum(email_lengths[spam_indices:])
```

```
nonspam_wc = numpy.sum(email_lengths[:nonspam_indices])
# print(spam_wc, nonspam_wc)

# Now the k-th entry of prob_tokens_spam represents phi_(k|y=1)
prob_tokens_spam = (numpy.sum(train_matrix[spam_indices:], axis = 0) +
1) / (spam_wc + numTokens)
# Now the k-th entry of prob_tokens_nonspam represents phi_(k|y=0)
prob_tokens_nonspam = (numpy.sum(train_matrix[:nonspam_indices], axis
= 0) + 1) / (nonspam_wc + numTokens)
# print(prob_tokens_spam)
# print(prob_tokens_nonspam)



# from scipy import sparse
# import numpy
import math

#  打开文件 test-features 并将其数据存入矩阵中
with open('test-features.txt', 'r') as f:
    ele = f.read().split()
    row = [int(x) - 1 for x in ele[::3]]
    col = [int(x) - 1 for x in ele[1::3]]
    data = [int(x) for x in ele[2::3]]
    spmatrix = sparse.coo_matrix((data, (row, col)))
    test_matrix = spmatrix.toarray()
    # print(spmatrix)
    # print(test_matrix)

numTestDocs = len(test_matrix)
numTokens = len(test_matrix[0])
# print(numTestDocs,numTokens)

output = list()
log_a = numpy.dot(test_matrix,(list(map(math.log,prob_tokens_spam)))) +
math.log(prob_spam)
log_b = numpy.dot(test_matrix,(list(map(math.log,prob_tokens_nonspam))))
+ math.log(1 - prob_spam)
for i in range(len(log_a)):
    if(log_a[i] > log_b[i]):
        output.append(1)
    else:
        output.append(0)
# print(output)
```

```
#  打开 test-labels 文件并将其中数据存入 test_labels 中
with open('test-labels.txt', 'r') as f:
    ele = f.read().split()
    test_labels = list()
    for i in ele:
        test_labels.append(int(i))
# print(test_labels)

#  计算判断错误的邮件的数量和比例
numdocs_wrong = 0
for i in range(len(output)):
    if(output[i] ^ test_labels[i] == 1):
        numdocs_wrong += 1
print('numdocs_wrong =', numdocs_wrong)
fraction_wrong = numdocs_wrong / numTestDocs
print('fraction_wrong =', fraction_wrong)
```