模式识别

Lab1

姓名：郝裕玮

班级：计科1班

学号：18329015

目录

[1 实验5.2 3](#_Toc98864456)

[1.1 题目内容 3](#_Toc98864457)

[1.2 实验结果分析 4](#_Toc98864458)

[2 实验5.3 8](#_Toc98864459)

[2.1（a）题 8](#_Toc98864460)

[2.2（b）题 8](#_Toc98864461)

[2.3（c）题 9](#_Toc98864462)

[2.4（d）题 10](#_Toc98864463)

[3 实验6.6 11](#_Toc98864464)

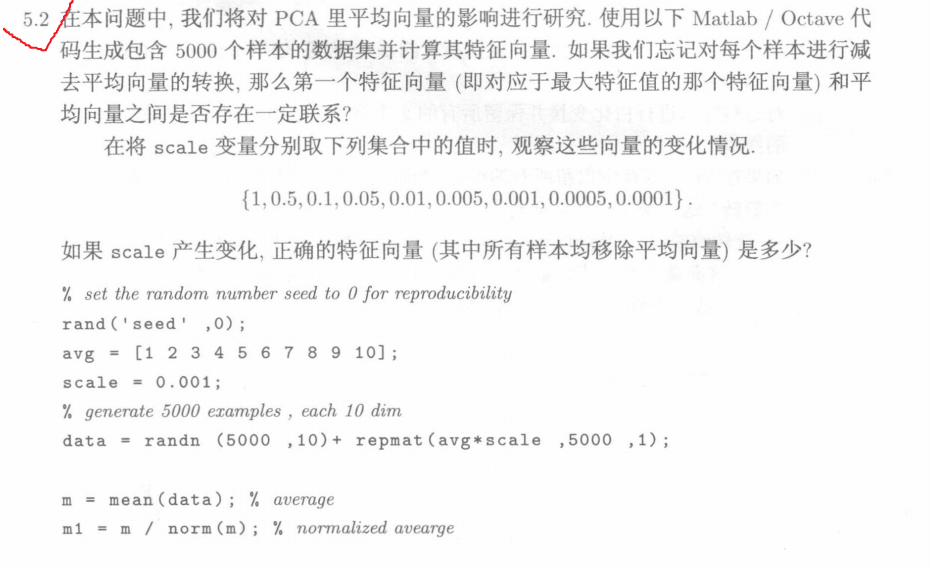
[3.1（c）题 11](#_Toc98864465)

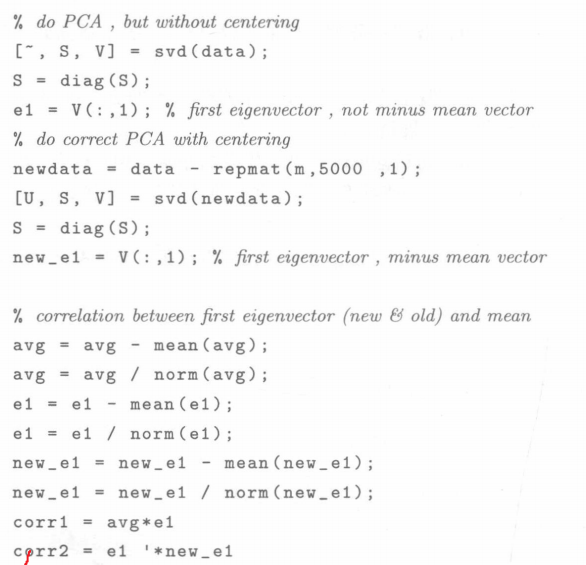
[3.2（d）题 13](#_Toc98864466)

[3.3 补充题目 14](#_Toc98864467)

# 1 实验5.2

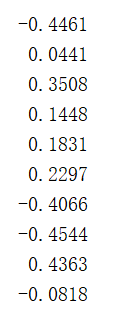
## 1.1 题目内容





## 1.2 实验结果分析

（1）运行代码可得特征向量结果如下所示：

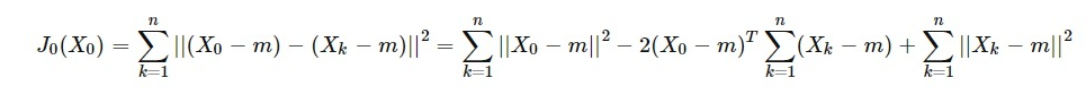


问：如果我们忘记对每个样本进行减去平均向量的转换，那么第一个特征向量(即对应于最大特征值的那个特征向量)和平均向量之间是否存在一定联系？

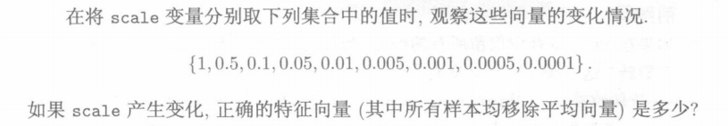
答：减去均值等同于坐标移动，这样就能把原始数据点的中心移到与原点重合，此举有利于很多表达，比如数据的协方差矩阵可以直接写成X\*X'，若没有减去均值，则每两个特征之间都要进行(X-X均值)\*（Y-Y均值）运算，再组合成协方差矩阵。

从线性变换的本质来说，PCA就是在线性空间做一个旋转（数据矩阵右乘协方差矩阵的特征向量矩阵），然后取低维子空间（实际上就是前n\_components个特征向量张成的子空间）上的投影点来代替原本的点，以达到降维的目的。正因为只做了旋转，没有平移。所以我们要保证原本空间里的点是以原点为中心分布的，这就是我们对数据进行均值化的原因。

同时，只有原始数据点居中时，得到的均方误差才最小。

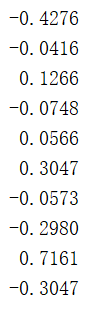


上式中第二项为0，第三项是与无关的常数，所以=**时，均方误差最小。

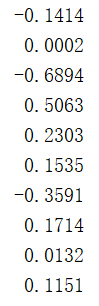


正确的特征向量变化情况如下：

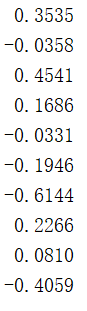
scale = 1：



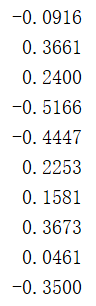
scale = 0.5：



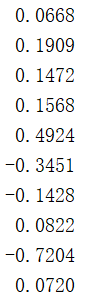
scale = 0.1（见下页）：



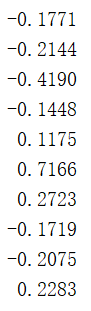
scale = 0.05：



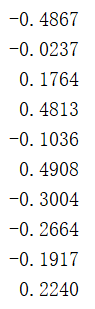
scale = 0.01：



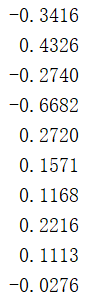
scale = 0.005：



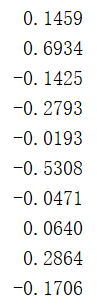
scale = 0.001：



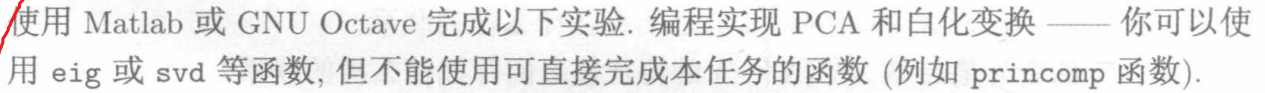
scale = 0.0005：



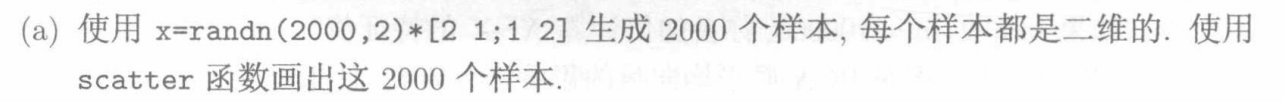
scale = 0.0001：



# 2 实验5.3



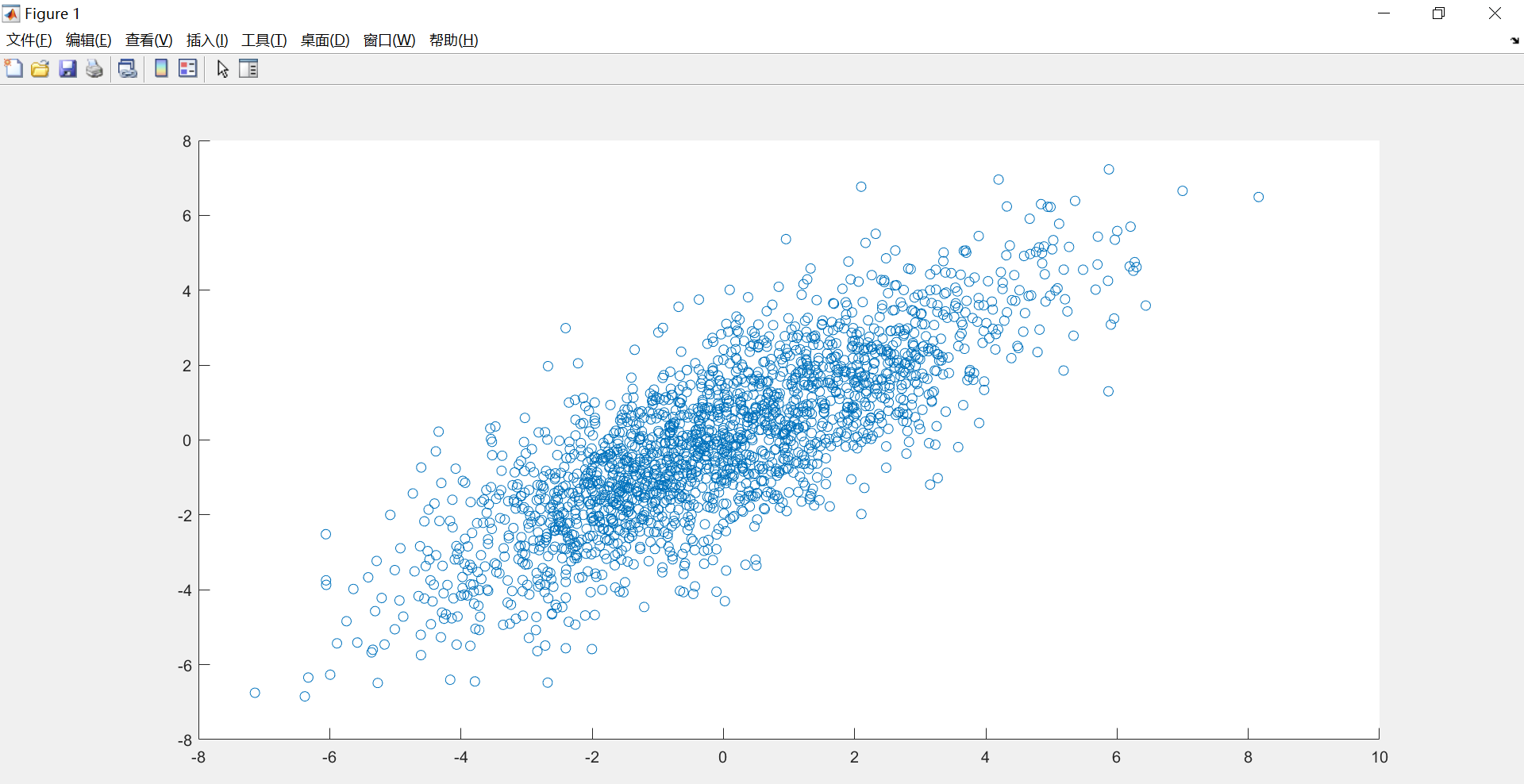
## 2.1（a）题



x = randn(2000,2) \* [2 1;1 2];

figure(1);

scatter(x(:,1), x(:,2));



## 2.2（b）题



x = randn(2000,2) \* [2 1;1 2];

mean\_x = mean(x); % 均值

cov\_x = cov(x); % 协方差矩阵

[ev,ed] = eigs(cov\_x); % 特征向量

[row,col] = size(x);

u = repmat(mean\_x , row, 1);

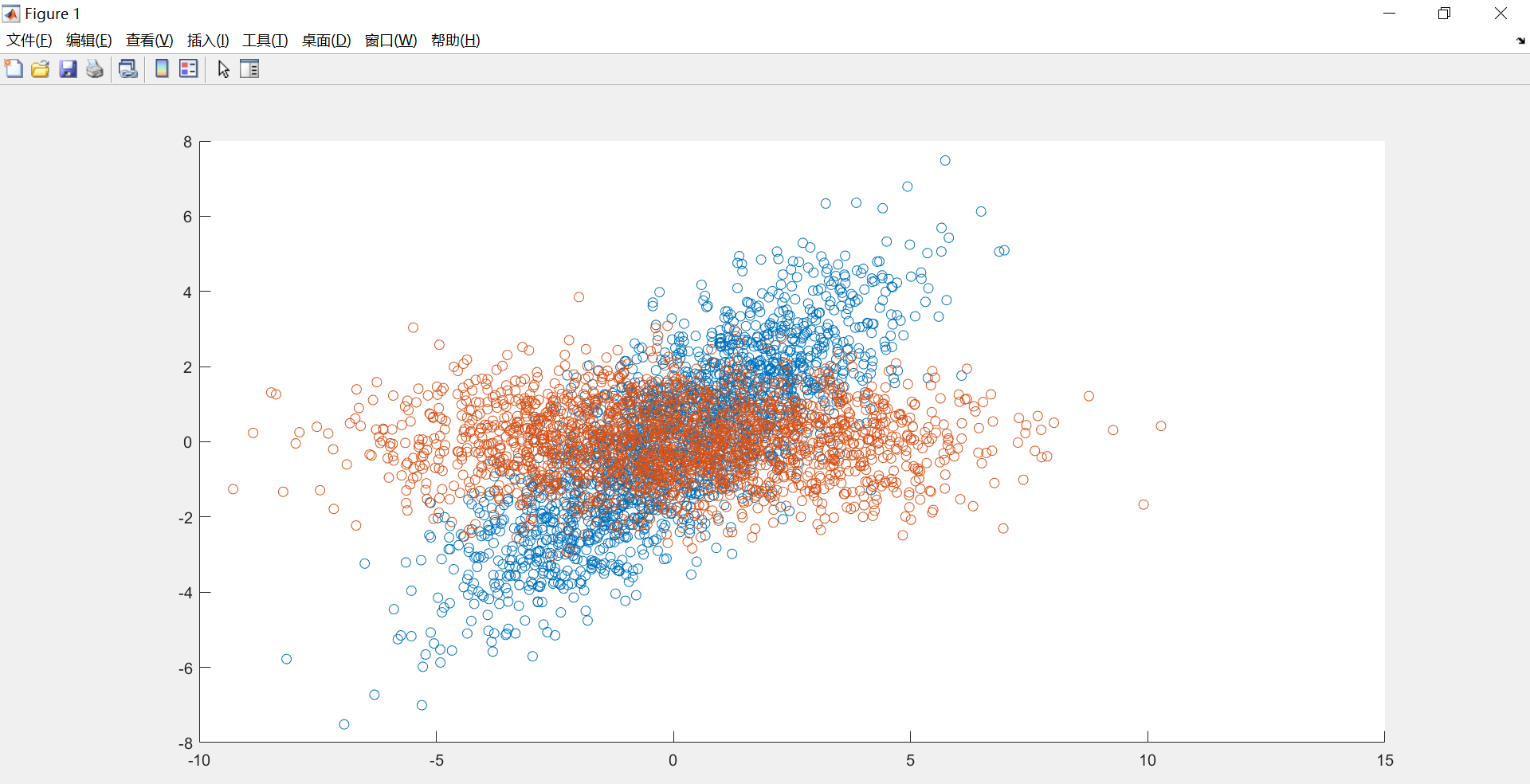
figure(1);

x1 = (x - u) \* ev;

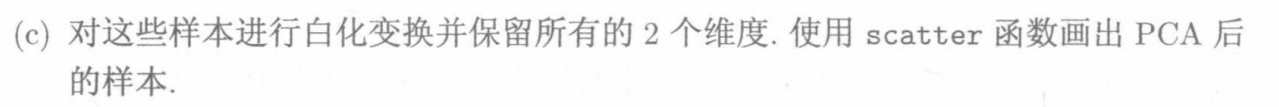
hold on;

scatter(x(:,1), x(:,2));

scatter(x1(:, 1), x1(:, 2));



## 2.3（c）题



x = randn(2000,2) \* [2 1;1 2];

mean\_x = mean(x); % 均值

cov\_x = cov(x); % 协方差矩阵

[ev,ed] = eigs(cov\_x); % 特征向量

[row,col] = size(x);

u = repmat(mean\_x , row, 1);

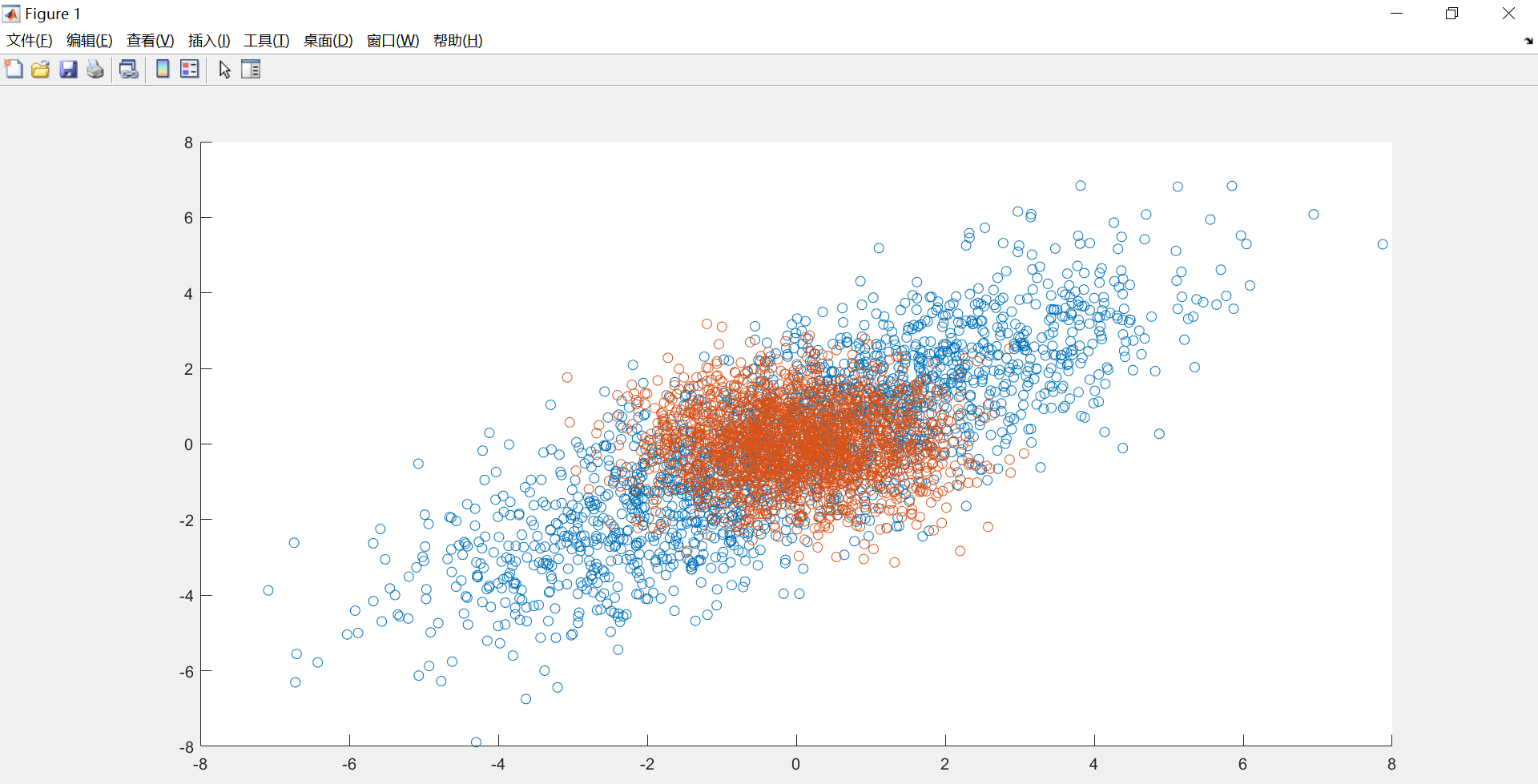
x1 = (x - u) \* ev \* inv(sqrt(ed));

hold on;

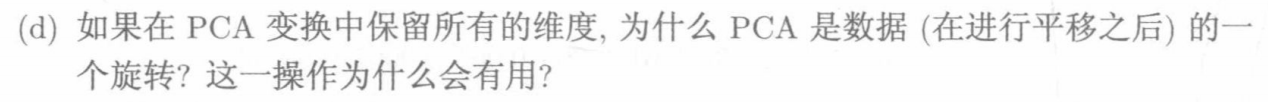
scatter(x(:,1), x(:,2));

scatter(x1(:, 1), x1(:, 2));

结果见下页：



## 2.4（d）题



答：PCA的核心步骤有2步：

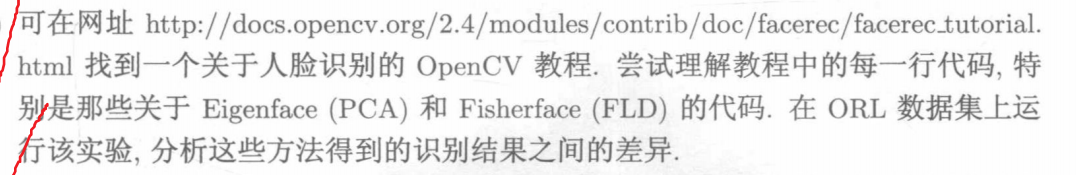
（1）中心化：原始数据集减去均值，这里的矩阵减法相当于数据平移；

（2）用前k个特征向量构成的矩阵乘原始矩阵实现矩阵降维。若保留所有维度，则相当于将所有特征向量组成的矩阵乘原始矩阵，这样的操作实际上就是基变换，将矩阵进行旋转。

这一操作有用的原因：向投影之后方差最大的维度进行投影，投影后的数据能最大程度上的保留原始数据的信息。

# 3 实验6.6

## 3.1（c）题



运行后结果如下：

Eigenface（PCA）：

Predicted class = 39 / Actual class = 39.

Eigenvalue #0 = 2819989.11572

Eigenvalue #1 = 2062664.43241

Eigenvalue #2 = 1096663.64953

Eigenvalue #3 = 894183.32702

Eigenvalue #4 = 818967.15747

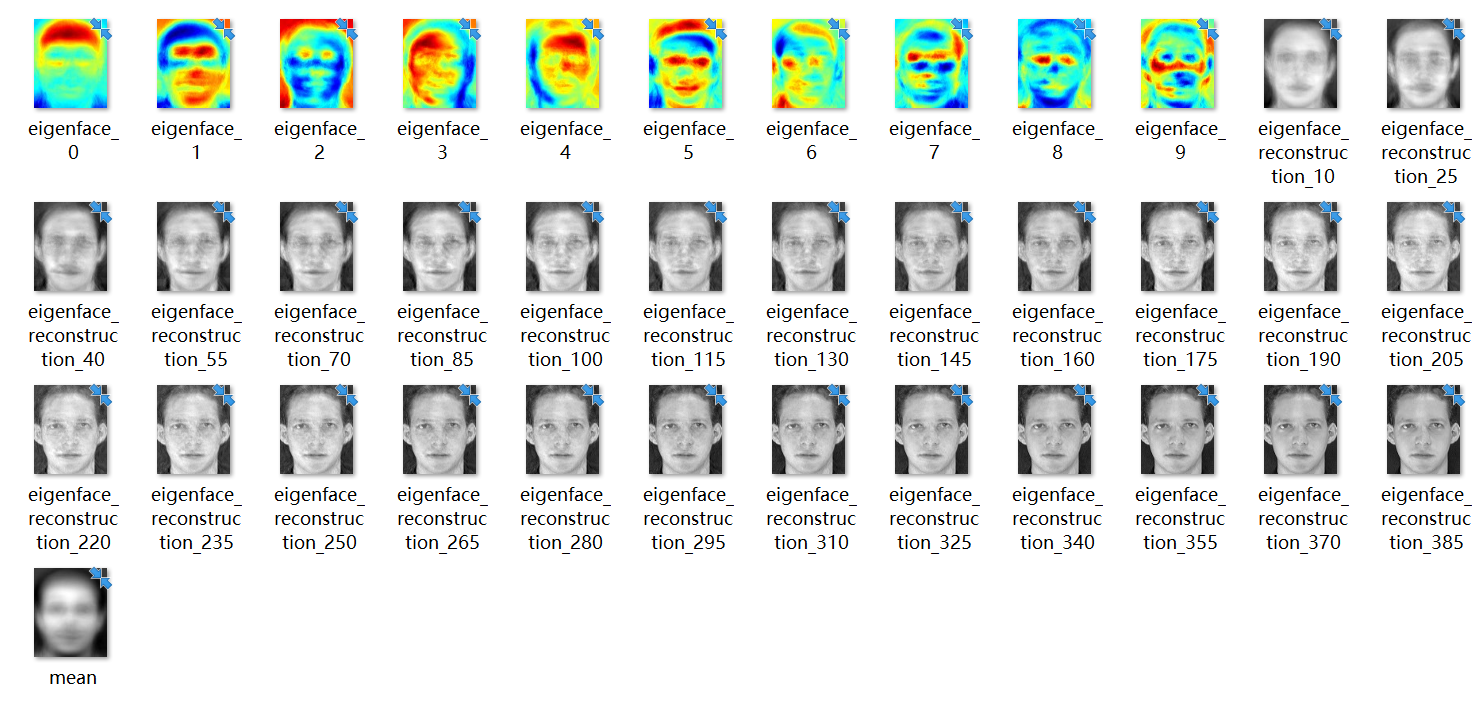
Eigenvalue #5 = 537972.44192

Eigenvalue #6 = 392438.01898

Eigenvalue #7 = 372326.54572

Eigenvalue #8 = 313616.90351

Eigenvalue #9 = 288570.19349



Fisherface（FLD）：

Predicted class = 39 / Actual class = 39.

Eigenvalue #0 = 46343770.45151

Eigenvalue #1 = 12526.75725

Eigenvalue #2 = 2156.94929

Eigenvalue #3 = 1202.72304

Eigenvalue #4 = 643.75567

Eigenvalue #5 = 415.09794

Eigenvalue #6 = 364.10602

Eigenvalue #7 = 216.88946

Eigenvalue #8 = 160.48730

Eigenvalue #9 = 134.74498

Eigenvalue #10 = 87.77821

Eigenvalue #11 = 76.46510

Eigenvalue #12 = 49.96221

Eigenvalue #13 = 47.58930

Eigenvalue #14 = 42.38283

Eigenvalue #15 = 34.96763



Fisherfaces算法和Eigenfaces算法有相同的地方，也有不相同的地方。相同：两者均可以对数据进行降维；两者在降维时均使用了矩阵特征分解的思想。不同：Fisherfaces是有监督的降维方法，而是Eigenfaces无监督的降维方法；Fisherfaces除了可以用于降维，还可以用于分类。

Eigenfaces:

1、 训练图像；

2、 求出平均脸；

3、 获得特征子脸；

4、 进行图像重构；

5、 寻找相似度高的人脸图像。

Fisherfaces：

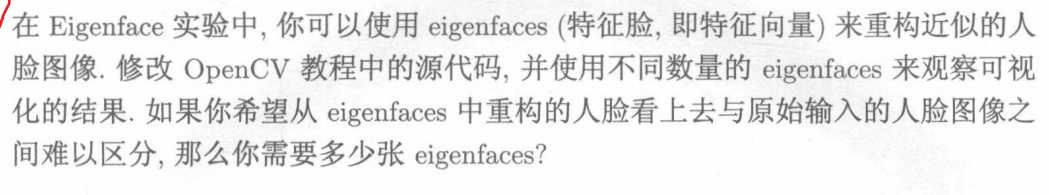
1、 获得人脸图像数据，然后求出人脸的均值；

2、 观察各个人脸的特征值；

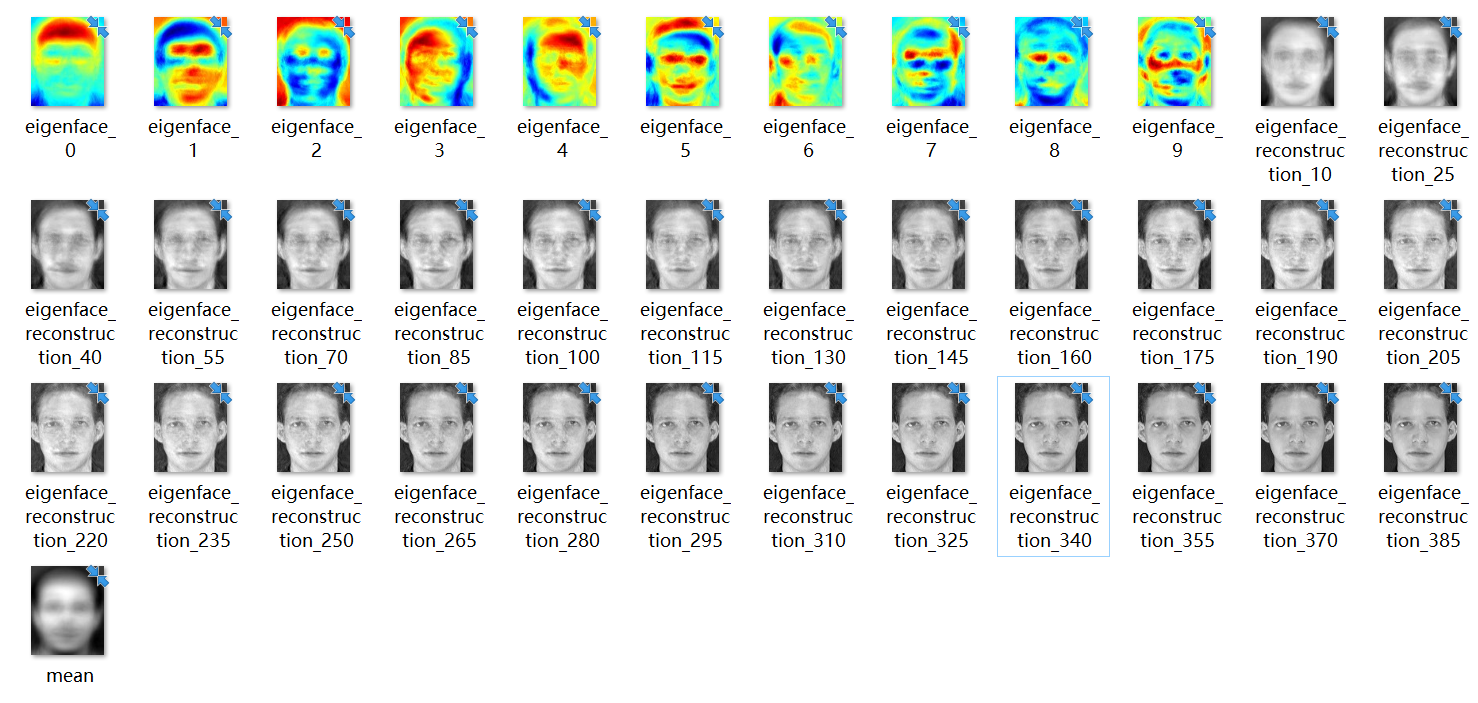
3、 进行人脸鉴定，观察人脸特征，判断是否是个人；

4、 最后进行人脸识别。

## 3.2（d）题



因为难以区分是主观标准，所以经过个人实验对比，需要10张eigenfaces即可。



## 3.3 补充题目

将PCA和FLD中的特征向量resize成原图尺寸，并显示前10个Eigenface和Fisherface。

（1）对于Eigenface：

源代码：

    for(int num\_components = min(W.cols, 10); num\_components < min(W.cols, 300); num\_components+=15)

修改为：

    for(int num\_components = min(W.cols, 10); num\_components < W.cols; num\_components+=15)

（2）对于Fisherface：

源代码：

for(int num\_component = 0; num\_component < min(16, W.cols); num\_component++) {

修改为：

for(int num\_component = 0; num\_component < W.cols; num\_component++) {

展示结果如下：



