正　文　部　分

一. 摘要

(1) 本项目在 Rice Leaf Diseases 数据集上完成了如下的实验内容

1. 提取图像的 HOG 特征。通过修改 HOG 参数，每张图像提取了 144 维的特征。
2. 分别利用 PCA，LDA，KPCA, KLDA 对 HOG 特征进行降维操作，降为 2 维特征。
3. 不同降维操作后的特征进行 FCM 聚类
4. 不同降维操作后的特征进行 SVM，逻辑回归和 ANN 分类
5. 利用逻辑回归对原始图像数据进行分类

(2) 代码所在位置

```
./machine_learning_homework
    ./dataset          % Rice Leaf Diseases 数据集
    ./hogExtractor     % hog 特征提取
    ./dimReduction     % PCA，LDA，KPCA, KLDA 降维操作
    ./cluster          % FCM 聚类操作
    ./mulClassSvm      % SVM 分类
    ./logRegression    % 逻辑回归
    ./ANN              % ANN 分类
```

(1) 运行环境和如何运行

环境：Matlab R2019a

运行：运行相应文件夹下的主文件即可，下面章节报告内容中指明了主文件名字和运行顺序。

二. 数据集介绍

该数据集是关于三类水稻病害（细菌性叶枯病、褐斑病、叶黑穗病）的数据集。其中，每类病害各包含 40 个样本图片，统共包含 120 个样本图片。利用这

些样本图片，通过预处理算法和分类聚类算法，实现三种病害的分类和聚类处理。

在数据集中，细菌性叶枯病样本图片保存在“Bacterialleafblight”文件中，褐斑病样本图片保存在“Brownspot”文件中，叶黑穗病样本图片保存在“Leafsmut”文件中。图 1 展示了上述三种水稻病害的样本图片，可以看出三者具有明显不同的图像特征，可以对其进行分类和聚类处理。

(a)

(b)

(c)

图 1　(a) 细菌性叶枯病，(b) 叶黑穗病，(c) 褐斑病

三． HOG 特征提取

（1）实现过程

第一步：读取图像并插值成统一大小

第二步：变为灰度图像并利用 extractHOGFeatures 函数提取特征

```
imgData = imread(BacterialPathes + imgDir(i).name);
imgDataResized = imresize(imgData,[200,400]);
imgDataResizedGray = rgb2gray(imgDataResized)
imgDataResizedGray_ada = adapthisteq(imgDataResizedGray,'NumTiles',[8 8],'ClipLimit',0.01,'Distribution','uniform');
[HogFeature,vision] = extractHOGFeatures(double(imgDataResizedGray_ada),'CellSize',[80 80])
```

代码位置说明：

./ hogExtractor

./ datasetHogProcess1.m　　% 第一个类图像进行 hog 特征提取并保存

./ datasetHogProcess2.m　　% 第二个类图像进行 hog 特征提取并保存

./ datasetHogProcess3.m　　% 第三个类图像进行 hog 特征提取并保存

（2）实验结果

图 2 展示了提取的 HOG 特征（可视化代码也在提供的代码中）。

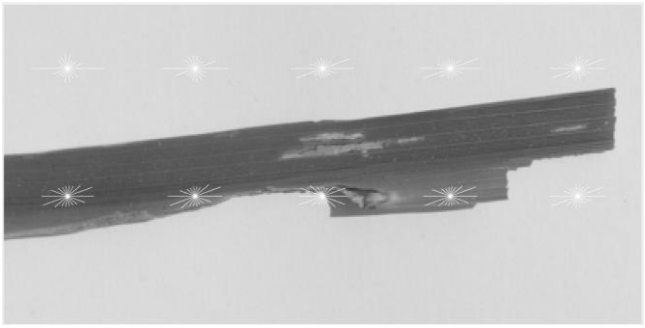


图 1 Hog 特征

四. HOG 特征降维操作

(1) PCA

1. 实现过程

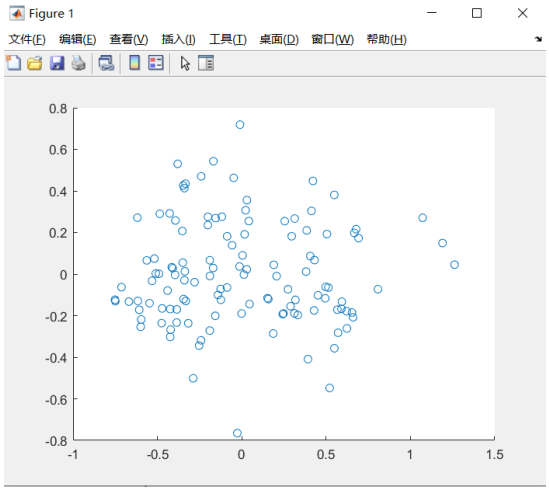
通过把 hog 特征投影到方差最大的方向(这里我选择了两个最大的方向）实现降维。代码实现如下：

./ dimReduction

./pca_code　　% 包含 PCA 和 kPCA 函数

./PCAProcess.m　　% PCA 降维主函数（保存 PCA 结果）

2. 实验结果



(2) kPCA

1. 实现过程

先利用核函数把hog特征变换到高维的空间,然后再进行PCA降维，投影到方差最大的方向实现降维(这里我选择了两个最大的方向)。

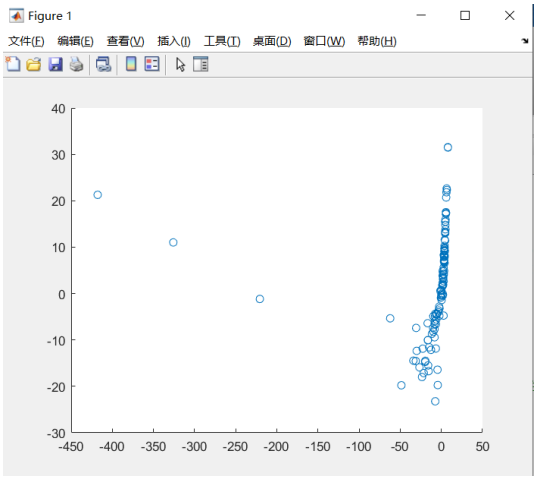
代码实现如下：

./ dimReduction

./ pca_code　　% 包含 PCA 和 kPCA 函数

./ kPCAProcess.m　　% kPCA 降维主函数（保存 kPCA 结果）

2. 实验结果



(3) LDA

1. 实现过程

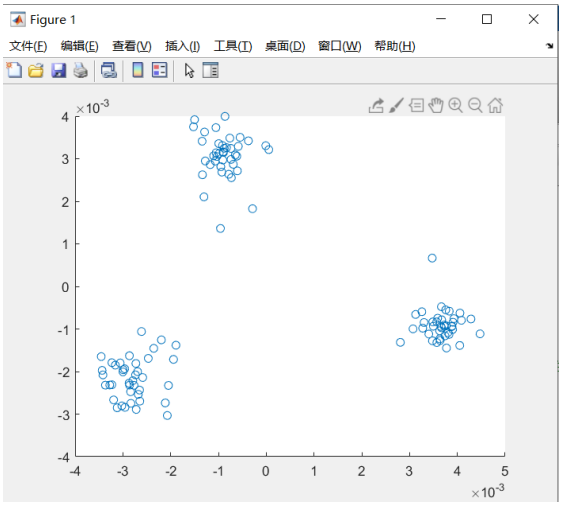
通过把 hog 特征投影到使类内距离最小，类间距离最大的方向（这里我选择了两个最大的方向）实现降维。代码实现如下：

./ dimReduction

./ lda_code　　% 包含 LDA 和 kLDA 函数

./ LDAProcess.m　　% LDA 降维主函数（保存 LDA 结果）

2. 实验结果



(4) kLDA

1. 实现过程

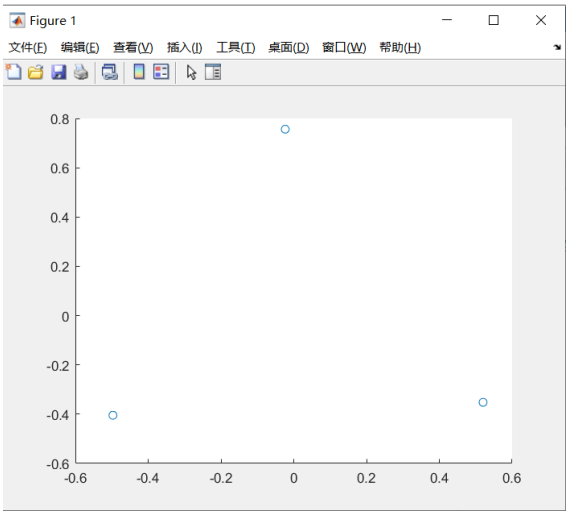
首先通过核函数把hog特征变换到高维特征空间,然后再LDA降维，投影到使类内距离最小，类间距离最大的方向（这里我选择了两个最大的方向）实现降维。代码实现如下：

./ dimReduction

./ lda_code % 包含 LDA 和 kLDA 函数

./ kLDAProcess.m % kLDA 降维主函数（保存 kLDA 结果）

2. 实验结果



五. FCM 聚类

(1) FCM 聚类实现

因为数据集分为 3 类，所以我们设置聚类中心为 3。通过迭代优化使每个样本点到中心点的加权（也就是样本到每个中心的概率）距离最小。从而实现聚类。代码实现如下(说明一下聚类精度代码有问题，因为聚类出来的类别和真实类别可能顺序不一样)：

./ cluster

./ FCMCluster.m % FCM 聚类主函数

(2) PCA 特征 FCM 聚类结果

1. 可视化结果

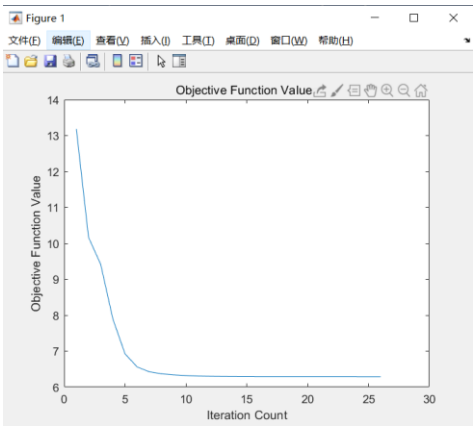


图 1. 随着迭代目标函数的值

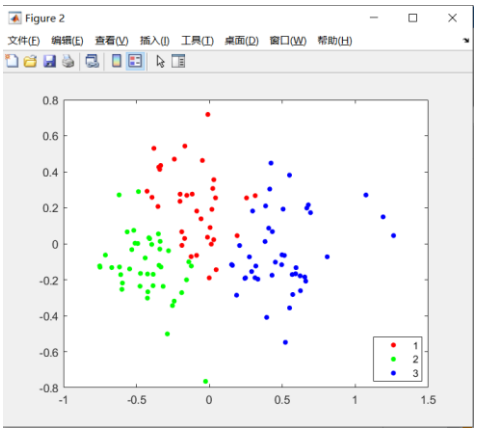


图 2. FCM 聚类结果

2. 聚类精度

allClassAccurate = 0.3750

BacterialAccurate = 0.2250

BrownspotAccurate = 0.2250

LeafsmutAccurate = 0.6750

(3) kPCA 特征 FCM 聚类结果

1. 可视化结果

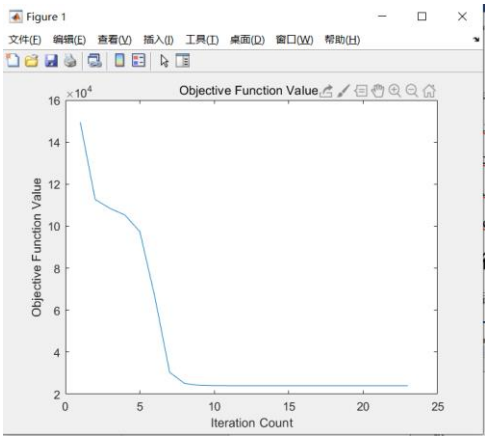


图 1. 随着迭代目标函数的值

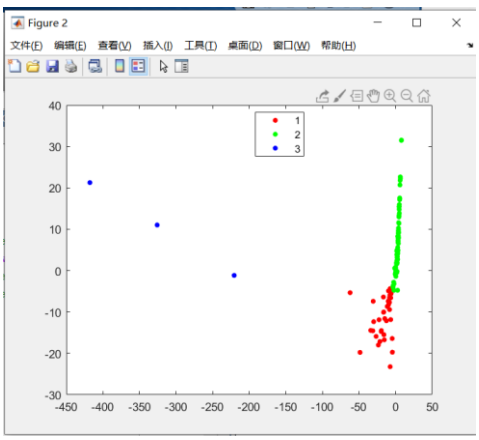


图 2. FCM 聚类结果

2. 聚类精度

allClassAccurate = 0.2667

BacterialAccurate = 0.0500

BrownspotAccurate = 0.6750

LeafsmutAccurate = 0.0750

(4) LDA 特征 FCM 聚类结果

1. 可视化结果

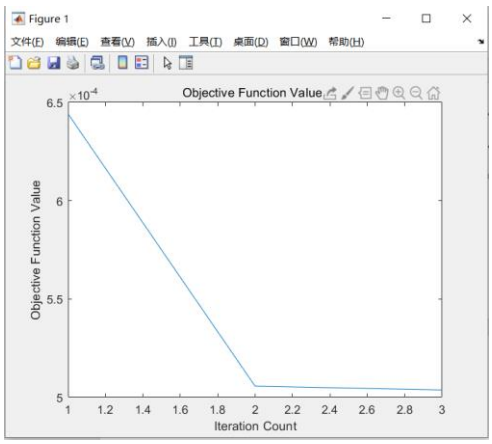


图 1. 随着迭代目标函数的值

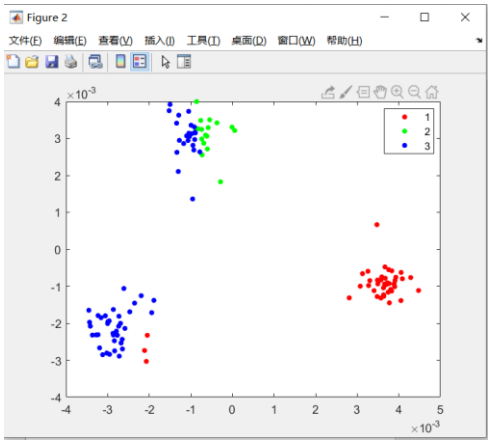


图 2. FCM 聚类结果

2. 聚类精度

allClassAccurate = 0.1667

BacterialAccurate = 0.0750

BrownspotAccurate = 0.4250

LeafsmutAccurate = 0.0000

(5) kLDA 特征 FCM 聚类结果

1. 可视化结果

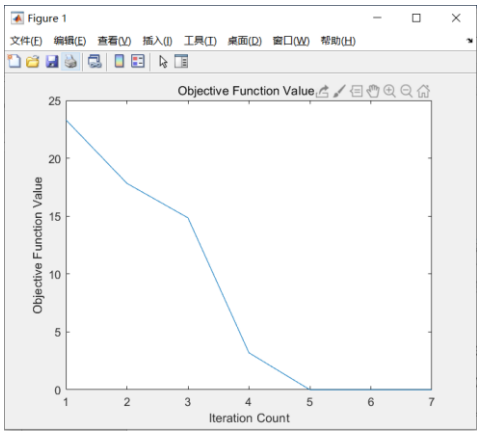


图 1. 随着迭代目标函数的值

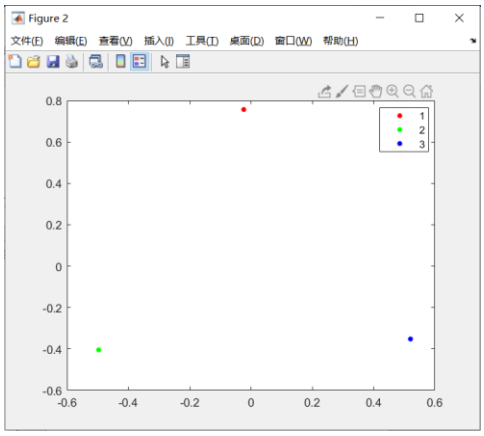


图 2. FCM 聚类结果

2. 聚类精度

allClassAccurate = 0.0000

BacterialAccurate = 0.0000

BrownspotAccurate = 0.0000

LeafsmutAccurate = 0.0000

(6) 原始图像 FCM 聚类结果

1. 可视化结果

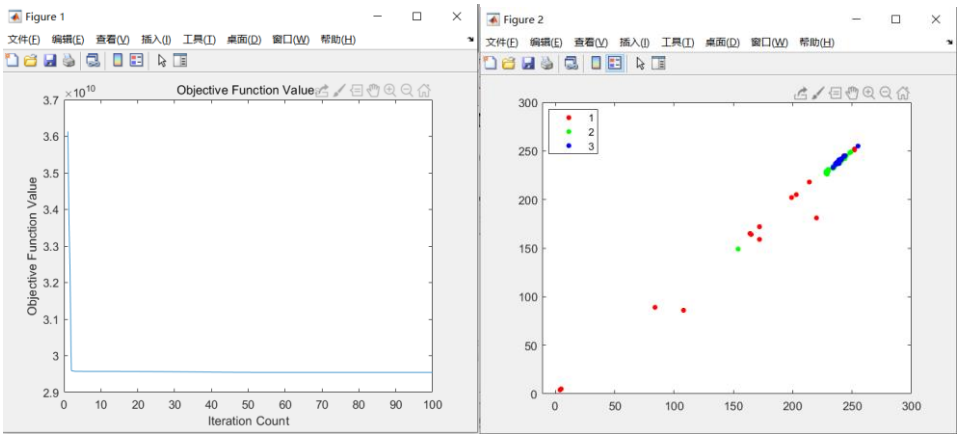


图 1. 随着迭代目标函数的值　　图 2. FCM 聚类结果（前两维）

2. 聚类精度

allClassAccurate = 0.1833

BacterialAccurate = 0.1000

BrownspotAccurate = 0.3250

LeafsmutAccurate = 0.1250

说明：首先使用./machine_learning_homework/dataset/img2Mat.mat 生成原始数据（由于图像大小不一样，这里我们把图像插值成同样大小[200,400,3]）。

六. 分类

(1) 数据集划分

通过如下三步进行数据集划分

第一步：根据数据集中样本数量生成随机数并保存该随机数

第二步：利用该随机数打乱数据集

第三步：从打乱的数据集中选择 80%的样本用于训练，剩余 20%用于测试

(2) SVM 分类

1. 实现过程

这里使用了硬 SVM 分类，没有使用核函数，使用 SMO 算法进行优化。

代码实现如下：

./ mulClassSvm

./其他代码　　% SVM 组成代码

./ main.m　　% SVM 多分类主函数

./ rand_indices.mat　　% 划分数据集时的随机数

2. 实验结果

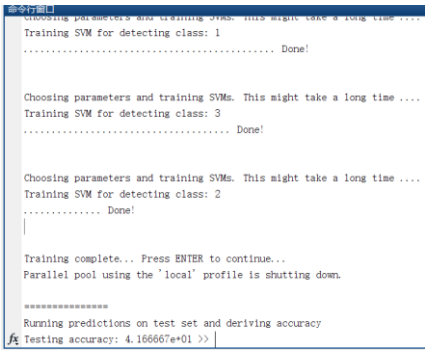


图 1. PCA 特征分类精度

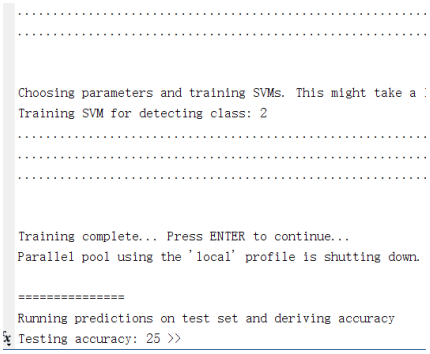


图 2. kPCA 特征分类精度

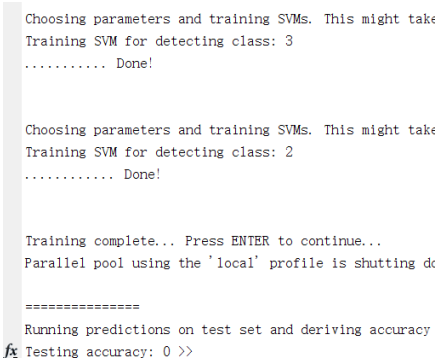


图 3. LDA 特征分类精度

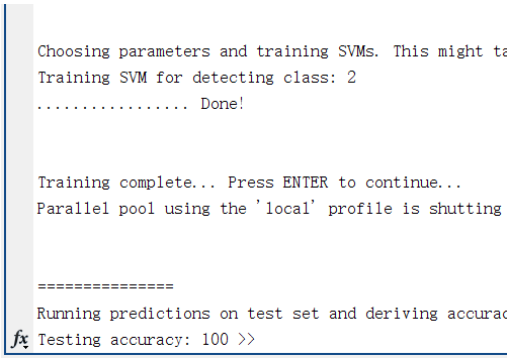


图 4. kLDA 特征分类精度

(3) 逻辑回归分类

1. 实现过程

使用迭代重加权最小二乘实现逻辑回归。代码实现如下：

./ logRegression

./其他代码　　% 逻辑回归组成代码

./ main.m　　% 逻辑回归主函数

./ rand_indices.mat　　% 划分数据集时的随机数

2. 实验结果

```
objectID=   15, predicted=  0, probability = 0.6822, true=  0, accuracy=1.00
objectID=   16, predicted=  0, probability = 0.8901, true=  0, accuracy=1.00
objectID=   17, predicted=  0, probability = 0.7309, true=  1, accuracy=0.00
objectID=   18, predicted=  1, probability = 0.8956, true=  0, accuracy=0.00
objectID=   19, predicted=  1, probability = 0.5693, true=  1, accuracy=1.00
objectID=   20, predicted=  1, probability = 0.8499, true=  1, accuracy=1.00
objectID=   21, predicted=  0, probability = 0.9473, true=  0, accuracy=1.00
objectID=   22, predicted=  0, probability = 0.9383, true=  0, accuracy=1.00
objectID=   23, predicted=  1, probability = 0.5114, true=  0, accuracy=0.00
classification accuracy=0.7083
```

图 1. PCA 特征分类精度

```
objectID=   17, predicted=  0, probability = 0.5495, true=  1, accuracy=0.00
objectID=   18, predicted=  1, probability = 0.8318, true=  0, accuracy=0.00
objectID=   19, predicted=  1, probability = 0.5470, true=  1, accuracy=1.00
objectID=   20, predicted=  1, probability = 0.6999, true=  1, accuracy=1.00
objectID=   21, predicted=  0, probability = 0.9789, true=  0, accuracy=1.00
objectID=   22, predicted=  0, probability = 0.9667, true=  0, accuracy=1.00
objectID=   23, predicted=  1, probability = 0.5508, true=  0, accuracy=0.00
classification accuracy=0.7083
fx >>
```

图 2. kPCA 特征分类精度

```
objectID=   16, predicted=  0, probability = 1.0000, true=  0, accuracy=1.00
objectID=   17, predicted=  1, probability = 1.0000, true=  1, accuracy=1.00
objectID=   18, predicted=  0, probability = 1.0000, true=  0, accuracy=1.00
objectID=   19, predicted=  1, probability = 0.9999, true=  1, accuracy=1.00
objectID=   20, predicted=  1, probability = 1.0000, true=  1, accuracy=1.00
objectID=   21, predicted=  0, probability = 1.0000, true=  0, accuracy=1.00
objectID=   22, predicted=  0, probability = 1.0000, true=  0, accuracy=1.00
objectID=   23, predicted=  0, probability = 1.0000, true=  0, accuracy=1.00
classification accuracy=1.0000
```

图 3. LDA 特征分类精度

```
objectID=   17, predicted=  1, probability = 1.0000, true=  1, accuracy=1.00
objectID=   18, predicted=  0, probability = 1.0000, true=  0, accuracy=1.00
objectID=   19, predicted=  1, probability = 1.0000, true=  1, accuracy=1.00
objectID=   20, predicted=  1, probability = 1.0000, true=  1, accuracy=1.00
objectID=   21, predicted=  0, probability = 1.0000, true=  0, accuracy=1.00
objectID=   22, predicted=  0, probability = 1.0000, true=  0, accuracy=1.00
objectID=   23, predicted=  0, probability = 1.0000, true=  0, accuracy=1.00
classification accuracy=1.0000
fx >>
```

图 4. kLDA 特征分类精度

(4) ANN 分类

1. 实现过程

设置人工神经网络 3 层，每层 10 个神经元，反向传播 700 次进行优化。代码实现如下：

./ ANN

```
./neural_network          % ANN 代码
./main.m                  % ANN 主函数
./rand_indices.mat        % 划分数据集时的随机数
```

2. 实验结果

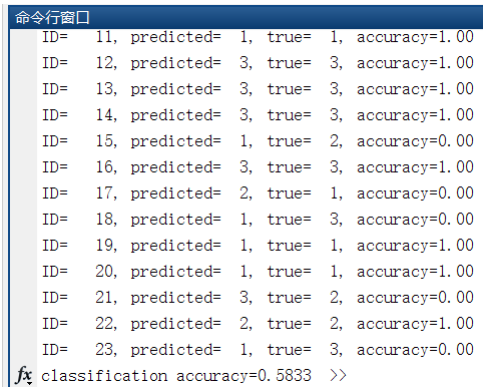


图 1. PCA 特征分类精度

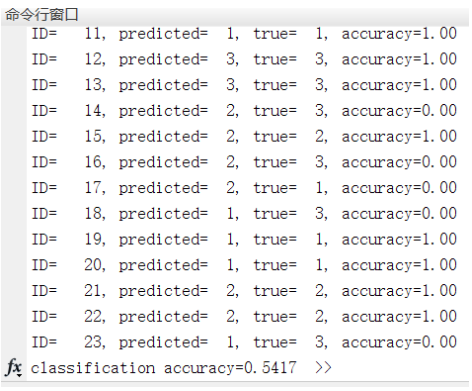


图 2. kPCA 特征分类精度

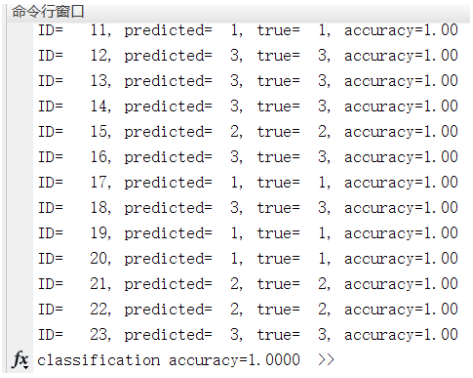


图 3. LDA 特征分类精度

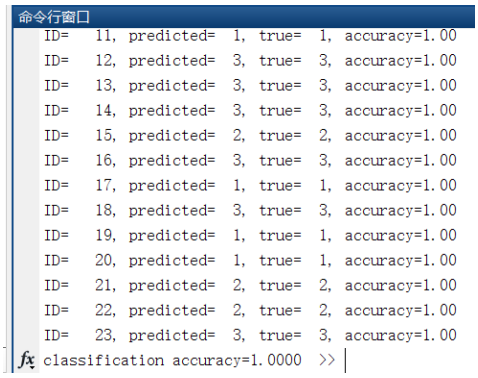
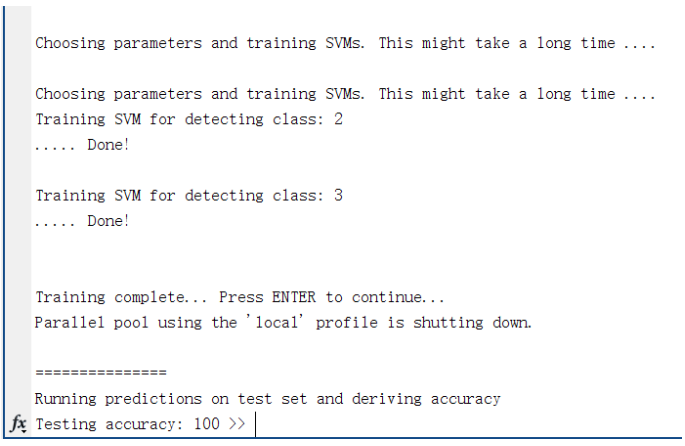


图 4. kLDA 特征分类精度

(5) 原始图像逻辑回归分类

实验结果



七. 结论与感想

通过该课程的学习，我了解了机器学习的处理流程：处理数据阶段（使用线性变换（PCA和LDA等）进行数据降维和使用非线性变换对数据进行降维（kPCA和kLDA等）），然后可以选择聚类方法（kmeans和FCM等）对数据进行分簇，也可以选择分类算法

(SVM,ANN,逻辑回归等)对数据进行分类。通过唐老师的仔细讲解，同时也明白了每个算法的优势和劣势。最后，通过实际动手操作，对所学东西的理解更加透彻。当然实现过程中也会遇到各种意想不到的错误。通过该项目我发现：

- kLDA特征的聚类结果最好。
- 原始图像的聚类效果最差。
- kLDA特征的分类结果最好。
- 原始图像的分类效果却不是最差的。
- 分类器中逻辑回归效果最好。
- SVM效果最不稳定。

以上结论只是实验结果，可能特征选择的参数和分类器参数不同也会影响效果。