数字图像处理

基于 matlab 的人脸识别项目

学生姓名	WCW
学　　号	xxxxx332
专业班级	21 级 1 班
学　　院	网安
完成时间	2024.1.14

目录

一　项目描述

数字图像处理课程 Final Project—基于 PCA 的人脸识别 ：

1.算法 PCA 人脸识别或 Eigenfaces 人脸识别（见人脸识别课件）

2.采用数据库为剑桥大学 ORL 人脸数据库，包含 40 个人的 400 张人脸图像 （每人对应 10 张），图像为 92x112 灰度图像（256 灰度级），数据库：由主讲教师提供。

3.对于每个人的 10 张图像，随机选择 5 张用来训练，另外 5 张用于测试。对于每人的五张训练图像，可以将五张训练图像平均后作为一个特征图像再进行 PCA 特征抽取。

4.选择合适的特征维数，建议为 50-100；采用 2 范数（欧式距离）最小匹配。

5.对每个人的另外 5 张训练图像分别测试，共测试 5x40 个图像，计算识别系统的正确率 =（识别正确的图像数）/200。

6．可以使用 Matlab 或 Python 的工具库。

二　项目-算法原理（PCA 特征提取）

PCA（Principal Component Analysis，主成分分析）是一种常用于降维和特征提取的技术，而在人脸识别中，PCA 被广泛应用。以下是 PCA 在人脸识别中的基本原理：

1. 数据收集和准备： 收集大量的人脸图像数据集，确保每个图像都对齐和归一化，以减小光照和尺度变化的影响。

2. 数据降维： 对于每张人脸图像，将其像素矩阵展开成一个向量。将所有人脸图像的向量组合成一个矩阵，其中每一列代表一个人脸样本，而行表示像素。

3. 中心化： 对数据矩阵进行中心化，即减去每一列的均值，以去除平移对分析的影响。

4. 协方差矩阵计算： 计算中心化后的数据矩阵的协方差矩阵。协方差矩阵描述了数据中不同维度之间的关系。

5. 特征值分解： 对协方差矩阵进行特征值分解，得到特征值和特征向量。特征向量构成了原始特征空间的新基，而特征值表示了在每个新方向上的数据分散程度。

6. 特征向量排序： 将特征值按降序排列，选择前 k 个特征值对应的特征向量，其中 k 是希望保留的主成分数量。

7. 投影： 将数据投影到由选定的主成分构成的子空间上。这可以通过将数据矩阵

乘以选定的特征向量矩阵来实现。

8. 特征脸： 主成分向量通常被称为特征脸。这些特征脸是通过 PCA 学到的最能代表人脸数据变异的方向，它们可以用于表示原始人脸图像的重要信息。

9. 分类： 对投影后的数据进行分类，可以使用各种分类器，如最近邻分类器等。

通过 PCA，我们能够在保留主要信息的同时减少数据的维度，提高计算效率，并且对于人脸识别等应用，可以更好地捕捉人脸图像中的关键特征。

三 算法流程

这里以训练过程的算法流程为例，对于测试过程的算法流程类似：

1.计算 40 个人每个人的特征人脸图的均值图

```
eigen_matrix_avg = mean(eigen_matrix,2);
```

2. 所有 40 个人的人脸图都减去这个均值图

```
dif = eigen_matrix - repmat(eigen_matrix_avg,1,40);
```

3. 计算协方差矩阵 L_matrix，原来 dif:mn*40 维度，计算出 L_matrix：40*40 维度

```
L_matrix = dif'*dif;
```

4. 计算协方差矩阵的特征向量矩阵 W 和特征值矩阵 G

```
[W, G] = eig(L);
```

注意：协方差矩阵 L 最多有 39 个特征值

5. 将 dif 投射到所有特征向量生成的空间 W 中

```
dif_W = dif*W ;
```

6. 找到前 dif_W 中的对应位置的 k 列个 mn 维列向量

```
dif_Wk = dif_W(:,k_index);
```

7. 将 dif_Wk（mn*k 维度）矩阵的每一列都进行归一化处理

```
dif_Wk(:,i) = dif_Wk(:,i)./norm(dif_Wk(:,i));
```

8. 把 dif 投射到 dif_Wk 空间，得到 k*40.这样每个人只要 k 维度的向量就可以表示其特征了

```
eigen_dif_matrix = dif_Wk'*dif;
```

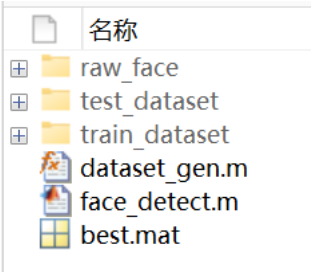
另外测试过程还有“特征图距离”的计算：

利用 dif_Wk 计算这张图片的特征空间的系数——>计算这个 score_vec 和所有 40 个列向量的差值——>计算二范数距离

```
score_vec = dif_Wk'*(tmp_img-eigen_matrix_avg);
score_dif = eigen_dif_matrix - repmat(score_vec,1,40);
value_dif = zeros(1,40);
```

四　安装使用运行说明

1. **项目的文件布局如下：**（raw_face，test_dataset 和 train_dataset 文件夹需要保留）



其中，

（1） raw_face 文件夹：就是老师提供的 400 张图像文件夹

（2） test_dataset：用于存放划分后的 200 张测试图像集（运行时更改）

（3） train_dataset：用于存放划分后的 200 张训练图像集（运行时更改）

（4） “best.mat”文件，里面存放了训练好的模型参数，这是我选取的效果最好的一个

（5） 两个代码文件：dataset_gen.m —— 用于随机给每个人分 5 张训练和测试图

Face_detect.m —— 包含 训练 + 测试 + 正确率输出

2. 运行使用：

使用一：使用最好的 model 进行测试

在包含 3 个数据相关文件夹（后两个可以为空）的前提下，直接运行 face_detect.m 代码即可输出随机划分的测试集的正确率

使用二：想要自己重新 train 模型——请注释掉 load 代码即可，可以更改全局参数 k 进行调整

```
load('best.mat');
```
这行代码——需要注释

五　代码展示与结果分析

下面，我将按照 face_detect.m 代码文件中的各个部分进行展示

1.代码逐部分展示：

（1）第一部分，全局参数的设置，主要是 k 值的选取，最终每个图像的特征长度由 k 决定

```
%-----------------------------------------part1:全局参数设置:
m = 112;   %行
n = 92;    %列
k = 36;    %特征长度
```

（2）第二部分，运行 dataset_gen.m 函数，将原来 raw_face 文件夹下面的 40 个人的图像进行随机的 5-5 划分，分别存到 train_dataset 和 test_dataset 文件夹下。

```
%-----------------------------------------part2:调用dataset_gen.
%随机划分train_dataset和test_dataset:
warning('off')
dataset_gen()
```

Dataset_gen.m 中的核心代码如下:

```
%最后，只要把raw_face\\s1-s40
%拆分到train_dataset\\s1-40和test_dataset\\s1-40即可
for j = 1:10
    if j < 6
        %把前1-5的random_1_10(j)的图片给到train_dataset
        train_dir = sprintf('train_dataset\\s%d\\%d.pgm',[i,j]);
        raw_dir = sprintf('raw_face\\s%d\\%d.pgm',[i,random_1_10(j)]);
        copyfile(raw_dir,train_dir);

    else
        %把后6-10的ranm_1_10(j)的图片给到test_dataset
        test_dir = sprintf('test_dataset\\s%d\\%d.pgm',[i,j-5]);
        raw_dir = sprintf('raw_face\\s%d\\%d.pgm',[i,random_1_10(j)]);
        copyfile(raw_dir,test_dir);
    end
end
```

（3）第三部分，train 训练部分，通过对 train_dataset 文件下的人脸用 PCA 算法提取得到最终的 dif_Wk，eigen_dif_matrix 和 eigen_matrix 和 eigen_matrix_avg 这三个用在 test 中的量。

```
%-----------------------------------------part3:train训练过程:
%总共40个人，每个人有5张图像，最后取平均，就得到了我们所需要的eigen_matrix_avg
eigen_matrix = [];
for i = 1:40
    total_img = zeros(m*n,1);
    for j = 1:5
        tmp_dir = sprintf('train_dataset\\s%d\\%d.pgm',[i,j]);
        tmp_img = im2double(imread(tmp_dir));
        tmp_img = reshape(tmp_img,m*n,1);
        total_img = total_img + tmp_img;
    end
    eigen_matrix = cat(2,eigen_matrix,total_img./5);
end

%最终eigen_matrix的维度是 mn*40
%下面这个eigen_matrix_avg的维度是 mn*1
eigen_matrix_avg = mean(eigen_matrix,2);              %计算所有人的平均特征一个列向量
dif = eigen_matrix - repmat(eigen_matrix_avg,1,40);  %减去均值后的矩阵
L_matrix = dif'*dif;            %计算协方差矩阵 L_matrix = 40*40
[W,G] = eig(L_matrix);          %协方差矩阵最多39个特征值

dif_W = dif*W ;                 %将dif投射到所有特征向量生成的空间W中
```

```
G_arr = diag(G,0);                    %去除所有的特征值组成一个一维的向量

%找出前k个最大的特征值的index下标
[values,indexes] = sort(G_arr,'descend');
k_index = indexes(1:k);

%找到前dif_W中的对应位置的k列个mn维列向量
dif_Wk = dif_W(:,k_index);
%按"列"数据进行归一化操作:
for i = 1:k
    dif_Wk(:,i) = dif_Wk(:,i)./norm(dif_Wk(:,i));
end

%最后，只要将原来的dif: mn*40 ，与dif_Wk:mn*k
%把dif投射到dif_Wk空间，得到k*40.这样每个人只要k维度的向量就可以表示其特征
eigen_dif_matrix = dif_Wk'*dif;
```

将 V_k：mn*k 拆成 k 张 m*n 图像就可以输出 k 张 eigen_face 的结果

```
%生成k个特征脸--把dif_Wk拆开成k个特征图像
figure
for i = 1:k
    I = reshape(dif_Wk(:, i), m, n);
    subplot(5, 8, i), imshow(I, []), title(['no.', num2str(i), ''])
end
```

（4）第四部分，test 测试部分，利用 train 好的 model 参数，先计算需要测试的图像投影到选取的 dif_Wk 空间后的 k*1 的特征向量，与已经计算好的 40 个人各自的特征模板进行二范数距离，取最小的那个作为预测结果。

```
load('best.mat');

acc = 0;
for i = 1:40
    for j = 1:5
        %读取test_dataset中的si文件夹下的第j张pgm
        tmp_dir = sprintf('test_dataset\\s%d\\%d.pgm',[i,j]);
        tmp_img = im2double(imread(tmp_dir));
        tmp_img = reshape(tmp_img,m*n,1);
        %利用dif_Wk —— 计算这张图片的特征空间的系数:
        score_vec = dif_Wk'*(tmp_img-eigen_matrix_avg);
        %计算这个score_vec和所有40个列向量的差值
        score_dif = eigen_dif_matrix - repmat(score_vec,1,40);
        %计算二范数距离
        value_dif = zeros(1,40); %初始化一个1*40个数组，每个位置存一个距离值
        for k = 1:40
            value_dif(k) = sum(score_dif(:,k).*score_dif(:,k));
        end
        %根据这个value_dif数组每个位置的距离值最小的那个，作为预测的结果
        [tmp_value,tmp_index] = min(value_dif);
        acc = acc+sum(tmp_index == i);

    end
end
```

（5）第五部分，计算平均 acc，并输出

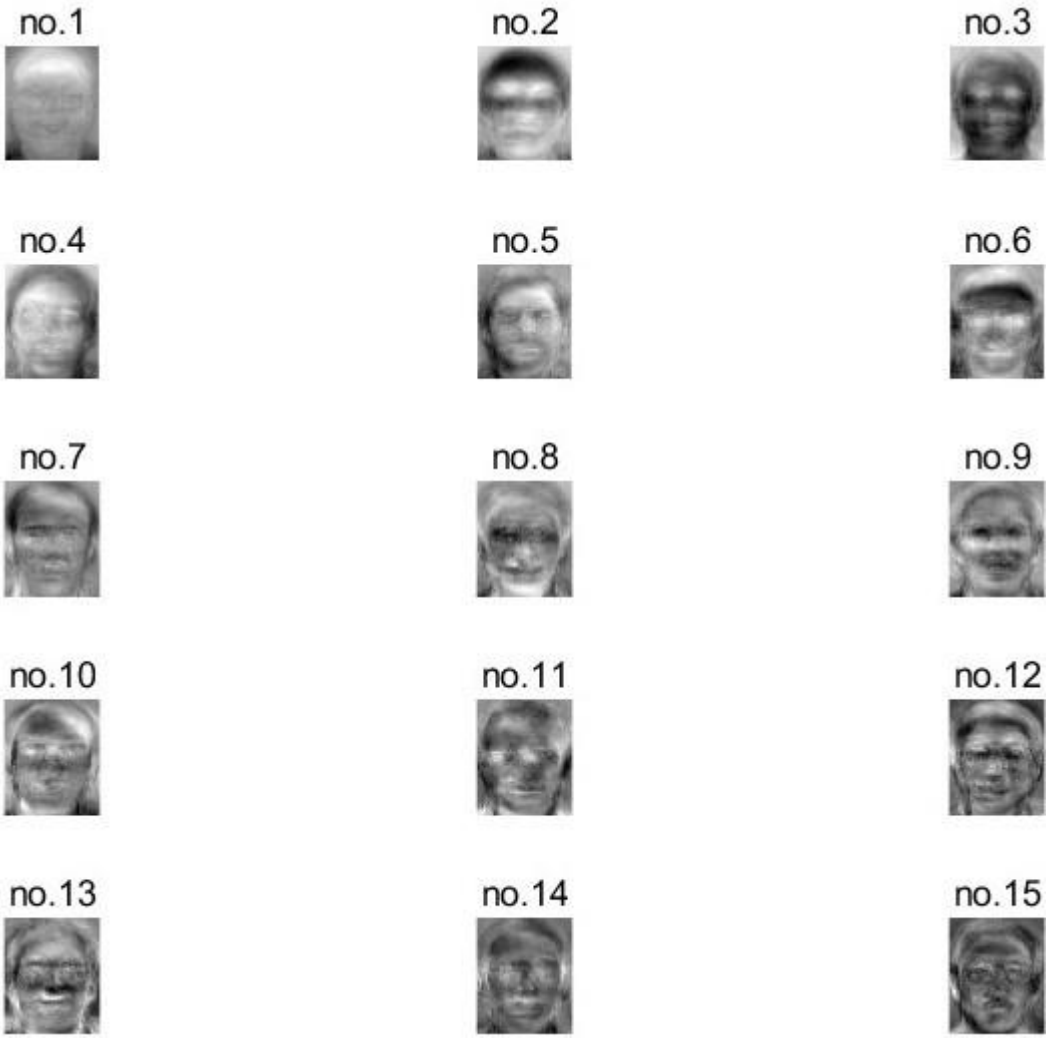
```
avg_acc = acc/200;
disp(avg_acc)
```

窗口

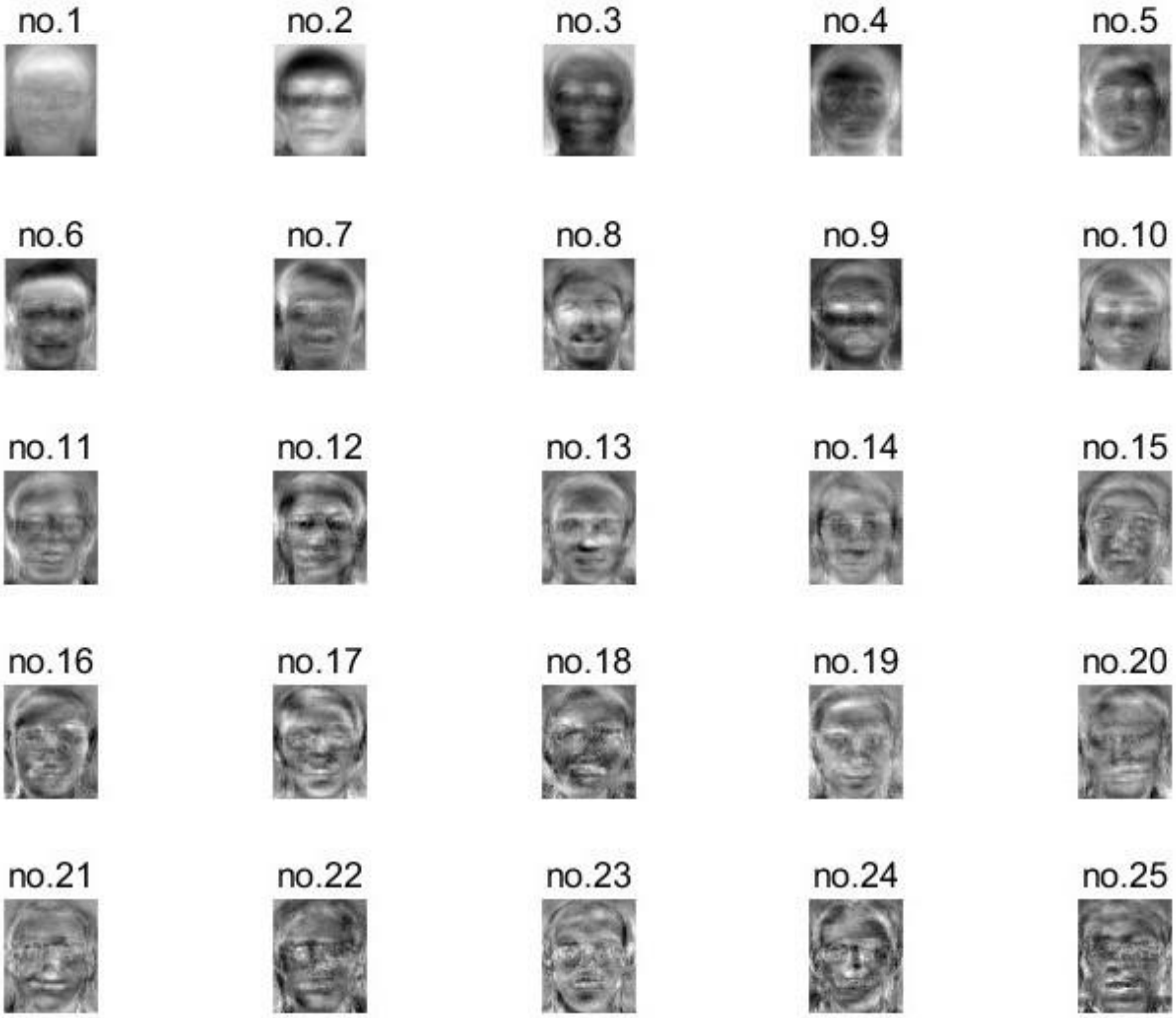
```
face_detect
 0.9800
```

2.eigen_face 结果展示与分析：

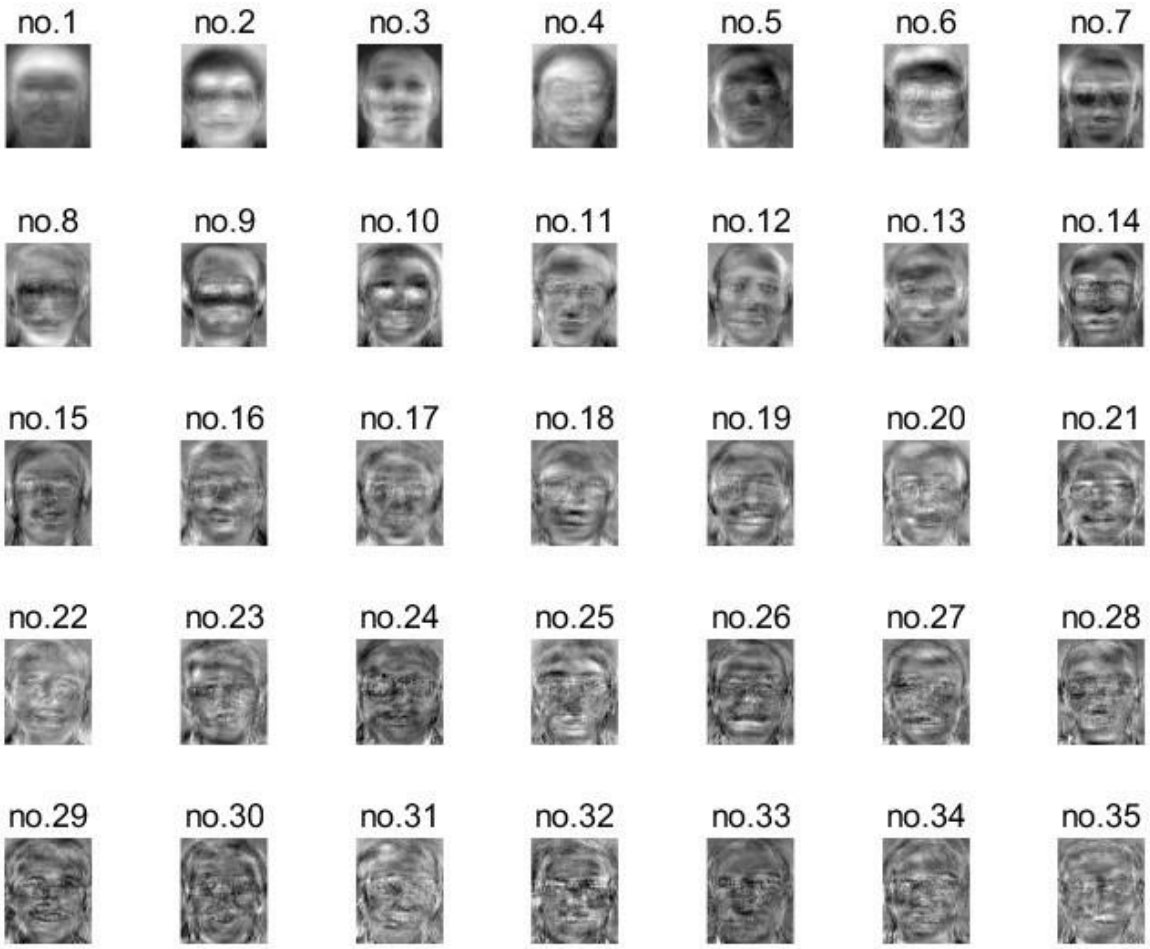
（1）k=15


no.1
no.2
no.3
no.4
no.5
no.6
no.7
no.8
no.9
no.10
no.11
no.12
no.13
no.14
no.15

(2) k=25


no.1
no.2
no.3
no.4
no.5
no.6
no.7
no.8
no.9
no.10
no.11
no.12
no.13
no.14
no.15
no.16
no.17
no.18
no.19
no.20
no.21
no.22
no.23
no.24
no.25

(3) k=35



分析：eigen_face 主要保留了 k 个该空间的“基图像”，最终用它们的组合可以恢复得到人脸的主要部分。

3.正确率表格与分析：

（1）k=15，25，35，38，39

K 值	正确率—均值	截图
K=15	84%	>> face_detect 0.8750
K=25	90%	>> face_detect 0.8750
K=35	93%	>> face_detect 0.9200

K=38	95%	>> face_detect 0.9600
K=39	95%	>> face_detect 0.9850

（2）分析：正确率——既受到 k 值的影响，同时，也会受到图像集随机划分的影响，一般情况下，都是 k 越大，分类的正确率越大。