**华中科技大学计算机科学与技术学院**

**机器学习报告**



专 业： 计算机科学与技术

班 级： 计算机1805班

学 号： U201814780

姓 名： 叶蕴锋

成 绩：

指导教师： 邹复好

**完成日期： 2020年6 月21**

# 机器学习结课报告

## 一、实验题目：基于朴素贝叶斯分类器的语音性别识别

## 二、实验要求

用朴素贝叶斯分类器进行数字手写体识别(基于MINIST数据集)。通过朴素贝叶斯方法，可以先对所有特征值做统计，并且通过离散特征方法得到参数。之后使用预测函数预测测试集。

其中数据集可自行在<https://www.kaggle.com/primaryobjects/voicegender>下载。这个数据集是基于对男女语音段进行合理的声音预处理而得到的语音特征(并不包含原始语音段)。集合中共有3168条数据，男女各1584条，每条数据可视作一个长度为21的一维数组。其中前20个数值是这条语音的20个特征值，这些特征值包括了语音信号的长度、基频、标准差、频带中值点/一分位频率/三分位频率等；最后一个数值是性别标记。元数据集中直接以字符串,即male和female进行标注。

基于朴素贝叶斯分类器的语音性别识别结果展示方式：

任务1：对于缺失数据处理的结果

任务2：训练集为2000例，分别进行50次测试，输出测试结果，并绘制50次结果的曲线图

任务3：输出不同数量的训练集测试结果，并绘制曲线图

## 算法设计

本次参数估计采用的是离散特征

算法实现：

### 数据的读入与处理

本次实验采用的数据集格式为csv格式，所以需要在头文件中加入import csv以便于后续csv文件的读入。其次在csv文件读入后，我们去观察csv文件内容，可以很清晰的看见在部分位置的数据是为0的，我们认为这里是数据缺失。为了避免由于数据缺失在这里造成的影响，我们对如何处理数据缺失问题采取了几种尝试：

1. 忽略数据缺失，继续进行实验
2. 用数据平均值对数据进行填充
3. 分别用男性平均值和女性平均值对数据进行填充
4. 用KNN算法对缺失数据进行填充
5. 删除缺失数据

几种方法尝试下来，可以发现忽略数据缺失，继续进行实验和用数据平均值对数据进

行填充的准确度大概维持在87%-89%之间，而用KNN算法对缺失数据进行填充虽随着k值变化而有所不同，但准确度依旧不超过91%，而分别用男性平均值和女性平均值对数据进行填充和删除缺失数据的准确度都在91%左右，但删除缺失数据往往会对实验产生影响，最终选择的方法是分别用男性平均值和女性平均值对数据进行填充。

### 切分训练集和测试集

本数据集共有3168条数据，为了让男性声音样本和女性声音样本随机地分布到训练和测试数据集，这里采用随机算法随机选取训练数据集。同时为了方便后续对训练集样本数量对于准确率的研究，这里数据集数量分别采取了20个点，由100至2000，每隔100取值，而剩余样本将作为测试集进行测试。

### 朴素贝叶斯算法的实现

由于csv文件中有20个特征值，在研究20个特征值时我们不妨将其分类为F1，F2 ,F3 ......,F20 其中F1i1表示特征F1取第i1个分类。

则一个测试样本向量可以表示如下：

test\_vector = [F1i1 , F2i2 , F3i3  .....  F19i19 , F20i20]

则其是男性的概率为：

P(男 | F1i1 ✕ F2i2 ✕ F3i3  .....  F19i19 ✕ F20i20)

= P(F1i1 ✕ F2i2 ✕ F3i3  .....  F19i19 ✕ F20i20 | 男) \* P(男) / P(F1i1 ✕ F2i2 ✕ F3i3  .....  F19i19 ✕ F20i20)

同理其是女性的概率为

P(女 | F1i1 ✕ F2i2 ✕ F3i3  .....  F19i19 ✕ F20i20)

= P(F1i1 ✕ F2i2 ✕ F3i3  .....  F19i19 ✕ F20i20 | 女) \* P(女) / P(F1i1 ✕ F2i2 ✕ F3i3  .....  F19i19 ✕ F20i20)

最后，只需要比较男性的概率与女性的概率即可。

### 结果展示

本次实验结果展示分为6张图片，前三张分别为2000个训练集，1168个测试集进行测试，共测试五十次的情况。后三张图片则是不同数量训练集，程序准确率比较的折线图。

## 四、实验环境与平台

本实验环境为：

硬件环境：Intel® CoreTM i7

软件环境：Python 3.7.1

实验平台为：Pycharm

## **五、程序实现**

def native\_bayes(train\_set,test\_set,new\_data\_set,list\_class):

#train\_set 训练集数据

#test\_set 测试集数据  
 list\_classes\_1 = []  
 train\_data\_1 = []  
 list\_classes\_0 = []  
 train\_data\_0 = []  
 train\_mat = []  
 train\_classes = []  
 for index in train\_set:  
 train\_mat.append(new\_data\_set[index])  
 train\_classes.append(list\_class[index])  
 test\_mat = []  
 test\_classes = []  
 for index in test\_set:  
 test\_mat.append(new\_data\_set[index])  
 test\_classes.append(list\_class[index])  
 num\_train\_data = len(train\_mat)  
 num\_feature = len(train\_mat[0])  
 p\_1\_class = sum(train\_classes) / float(num\_train\_data) # p\_1\_class: 任一样本分类为1的概率  
 p\_0\_class = 1 - p\_1\_class #p\_0\_class: 任一样本分类为0的概率  
 n = N + 1  
 for i in list(range(num\_train\_data)):  
 if train\_classes[i] == 1:  
 list\_classes\_1.append(i)  
 train\_data\_1.append(train\_mat[i])  
 else:  
 list\_classes\_0.append(i)  
 train\_data\_0.append(train\_mat[i])  
 train\_data\_1 = np.matrix(train\_data\_1)  
 p\_1\_feature = {}  
 for i in list(range(num\_feature)):  
 feature\_values = np.array(train\_data\_1[:, i]).flatten()  
  
 feature\_values = feature\_values.tolist() + list(range(n))  
 p = {}  
 count = len(feature\_values)  
 for value in set(feature\_values):  
 p[value] = np.log(feature\_values.count(value) / float(count))  
 p\_1\_feature[i] = p #p\_1\_feature: 类别为1的情况下所有特征所有取值的概率  
  
  
 train\_data\_0 = np.matrix(train\_data\_0)  
 p\_0\_feature = {}  
 for i in list(range(num\_feature)):  
 feature\_values = np.array(train\_data\_0[:, i]).flatten()  
 feature\_values = feature\_values.tolist() + list(range(n))  
 p = {}  
 count = len(feature\_values)  
 for value in set(feature\_values):  
 p[value] = np.log(feature\_values.count(value) / float(count))  
 p\_0\_feature[i] = p #p\_0\_feature: 类别为0的情况下所有特征所有取值的概率  
 return p\_1\_feature, p\_1\_class, p\_0\_feature, p\_0\_class  
  
  
def test\_bayes(a):  
 file\_name = 'data/voice.csv'  
 data = []  
 male\_data = []  
 female\_data = []  
 list\_class = []  
 train\_mat = []  
 test\_mat = []  
 train\_classes = []  
 test\_classes = []  
 csv\_reader = csv.DictReader(open(file\_name, encoding='utf-8'))  
 label\_name = list(csv\_reader.fieldnames)  
 num = len(label\_name) - 1  
 male\_count=0  
 female\_count=0  
 for line in csv\_reader.reader:  
 data.append(line[:num])  
 if line[-1][0] == 'm':  
 gender = 1.0  
 male\_count+=1  
 male\_data.append(line[:num])  
 else:  
 gender = 0.0  
 female\_count+=1  
 female\_data.append(line[:num])  
 list\_class.append(gender)  
  
 data = np.array(data).astype(float)  
 male\_data=np.array(male\_data).astype(float)  
 female\_data=np.array(female\_data).astype(float)  
 min\_vector = data.min(axis=0)  
 max\_vector = data.max(axis=0)  
 diff\_vector = max\_vector - min\_vector  
 diff\_vector /= 9  
 male\_sum\_vector = np.sum(male\_data, axis=0)  
 male\_mean\_vector = male\_sum\_vector / male\_count  
 female\_sum\_vector = np.sum(female\_data, axis=0)  
 female\_mean\_vector = female\_sum\_vector / female\_count

#缺失数据填充  
 for row in range(len(data)):  
 for col in range(num):  
 if data[row][col] == 0.0:  
 if list\_class[row]==0.0:  
 data[row][col] = male\_mean\_vector[col]  
 else:  
 data[row][col] = female\_mean\_vector[col]

new\_data\_set = []  
 for i in range(len(data)):  
 line = np.array((data[i] - min\_vector) / diff\_vector).astype(int)  
 new\_data\_set.append(line)  
 test\_set = list(range(len(new\_data\_set)))  
 train\_set = []

# 随机划分数据集为训练集 和 测试集  
 for i in range(a):  
 random\_index = int(np.random.uniform(0, len(test\_set)))  
 train\_set.append(test\_set[random\_index])  
 del test\_set[random\_index]

# 训练数据集  
 for index in train\_set:  
 train\_mat.append(new\_data\_set[index])  
 train\_classes.append(list\_class[index])

# 测试数据集  
 for index in test\_set:  
 test\_mat.append(new\_data\_set[index])  
 test\_classes.append(list\_class[index])

p\_1\_feature, p\_1\_class, p\_0\_feature, p\_0\_class = native\_bayes(train\_set,test\_set,new\_data\_set,list\_class)  
 male\_accurate = 0.0  
 male\_wrong = 0.0  
 female\_accurate = 0.0  
 female\_error = 0.0  
 male\_num = 0.0  
 female\_num = 0.0  
 for i in list(range(len(test\_mat))):  
 test\_vector = test\_mat[i]  
 sum\_1 = 0.0  
 sum\_0 = 0.0  
 for j in list(range(len(test\_vector))):  
 sum\_1 += p\_1\_feature[j][test\_vector[j]]  
 sum\_0 += p\_0\_feature[j][test\_vector[j]]  
 p1 = sum\_1 + np.log(p\_1\_class)  
 p0 = sum\_0 + np.log(p\_0\_class)  
 if p1 > p0:  
 result=1  
 else:  
 result=0  
  
 if test\_classes[i] == 1:  
 male\_num += 1  
 if result == test\_classes[i]:  
 male\_accurate += 1  
 else:  
 male\_wrong += 1  
 else:  
 female\_num += 1  
 if result == test\_classes[i]:  
 female\_accurate += 1  
 else:  
 female\_error += 1

#准确率计算  
 male\_accurate\_rate.append(male\_accurate / male\_num)  
 male\_wrong\_rate.append(male\_wrong / male\_num)  
 female\_accurate\_rate.append(female\_accurate / female\_num)  
 female\_error\_rate.append(female\_error / female\_num)  
 total\_accurate\_rate.append((male\_accurate + female\_accurate) / (male\_num + female\_num))  
 return (male\_accurate + female\_accurate) / (male\_num + female\_num)

#准确率折线图  
N = 10  
a = np.array([100,200,300,400,500,600,700,800,900,1000,1100,1200,1300,1400,1500,1600,1700,1800,1900,2000])  
male\_accurate\_rate = []  
male\_wrong\_rate = []  
female\_accurate\_rate = []  
female\_error\_rate = []  
total\_accurate\_rate = []  
for k in a:  
 test\_bayes(k)  
i=1  
for i in range(20):  
 print("训练集数量为%d" %(a[i]))  
 print("男性正确率：", male\_accurate\_rate[i-1])  
 print("男性错误率：", male\_wrong\_rate[i-1])  
 print("女性正确率：", female\_accurate\_rate[i-1])  
 print("女性错误率：", female\_error\_rate[i-1])  
 print("总正确率：", total\_accurate\_rate[i-1])  
  
plt.figure()  
plt.plot(a,total\_accurate\_rate)  
plt.title('total\_accurate\_rate')  
plt.xlabel('train\_num')  
plt.ylabel('rate')  
plt.tick\_params(axis='both',labelsize=14)  
  
plt.figure()  
plt.plot(a,male\_accurate\_rate)  
plt.title('male\_accurate\_rate')  
plt.xlabel('k')  
plt.ylabel('train\_num')  
plt.tick\_params(axis='both', labelsize=14)  
  
plt.figure()  
plt.plot(a,female\_accurate\_rate)  
plt.title('female\_accurate\_rate')  
plt.xlabel('train\_num')  
plt.ylabel('rate')  
plt.tick\_params(axis='both', labelsize=14)  
  
i = 0  
N = 10  
k=2000  
male\_accurate\_rate = []  
male\_wrong\_rate = []  
female\_accurate\_rate = []  
female\_error\_rate = []  
total\_accurate\_rate = []  
  
a=0.0  
for i in range(50):  
 a+=test\_bayes(k)  
a=a/50  
print(a)  
  
for i in range(50):  
 print("第%d次：" % (i + 1))  
 print("男性正确率：", male\_accurate\_rate[i])  
 print("男性错误率：", male\_wrong\_rate[i])  
 print("女性正确率：", female\_accurate\_rate[i])  
 print("女性错误率：", female\_error\_rate[i])  
 print("总正确率：", total\_accurate\_rate[i])  
  
plt.figure()  
plt.plot(total\_accurate\_rate)  
plt.title('total\_accurate\_rate')  
plt.xlabel('times')  
plt.ylabel('rate')  
plt.tick\_params(axis='both',labelsize=14)  
  
plt.figure()  
plt.plot(male\_accurate\_rate)  
plt.title('male\_accurate\_rate')  
plt.xlabel('times')  
plt.ylabel('rate')  
plt.tick\_params(axis='both', labelsize=14)  
  
plt.figure()  
plt.plot(female\_accurate\_rate)  
plt.title('female\_accurate\_rate')  
plt.xlabel('times')  
plt.ylabel('rate')  
plt.tick\_params(axis='both', labelsize=14)  
plt.show()

## 实验结果

### 训练集数量对准确率的影响

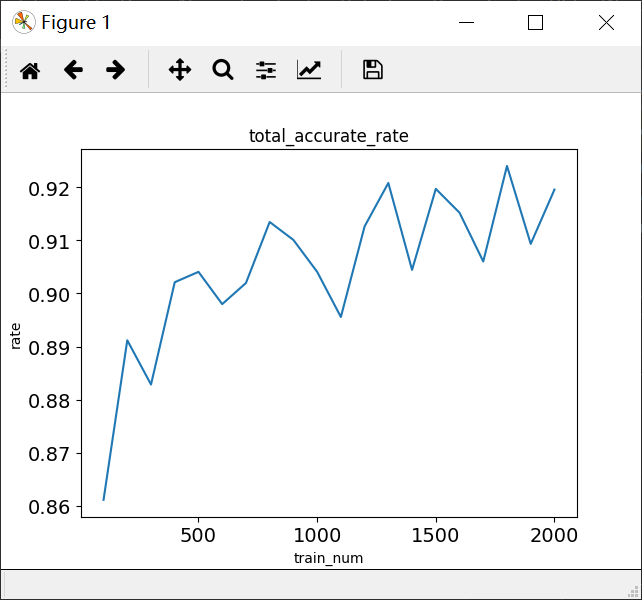


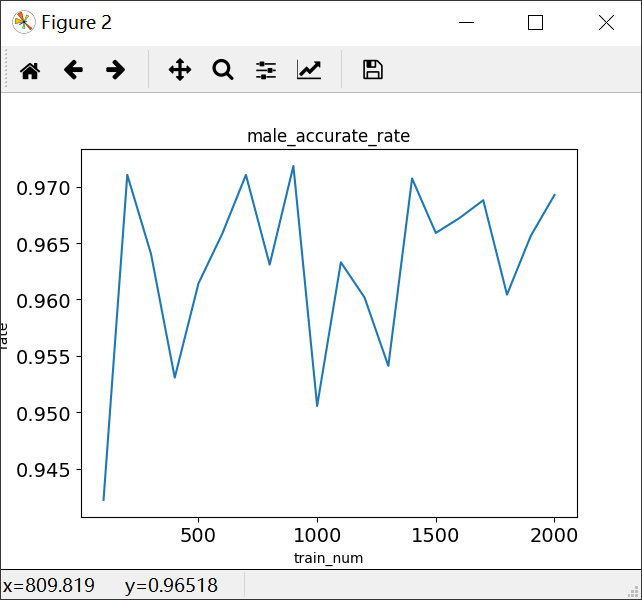
图 1 总准确率与测试集数量的关系

图 2 男性准确率与测试集数量的关系

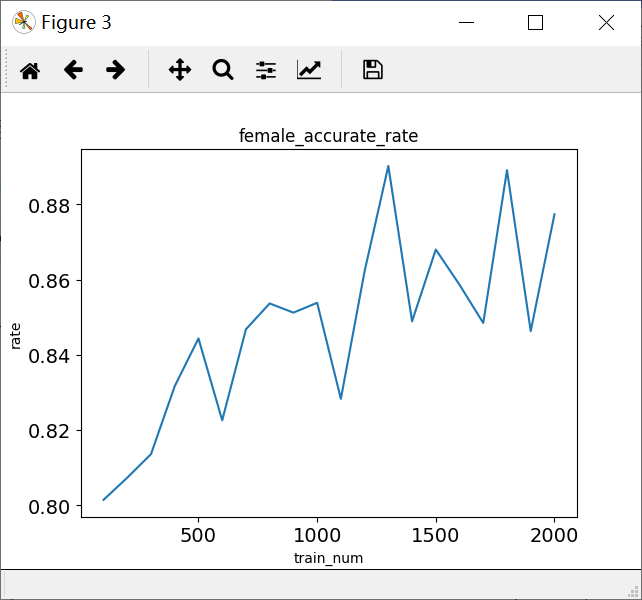


图 3 女性准确率与测试集数量的关系

### 二．五十次测试的准确率结果

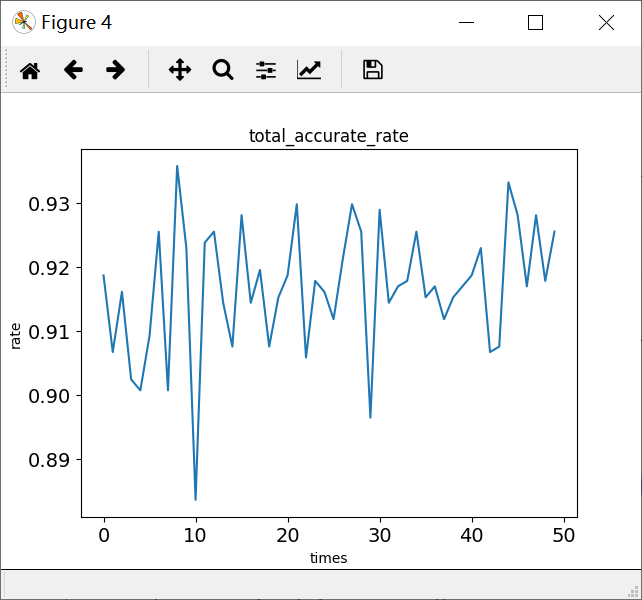


图 4 50次测试的总准确率

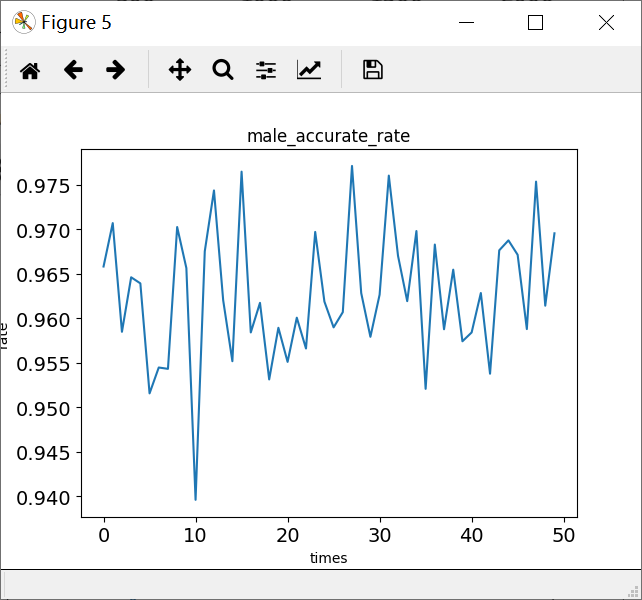


图 5 50次测试的男性准确率

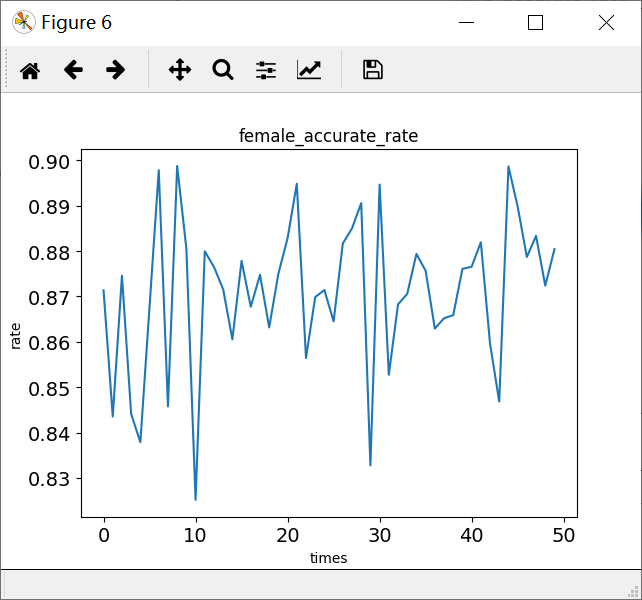


图 6 50次测试的女性准确率

## 结果分析

实验分析：

1.从准确率来看，总准确率在92%左右，男性准确率在96%左右，而女性准确率在87%左右，最终得出结果是男性准确率远高于女性准确率。我们研究其中的缘由，从部分文集中查阅可以了解到女性语音的基频较之更高，所以采样更稀疏，相当于丢失了更多的信息，所以识别准确率更低。

2.从训练集数量可以看到，当给与程序更多训练集数量后，其准确率会大幅增加，特征提取会更加精确，猜想当训练集数量足够多时，其准确率可能会最终趋近于某个值。

误差分析：

本次实验我认为主要的误差和改进的地方在于部分数据的缺失上，在本次实验中，我采用的是分别用男性和女性的平均值去代替缺失地方的数据，但其实仍然会带来相当大的误差。我认为为了提高最后的准确率，还可以尝试采用特征加权、增加特征值离散程度等措施来对我们的实验进行改进。