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
1.1 题目内容 3
1.2 实验结果分析 4
2 实验 5.3 8
2.1（a）题 8
2.2（b）题 8
2.3（c）题 9
2.4（d）题 10
3 实验 6.6 11
3.1（c）题 11
3.2（d）题 13
3.3 补充题目 14

1 实验 5.2

1.1 题目内容

5.2 在本问题中，我们将对 PCA 里平均向量的影响进行研究. 使用以下 Matlab / Octave 代码生成包含 5000 个样本的数据集并计算其特征向量. 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?

在将 scale 变量分别取下列集合中的值时，观察这些向量的变化情况.

$$\{1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}.$$

如果 scale 产生变化，正确的特征向量 (其中所有样本均移除平均向量) 是多少?

```
% set the random number seed to 0 for reproducibility
rand('seed' ,0);
avg = [1 2 3 4 5 6 7 8 9 10];
scale = 0.001;
% generate 5000 examples , each 10 dim
data = randn (5000 ,10)+ repmat(avg*scale ,5000 ,1);

m = mean(data); % average
m1 = m / norm(m); % normalized avearge
```

```
% do PCA , but without centering
[~, S, V] = svd(data);
S = diag(S);
e1 = V(:,1); % first eigenvector , not minus mean vector
% do correct PCA with centering
newdata = data - repmat(m,5000 ,1);
[U, S, V] = svd(newdata);
S = diag(S);
new_e1 = V(:,1); % first eigenvector , minus mean vector

% correlation between first eigenvector (new & old) and mean
avg = avg - mean(avg);
avg = avg / norm(avg);
e1 = e1 - mean(e1);
e1 = e1 / norm(e1);
new_e1 = new_e1 - mean(new_e1);
new_e1 = new_e1 / norm(new_e1);
corr1 = avg*e1
corr2 = e1 '*new_e1
```

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：

```
-0.4461
 0.0441
 0.3508
 0.1448
 0.1831
 0.2297
-0.4066
-0.4544
 0.4363
-0.0818
```

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。

$$J_0(X_0)=\sum_{k=1}^{n}||(X_0-m)-(X_k-m)||^2=\sum_{k=1}^{n}||X_0-m||^2-2(X_0-m)^T\sum_{k=1}^{n}(X_k-m)+\sum_{k=1}^{n}||X_k-m||^2$$

上式中第二项为 0，第三项是与X_0无关的常数，所以X_0=m时，均方误差最小。

在将 scale 变量分别取下列集合中的值时，观察这些向量的变化情况.

$$\{1,0.5,0.1,0.05,0.01,0.005,0.001,0.0005,0.0001\}.$$

如果 scale 产生变化，正确的特征向量（其中所有样本均移除平均向量）是多少?

正确的特征向量变化情况如下：

scale = 1：

```
-0.4276
-0.0416
 0.1266
-0.0748
 0.0566
 0.3047
-0.0573
-0.2980
 0.7161
-0.3047
```

scale = 0.5：

```
-0.1414
 0.0002
-0.6894
 0.5063
 0.2303
 0.1535
-0.3591
 0.1714
 0.0132
 0.1151
```

scale = 0.1（见下页）：

0. 3535
-0. 0358
0. 4541
0. 1686
-0. 0331
-0. 1946
-0. 6144
0. 2266
0. 0810
-0. 4059

scale = 0.05：

-0. 0916
0. 3661
0. 2400
-0. 5166
-0. 4447
0. 2253
0. 1581
0. 3673
0. 0461
-0. 3500

scale = 0.01：

0. 0668
0. 1909
0. 1472
0. 1568
0. 4924
-0. 3451
-0. 1428
0. 0822
-0. 7204
0. 0720

scale = 0.005：

-0. 1771
-0. 2144
-0. 4190
-0. 1448
0. 1175
0. 7166
0. 2723
-0. 1719
-0. 2075
0. 2283

scale = 0.001 :

```
-0. 4867
-0. 0237
 0. 1764
 0. 4813
-0. 1036
 0. 4908
-0. 3004
-0. 2664
-0. 1917
 0. 2240
```

scale = 0.0005 :

```
-0. 3416
 0. 4326
-0. 2740
-0. 6682
 0. 2720
 0. 1571
 0. 1168
 0. 2216
 0. 1113
-0. 0276
```

scale = 0.0001 :

```
 0. 1459
 0. 6934
-0. 1425
-0. 2793
-0. 0193
-0. 5308
-0. 0471
 0. 0640
 0. 2864
-0. 1706
```

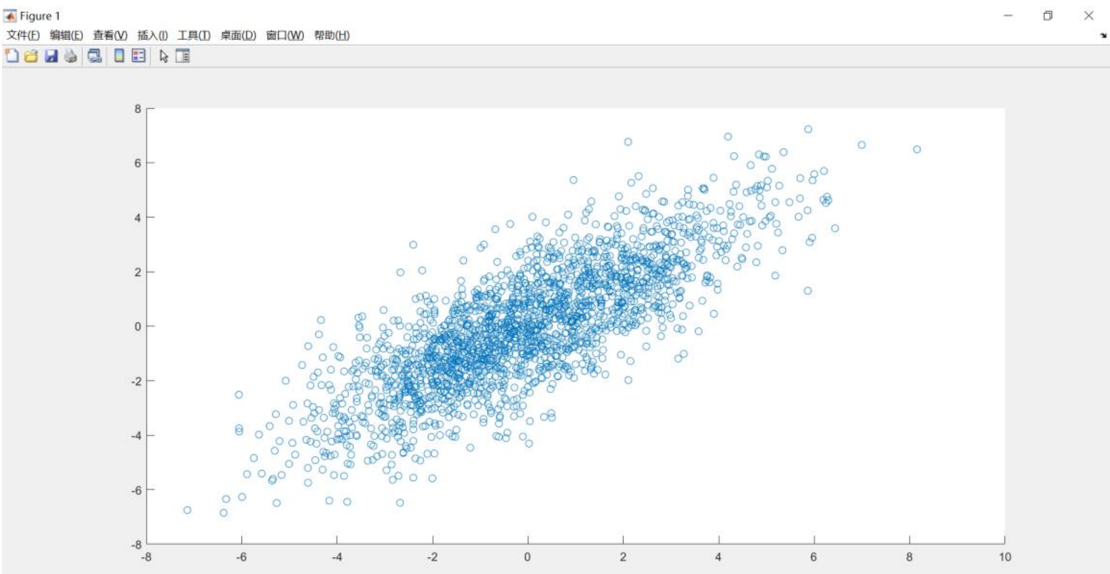
2 实验 5.3

使用 Matlab 或 GNU Octave 完成以下实验. 编程实现 PCA 和白化变换 —— 你可以使用 eig 或 svd 等函数, 但不能使用可直接完成本任务的函数 (例如 princomp 函数).

2.1（a）题

(a) 使用 x=randn(2000,2)*[2 1;1 2] 生成 2000 个样本, 每个样本都是二维的. 使用 scatter 函数画出这 2000 个样本.

```
x = randn(2000,2) * [2 1;1 2];
figure(1);
scatter(x(:,1), x(:,2));
```



2.2（b）题

(b) 对这些样本进行 PCA 变换并保留所有的 2 个维度. 使用 scatter 函数画出 PCA 后的样本.

```
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

(c) 对这些样本进行白化变换并保留所有的 2 个维度. 使用 scatter 函数画出 PCA 后的样本.

```
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

(d) 如果在 PCA 变换中保留所有的维度，为什么 PCA 是数据 (在进行平移之后) 的一个旋转？这一操作为什么会有用？

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

可在网址 http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html 找到一个关于人脸识别的 OpenCV 教程. 尝试理解教程中的每一行代码, 特别是那些关于 Eigenface (PCA) 和 Fisherface (FLD) 的代码. 在 ORL 数据集上运行该实验, 分析这些方法得到的识别结果之间的差异.

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

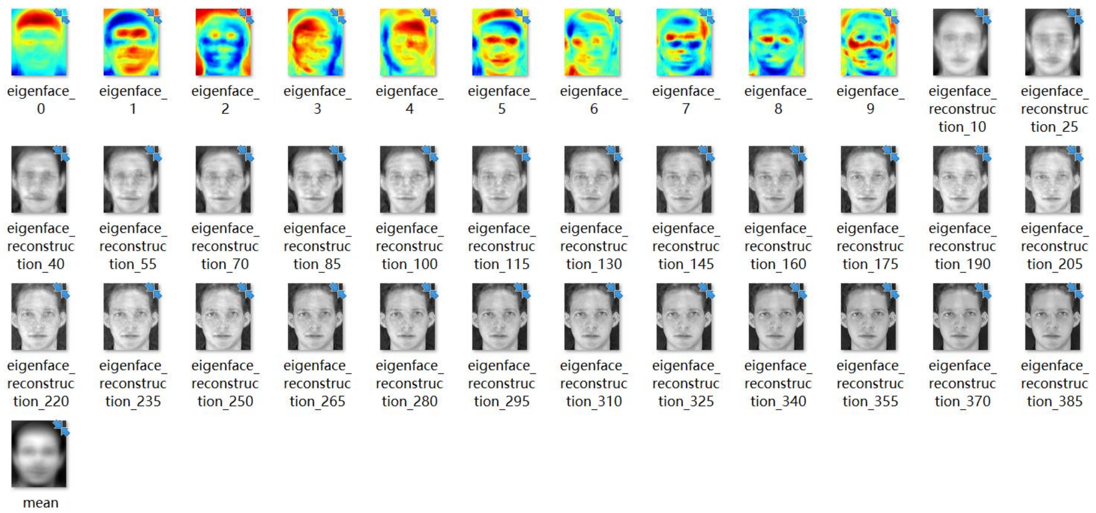
Eigenvalue #5 = 537972.44192

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

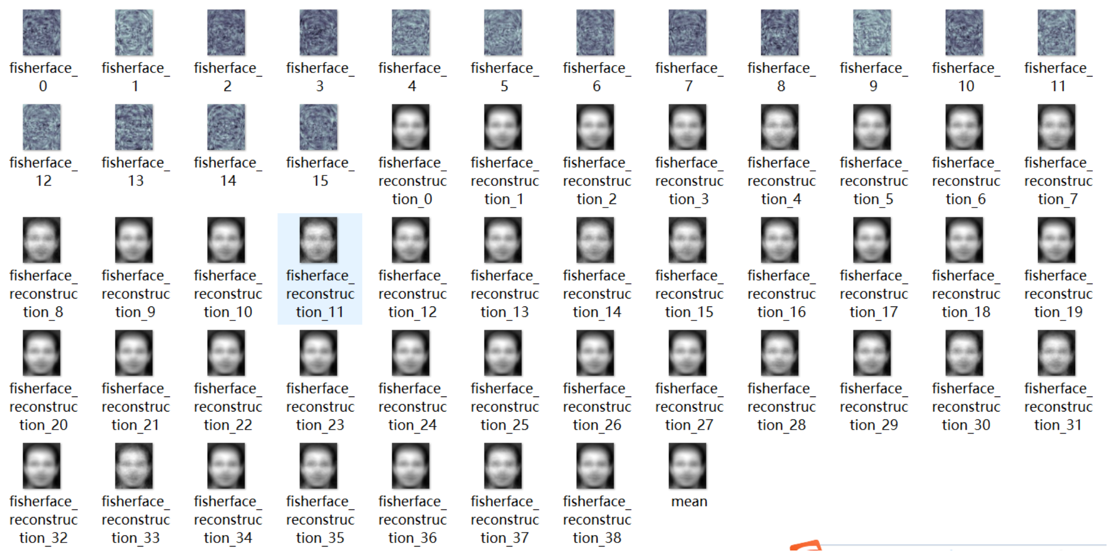
Eigenvalue #11 = 76.46510

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：Fisherfac es 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；

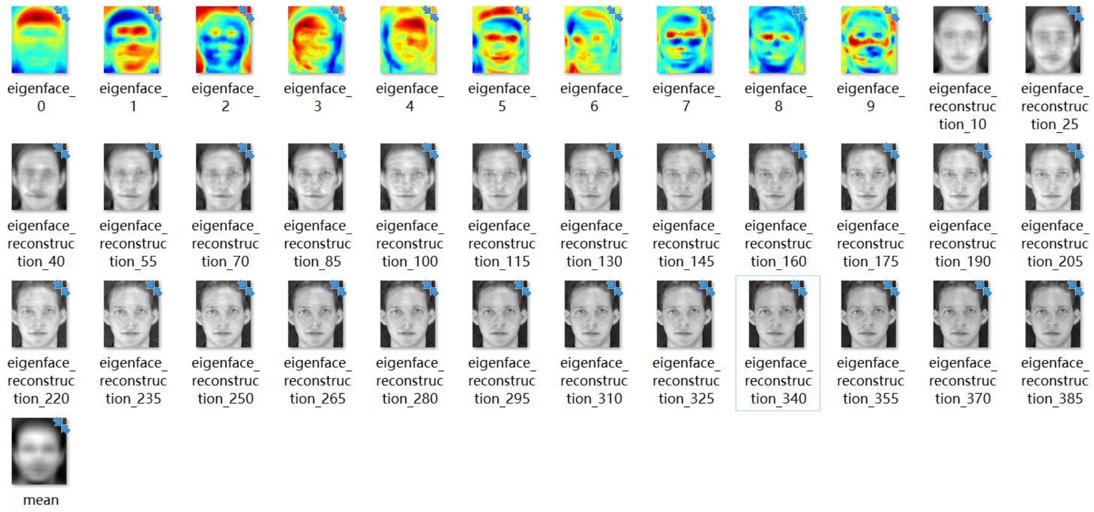
3、 进行人脸鉴定，观察人脸特征，判断是否是个人；

4、 最后进行人脸识别。

3.2（d）题

> 在 Eigenface 实验中，你可以使用 eigenfaces (特征脸，即特征向量) 来重构近似的人脸图像. 修改 OpenCV 教程中的源代码，并使用不同数量的 eigenfaces 来观察可视化的结果. 如果你希望从 eigenfaces 中重构的人脸看上去与原始输入的人脸图像之间难以区分，那么你需要多少张 eigenfaces?

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：


eigenface_0
eigenface_1
eigenface_2
eigenface_3
eigenface_4
eigenface_5
eigenface_6
eigenface_7
eigenface_8
eigenface_9


fisherface_0
fisherface_1
fisherface_2
fisherface_3
fisherface_4
fisherface_5
fisherface_6
fisherface_7
fisherface_8
fisherface_9