Face Recognition

1．算法描述

Principal Components Analysis (PCA)

定义：

N 个向量 xi，每个向量有 d 个元素，用列向量表示，每个 xi 都代表一个训练集图片。N 就是训练集图片数量，40 个人，每个人 7 张训练集图片，N=7*40=280。d 是每张图片的像素数量，d= 92x112。

xi ∈ R^d

我们的目标是，将原向量从 d 维空间投影到 k 维空间，称为降维，我们使用 PCA 降维。

PCA 算法的步骤

训练阶段。

1. 计算表示训练集图片像素的列向量的均值，就是计算出 xi 的均值。

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i, \mathbf{x}_i \in R^d, \overline{\mathbf{x}} \in R^d$$

2. 每个列向量 xi 都减去均值。

$$\overline{\mathbf{x}}_\mathbf{i} = \mathbf{x}_\mathbf{i} - \overline{\mathbf{x}}$$

3. 计算步骤 2 得到的向量的协方差矩阵 C。我们求的是协方差的无偏估计，由于减去了均值，故除以(N-1)。

$$\mathbf{C} = \frac{1}{N-1}\sum_{i=1}^{N}\bar{\mathbf{x}}_i\bar{\mathbf{x}}_i^T = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x_i}-\bar{\mathbf{x}})(\mathbf{x_i}-\bar{\mathbf{x}})^T, \textit{Note}: \mathbf{C} \in R^{d\times d}$$

$$\textit{Note}: \mathbf{C} \text{ is a symmetric matrix, and it is positive-semidefinite}$$

4. 求协方差矩阵 C 的特征向量。V 是特征向量矩阵，每一列都是一个特征向量。

$$\mathbf{CV} = \mathbf{V\Lambda}, \mathbf{V} \in R^{d\times d}, \mathbf{\Lambda} \in R^{d\times d}$$

$\mathbf{V}$ – matrix of eigenvectors (each column is an eigenvector),

$\mathbf{\Lambda}$ – diagonal matrix of eigenvalues V: d维，共d个

Note: $\mathbf{V}$ is an orthonormal matrix (i.e. $\mathbf{VV}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$), as $\mathbf{C}$ is a covariance matrix and hence it is symmetric.

Note: $\mathbf{\Lambda}$ contains non-negative values on the diagonal (eigen-values)

5. 根据协方差矩阵 C 的特征值的大小，提取最大的 k 个特征值对应的 k 个特征向量，称为 eigenspace。

Extract the *k* eigenvectors corresponding to the *k* largest eigenvalues. This is called the extracted **eigenspace:**

V_k: d维，k个

$$\hat{\mathbf{V}}_\mathbf{k} = \mathbf{V}(:,1:k)$$

There is an implicit assumption here that the first *k* indices indeed correspond to the *k* largest eigenvalues. If that is not true, you would need to pick the appropriate indices.

6. 将 $\bar{\mathbf{x}}_\mathbf{i}$ 投影到 eigenspace，对于每个 $\bar{\mathbf{x}}_\mathbf{i}$，得到一个向量αik，αik 有 k 个元素，每个元素称为 eigen-coefficients。我们将 $\bar{\mathbf{x}}_\mathbf{i}$ 表示成 k 个特征值最大的特征向量的线性组合，线性组合的系数就是 eigen-coefficients。

$$\boldsymbol{\alpha}_{ik} = \hat{\mathbf{V}}_{\mathbf{k}}^{\mathbf{T}} \overline{\mathbf{x}}_{\mathbf{i}}, \boldsymbol{\alpha}_{ik} \in R^k; \boldsymbol{\alpha}_i = \mathbf{V}^T \overline{x}_i, \boldsymbol{\alpha}_i \in R^d$$

As $\mathbf{V}$ is orthonormal, we have

$$\overline{\mathbf{x}}_{\mathbf{i}} = \mathbf{V}\boldsymbol{\alpha}_{\mathbf{i}} = \mathbf{V}(:,1)\boldsymbol{\alpha}_i(1) + \mathbf{V}(:,2)\boldsymbol{\alpha}_i(2) + \ldots + \mathbf{V}(:,d)\boldsymbol{\alpha}_i(d)$$

$$\approx \hat{\mathbf{V}}_{\mathbf{k}}\boldsymbol{\alpha}_{ik} = \hat{\mathbf{V}}_{\mathbf{k}}(:,1)\boldsymbol{\alpha}_{ik}(1) + \hat{\mathbf{V}}_{\mathbf{k}}(:,2)\boldsymbol{\alpha}_{ik}(2) + \ldots + \hat{\mathbf{V}}_{\mathbf{k}}(:,\underset{k}{\cancel{d}})\boldsymbol{\alpha}_{ik}(k)$$

We are representing each face as a linear combination of the *k* eigenvectors corresponding to the *k* largest eigenvalues. The coefficients of the linear combination are the eigen-coefficients.

Note that $\boldsymbol{\alpha}_{\mathbf{ik}}$ is a vector of the eigencoefficients of the *i*-th sample point, and it has *k* elements. The j-th element of this vector is denoted as $\boldsymbol{\alpha}_{\mathbf{ik}}$ (j).

7. 保存每个测试集图像的 eigen-coefficient 和此图像对应的人的身份到数据库中。保存 $\hat{\mathbf{V}}_{\mathbf{k}}, \overline{\mathbf{x}}$ 到数据库中。

测试阶段。

测试阶段,以列向量的形式给你一个测试图像 Zp,Zp 有 d 个元素。(Zp 和 xi 的形式一样)。

1. Zp 减去训练阶段步骤 1 得到的均值。

$$\overline{\mathbf{z}}_{\mathbf{p}} = \mathbf{z}_{\mathbf{p}} - \overline{\mathbf{x}}$$

2. 将 $\overline{\mathbf{z}}_{\mathbf{p}}$ 投影到 eigen-space，得到 eigen-coefficients $\boldsymbol{\alpha}_{\mathbf{p}}$

$$\boldsymbol{\alpha}_{\mathbf{p}} = \hat{\mathbf{V}}_{\mathbf{k}}^{\mathbf{T}}\overline{\mathbf{z}}_{\mathbf{p}} \longrightarrow \text{Eigen-coefficients of the probe image } \mathbf{z}_p.$$

3. 将αp 和αik 对比，计算αp 和αik 的欧几里得距离（平方距离），采用 2 范数最小匹配，

   当欧几里得距离最小时，得到αik，αik 对应的人，就是人脸识别匹配到的人。

PCA 算法的优化

当 N 远小于 d 的时候,训练集图片数量远小于每张图片的像素数量时,有如下的优化方法。

考虑协方差矩阵 C。

$$\mathbf{C} = \frac{1}{N-1}\sum_{i=1}^{N}\overline{\mathbf{x}}_i\overline{\mathbf{x}}_i^T \propto \mathbf{X}\mathbf{X}^T, \text{where}$$

$$\mathbf{X} = [\overline{\mathbf{x}}_1 \mid \overline{\mathbf{x}}_2 \mid \ldots \mid \overline{\mathbf{x}}_N] \in R^{d\times N}$$

考虑矩阵$\mathbf{X}^T\mathbf{X}$，大小为 N*N，而不考虑矩阵$\mathbf{X}\mathbf{X}^T$，大小为 d*d。

$\mathbf{X}^T\mathbf{X}$的特征向量有如下形式。

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \lambda\mathbf{w}, \mathbf{w} \in R^N$$

等式两边左乘 X，得 $XX^T(Xw) = \lambda(Xw)$，将 Xw 看做一个整体，就是 Ax=λx 的形式。

我们就得到了 XX^T 的特征向量 Xw，就是协方差矩阵 C 的特征向量。

这种方法计算协方差矩阵 C 的特征向量的时间复杂度，等于

计算矩阵 X^TX 的特征向量的时间复杂度 $O(N^3)$+对于每个 w，计算 Xw 的时间复杂度 O(N*dN)

等于 $O(N^3+dN^2)$，远远小于原来的时间复杂度 $O(d^3)$。

优化后的算法步骤

训练阶段

1. 计算表示训练集图片像素的列向量的均值，就是计算出 xi 的均值。

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i, \mathbf{x}_i \in R^d, \overline{\mathbf{x}} \in R^d$$

2. 每个列向量 xi 都减去均值。

$$\overline{\mathbf{x}}_\mathbf{i} = \mathbf{x}_\mathbf{i} - \overline{\mathbf{x}}$$

3. 计算矩阵 L。

$$\mathbf{L} = \mathbf{X}^T\mathbf{X}, \mathbf{L} \in R^{N\times N}, \mathbf{X} = [\overline{\mathbf{x}}_1 \mid \overline{\mathbf{x}}_2 \mid \ldots \mid \overline{\mathbf{x}}_N] \in R^{d\times N}$$

$$\mathit{Note}: \mathbf{L} \text{ is a symmetric matrix, and it is positive-semidefinite}$$

4. 计算 L 的特征向量矩阵 W。

$$\mathbf{LW} = \mathbf{W\Gamma}, \mathbf{W} - eigenvectors, \mathbf{\Gamma} - eigenvalues$$

$$\mathbf{WW}^T = \mathbf{I}$$

5. 根据 W 得到协方差矩阵 C 的特征向量。

$$\mathbf{V} = \mathbf{XW}, \mathbf{X} \in R^{d\times N}, \mathbf{W} \in R^{N\times N}, \mathbf{V} \in R^{N\times N}$$

d

6. 将 V 的列向量归一化。

7. 根据协方差矩阵 C 的特征值的大小，提取最大的 k 个特征值对应的 k 个特征向量，称为 eigenspace。

8. 将 $\bar{\mathbf{x}}_\mathbf{i}$ 投影到 eigenspace，对于每个 $\bar{\mathbf{x}}_\mathbf{i}$，得到一个向量αik，αik 有 k 个元素，每个元素称为 eigen-coefficients。我们将 $\bar{\mathbf{x}}_\mathbf{i}$ 表示成 k 个特征值最大的特征向量的线性组合，线性组合的系数就是 eigen-coefficients。

9. 保存每个测试集图像的 eigen-coefficient 和此图像对应的人的身份到数据库中。保存 $\hat{\mathbf{V}}_\mathbf{k}, \bar{\mathbf{X}}$到数据库中。

测试阶段不变。

2. 代码

generate_random_sequence_to_define_training_and_test_images.m 文件，里面有 generate_random_sequence_to_define_training_and_test_images 函数。

对于每个人的 10 张图像，随机选择 7 张用来训练，另外 3 张用于测试。

生成 1-10 的不重复的随机数列，前 7 个数作为训练图像，后 3 个数作为测试图像。一共生成 40 个这样的数列。

training_set_index 是 40*7 的矩阵。第 i 行表示第 i 个人的 7 个训练图像的索引。

例如，第 5 行是 1 3 4 5 8 9 10，则第 5 个人的训练图像是图 1 3 4 5 8 9 10。

test_set_index 是 40*3 的矩阵。第 i 行表示第 i 个人的 3 个测试图像的索引。

```
function [training_set_index, test_set_index] = generate_random_sequence_to_define_training_and_test_images()
    % 生成不重复的随机整数序列,用来随机选择训练集和测试集
    % training_set_index 是训练集
    % test_set_index 是测试集
    global num_training_face num_trainingpic_per_face
    global num_test_face num_testpic_per_face
    training_set_index = zeros(num_training_face, num_trainingpic_per_face);
    test_set_index = zeros(num_test_face, num_testpic_per_face);
    generate_times = num_training_face;
    generate_number = num_trainingpic_per_face + num_testpic_per_face;
    for i=1:generate_times
        random_index = randperm(generate_number);
        training_set_index(i, :) = sort( random_index(1: num_trainingpic_per_face) );
        test_set_index(i, :) = sort( random_index(num_trainingpic_per_face + 1: generate_number) );
    end
end
```

Face_Recognition_main.m 文件是主文件。

```
for i = 1:test_times
    [training_set_index, test_set_index] = generate_random_sequence_to_define_training_and_test_images();
    k_index = 0;
    test_info_saved_index = i;
    for k = k_start : k_end
        k_index = k_index + 1;
        Face_Recognition();
    end
end
```

总共测试 test_times 轮。每一轮先生成训练集和测试集索引，然后对于[k_start,k_end]区间内的每个 k 值都进行测试。

第 i 轮，提取的特征向量数量为 k 时，正确率保存在 correct_rate_list(i,k)中。

```
k_rate_list = sum(correct_rate_list) / test_times;
[k_rate_sort,k_rate_I] = sort(-k_rate_list);
k_rate_sort = -k_rate_sort;
k_index_list = k_index_list(k_rate_I);
fprintf('best k :%d , correct rate: %.4f\n', k_index_list(1), k_rate_sort(1));
```

sum(correct_rate_list) / test_times 得到，提取的特征向量数量为 k 时，test_times 次测试的平均正确率。排序后得到正确率最高的 k。

correct_rate_list.xlsx 保存了 correct_rate_list 的一个样例。

Face_Recognition.m 文件，实现了 PCA 优化后的算法。代码较长，只说关键步骤。

function [pic, pic_identity, avg_training_pic] =

read_training_images_and_deduct_mean_from_images()

此函数实现功能：读入训练集图片，求出像素均值，每个图片像素值都减去均值

定义：

m*n 是图片的像素。

pic 用来存放所有训练集图片 m*n*k

k 是训练集图片数量，一个特定的 k0 表示 m*n 大小的一张图片。

pic(:, i) 是列向量 xi。

pic_identity(i) 是第 i 个图片的身份信息。

sum_training_pic 训练集图片的像素值之和。

```
% Compute the mean of the given images
avg_training_pic = sum_training_pic / num_training_pic;

% Deduct the mean from each point:
for i=1:num_training_pic
    pic(:, 1, i) = pic(:, 1, i) - avg_training_pic;
end
```

function [L, picX] = compute_XTX(pic)

此函数实现功能：快速计算特征向量 计算矩阵 L = X^{T}X。

定义：picX 就是 X。 $\mathbf{X} = [\bar{\mathbf{x}}_1 \mid \bar{\mathbf{x}}_2 \mid \ldots \mid \bar{\mathbf{x}}_N] \in R^{d \times N}$

```
function [L, picX] = compute_XTX(pic)
    % 快速计算特征向量 计算矩阵 L = X^TX
    % X^T是X的转置
    global num_training_pic
    global p
    % Compute X^TX size N*N reducing computational complexity
    picX = zeros(p, num_training_pic);
    for i=1:num_training_pic
        picX(:,i) = pic(:, 1, i);
    end
    L = picX' * picX ;
    %L = L / (num_training_pic - 1);
end
```

function [V,D] = calculate_normalized_eigenvectors_and_eigenvalues_of_C(L, picX)

此函数实现功能：求 L 的特征向量 W，特征值 D，根据 W 求协方差矩阵 C 的特征向量 V，V = XW。然后将 V 单位化。

```
function [V,D] = calculate_normalized_eigenvectors_and_eigenvalues_of_C(L, picX)
    % 求L的特征向量 W ， 特征值D, V = XW, V是协方差矩阵C的特征向量
    % 然后将V单位化
    global num_training_pic
    % Find the eigenvectors W of L
    [W,D] = eig(L);
    % Obtain the eigenvectors of C from those of L
    V = picX * W;
    % Unit-normalize the columns of V
    for i=1:num_training_pic
        V(:, i) = V(:, i) ./ norm( V(:,i) );
    end
end
```

function [Veig, VeigTrans] =

pick_eigenvectors_corresponding_to_k_largest_eigenvalues(V, D)

此函数实现功能：根据特征值大小降序排列，对应的特征向量也重新排列，选出 k 个最大的特征值对应的特征向量，组成 eigenspace

定义：Veig 是 eigenspace,VeigTrans 是 Veig 的转置（方便计算）

```
function [Veig, VeigTrans] = pick_eigenvectors_corresponding_to_k_largest_eigenvalues(V, D)
    % 根据特征值大小降序排列，对应的特征向量也重新排列，选出k个最大的特征值对应的特征向量，组成eigenspace
    % Veig是eigenspace,VeigTrans是Veig的转置
    global k
    [Dsort,I] = sort(diag(-D));
    % -Dsort才是按照降序排列的C的特征值D
    Dsort=-Dsort;
    % 特征向量根据特征值降序排列
    V = V(:, I);
    % Extract the k eigenvectors corresponding to the k largest eigenvalues. This is called the extracted eigenspace
    Veig = V(:, 1:k);
    VeigTrans = Veig';
end
```

function coefficients_eig = project_image_onto_eigenspace(VeigTrans, pic)

此函数实现功能：将 $\overline{\mathbf{x}}_i$ 投影到 eigenspace。

定义：coefficients_eig 是 eigen-coefficients。列向量 i 表示 $\overline{\mathbf{x}}_i$ 投影到 eigenspace 的结果αik。

```
function coefficients_eig = project_image_onto_eigenspace(VeigTrans, pic)
    % 将每个图片减去像素平均值之后的向量映射到eigenspace,得到eigen-coefficients
    % coefficients_eig 是 eigen-coefficients 是老师pdf中的aik
    % Project each point onto the eigenspace, giving a vector of k eigen-coefficients for that point
    % global num_training_face num_trainingpic_per_face num_training_pic
    global num_training_pic
    global k
    coefficients_eig = zeros(k, num_training_pic);

    for i=1:num_training_pic
        coefficients_eig(:, i) = VeigTrans * pic(:, 1, i);
    end
end
```

function [result, correct_rate_info] =

test_phase(pic_identity, VeigTrans, coefficients_eig, avg_training_pic)

此函数实现功能：

测试阶段，先将测试图像从 n*m 矩阵变为 p*1 列向量 Zp p=n*m

Zp 减去训练阶段求出的图像各个坐标的平均像素值

映射 Zp 到 eigenspace,求出 Zp 的 eigen-coefficients

求 Zp 的 eigen-coefficients 和训练集图片的 eigen-coefficients 的

欧几里得距离的平方 jp

jp 最小值对应的训练集图片就是匹配图片。

定义:

num_test_face 提供图片的测试者数量

num_testpic_per_face 每个测试者的测试图片数量

success_match 成功匹配的数量 fail_match 错误匹配的数量

sum_diff 和 difference 是测试图像和训练图像的欧几里得距离的平方

test_index 测试图像下标

下面两个函数实现的是优化前的方法。

function [covariance] = covariance_matrix(pic)

此函数实现功能：求协方差矩阵 C

定义：covariance 是协方差矩阵。

```
function [covariance] = covariance_matrix(pic)
    global num_training_face num_trainingpic_per_face num_training_pic
    global p
    % Compute the covariance matrix of these mean-deducted points
    % use XX^T size d*d
    covariance = zeros(p,p);
    for i=1:num_training_face
        for j=1:num_trainingpic_per_face
            kk = (i-1) * num_trainingpic_per_face + j;
            covariance = covariance + pic(:, 1, kk) * pic(:, 1, kk)';
        end
    end
    covariance = covariance / (num_training_pic - 1);
end
```

function [V,D] = get_eigenvectors_of_covariance_matrix(covariance)

此函数实现功能：求协方差矩阵 C 的特征向量 V 和特征值 D

```
function [V,D] = get_eigenvectors_of_covariance_matrix(covariance)
    % 求协方差矩阵C的特征向量V和特征值D
    % Find the eigenvectors of C covariance
    [V,D] = eig(covariance);
end
```

外层函数 function [] = Face_Recognition()

此函数实现功能：整个 PCA 优化后算法。如果想用优化前的方法，将优化后的方法后面两行加上注释，将优化前的方法后面两行的注释取消。

```
[pic, pic_identity, avg_training_pic] = read_training_images_and_deduct_mean_from_images();

% 优化后的方法
[L, picX] = compute_XTX(pic);
[V,D] = calculate_normalized_eigenvectors_and_eigenvalues_of_C(L, picX);

% 优化前的方法
% covariance = covariance_matrix(pic);
% [V,D] = get_eigenvectors_of_covariance_matrix(covariance);

[~, VeigTrans] = pick_eigenvectors_corresponding_to_k_largest_eigenvalues(V, D);
coefficients_eig = project_image_onto_eigenspace(VeigTrans, pic);
[result, correct_rate_info] = test_phase(pic_identity, VeigTrans, coefficients_eig, avg_training_pic);
```

运行方法

打开 Face_Recognition_main.m 文件，F5 运行。

可以调整的参数，test_times 测试轮数，测试 k 值的最小值 k_start，最大值 k_end。

效率对比。

k 取 50。

使用优化前的方法，完成一次测试花费的时间：319.7237s

由于优化前的方法时间复杂度较高，不重复测试求平均值。

其中，计算协方差矩阵 C 花费时间 170.0099s。

计算协方差矩阵 C 的特征向量和特征值花费时间 146.5489s。

说明优化前的方法，计算协方差矩阵 C 和计算 C 的特征向量、特征值花费大量时间。

使用优化后的方法，完成 10 次测试的总时间：10.9299s

平均一次测试花费的时间：1.0930s

完成每次测试花费的时间：

1.2975	1.1279	1.1282	1.066	1.061	1.0551	1.0559	1.0793	1.0202	1.0388

可以看出，优化后的方法大大减少了时间复杂度，减少了花费的时间。

使用优化后的方法，完成 10 次测试的平均正确率：0.9775

每次测试的正确率：

0.9583	0.9833	0.9667	0.9667	1	0.9833	0.975	0.975	0.9833	0.9833

可以看到，测试的正确率波动较大，训练集和测试集的选择对正确率的影响较大。甚至出现过正确率为 1 的情况。

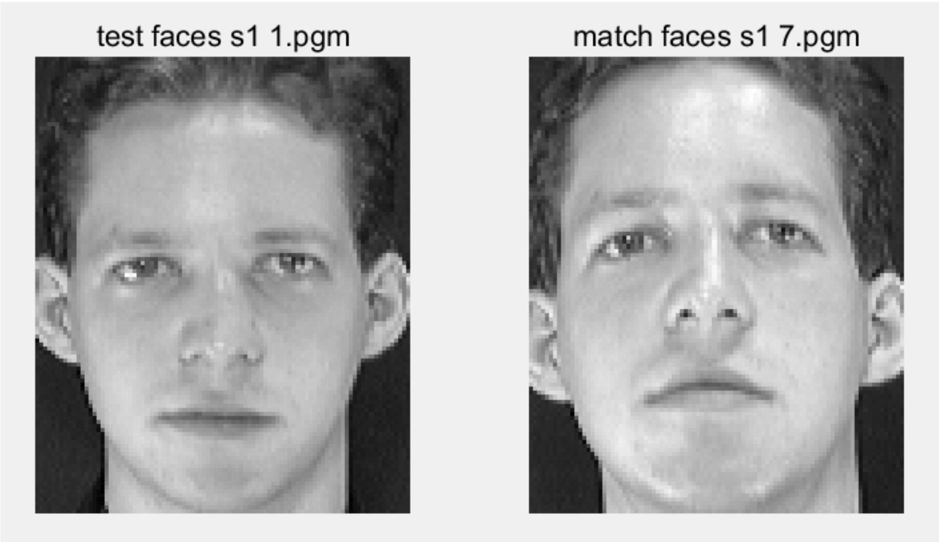
选取最好的 k，k=50 最好，正确率是 0.9658

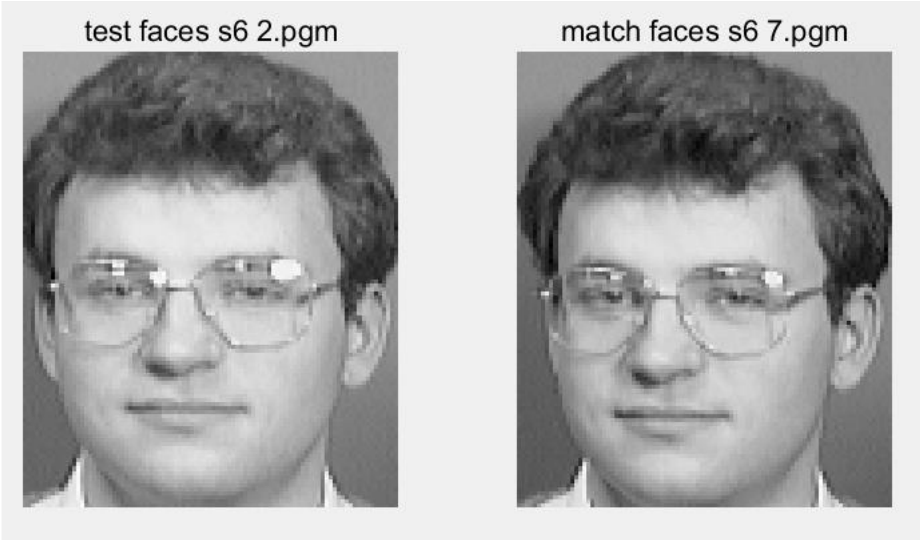
对于[50,100]之间的 k，总共进行 10 轮测试，每轮测试，不同的 k 的训练集和测试集相同。得到平均正确率，在 correct_rate_list.xlsx 文件中，由于结果太大，只展示[50,60]之间的 k 的平均正确率。

0.9658	0.965	0.965	0.9633	0.9633	0.9633	0.9633	0.9633	0.9633	0.9641	0.9641

最好的 k=50，正确率是 0.9658。

部分人脸识别结果（左图是测试图片，右图是匹配图片）





细节。

老师在 pdf 中提到了。

3. 对于每个人的10张图像，随机选择7张用来训练，另外3张用于测试。对于每人的7张训练图像，可以将7张训练图像平均后作为一个特征图像再进行PCA特征抽取。

按照这种方法，以将 7 张训练图像平均后作为一个特征图像再进行 PCA 特征抽取，用于 PCA 特征抽取的图像数量 N 是 40。

Eigen-faces: Algorithm (*N* << *d* case)

4. Find the eigenvectors of **L**:

$$\mathbf{LW} = \mathbf{W\Gamma}, \mathbf{W} - eigenvectors, \mathbf{\Gamma} - eigenvalues,$$

$$\mathbf{WW}^T = \mathbf{I}$$

可以验证一下对于L最大的k个特征向量是否对应C的最大的特征向量

5. Obtain the eigenvectors of **C** from those of **L**:

$$\mathbf{V} = \mathbf{XW}, \mathbf{X} \in R^{d\times N}, \mathbf{W} \in R^{N\times N}, \boxed{\mathbf{V} \in R^{\underset{d}{\cancel{N}}\times N}}$$

1. 得到最大的k个V_i
2. 对于每个X_j，投影到k个特征向量得到其特征

6. Unit-normalize the columns of **V**.
7. **C** will have at most only *N* eigenvectors corresponding to non-zero eigen-values*. Out of these you pick the top $\boxed{k\ (k < N)}$ corresponding to the largest eigen-values.

* Actually this number is at most *N*-1 – this is due to the mean subtraction, else it would have been at most *N*.

当 N=40 时，求出来的协方差矩阵 C 的特征向量矩阵大小是 d*N，我们得到的特征向量的数量是 N=40 个。

而老师在 pdf 中又提到，建议 k 取 50-100，(k < N)，此时我们得到的特征向量只有 40 个，无法满足 k 取 50-100， 感觉将 7 张训练图像平均后作为一个特征图像再进行 PCA 特征抽取，会忽略掉同一个人的不同训练图像之间区别，降低人脸识别准确率。所以我没有用这种方法。

4. 选择合适的特征维数，建议为$\boxed{50\text{-}100}$；采用2范数最小匹配。