

数字图像处理课程Final Project

基于PCA的人脸识别

学 校: 中山大学

学 院: 数据科学与计算机学院

专 业: 计算机科学与技术

姓 名: 郑 康 泽

学 号: 17341213

一. 项目描述

1. 算法PCA人脸识别或Eigenfaces人脸识别。
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（每个人对应10张），图像为 92×112 灰度图像（256灰度级）。
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5张训练图像，可以将5张训练图像平均后作为一个特征图像再进行PCA特征提取。
4. 选择合适的特征维数，建议为50 ~ 100；采用2范数最小匹配。
5. 对每个人的另外5张训练图像分别测试，共测试 5×40 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的工具库。

二. PCA原理及方法

1. Principle Components Analysis(PCA)介绍

考虑 N 个向量或者点 \mathbf{x}_i ，每个向量（点）是 d 维的，表示成列向量的形式，即 $\forall i, \mathbf{x}_i \in \mathbb{R}^d$ 。因为 d 可能会很大，所以我们想要从 \mathbf{x}_i 的 d 维特征中中抽取出 k 维特征，来表示 \mathbf{x}_i ，其中 $k \ll d$ 。通常我们有效地将原来的向

量（点）从 d 维空间映射到 k 维空间（有效指的是样本之间区分度要足够高，即特征要足够明显），这就叫做降维，而PCA就是一种降维的方法。

2. $k = 1$ 时 \vec{e} 的取值

我们的目标是找到一条通过样本均值点 $\bar{\mathbf{x}}$ 的线 \vec{e} ($\|\vec{e}\| = 1$)，使得任意的点减去均值($\mathbf{x}_i - \bar{\mathbf{x}}$)后，投影到 \vec{e} 上的投影最精确地近似于 $\mathbf{x}_i - \bar{\mathbf{x}}$ 。

$\mathbf{x}_i - \bar{\mathbf{x}}$ 投影到 \vec{e} 的投影为 $a_i \vec{e}$ ，其中 $a_i = \vec{e}^t (\mathbf{x}_i - \bar{\mathbf{x}})$ ， $a_i \in \mathbb{R}$ （根据余弦公式可以得出）。这种近似的误差为 $\|a_i \vec{e} - (\mathbf{x}_i - \bar{\mathbf{x}})\|^2$ ，那么对于所有的样本，总体误差为：

$$\begin{aligned}
 J(\vec{e}) &= \sum_{i=1}^N \|a_i \vec{e} - (\mathbf{x}_i - \bar{\mathbf{x}})\|^2 \\
 &= \sum_{i=1}^N \|a_i \vec{e}\|^2 + \sum_{i=1}^N \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 - 2 \sum_{i=1}^N a_i \vec{e}^t (\mathbf{x}_i - \bar{\mathbf{x}}) \\
 &= \sum_{i=1}^N a_i^2 + \sum_{i=1}^N \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 - 2 \sum_{i=1}^N a_i \vec{e}^t (\mathbf{x}_i - \bar{\mathbf{x}}) \\
 &= \sum_{i=1}^N a_i^2 + \sum_{i=1}^N \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 - 2 \sum_{i=1}^N a_i^2, (\because a_i = \vec{e}^t (\mathbf{x}_i - \bar{\mathbf{x}})) \\
 &= - \sum_{i=1}^N a_i^2 + \sum_{i=1}^N \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 \\
 &= - \sum_{i=1}^N (\vec{e}^t (\mathbf{x}_i - \bar{\mathbf{x}}))^2 + \sum_{i=1}^N \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 \\
 &= - \sum_{i=1}^N \vec{e}^t (\mathbf{x}_i - \bar{\mathbf{x}}) (\mathbf{x}_i - \bar{\mathbf{x}})^t \vec{e} + \sum_{i=1}^N \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 \\
 &= - \vec{e}^t \mathbf{S} \vec{e} + \sum_{i=1}^N \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 \\
 &= - \vec{e}^t \cdot (N-1) \mathbf{C} \cdot \vec{e} + \sum_{i=1}^N \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2, \\
 &\text{where } \mathbf{C} = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}}) (\mathbf{x}_i - \bar{\mathbf{x}})^t
 \end{aligned}$$

因为 $\sum_{i=1}^N \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2$ 与 \vec{e} 无关，所以最小化 $J(\vec{e})$ 等价于最大化 $\vec{e}^t \mathbf{S} \vec{e}$ 。同时有约束 $\vec{e}^t \vec{e} = 1$ ，利用拉格朗日乘数法构建拉格朗日函数：

$$L(\vec{e}, \lambda) = \vec{e}^t \mathbf{S} \vec{e} - \lambda (\vec{e}^t \vec{e} - 1)$$

求 $L(\vec{e}, \lambda)$ 对 \vec{e} 的偏导，然后令该偏导为零，得到 $\mathbf{S} \vec{e} = \lambda \vec{e}$ 。所以 \vec{e} 是 \mathbf{S} 的特征向量， λ 是对应的特征值。因为 $\mathbf{S} \vec{e} = \lambda \vec{e}$ ，所以 $\vec{e}^t \mathbf{S} \vec{e} = \lambda$ ，所以为了最大化 $\vec{e}^t \mathbf{S} \vec{e}$ ，我们选择最大的特征值 λ_0 对应的特征向量 \vec{e}_0 。这样我们就找到能最小化 $J(\vec{e})$ 的 \vec{e} 。

3. $k > 1$ 时 $\{\mathbf{e}_j\}_{j=1}^k$ 的取值

对于减去均值的样本点 $\mathbf{x}_i - \bar{\mathbf{x}}$ ，我们将它们从 d 维投影到 k 维，并且使得投影前的样本点与投影后的样本点之间的平方误差和最小。总体的平方误差为：

$$\begin{aligned} J(\{\mathbf{e}_j\}_{j=1}^k) &= \sum_{i=1}^N \left\| (\mathbf{x}_i - \bar{\mathbf{x}}) - \sum_{j=1}^k \mathbf{e}_j^t (\mathbf{x}_i - \bar{\mathbf{x}}) \mathbf{e}_j \right\|_2^2 \\ &= - \sum_{j=1}^k \mathbf{e}_j^t \mathbf{S} \mathbf{e}_j + \sum_{i=1}^N \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 \end{aligned}$$

利用归纳法可以证明，选择 \mathbf{S} 的前 k 大的特征值对应的 k 个特征向量作为 $\{\mathbf{e}_j\}_{j=1}^k$ ，即可使得 $J(\{\mathbf{e}_j\}_{j=1}^k)$ 最小。

4. PCA算法流程

1. 计算样本点的均值：

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

2. 所有样本点减去均值得到新的样本点：

$$\bar{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$$

3. 计算新的样本点的协方差矩阵：

$$\mathbf{C} = \frac{1}{N-1} \sum_{i=1}^N \bar{\mathbf{x}}_i \bar{\mathbf{x}}_i^t = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^t, \mathbf{C} \in \mathbb{R}^{d \times d}$$

4. 计算 \mathbf{C} 的特征值和特征向量：

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

\mathbf{V} 是特征向量矩阵（列向量是特征向量），并且 \mathbf{V} 是正交的，即 $\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$ ； $\mathbf{\Lambda}$ 是特征值矩阵，特征值在 $\mathbf{\Lambda}$ 的对角线上，且因为协方差矩阵是对称的，所以特征值都不小于0。

5. 抽取出前 k 大的特征值对应的 k 个特征向量：

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

这里假设恰好前 k 列就是前 k 大的特征值对应的 k 个特征向量。这 k 个特征向量组成抽取后的特征空间的基向量。

6. 将新的样本点映射到抽取后的特征空间中，并计算各个的基向量的系数：

$$\mathbf{a}_{ik} = \hat{\mathbf{V}}_k^T \bar{\mathbf{x}}_i, \mathbf{a}_{ik} \in \mathbb{R}^k$$

7. 对于要预测的点 \mathbf{z}_p ，首先减去样本点的均值，得到 $\bar{\mathbf{z}}_p = \mathbf{z}_p - \bar{\mathbf{x}}$ 。然后计算 $\bar{\mathbf{z}}_p$ 在抽取后的特征空间中的系数： $\mathbf{a}_p = \hat{\mathbf{V}}_k^T \bar{\mathbf{z}}_p$ ，接着与每个新的样本点在抽取后的特征空间的系数向量 \mathbf{a}_{ik} 计算距离，找到与其中一个新的样本点之间距离最小的新的样本点，并预测该点的类别为

找到的样本点的类别。其中距离的计算方法有多种，如下：

- $j_p = \arg \min_i \|\mathbf{a}_p - \mathbf{a}_{ik}\|_2^2$
- $j_p = \arg \min_i \|\mathbf{a}_p - \mathbf{a}_{ik}\|_1$
- $j_p = \arg \min_i \frac{\mathbf{a}_p \cdot \mathbf{a}_{ik}}{\|\mathbf{a}_p\| \|\mathbf{a}_{ik}\|}$

5. 上述算法的缺点

因为 \mathbf{C} 是 $d \times d$ ，所以计算 \mathbf{C} 的特征值及特征向量需要 $O(d^3)$ 的时间复杂度，而 d 如果是一张图的像素点的个数的话，即 d 可能是10,000甚至更多，那么这个计算将是非常耗时；同时可以看出，储存这个协方差矩阵 \mathbf{C} 也是十分耗内存的，因此上述算法是不可取的。

6. 改良算法（当 $N \ll d$ ）

因为只有 N 个样本点，所以协方差矩阵 \mathbf{C} 的秩最多为 $N - 1$ ，即 \mathbf{C} 最多有 $N - 1$ 个互异的非零的特征值，因此我们不需要计算出 d 个特征向量（其中会有重复）。观察到 $\mathbf{C} = \frac{1}{N-1} \sum_{i=1}^N \bar{\mathbf{x}}_i \bar{\mathbf{x}}_i^t \propto \mathbf{X} \mathbf{X}^T$ ，其中 $\mathbf{X} = [\bar{\mathbf{x}}_1 \ \bar{\mathbf{x}}_2 \ \dots \ \bar{\mathbf{x}}_N] \in \mathbb{R}^{d \times N}$ ，而 $\mathbf{X} \mathbf{X}^T$ 的协方差矩阵可以由 $\mathbf{X}^T \mathbf{X}$ 的协方差矩阵计算得到，而计算 $\mathbf{X}^T \mathbf{X}$ 的协方差矩阵的时间复杂度为 $O(N^3)$ ，这就大大降低时间复杂度。推导过程如下：

$\mathbf{X}^T \mathbf{X}$ 的特征值 λ 和特征向量 \mathbf{w} 关系为： $\mathbf{X}^T \mathbf{X} \mathbf{w} = \lambda \mathbf{w}$ ， $\mathbf{w} \in \mathbb{R}^N$ ，两边同时乘以 \mathbf{X} ，得到 $\mathbf{X} \mathbf{X}^T (\mathbf{X} \mathbf{w}) = \lambda (\mathbf{X} \mathbf{w})$ ，即 $\mathbf{X} \mathbf{w}$ 是矩阵 $\mathbf{X} \mathbf{X}^T$ 的特征向量， λ 为对应的特征值。所以对于矩阵 $\mathbf{X}^T \mathbf{X}$ 的每一个特征向量 \mathbf{w} ，左乘以一个 \mathbf{X} 后就是矩阵 $\mathbf{X} \mathbf{X}^T$ 的特征向量。利用以上方法求得协方差矩阵 \mathbf{C} 的特征值和特征向量的时间复杂度为 $O(N^3 + dN^2) = O(dN^2)$ ，这是远远小于直接算协方差矩阵 \mathbf{C} 的时间复杂度 $O(d^3)$ 的。

7. 改良PCA算法流程

1. 计算样本点的均值：

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

2. 所有样本点减去均值得到新的样本点：

$$\bar{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$$

3. 计算下面这个矩阵：

$$\mathbf{L} = \mathbf{X}^T \mathbf{X}, \mathbf{L} \in \mathbb{R}^{N \times N}, \mathbf{X} = [\bar{\mathbf{x}}_1 \ \bar{\mathbf{x}}_2 \ \dots \ \bar{\mathbf{x}}_N] \in \mathbb{R}^{d \times N}$$

4. 计算矩阵 \mathbf{L} 的特征值和特征向量：

$$\mathbf{L} \mathbf{W} = \mathbf{W} \mathbf{\Gamma}$$

\mathbf{W} 是特征向量矩阵， $\mathbf{\Gamma}$ 是特征值矩阵。

5. 利用 \mathbf{W} 计算矩阵 $\mathbf{C} = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^t$ 的特征向量：

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

对应的特征值是 $\mathbf{\Gamma}$ 。

6. 将 \mathbf{V} 的每个列向量归一化，即每个列向量的模长都化为1。

7. 抽取出前 k 大的特征值对应的 k 个特征向量（ $k < N$ ）：

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

这里假设恰好前 k 列就是前 k 大的特征值对应的 k 个特征向量。这 k 个特征向量组成抽取后的特征空间的基向量。

8. 将新的样本点映射到抽取后的特征空间中，并计算各个的基向量的系数：

$$\mathbf{a}_{ik} = \hat{\mathbf{V}}_k^T \bar{\mathbf{x}}_i, \mathbf{a}_{ik} \in \mathbb{R}^k$$

9. 对于要预测的点 \mathbf{z}_p ，首先减去样本点的均值，得到 $\bar{\mathbf{z}}_p = \mathbf{z}_p - \bar{\mathbf{x}}$ 。然后计算 $\bar{\mathbf{z}}_p$ 在抽取后的特征空间中的系数： $\mathbf{a}_p = \hat{\mathbf{V}}_k^T \bar{\mathbf{z}}_p$ ，接着与每个新的样本点在抽取后的特征空间的系数向量 \mathbf{a}_{ik} 计算距离，找到与其中一个新的样本点之间距离最小的新的样本点，并预测该点的类别为找到的样本点的类别。距离的计算方法在4.7中有提到。

三. 实验步骤及代码展示

1. 划分数据集

根据要求，每个人的5张图像用作训练，另外5张图像用作测试。可以利用 `randperm(n)` 函数，产生一个随机打乱顺序的1~10的数字序列，然后前5个数字对应的图像用作训练，后5个数字对应的图像用作测试。实现函数如下：

```
function divide_data(class_num, per_class_pic_num, pattern)
% class_num: 类别数量
% per_class_pic_num: 每个类别的训练集或者测试集数量
% pattern: 数据集的路径
% 本函数的功能: 划分数据集

% 创建训练集和测试集文件夹
if ~exist('train_data', 'dir')
    mkdir('train_data');
end
if ~exist('test_data', 'dir')
    mkdir('test_data');
end

% 每个类别的数据集一半作为训练集，一般作为测试集
for i = 1:class_num
```

```

% 随机打乱序号
random_index = randperm(per_class_pic_num * 2);

% 训练集
path = sprintf('train_data\\s%d', i);
if ~exist(path, 'dir')
    mkdir(path);
end

for j = 1:per_class_pic_num;
    des = sprintf('train_data\\s%d\\%d.pgm', [i, j]);
    copyfile(sprintf(pattern, [i, random_index(j)]), des)
end

% 测试集
path = sprintf('test_data\\s%d', i);
if ~exist(path, 'dir')
    mkdir(path);
end

for j = 1:per_class_pic_num
    des = sprintf('test_data\\s%d\\%d.pgm', [i, j]);
    copyfile(sprintf(pattern, ...
        [i, random_index(per_class_pic_num + j)]), des)
end

end
end

```

2. 训练

根据要求，将每个人的5张图像平均后当作一张特征图像 \mathbf{I}_i ，那么就有40张特征图像。将这些特征图像的矩阵表示 \mathbf{I}_i 变形为列向量表示 \mathbf{x}_i ，并将40个列向量拼在一起，然后每一个列向量 \mathbf{x}_i 减去40个列向量的均值 $\bar{\mathbf{x}}$ ，形成算法中的 \mathbf{X} ，接着算出矩阵 $\mathbf{X}^T \mathbf{X}$ 的特征值矩阵 $\mathbf{\Gamma}$ 和特征向量矩阵 \mathbf{W} ，计算协方差矩阵 \mathbf{C} 的特征向量矩阵 $\mathbf{V} = \mathbf{XW}$ 并对每列归一化，并根据特征值矩阵 $\mathbf{\Gamma}$ ，从特征向量矩阵 \mathbf{V} 选出前 k 大特征值对应的 k 个特征向量形成一个 $d \times k$ 的矩阵 $\hat{\mathbf{V}}_k$ ，作为抽取后的特征空间的基向量。然后计算每一个减去均值后的列向量 $\bar{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$ 在抽取后的特征空间中系数向量 $\mathbf{a}_{ik} = \hat{\mathbf{V}}_k^T \bar{\mathbf{x}}_i$ 。实现函数如下：

```

function [character_pics_mean, v_k, A_k] = ...
    train(m, n, k, class_num, per_class_pic_num, pattern)
% m: 图像的行数
% n: 图像的列数
% k: 选取的特征向量个数
% class_num: 类别数量
% per_class_pic_num: 每个类别的训练集数量
% pattern: 训练集的路径
% 本函数的功能: 返回特征图像的均值, 特征图像的协方差矩阵的k个特征向量,
% 以及特征图像在k个特征向量组成的特征空间的系数

```

```

% 每个人的五张图像平均后作为一个特征图像
character_pics = [];
for i = 1:class_num
    all_images = zeros(m * n, 1);
    for j = 1:per_class_pic_num
        I = im2double(imread(sprintf(pattern, [i, j])));
        I = reshape(I, m * n, 1);
        all_images = all_images + I;
    end
    character_pics = cat(2, character_pics, ...
        all_images ./ per_class_pic_num);
end

% 特征图像的均值
character_pics_mean = mean(character_pics, 2);

% X
X = character_pics - repmat(character_pics_mean, 1, class_num);

% L = X^T * X
L = X' * X;
[w, G] = eig(L);

% v是协方差矩阵的特征向量矩阵
V = X * w;

% 找到特征值前k大的索引
g = diag(G, 0);
index = find_k_max(g, k);

% 取特征值前k大的对应的特征向量
V_k = V(:, index);
for i = 1:k
    V_k(:, i) = V_k(:, i) ./ norm(V_k(:, i));
end

% 特征图像在特征空间的系数
A_k = V_k' * X;

% eigenfaces
figure
for i = 1:k
    I = reshape(V_k(:, i), m, n);
    subplot(5, k/5, i), imshow(I, []), title(['第', num2str(i), '个',
eigenface'])
end

end

function index = find_k_max(A, k)
% A: 向量
% k: int
% 本函数的功能：找到A中数字前k大的对应的索引

[~, indices] = sort(A, 'descend');
index = indices(1:k);
end

```

3. 测试

对于每一个测试图像，先将矩阵表示 \mathbf{I} 变形为列向量表示 \mathbf{z}_p ，然后减去训练步骤中得到 $\bar{\mathbf{x}}$ ，得到 $\bar{\mathbf{z}}_p = \mathbf{z}_p - \bar{\mathbf{x}}$ ，将训练步骤中得到的 $\hat{\mathbf{V}}_k$ 乘上 $\bar{\mathbf{z}}_p$ 得到 $\bar{\mathbf{z}}_p$ 在抽取后的特征空间中系数向量 \mathbf{a}_p ，然后计算 \mathbf{a}_p 与每一个 \mathbf{a}_{ik} 的距离，距离最小的 i 对应的 \mathbf{I}_i 是哪个人的图像，就预测 \mathbf{I} 是那个人的图像。实现函数如下：

```
function accuracy = test(m, n, class_num, per_class_pic_num, pattern,...
    character_pics_mean, v_k, A_k)
% m: 图像的行数
% n: 图像的列数
% k: 选取的特征向量个数
% class_num: 类别数量
% per_class_pic_num: 每个类别的训练集数量
% pattern: 测试集的路径
% character_pics_mean: 特征图像的均值
% v_k: 特征图像的协方差矩阵的k个特征向量
% A_k: 以及特征图像在k个特征向量组成的特征空间的系数
% 本函数的功能: 计算测试集的准确率

% 初始化准确率
accuracy = 0;

for i = 1:class_num
    for j = 1:per_class_pic_num
        I = im2double(imread(sprintf(pattern, [i, j])));
        I = reshape(I, m * n, 1);

        % 该图像在特征空间的系数
        a = v_k' * (I - character_pics_mean);

        % 得到的系数与特征图像的系数作差
        dif = A_k - repmat(a, 1, class_num);

        % 根据差算出二范数
        dist = zeros(1, class_num);
        for k = 1:class_num
            dist(k) = sum(dif(:, k) .* dif(:, k));
        end

        % 取二范数最小值对应的类别作为预测类别
        [~, prediction] = min(dist);

        % 预测正确
        if prediction == i
            accuracy = accuracy + 1;
        end
    end
end

% 最终的准确率
accuracy = accuracy / (class_num * per_class_pic_num);
```



```
end
```

4. 主函数

为了不用每次都训练，可以将训练得到的参数保存进mat文件中，然后在测试的时候导入就行。主函数如下：

```
clc
clear

m = 112;
n = 92;
k = 30;
class_num = 40;
per_class_pic_num = 5;
pattern1 = 'ORL_faces\\s%d\\%d.pgm';
pattern2 = 'train_data\\s%d\\%d.pgm';
pattern3 = 'test_data\\s%d\\%d.pgm';

% 训练
if ~exist('model.mat', 'file')
    % 划分数据集
    divide_data(class_num, per_class_pic_num, pattern1)

    % 计算参数
    [character_pics_mean, v_k, A_k] = ...
        train(m, n, k, class_num, per_class_pic_num, pattern2);

    % 保存参数
    save('model.mat', 'character_pics_mean', 'v_k', 'A_k');
end

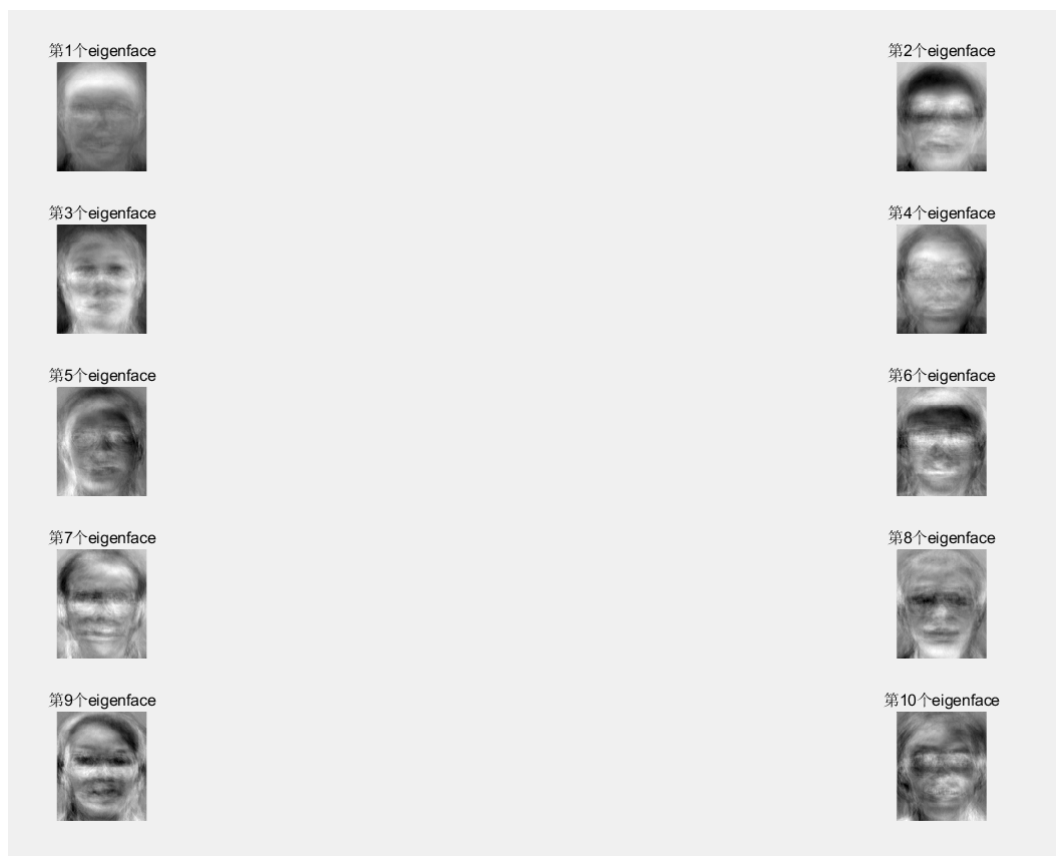
% 导入参数
load('model.mat');

% 测试
accuracy = test(m, n, class_num, per_class_pic_num, pattern3, ...
    character_pics_mean, v_k, A_k);
disp(accuracy);
```

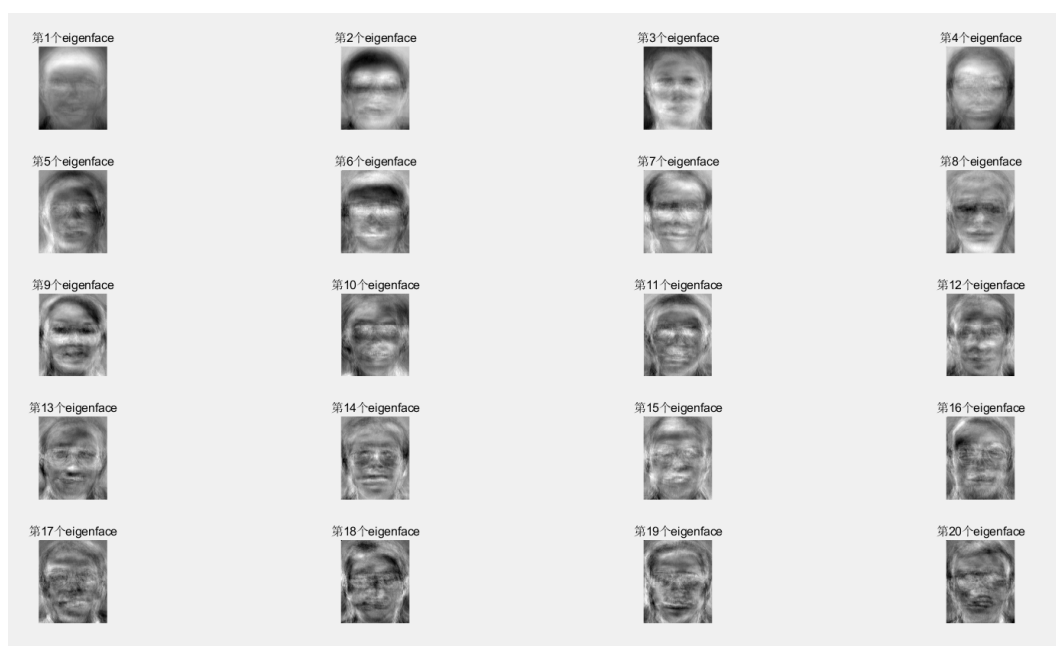
四. 实验结果展示

1. eigenfaces展示

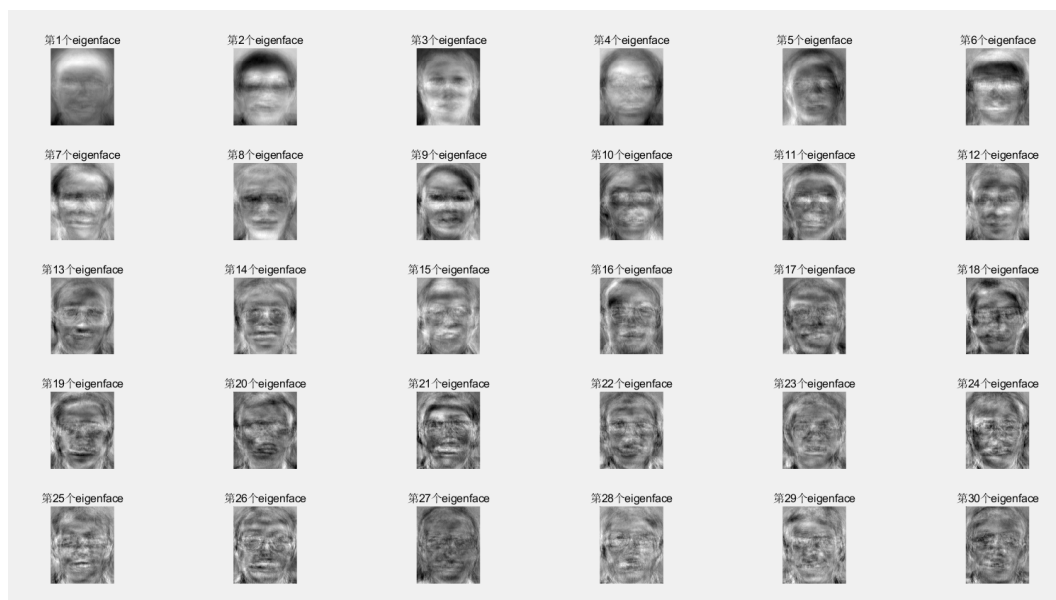
1. $k = 10$ 时10个eigenfaces



2. $k = 20$ 时20个eigenfaces



3. $k = 10$ 时10个eigenfaces



2. 测试性能表格

k 的取值范围为 $(0, N)$ 即 $(0, 40)$ 。

k	准确率
10	85.5%
20	91.5%
30	94%
35	94.5
37	95%
39	95%

准确率不仅会受 k 值的影响，也会受数据集划分的影响。当 k 越来越接近39，准确率会越来越高。固定 $k = 39$ ，测试不同划分下的准确率，经测试，准确率在87%到95%之间。