2019 模式识别

大作业报告

学生姓名：吴　迪

学生学号：　(此处省略)

前言

问题描述

作业 2019-09-30. 用多种数据进行性别分类的实验（一）

用 dataset3 作为训练数据，用 dataset4 作为测试数据，采用不同的特征、训练样本数、分类方法进行比较实验，观察、分析实验结果的异同。

– 要求采用的特征组合：a) 10 个特征都用；b) 任意选取其中两列特征（要在报告中说明是哪两个特征）。

– 要求试验的训练样本：a) 从 dataset3 中任选 20 个训练样本（男女各 10 例）； b) dataset3 中的全部训练样本。

– 要求试验的分类器方法：a) 最小错误率贝叶斯分类器（假设正态分布，先验概率各 50%）；b) Fisher 线性判别（FLD）；c) SVM（核函数自定）；d) 采用 BP 算法的 MLP 神经网络（网络结构自定）

- 对测试错误率用下表汇总实验结果

训练样本数	特征数	Bayes	FLD	Linear SVM	MLP
10+10	10				
	2				
469+485	10				
	2				

- 用所选出的两维特征画出样本的分布，设法在其中画出几组实验得到的分类边界（如画在一起不清楚可以画在多幅图上）。结合实验观察和对各种方法特点的理解，尝试对训练样本数、特征维数以及所选用的方法与测试结果的关系进行分析和讨论。

- 与选用了不同特征的同学进行讨论，比较使用不同特征的结果，尝试分析其中的原因和规律。

- 提示：在某些样本数和特征维数下，正常结果可能会相当不好，请注意鉴别和分析是实验错误还是正常情况，如是正常情况请尝试分析原因。

目录

CONTENT

实验内容与原理

1. 特征选择

特征选择的准则

特征选择，能剔除不相关、冗余、没有差异刻画能力的特征，从而达到减少特征个数、减少训练或者运行时间、提高模型精确度的作用。

在机器学习的实际应用中，特征数量可能较多，其中可能存在不相关的特征，特征之间也可能存在相关性，容易导致如下的后果：

（1） 特征个数越多，分析特征、训练模型所需的时间就越长，模型也会越复杂。
（2） 特征个数越多，容易引起“维度灾难”，其**泛化能力**会下降。
（3） 特征个数越多，容易导致机器学习中经常出现的特征稀疏的问题，导致模型效果下降。

特征选取的方法

（1）**过滤式**：选择与目标变量相关性较强的特征。缺点：忽略了特征之间的关联性。

我选择了 x 2 （**卡方检验**）作为这一类型的代表特征选取方法。

经典的卡方检验是检验定性自变量对定性因变量的相关性。假设自变量有N种取值，因变量有M种取值，考虑自变量等于i且因变量等于j的样本频数的观察值与期望的差距，构建统计量：

$$x 2=\sum (A-E)2/E x 2$$

A为实际值， E为理论值，求和值为理论值与实际值的差异程度。

基本思想是根据样本数据推断总体的分布与期望分布是否有显著性差异，或者推断两个分类变量是否相关或者独立。某个特征的 x 2越大，则关联性越强，这个特征就越需要保留。

sklearn.feature_selection.SelectKBest和sklearn.feature_selection .chi2这两个包提供了卡方检验的方法，将参数k设为2，就可以选取两个最好的特征。

将dataset3中的十个特征分别命名为A-J这十个英文字母，经过卡方检验，得出与label关联性最强的两个的特征分别为**A和B**。

（2）**包裹式：**基于线性模型相关系数以及模型结果AUC逐步剔除特征。如果剔除相关系数绝对值较小特征后，AUC无大的变化，或降低，则可剔除。**从最终的学习器性能来看，包裹式特征选择比过滤式特征选择更好**。但是另一方面，由于在特征选择过程中需多次训练学习期，因此**包裹式特征选择的计算开销通常要大得多**

这里我选择了RFE,即递归消除特征法。它使用一个基模型来进行多轮训练，每轮训练后，消除若干权值系数的特征，再基于新的特征集进行下一轮训练。它使用模型精度来识别哪些属性（和属性组合)对预测目标属性的贡献最大。

调用sklearn.feature_selection.RFE和sklearn.svm.SVR，使用svm作为基模型，设置step = 1,每迭代一次剔除不重要的一个特征，经过几次迭代后，得到最重要的两个特征是A和E。

（3）**嵌入型：**利用模型提取特征，但是和包裹式选择不同的是学习器在学习的过程中会不断的进行特征选择，直到满足要求为止。

这里我选择了RandomForest来作为特征选择的学习器。输出每个特征的重要性，当调整RandomForest的两个重要参数n_estimators(树的个数)和max_features(每棵树所用的最大特征数)时，有四个特征的重要性存在交替领先的状态，并且它们的重要性远远大于其余六个特征，这四个特征分别是A B C E，重要性几乎相等，因此可以初步任选其中两个作为最重要的特征。

2. 特征选择

特征提取的原理

特征提取，作为机器学习中一个前处理步骤，在**降维、去除不相关数据**，增加学习精度和提高结果可理解性方面非常有效。原始数据常常是高维的，其中包含了许多冗余信息或者十分稀疏或者计算量大，拿原始数据来训练可行，但是往往直接训练是低效的。所以特征提取往往是必要的。

注：特征提取主要是为了解决下面三个问题，
（1）原始数据特征中的强相关性造成的冗余信息。
（2）原始数据十分稀疏。
（3）原始数据维度巨大。

实验内容与原理

特征提取的方法

1. 数据清洗

数据清洗是指发现并纠正数据文件中可识别的错误以及通过处理得到建模过程需要数据的过程。

数据清洗包含：

（1）缺失值处理。
（2）异常值检测与处理。
（3）调配样本比例和权重。

经大致观察，**dataset3中的数据无缺失值但是有噪声**，即异常值，所以需要对异常值经行处理。

2. 数据无量纲化

无量纲化使不同规格的数据转换到同一规格，在某些比较和评价的指标处理中经常会用到，去除数据的单位限制，将其转化为无量纲的纯数值，便于不同单位或量级的指标能够进行比较和加权。

数据无量纲化常用方法有：
（1）标准化方法。
（2）极值化方法。
（3）均值化方法。
（4）标准差化方法。

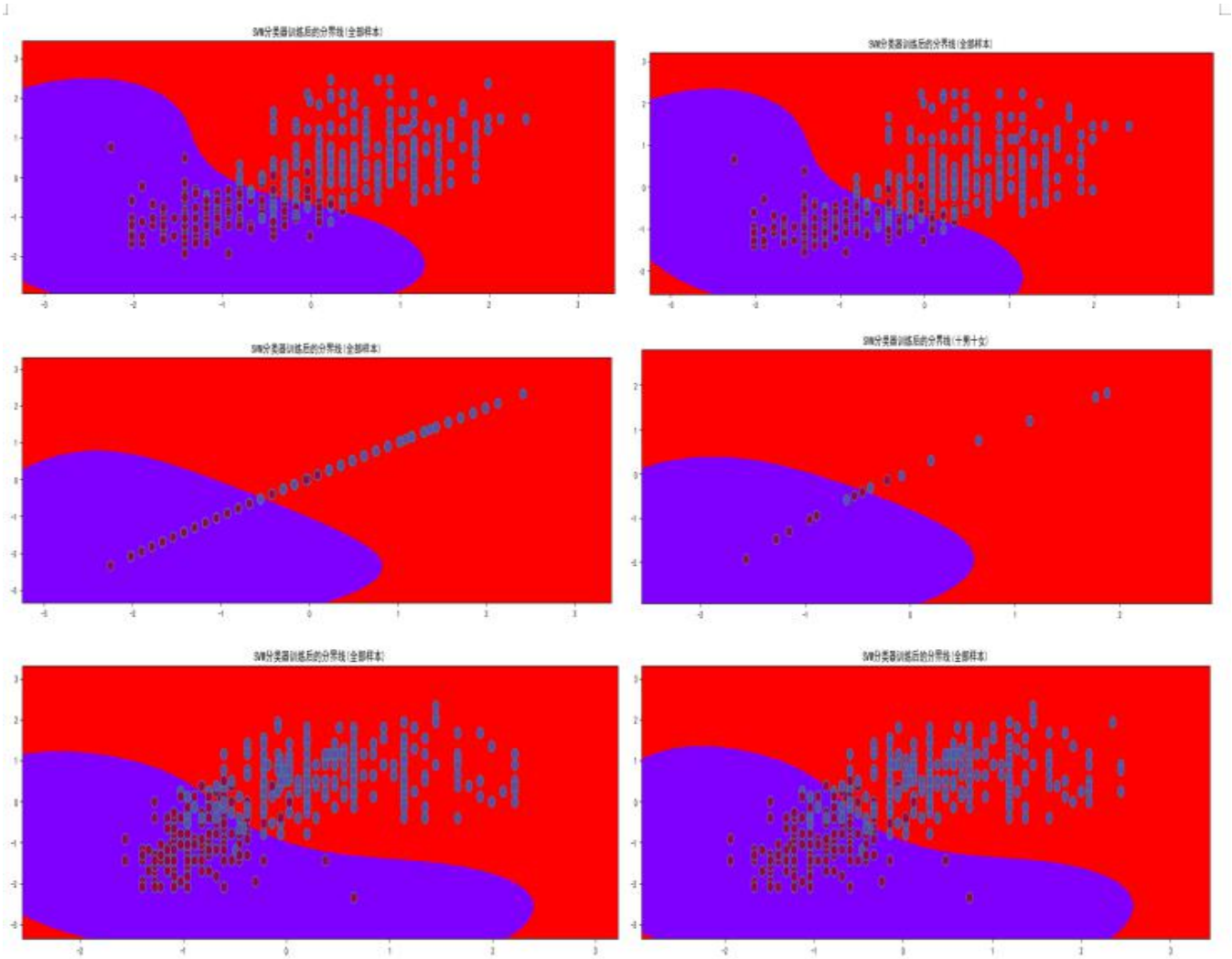
这里我将数据清洗和数据无量纲化放在一起处理

首先将dataset3中的每个特征的数据都分别做了**Z-score标准化**，即利用公式 $x_i' = \frac{x_i - \bar{x}}{s}$，把每个特征的数据子集都转化为均值为0，方差为1的数据子集。

然后根据根据**3σ原则**去修改异常数据，但是通过观察数据集，发现存在一些数据的绝对值在2.5~3之间，所以对于这个数据集，我采用了**2.5σ原则**，即修改绝对值大于2.5的数据，修改值选为每个特征数据子集的平均值。

3. 选取最好的两个特征

根据特征选择的方法，有四个特征A B C E，它们两两之间的组合都可能是最好的两个特征，所以在进行数据预处理即将所有数据标准化后，我选取了它们所有的两两组合来训练，结合SVM分类器画出它们样本点的分布以及决策线。



上述六张图从左到右，从上到下的特征组合分别为AB,AC,AE,BC,BE,CE。可见，中间两张图中特征之间的**线性相关性最强**。它们分别为AE,BC。这时比较它们在分类器上的**错误率**，分别为0.11855670103092786和0.134020618556701，所以选择错误率较小的**AE**组合W为最好的特征组合。

实验方法和设计

训练集和测试集的划分

1. 标签的处理

dataset3中的label有M,F和f，这不便于对原始数据进行训练，为了后续的方便，我将所有的**M用1替代，将F和f用0替代**。

```
df1.label = df1.label.apply(lambda x:1 if x == 'M' else 0)
```

2. 划分训练集和测试集

我将dataset3中所有的样本利用**Handout原理**进行了随机划分，**0.7倍**的样本数为训练集，其余为测试集。

```
x_train, x_test, y_train, y_test = train_test_split(df1[feature],
df1['label'], test_size = 0.3,random_state=42)

dataset3_2_train = pd.concat([x_train,y_train], axis=1)
dataset3_2_test = pd.concat([x_test,y_test], axis=1)
```

3. 十男十女样本的选择

将dataset3中所有标签为0的样本（女）和标签为1的样本（男）分别提取出来作为**两个表**。
然后再分别从两个表中**随机提取**十个样本作为另外两个表。最后将得到的另外两个拼接成一张表即得到**十男十女的随机样本**。

```
df1_F = df1[df1.label == 0]
df1_F10 = df1_F.sample(n = 10)
df1_M = df1[df1.label == 1]
df1_M10 = df1_M.sample(n = 10)
df1_10 = pd.concat([df1_F10,df1_M10])
```

4. 交叉验证（cross validation）

（1）我们将训练集（trainning set）分为三份，当然我们也可以分为五份，十份甚至更多，这个可以自己指定，我们就以三份为例。（实际程序中我选择了十份，也就是十折交叉验证）
（2）我们选择其中两份作为训练集，另外一份作为验证集，这样就产生了三种情况。1和2预测3,1和3预测2,2和3预测1，那么我们最终的结果就是三个模型的平均值，均值更能代表我们模型的准确性。这是一个边训练边测试的过程，能使我们更好的把握模型的准确性。

```
score2 = cross_val_score(model,df3[feature],df3['label'],cv = 10).mean()
```

模型的选择

1. 最小错误率贝叶斯

（1）原理

贝叶斯定理具体形式为：

$$P(B_i|A)=\frac{P(B_i)P(A|B_i)}{\sum_{j=1}^{n}P(B_j)P(A|B_j)}$$

P(A)是 A 的先验概率，P(A|B)是已知 B 发生后 A 的条件概率，也是类条件概率，P(B|A)是已知 A 发生后 B 的条件概率，也由于得自 A 的取值而被称作 B 的后验概率。

我们一般用下面的公式：

$P(A|Bj)=P(Bj)\prod P(Ai|Bj)$ 其中 i为元组 A 的列序号，$i=1,2,\ldots,n$ 公式②

求出$P(A|Bj)P(Bj)$之后，因为P(A)对每一个类别都相同，因此可通过$P(A|Bj)P(Bj)$对$P(Bj|A)$的数值进行比较，因此就有：$y=\max\{P(Bj|A)\}$得出概率最大的即为相应的预测类别。

（2）训练

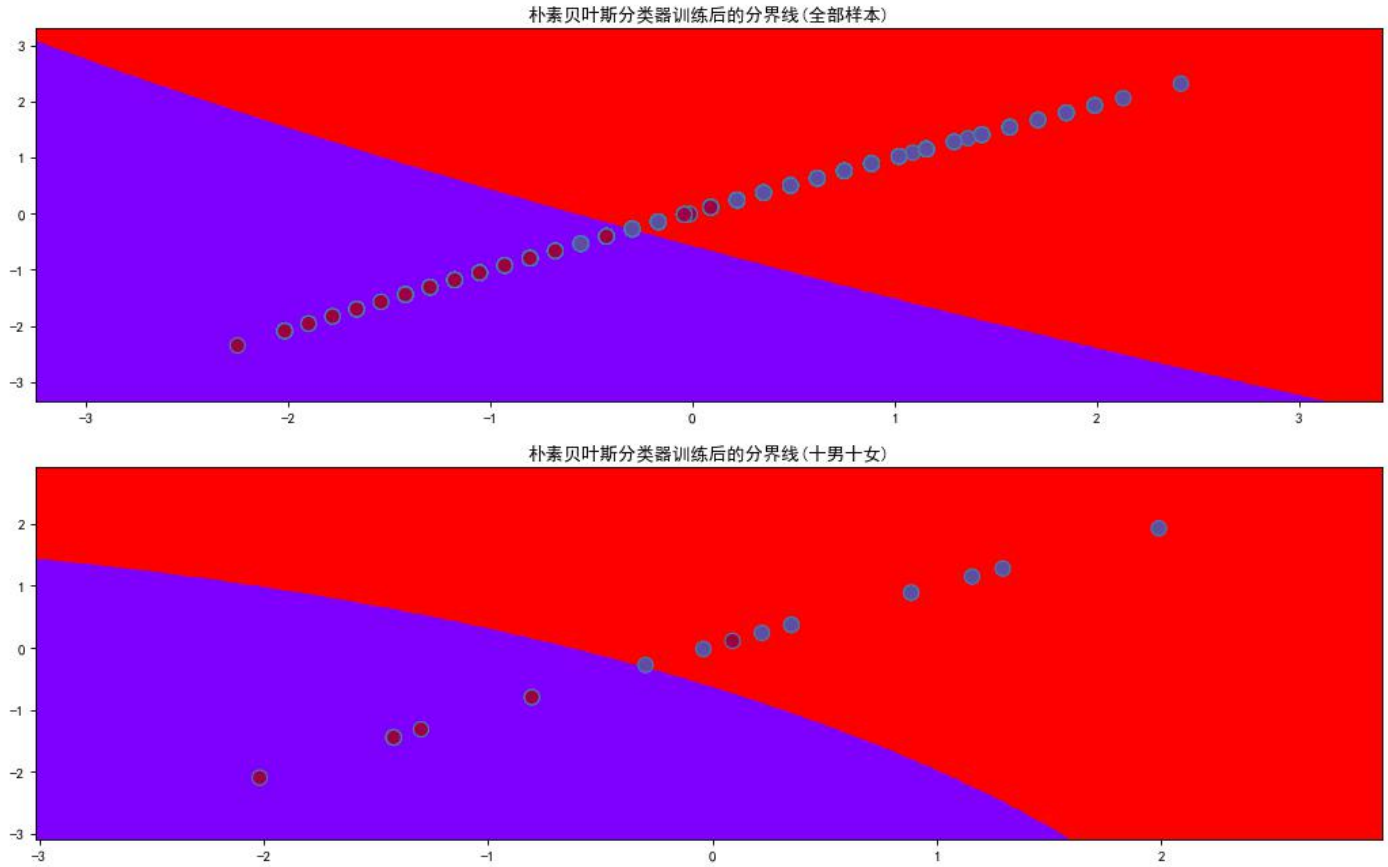
python中有对应最小错误率贝叶斯的包，GaussianNB，它的主要参数是**先验概率** priors，这个值默认不给出，如果不给出此时P(Y=Ck)=mk/m。其中m为训练集样本总数量，mk为输出为第k类别的训练集样本数。这里，我没有给出专门的先验概率，即默认用P(Y=Ck)=mk/m算出先验概率。

用最小概率贝叶斯对训练集进行训练。最后得到的错误率如下表。

表中一个格子有两个分数的，前一个分数为Handout得到的测试集的错误率，后一个为**交叉验证**得到的错误率。只有一个分数的为在对训练集训练并预测后的错误率，由于**十男十男是随机取样**，所以每次运行代码后的错误率并不一定一致。

训练样本数	特征数	Bayes
10+10	10	0.10
	2	0.050
469+485	10	0.098 0.102
	2	0.108 0.116

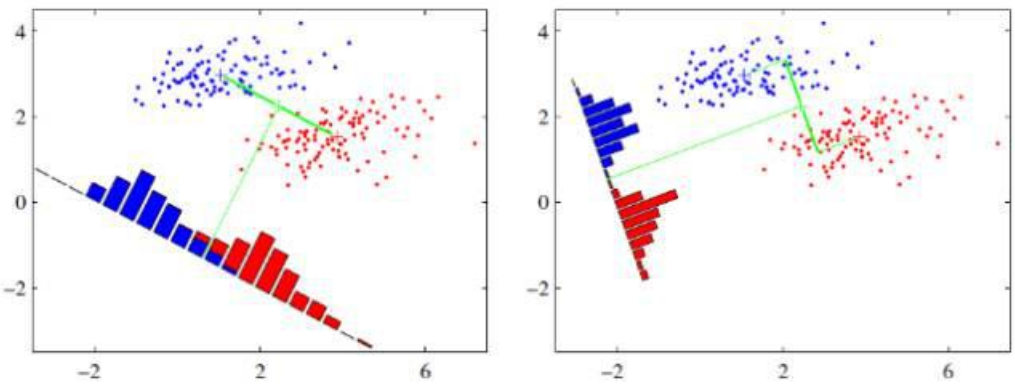
当选取A,E两个特征时，样本点的分布，以及分界线如下



2. Fisher 线性判别（FLD）

（1）原理：

假设有两类数据，分别为红色和蓝色，如下图所示，这些数据特征是二维的，希望将这些数据投影到一维的一条直线，让每一种类别数据的投影点尽可能的接近，而红色和蓝色数据中心之间的距离尽可能的大。



（2）训练：

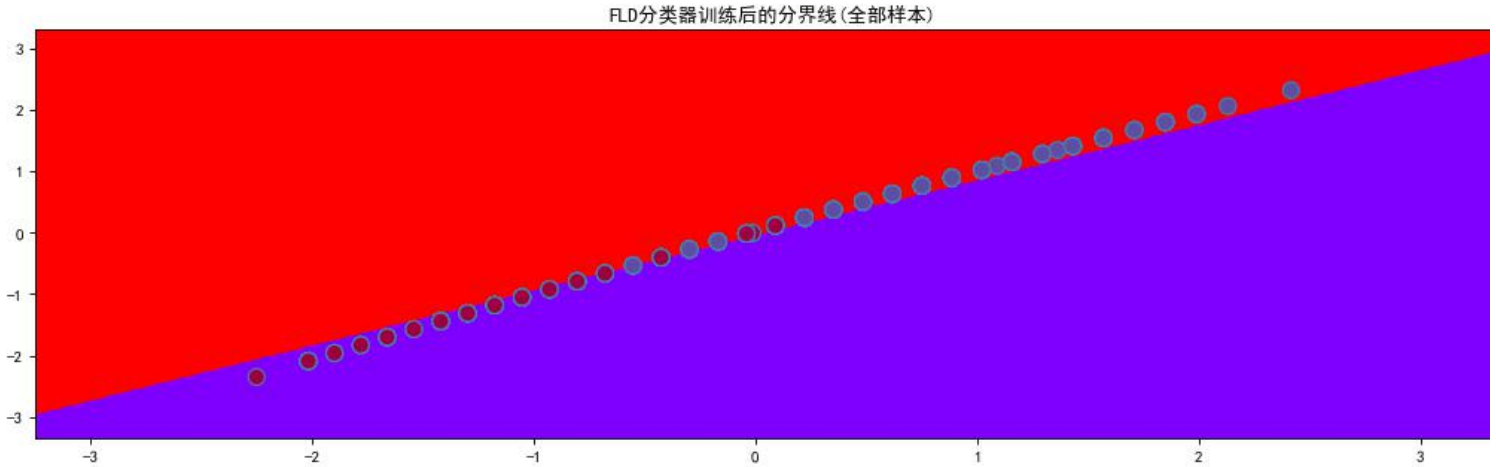
python中有对应的FLD包，LinearDiscriminantAnalysis，它的主要参数和最小错误率贝叶斯一样是先验概率，这里我也不填，即默认用P(Y=Ck)=mk/m算出先验概率。

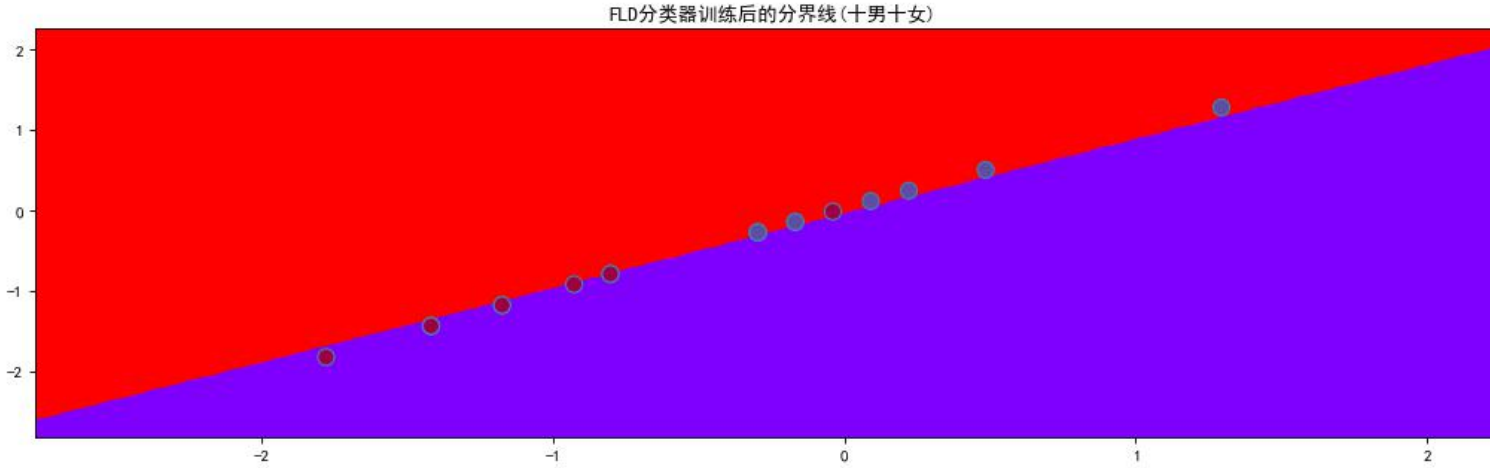
用Fisher 线性判别对训练集进行训练。最后得到的错误率如下表。

表中一个格子有两个分数的，前一个分数为Handout得到的测试集的错误率，后一个为**交叉验证**得到的错误率。只有一个分数的为在对训练集训练并预测后的错误率,由于**十男十男是随机取样**，所以每次运行代码后的错误率并不一定一致。

训练样本数	特征数	Fisher
10+10	10	0
	2	0.05
469+485	10	0.097 0.103
	2	0.118 0.121

当选取A,E两个特征时，样本点的分布，以及分界线如下





3. SVM

（1）原理

SVM的核心思想是尽最大努力使分开的两个类别有**最大间隔**，这样才使得分隔具有更高的可信度。而且对于未知的新样本才有很好的**泛化能力**，SVM的核心办法是：**让离分隔面最近的数据点具有最大的距离**。

（2）训练

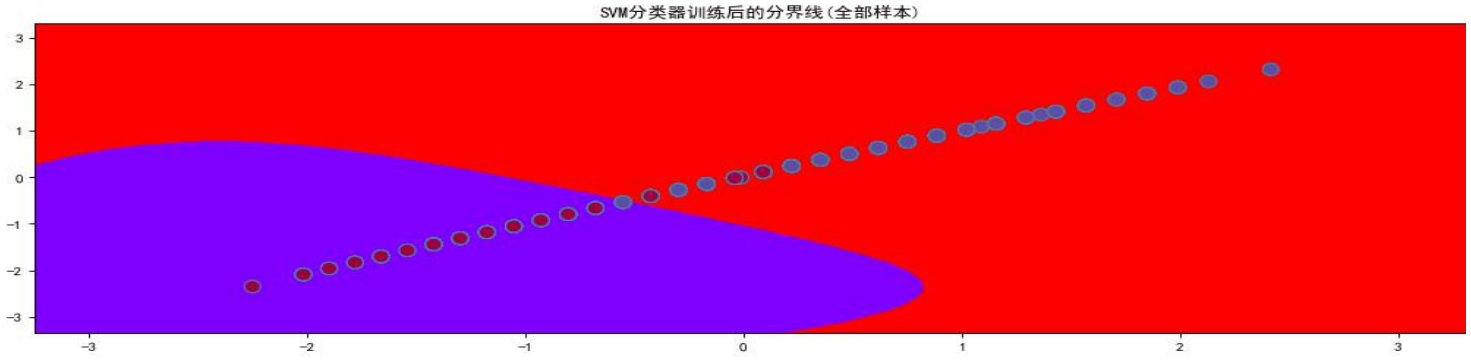
这里我选取了python中的包svm.SVC,其中最重要的参数是核函数，我选择了普遍来讲泛化能力比较好的**径向基核函数（Radial Basis Function）也叫高斯（Gaussian Kernel）**

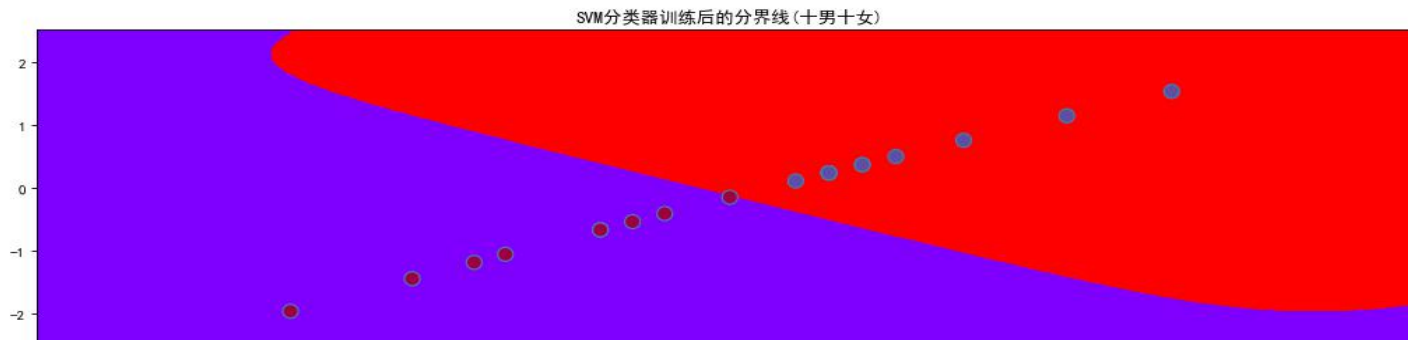
用SVM对训练集进行训练。最后得到的错误率如下表。

表中一个格子有两个分数的，前一个分数为Handout得到的测试集的错误率，后一个为**交叉验证**得到的错误率。只有一个分数的为在对训练集训练并预测后的错误率,由于**十男十男是随机取样**，所以每次运行代码后的错误率并不一定一致。

训练样本数	特征数	SVM
10+10	10	0
	2	0.05
469+485	10	0.087 0.089
	2	0.118 0.119

当选取A,E两个特征时，样本点的分布，以及分界线如下





4. 采用BP算法的MLP神经网络

（1）原理

BP神经网络是一种按误差逆传播算法训练的多层前馈网络，是目前应用最广泛的神经网络模型之一。BP网络能学习和存贮大量的 输入-输出模式映射关系，而无需事前揭示描述这种映射关系的数学方程。它的学习规则是使用最速下降法，通过反向传播来不断 调整网络的权值和阈值，使网络的误差平方和最小。每次根据训练得到的结果与预想结果进行误差分析，进而修改权值和阈值，一步一步得到能输出和预想结果一致的模型。

（2）训练

这里我选取了python中的包MLPClassifier,对于简单的二分类问题来说，我选择了MLPClassifier的默认参数来决定神经网络的结构

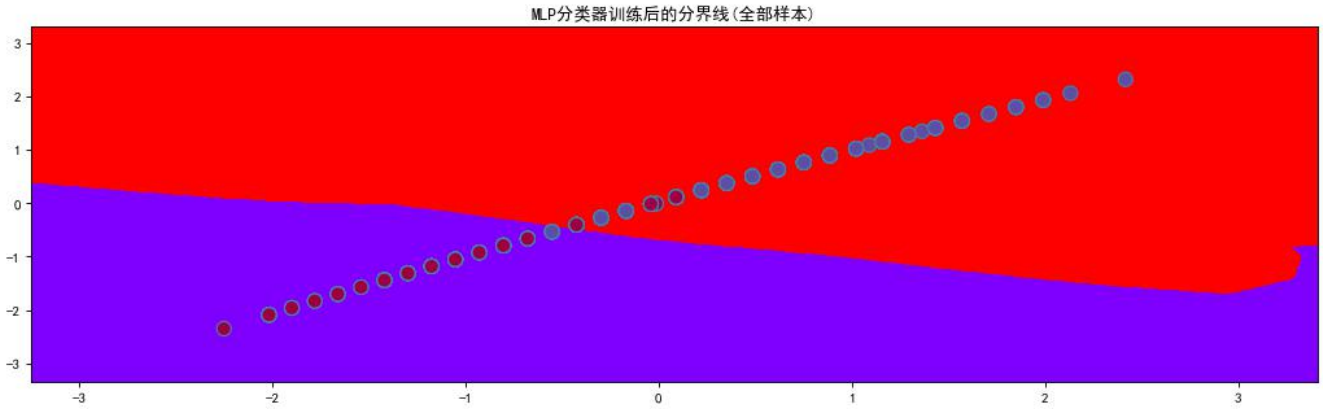
用BP算法的MLP神经网络对训练集进行训练。最后得到的错误率如下表。

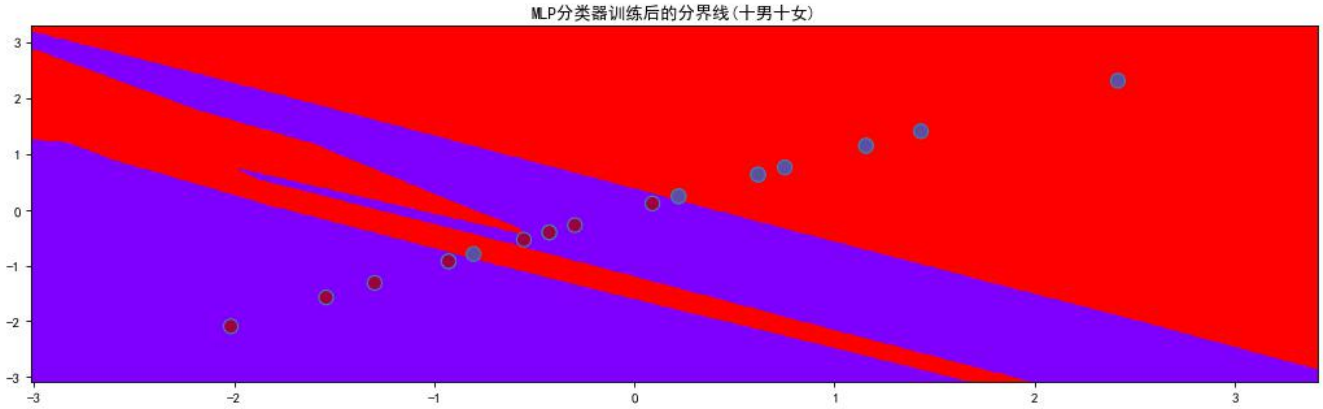
表中一个格子有两个分数的，前一个分数为Handout得到的测试集的错误率，后一个为**交叉验证**得到的错误率。只有一个分数的为在对训练集训练并预测后的错误率,由于**十男十男是随机取样**，所以每次运行代码后的错误率并不一定一致。

训练样本数	特征数	BP
10+10	10	0
	2	0.
469+485	10	0.15 0.13
	2	0.11 0.12

当选取A,E两个特征时，样本点的分布，以及分界线如下

实验方法和设计


MLP分类器训练后的分界线(全部样本)
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3


MLP分类器训练后的分界线(十男十女)
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

实验结果与讨论

1. 测试错误率汇总结果

表中一个格子有两个分数的，前一个分数为Handout得到的测试集的错误率，后一个为**交叉验证**得到的错误率。只有一个分数的为在对训练集训练并预测后的错误率,由于**十男十男是随机取样**，所以每次运行代码后的错误率并不一定一致。

训练样本数	特征数	Bayes	FLD	Linear SVM	MLP
10+10	10	0.10	0	0	0
	2	0.050	0.05	0.05	0.
469+485	10	0.098 0.102	0.097 0.103	0.087 0.089	0.15 0.13
	2	0.108 0.116	0.118 0.121	0.118 0.119	0.11 0.12

2. 结果分析

（1）结果呈现

在全部训练集上，四个模型的错误率相差不大，基本都在0.1附近。

在十男十女的样本上，MLP，SVM,,FLD在有些情况下甚至出现0错误率。

（2）分析

1）对于469+485样本，贝叶斯，FLD,SVM,MLP表现几乎一致，这也印证了“特征工程决定上限，模型只是逼近这个上限”这句话。并且通过cross validation(交叉验证),尽量避免了**overfitting（过拟合）**，使得模型的泛化能力得到保证。但是，可以发现，特征数为2的时候，模型的错误率基本要大于特征数为10时模型的错误率，这可能时**underfitting（欠拟合）**的现象。由于在大量数据中只用了两个特征，所以模型不能很好地拟合数据，就出现了欠拟合的现象。

2）在小样本上，模型错误率出现0的情况，很有可能是过拟合，由于样本数量太小，很容易出现**overfitting**的现象，尤其是当样本数少但特征数比较多的时候，但是当特征数为2时，四个模型中只有一个出现了0错误率，说明此时特征适当减少的时候，过拟合的现象有所缓解。

3. 得出结论

1）当样本数比较少而且特征数比较多的时候，模型进行拟合时容易出现overfitting，有效的解决方法是**适当减少样本特征来缓解**overfitting

2）当样本数比较多而且特征数比较少的时候，模型进行拟合容易出现underfitting，有效的解决方法是适**当增加样本特征来缓解**underfitting

3）**增加样本数量**和**适当增加样本特征数**可以在一定程度上**既避免**overfitting**又避免**underfitting。

程序运行方法说明

程序运行的顺序

1.特征选择
1）运行feature_select2.py,进行ｘ2 （卡方检验)提取特征。输出的两列特征，经过与dataset3对比可知为A,B这两个特征。

2）运行feature_select3.py，进行RFE特征提取，得到一个true和false列表，显示为true的位置对应的特征即为选出的特征，经对应，选出的特征是A,B。

3）运行feature_select4.py，进行RandomForestClassifier选取特征，得到十个特征的重要程度，改变RandomForestClassifier的参数，多次运行这个程序，发现前四个特征为A,B,C,E且重要程度几乎一样。

2.特征提取
1）运行preprocessing.py,进行数据预处理，并且划分测试集和验证集，处理过的总集，训练集和测试集分别被保存为了dataset3_2.csv，dataset3_2_train.csv，dataset3_2_test.csv。

2）运行SVM.py,将第17行的列表feature_中的元素分别改为A,B,C,E中两两的组合，总共运行6次，得到不同的样本分布图，最后确定A,E为最佳特征组合。

3.模型训练
分别运行nativebayes.py，FLD.py，SVM.py，MLP.py，得到相应的样本分布图以及模型错误率。

4.预测
运行predict.py,得到预测的csv文件dataset4_2.csv

程序运行结果截图

1. 特征选择

1）运行feature_select2.py

```
[[259.21  21.16]
 [256.    31.36]
 [265.69  25.  ]
 ...
 [313.29  39.69]
 [324.    54.76]
 [299.29  24.01]]
```

2）运行feature_select3.py

```
In [21]: runfile('D:/pattern recognition/code/code/feature_select3.py', wdir='D:/
pattern recognition/code/code')
[ True False False False  True False False False False False]
```

3）运行feature_select4.py

```
In [22]: runfile('D:/pattern recognition/code/code/fe
pattern recognition/code/code')
 1) B                            0.195568
 2) A                            0.187972
 3) C                            0.187512
 4) E                            0.187499
 5) D                            0.061160
 6) J                            0.042029
 7) I                            0.039319
 8) F                            0.034380
 9) H                            0.034339
10) G                            0.030222
```

```
In [23]: runfile('D:/pattern recognition/code/code/feature_select4.py', wdir='D:/
pattern recognition/code/code')
 1) B                            0.231522
 2) C                            0.223108
 3) E                            0.167844
 4) A                            0.124015
 5) D                            0.059744
 6) J                            0.052811
 7) H                            0.040196
 8) I                            0.038707
 9) F                            0.032275
10) G                            0.029778
```

```
In [24]: runfile('D:/pattern recognition/code/code/feature_select4.py', wdir='D:/
pattern recognition/code/code')
 1) C                            0.224884
 2) B                            0.200517
 3) A                            0.188628
 4) E                            0.147624
 5) D                            0.057467
 6) J                            0.041834
 7) I                            0.039507
 8) H                            0.034147
 9) F                            0.034008
10) G                            0.031387
```

```
In [25]: runfile('D:/pattern recognition/code/code/feature_select4.py', wdir='D:/
pattern recognition/code/code')
 1) B                            0.197414
 2) C                            0.194322
 3) A                            0.186621
 4) E                            0.177767
 5) D                            0.063323
 6) J                            0.042231
 7) I                            0.039352
 8) H                            0.035011
 9) F                            0.034203
10) G                            0.029756
```

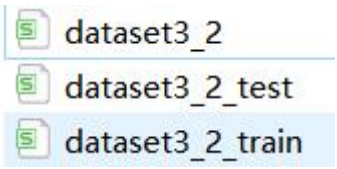
程序运行方法说明

2 特征提取

1）运行preprocessing.py

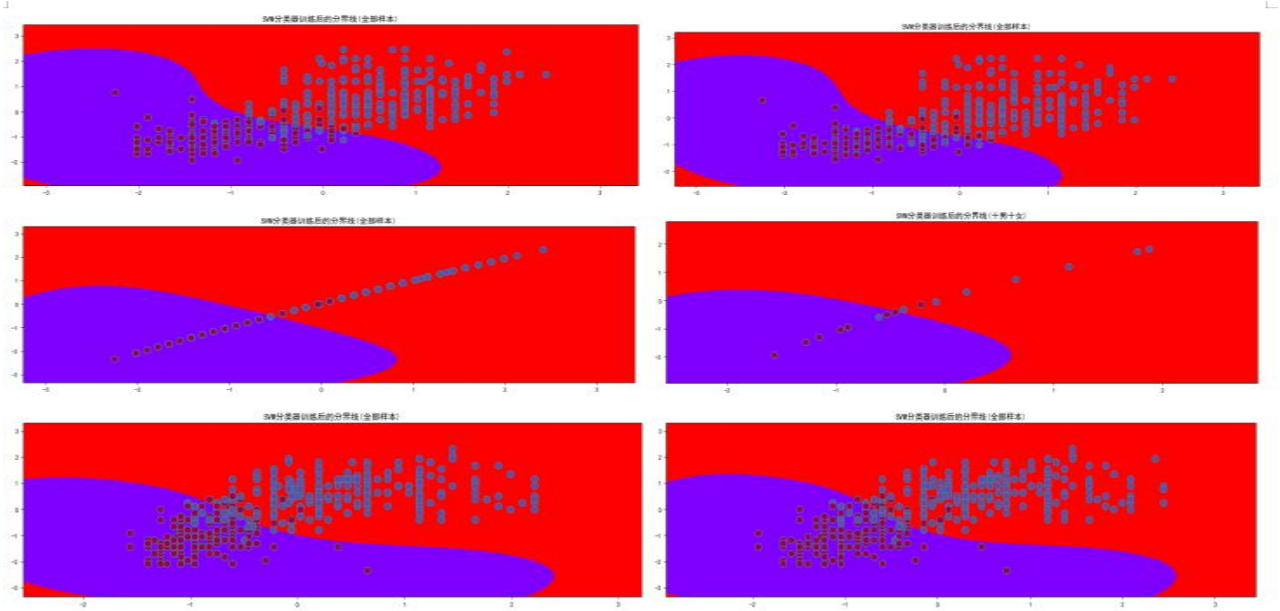
```
In [26]: runfile('D:/pattern recognition/code/code/preprocessing.py', wdir='D:/pattern
recognition/code/code')
D:\anaconda\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This
module was deprecated in version 0.18 in favor of the model_selection module into which
all the refactored classes and functions are moved. Also note that the interface of the
new CV iterators are different from that of this module. This module will be removed in
0.20.
  "This module will be removed in 0.20.", DeprecationWarning)

In [27]:
```



-1.052697	-0.958423	-1.042181	-0.748153	-1.053483	0.3473161	-1.05099	2.3720982	0.6481119	-0.93605	0
-0.297691	-0.678702	-0.684284	-0.71635	-0.276378	0.2253448	0.6434603	-0.327174	1.4657361	0.5646688	0
-1.419966	-1.090214	-1.221129	-0.772889	-1.442035	-0.26254	-0.694263	-1.504089	-1.124756	0.3981658	0
-1.298301	-0.45479	-0.415861	0.3720363	-1.312518	0.4692875	0.1975524	1.9202111	0.8262082	-0.536588	0
-1.419966	-0.958423	-1.042181	-0.497259	-1.442035	-1.360282	0.911005	-0.047004	-0.483205	-0.451146	0
-0.552392	-0.750649	-0.773758	-0.684546	-0.535413	1.4450579	-0.069992	-0.299057	-0.167489	-1.127382	0
-0.169204	-0.218776	-0.147439	-0.062611	-0.14686	0.2253448	-1.229353	-0.464749	0.1158466	0.206103	0
-1.175878	-0.958423	-1.042181	-0.666878	-1.183	-0.140569	-0.5159	1.8499175	-1.201662	-0.635906	0
-0.169204	-0.750649	-0.773758	-0.917772	-0.14686	1.8109719	0.4650971	1.2433845	-1.835118	0.4193438	0
-0.928758	-0.89051	-0.952707	-0.698681	-0.923965	-0.140569	0.7326419	-0.903582	-1.272496	-1.473533	0
-1.419966	-1.154093	-1.310604	-0.910704	-1.442035	-0.140569	0.7326419	0.1417841	0.1279895	-0.122521	0

2）运行SVM.py



程序运行方法说明

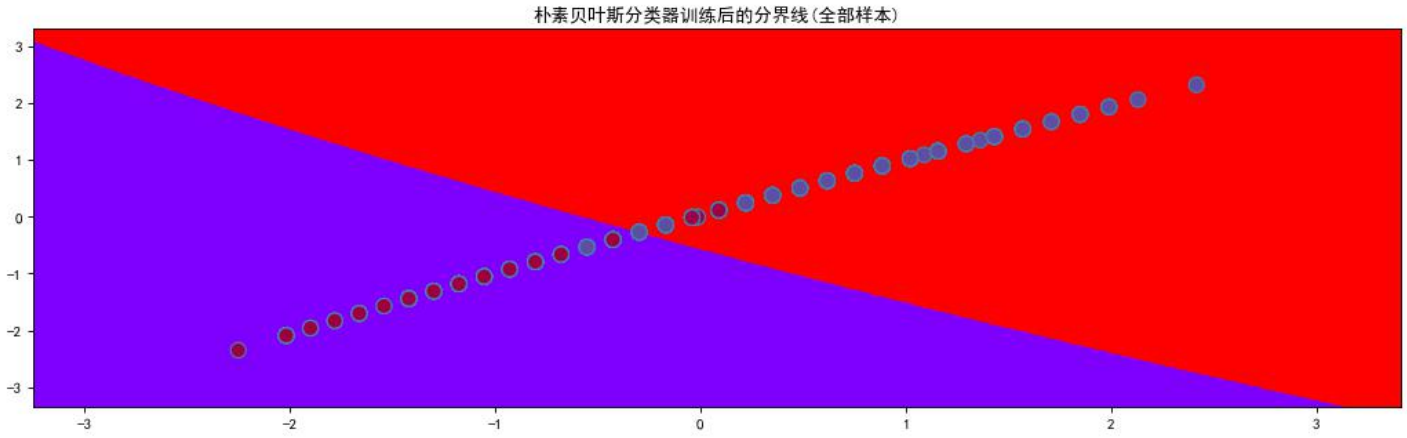
3.模型训练

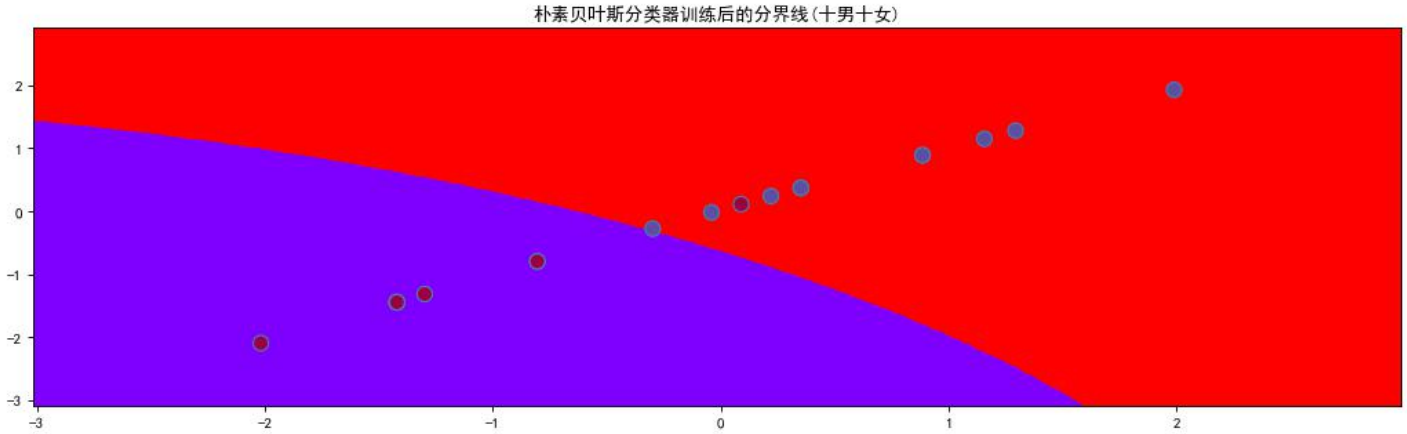
分别运行nativebayes.py，FLD.py，SVM.py，MLP.py

1）nativebayes.py

```
In [1]: runfile('D:/pattern recognition/code/code/nativebayes.py', wdir='D:/pattern
recognition/code/code')
测试集上Holdout验证错误率为（十个特征）0.09793814432989689
十折交叉验证错误率为（十个特征）0.09440486596736586
训练集上Holdout验证错误率为0.050000000000000044(十个特征，十男十女)
测试集上Holdout验证错误率为0.10824742268041232,使用的特征为（两个特征）A,E
十折交叉验证错误率为0.11601689976689966,使用的特征为（两个个特征）A,E
<Figure size 720x2880 with 0 Axes>
```

训练集上Holdout验证错误率为0.050000000000000044,使用的特征为（两个特征，十男十女）A,E



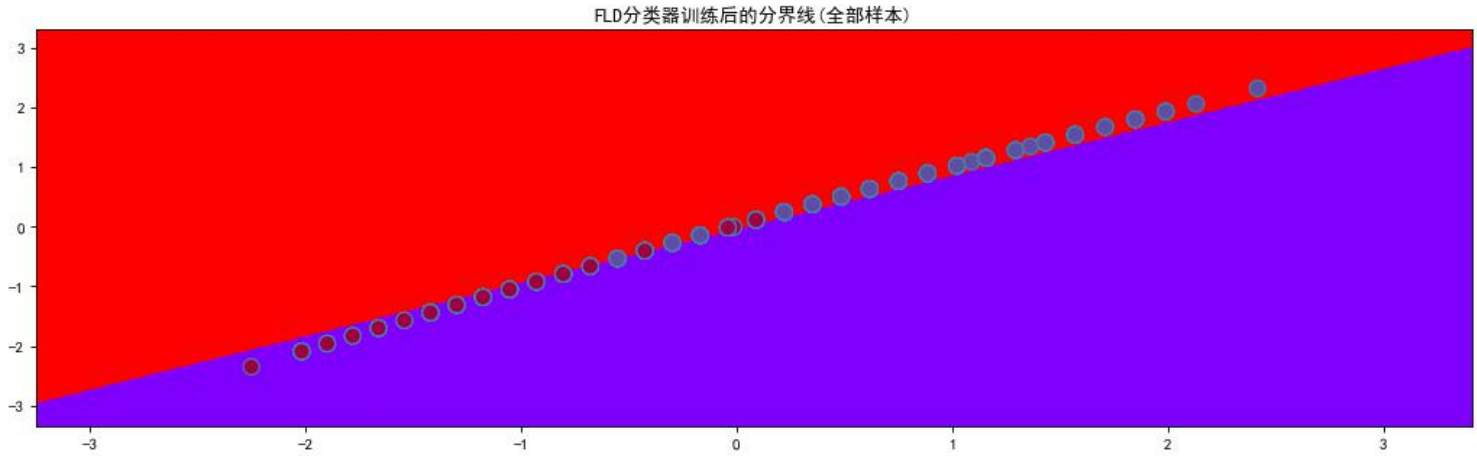


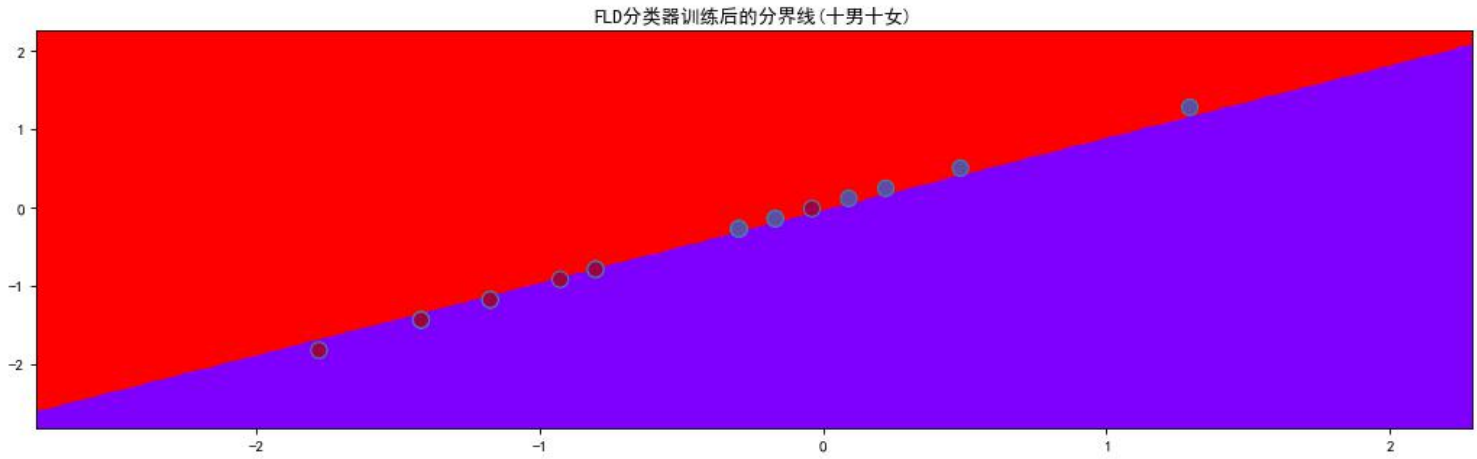
程序运行方法说明

2）FLD.py

```
In [2]: runfile('D:/pattern recognition/code/code/FLD.py', wdir='D:/pattern
recognition/code/code')
测试集上Holdout验证错误率为（十个特征）0.09793814432989689
十折交叉验证错误率为（十个特征）0.1037070221445221
训练集上Holdout验证错误率为0.0(十个特征，十男十女)
测试集上Holdout验证错误率为0.11855670103092786,使用的特征为（两个个特征）A,E
十折交叉验证错误率为0.12063228438228424,使用的特征为（两个个特征）A,E
<Figure size 720x2880 with 0 Axes>
```

训练集上Holdout验证错误率为0.050000000000000044,使用的特征为（两个特征，十男十女）A,E



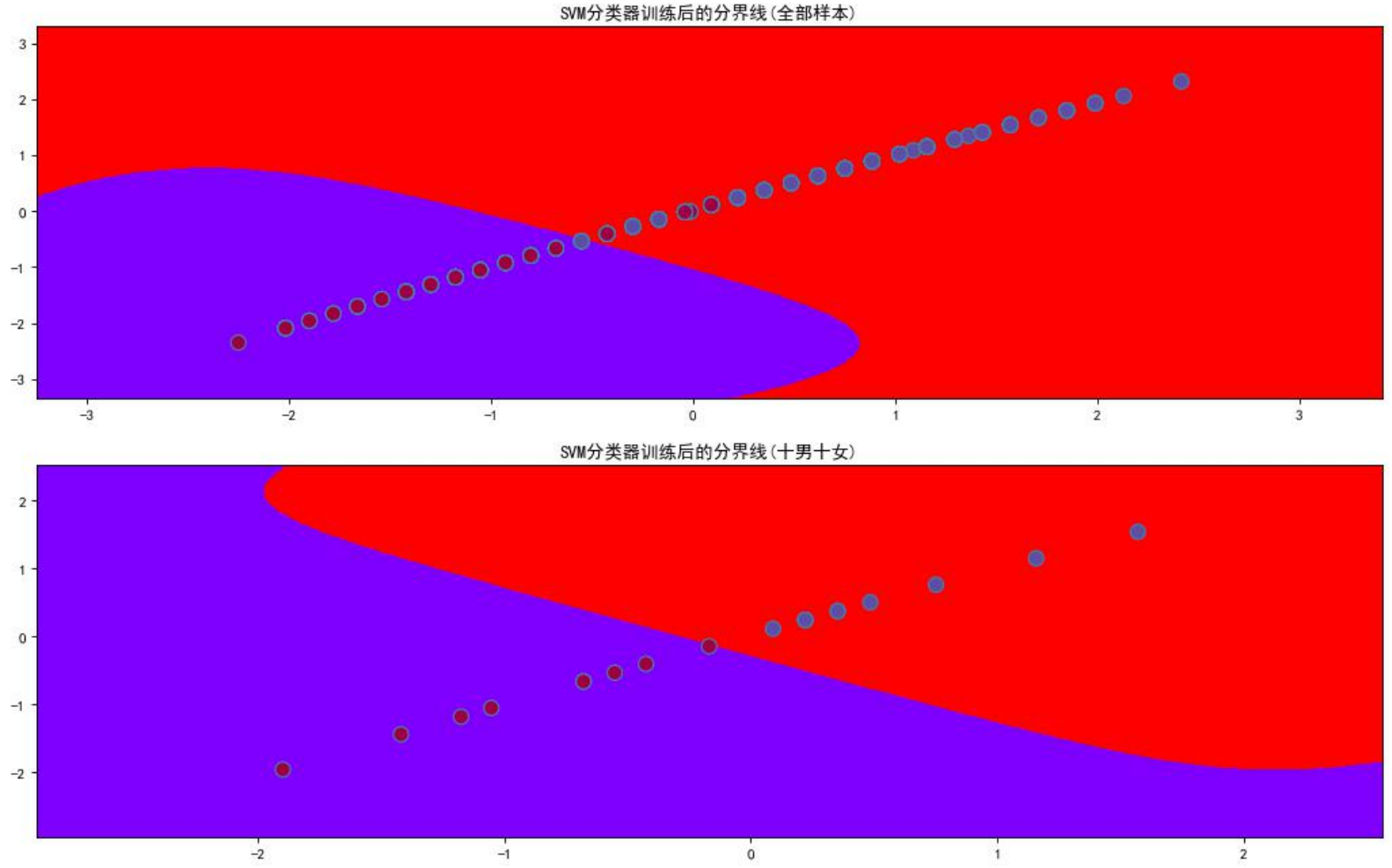


程序运行方法说明

3）SVM.py

```
In [3]: runfile('D:/pattern recognition/code/code/SVM.py', wdir='D:/pattern
recognition/code/code')
测试集上Holdout验证错误率为（十个特征）0.08762886597938147
十折交叉验证错误率为（十个特征）0.0897166375291375
训练集上Holdout验证错误率为0.0(十个特征，十男十女)
测试集上Holdout验证错误率为0.11855670103092786,使用的特征为（两个个特征）A,E
十折交叉验证错误率为0.11926063519813523,使用的特征为（两个个特征）A,E
<Figure size 720x2880 with 0 Axes>

训练集上Holdout验证错误率为0.050000000000000044,使用的特征为（两个个特征，十男十女）A,E
```



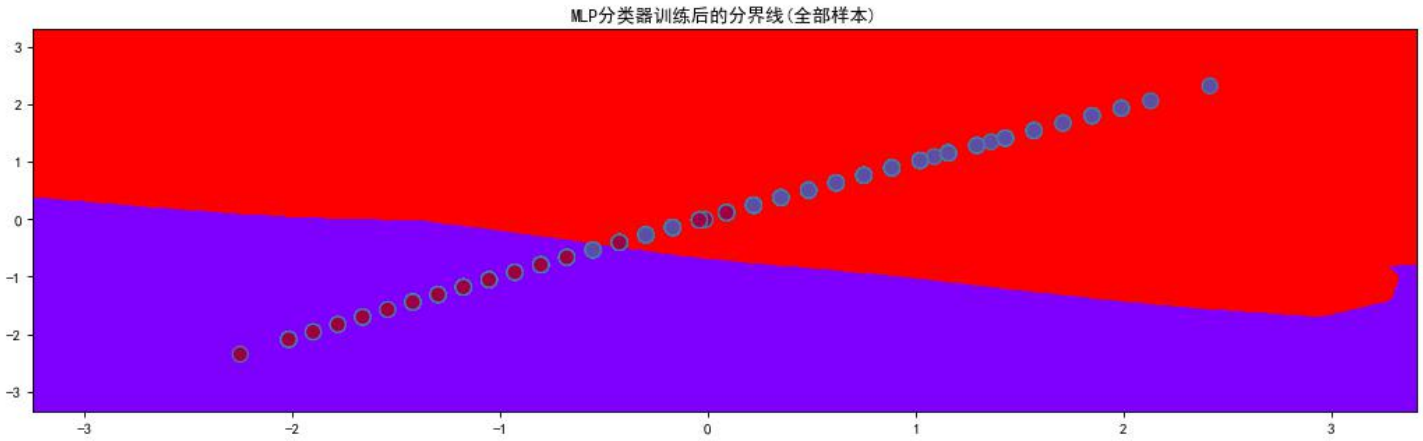
程序运行方法说明

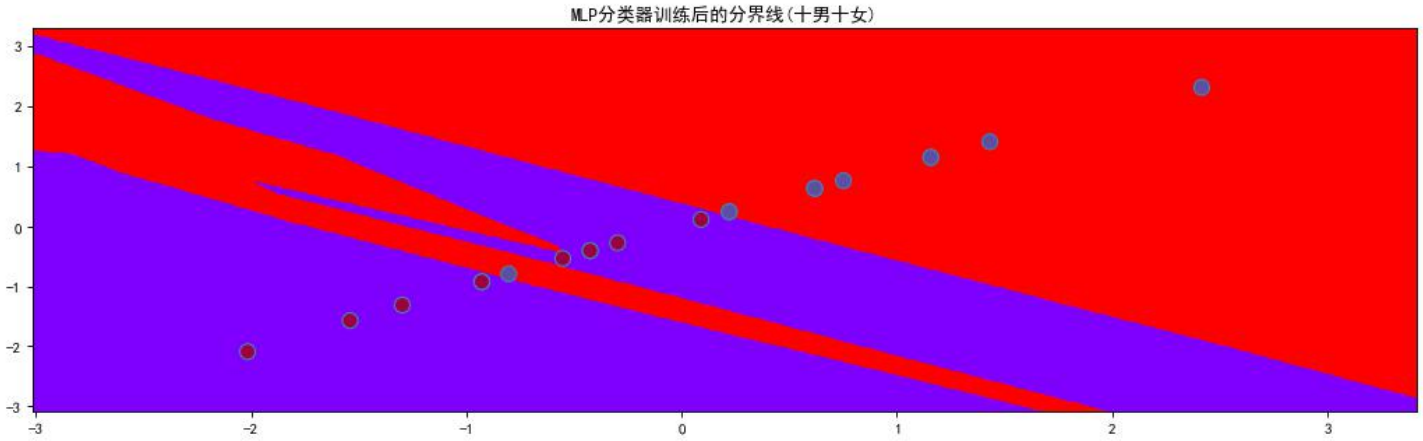
4）MLP.py

```
In [4]: runfile('D:/pattern recognition/code/code/MLP.py', wdir='D:/pattern
recognition/code/code')
```

测试集上Holdout验证错误率为（十个特征）0.14948453608247425
十折交叉验证错误率为（十个特征）0.13145177738927738
训练集上Holdout验证错误率为0.0(十个特征，十男十女)
测试集上Holdout验证错误率为0.11855670103092786,使用的特征为（两个个特征）A,E
十折交叉验证错误率为0.12380536130536124,使用的特征为（两个个特征）A,E
<Figure size 720x2880 with 0 Axes>

训练集上Holdout验证错误率为0.0,使用的特征为（两个个特征，十男十女）A,E





4.预测

运行predict.py

```
In [33]: runfile('D:/pattern recognition/code/code/predict.py', wdir='D:/pattern
recognition/code/code')

In [34]:
```

dataset4_2

-1.326454	-1.371248	-1.482598	-1.262801	-1.336692	-1.032446	-0.412015	-0.38958	-0.053554	-0.670229	F
-1.452109	-0.543336	-0.500223	0.5190476	-1.469502	-0.906016	-0.313398	-0.565644	1.0418626	0.6435387	F
-1.072796	-1.059563	-1.089648	-0.80116	-1.07107	0.358284	-1.10233	2.4907015	0.7048112	-0.93605	F

实验收获与小结

这次大作业我是从十月中旬开始启动，一周的时间完成了基本程序的编写工作，余下的时间在进行大作业报告的起草。在写大报告的过程中，我又对程序以及算法进行了几处小变动。现在，横跨半个多月的大作业即将画上句号。下面，我简要谈谈这个过程中我的心得体会。

首先，**这个过程我体会到了合作交流的重要性**，因为仅仅依靠我自己一个人根本不可能在这么短的时间里完成这么浩大的工程。最开始在读取数据的时候，我一直在寻找一个可以把txt文档转化为csv文档的方法，在网上苦苦寻找三天仍然没有找到合适的方法。最后我问了一个同学，他教我通过转化格式以及分隔的标准来将txt转化为csv，特别容易的一个办法。我先前总以为需要编写一段代码来实现，其实是一个很简单的过程。

其次，**这次的大作业强化了我的python编程和机器学习的基础知识**，在做数据预处理的时候，我编写了很长很复杂的函数来实现修改异常值等算法，在我不断优化我的代码的过程中，我发现使用匿名函数lambda以及apply和applymap函数可以极大地简化代码，并且可以提高程序的运行速度。在进行特征选取的时候，我以前都不知道如何进行特征选取，最开始我是自己编写的一个代码，通过计算每两个特征在不同分类器、不同情况下的错误率的平均值的大小来选择特征，后来才了解到有嵌入式，包裹式，嵌入式等特征选取方法，并且python有对应的toolbox。

第三，**这个过程培养了我的耐心**，因为在屡次debug的过程中，心情总是会十分焦躁，但是一旦发现了程序的问题并且对它进行改正后，我感觉精神得到了升华，耐心和付出也得到了回报。

第四，**我学会了要多向前人学习**，在编写如何画出决策面的代码的时候，我起初总是想通过自己的能力写出画样本分布的代码，但多次尝试得到的效果十分不好，最后我在网上找了许多代码，经过反复实验后找到了一个最合适的代码。我发现，前人已经为我们造了很多轮子，在学习提升的过程中，适当借鉴优秀的代码有利于提高我们的编程效率以及编程能力。

第五，**这个过程提高了我word文档编辑能力**，记得在做完《大数据与人工智能案例》这门课的大作业后，老师还说我们组的报告写得太丑了，她说一个月饼需要得到好的包装才会有好市场，所以这次，我就特别注意了报告这一块，也学到了很多word编辑技巧。

程序来源说明

本次大作业的程序编写中有一段程序我引用了他人的代码：

```
def border_of_classifier(sklearn_cl, x, y,n):
        ## 1 生成网格数据
        x_min, y_min = x.min(axis = 0) - 1
        x_max, y_max = x.max(axis = 0) + 1
        # 利用一组网格数据求出方程的值，然后把边界画出来。
        x_values, y_values = np.meshgrid(np.arange(x_min, x_max, 0.01),
        np.arange(y_min, y_max, 0.01))
        # 计算出分类器对所有数据点的分类结果 生成网格采样
        mesh_output = sklearn_cl.predict(np.c_[x_values.ravel(),
y_values.ravel()])
        # 数组维度变形
        mesh_output = mesh_output.reshape(x_values.shape)
        fig, ax = plt.subplots(figsize=(16,10), dpi= 80)
        ## 会根据 mesh_output结果自动从 cmap 中选择颜色
        plt.subplot(2,1,n)
        plt.pcolormesh(x_values, y_values, mesh_output, cmap =
'rainbow')
        plt.scatter(x[feature_[0]], x[feature_[1]], c = y, s=100,
edgecolors ='steelblue' , linewidth = 1, cmap = plt.cm.Spectral)
        plt.xlim(x_values.min(), x_values.max())
        plt.ylim(y_values.min(), y_values.max())
        # 设置x轴和y轴
        plt.xticks((np.arange(np.ceil(min(x[feature_[0]]) - 1),
np.ceil(max(x[feature_[0]]) + 1), 1.0)))
        plt.yticks((np.arange(np.ceil(min(x[feature_[1]]) - 1),
np.ceil(max(x[feature_[1]]) + 1), 1.0)))

        if(n==1):
            plt.title("朴素贝叶斯分类器训练后的分界线(全部样本)")
            plt.savefig("nball")
        else:
            plt.title("朴素贝叶斯分类器训练后的分界线(十男十女)")
            plt.savefig("nb10")
         plt.show()
```

本段代码为CSDN博主「Scc_hy」的原创文章
原文链接：https://blog.csdn.net/Scc_hy/article/details/91869095