EigenFace 方法人脸识别

一、 实验目的

1、熟悉常用特征降维方法，掌握 PCA 降维的基本原理；
2、熟悉 Eigenface 人脸识别的基本流程，实现 Eigenface 算法，对训练样本进行训练；
3、分析特征脸方法的优缺点；

二、 实验原理

1） PCA 降维

●为什么要降维：

原始观察空间中的样本具有极大的信息冗余；
样本的高维数引发分类器设计的“维数灾难”；
数据可视化、特征提取、分类与聚类等任务需求；

●线性降维

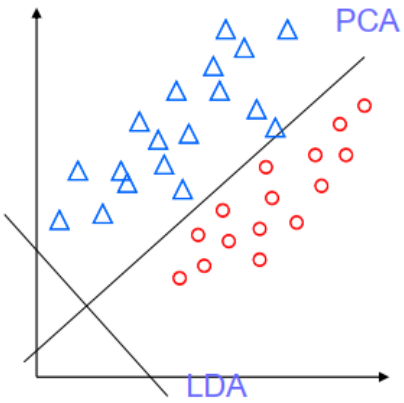
通过特征的线性组合来降维
本质上是把数据投影到低维线性子空间
线性方法相对比较简单且容易计算
代表方法有：主成分分析(PCA)，线性判别分析(LDA)，多维尺度变换(MDS)

●LDA

寻找最能把两类样本分开的投影，使投影后两类样本的均值之差与投影样本的总类散度的比值最大。把原问题转化为关于样本集总类内散度矩阵和总类间散度矩阵的广义特征值问题。

●PCA 降维整体思想

寻找能够保持采样数据方差的最佳投影子空间。对样本的散度矩阵进行特征值分解，所求子空间为经过样本均值，以最大特征值所对应的特征向量为方向的子空间。这样也就相当于只保留包含绝大多数方差的维度特征。实现数据特征的降维处理。PCA 对于椭球状分布的样本有很好的效果，学习所得的主方向就是椭球的主轴方向。但 PCA 是一种非监督算法，能找到很好代表所有样本的方向，但这个方向对于分类未必是最有利的。



PCA 与 LDA 降维方向比较

●PCA 算法步骤:

设有 m 条 n 维数据

① 将原始数据按列组合成 n 行 m 列的矩阵 A，每一列代表一个样本

② 将 A 的每一行（代表同一个属性的字段）进行零均值化，即减去这一行的均值

③ 求出协方差矩阵 C=$\frac{1}{m}A\cdot A^T$

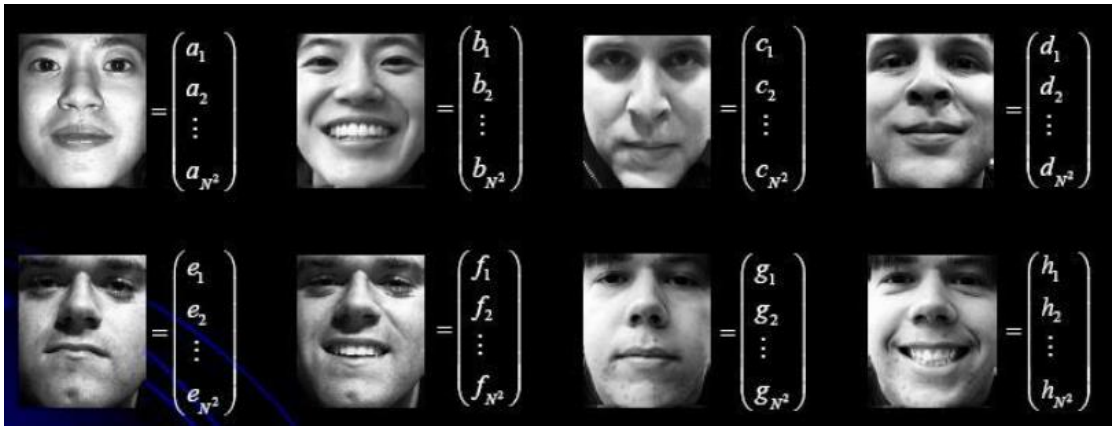
④ 求出协方差矩阵的特征值及对应特征向量

⑤将特征向量对应的特征值大小从上到小按行排列成矩阵，取前 k 行组成矩阵 P

⑥Y=P·A 即为降维到 k 维后的数据

2） Eigenface 算法

①将训练数据库中的每个人脸用一个列向量表示，该向量长为N^2=WH。如像素 128*128 的照片，用每个像素的灰度作为一个原始特征，N^2=128*128。M 是数据库中人脸的个数。



②计算平均脸，把每一个列向量按行求平均，即得平均脸的原始特征向量。

$$\mathbf{m}=\frac{1}{M}\begin{pmatrix} a_1+b_1+\cdots+h_1 \\ a_2+b_2+\cdots+h_2 \\ \vdots \\ a_{N^2}+b_{N^2}+\cdots+h_{N^2} \end{pmatrix}$$

③然后将训练集中的每个脸的的原始特征向量减去平均脸，并把这些向量按列组合成一个N^2*M 的矩阵[a，b，c，d，e，f，g，h]。

$$\mathbf{a}_m=\begin{pmatrix} a_1-m_1 \\ a_2-m_2 \\ \vdots \\ a_{N^2}-m_{N^2} \end{pmatrix} \quad \mathbf{b}_m=\begin{pmatrix} b_1-m_1 \\ b_2-m_2 \\ \vdots \\ b_{N^2}-m_{N^2} \end{pmatrix} \quad \mathbf{c}_m=\begin{pmatrix} c_1-m_1 \\ c_2-m_2 \\ \vdots \\ c_{N^2}-m_{N^2} \end{pmatrix} \quad \mathbf{d}_m=\begin{pmatrix} d_1-m_1 \\ d_2-m_2 \\ \vdots \\ d_{N^2}-m_{N^2} \end{pmatrix}$$

$$\mathbf{e}_m=\begin{pmatrix} e_1-m_1 \\ e_2-m_2 \\ \vdots \\ e_{N^2}-m_{N^2} \end{pmatrix} \quad \mathbf{f}_m=\begin{pmatrix} f_1-m_1 \\ f_2-m_2 \\ \vdots \\ f_{N^2}-m_{N^2} \end{pmatrix} \quad \mathbf{g}_m=\begin{pmatrix} g_1-m_1 \\ g_2-m_2 \\ \vdots \\ g_{N^2}-m_{N^2} \end{pmatrix} \quad \mathbf{h}_m=\begin{pmatrix} h_1-m_1 \\ h_2-m_2 \\ \vdots \\ h_{N^2}-m_{N^2} \end{pmatrix}$$

④找出矩阵协方差矩阵的特征值，Cov=$A \cdot A^T$。
显然这个矩阵维数将达到$N^2 * N^2$，计算量非常庞大。
但我们只对之多 M 个特征值感兴趣。
所以可以计算 L=A^T*A，Cov 的特征向量与 L 的特征向量是等价的。
而 L 是 M*M 矩阵，运算量小很多。从 L 得到特征向量 V。
⑤U=A*V，得到特征空间的基。
⑥计算每一张脸映射到特征空间的表示。

$$\Omega_1 = U^T(a_m) \quad \Omega_2 = U^T(b_m) \quad \Omega_3 = U^T(c_m)$$
$$\Omega_4 = U^T(d_m) \quad \Omega_5 = U^T(e_m) \quad \Omega_6 = U^T(f_m)$$
$$\Omega_7 = U^T(g_m) \quad \Omega_8 = U^T(h_m)$$

计算数据库的门槛

$$\theta = \frac{1}{2} max\left(\left\|\Omega_i - \Omega_j\right\|\right) \quad for\ j=1,2,\ldots,M$$

⑦识别一张脸:
得到该脸的列向量表示，将表达该脸的列向量减去训练集的平均脸。

$$\mathbf{r}_m = \begin{pmatrix} r_1 - m_1 \\ r_2 - m_2 \\ \vdots \\ r_{N^2} - m_{N^2} \end{pmatrix}$$

计算它映射到特征空间中的结果

$$\Omega = U^T(\mathbf{r}_m)$$

然后在特征空间上计算它与数据库所有已知脸的距离

$$\varepsilon_i^2 = \left\| \Omega - \Omega_i \right\|^2 \qquad for\ i = 1..M$$

从特征脸中重构这张脸，并计算重构结果与原始脸的距离

$$\mathbf{s} = U\Omega \qquad \xi^2 = \left\| \mathbf{r}_m - \mathbf{s} \right\|^2$$

若，ξ≥θ，说明这不是一张脸
若，ξ<θ 且 $\varepsilon_i \geq \theta$，i=1,……,M，说明这是一张新脸
若，ξ<θ 且 min{ε_i}<θ，说明这是一张已知的脸
Eigenface 流程：

训练流程

测试流程

三、 实验内容

本次实验使用 MATLAB 编程，因为 MATLAB 有很多方便使用的处理图像的函数。而且使用 MATLAB 进行矩阵运算非常方便。

MATLAB 代码:

```
clear all
s = dir('trainingset'); % 打开文件
s(1:2) = [];            % 因为前两项为“."".."",删去前两项
d = 128 * 128;          % 每张图片像素总数
M = length(s);          % 训练集内照片总数
A = zeros(d, M);        % d*n 矩阵
for k = 1:M
    image = strcat('trainingset/', s(k).name);%每次循环 image 为一张图片
    facek = imread(image);%从指定文件中读取图像,返回 M*N 数组
    facek = imresize(facek, [128, 128]);%改变图片的大小
    facek = reshape(facek, [d, 1]);%将 f 重新排序为矩阵 d*1
    facek = im2double(facek);%缩放到[0,1]
    A(:, k) = facek;%放入第 k 列
end

m = mean(A, 2);%返回每一行均值的列向量
aver=reshape(m,[128,128]);
aver=imresize(aver,[231,195]);%原图片为 231*195
subplot(1, 4, 1);
imshow((aver));
title('平均脸');

for k = 1:M
   A(:, k) = A(:, k) - m; %减去均值
end
[V, a, explained] = pcacov(A' * A);%V 特征向量，a 特征值，特征值占比
%特征值大小已经被排序
U = A * V;       %新空间的基
new_dim = 0;      %特征空间的维度
per = 0;
while(per<90)    %取特征值占比超过 95%的
    new_dim = new_dim + 1;
    per = per + explained(new_dim);
end

U = U(:, 1:new_dim);%选择前 new_dim 个特征向量组成特征空间
W=U;
for k = 1:new_dim
```

```
    U(:, k) = U(:, k)/norm(U(:, k)); %模长变为 1
end
eigenface = zeros(new_dim, M);   % 人脸图像在特征空间的表达
for k = 1:M
    eigenface(:, k) = U' * A(:, k);
end

%打开选择文件对话框
[file, image] = uigetfile({'*.*', 'All Files'}, '选择您要识别的图片: ');
image = strcat(image,file);
facek = imread(image);%从指定文件中读取图像,返回 M*N 数组
subplot(1, 4, 2);
imshow(facek);
title('要识别的图片');

facek = imresize(facek, [128, 128]);
rm = reshape(facek, [d, 1]);
rm = im2double(rm);
facek = rm - m;%减去平均脸
facek = U' * facek;  % 投影到特征空间上

S=U*facek;
aver=reshape(S,[128,128]);
aver=imresize(aver,[231,195]);%原图片为 231*195
subplot(1, 4, 3);
imshow((aver));
title('重构脸');

% 在数据库中找寻与识别图片最相近的图片
distance = Inf;%初始化为无穷大
best_k = 0;
for k = 1:M
    d = norm(facek - eigenface(:, k));%取模
    if(distance > d)
        best_k = k;
        distance = d;
    end
end

image = strcat('trainingset/', s(best_k).name);
subplot(1, 4, 4);
imshow(image);
title('训练集中最接近的图片');

```

```
%特征向量还原为像素排列，生成特征脸
figure;
for i=1:4
    for j=1:8
        aver=reshape(W(:,8*(i-1)+j),[128,128]);
        aver=imresize(aver,[231,195]);%原图片为 231*195
        subplot(4, 8,8*(i-1)+j );
        imshow((aver));
    end
end
```

这里分析几个重点函数:

imread(image);

从指定的文件中读取图片，返回一个 M*N 的矩阵，若为黑白图片矩阵为二维，若是彩色图片，矩阵为三维。这是 matlab 中的函数，运算速度很快，可以直接将返回的矩阵拉长为一维列向量表示该图片。

reshape(matrix,[x,y]);

将矩阵 matrix 转换为行数为 x 列数为 y 的矩阵。本实验中可以用来将矩阵拉长为一维列向量。

[V,a,c]=pcacov(A'·A);

计算协方差矩阵的特征向量，特征值，特征值能量占比，分别返回到 V，a，c 向量中，此处该函数返回值以对特征值大小进行排序。

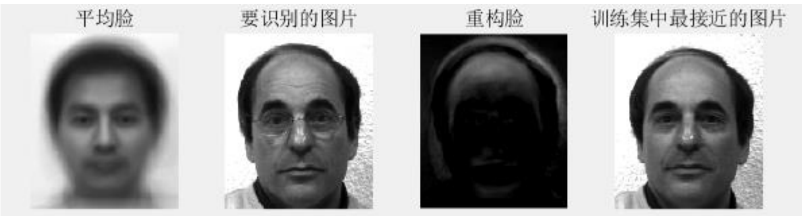
imshow(matrix);

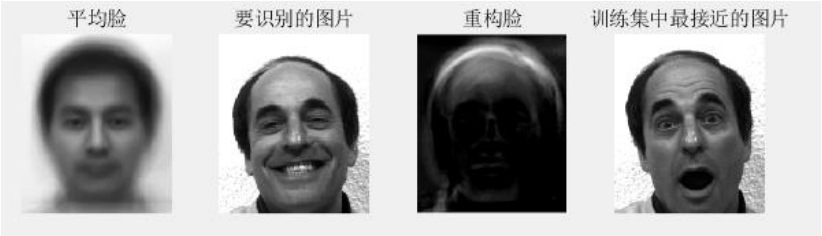
按矩阵 matrix 中的数据，输出图像。

程序运行结果:

特征脸:

testset 中用于识别图像的识别结果:





结果分析与总结：

数据库中大概有一百三十多张，识别结果可以看到代码测试还是很成功的。重构脸普遍比较模糊或灰度较大，一般来说，特征空间特征向量需要归一化，否则会影响分解和重构，但不会影响识别。实验中进行了归一化，但重构脸并不是很理想。初步估计是imshow(),图像输出函数的输入格式的问题。

可以在训练集中加入部分彩色图片，并使用 rgb2gray( );对彩色图片进行灰度化。我测试了一部分彩色图片，结果表示彩色图片的识别错误率会高很多。一方面原因应该是自己搜集的图片人脸在其中占比太小，这样对于人脸识别的正确率自然就会下降。另一方面彩色图片对光照更敏感，而 eigenface 人脸识别本身对于光照、头位、对齐、面部表情都会存在一定问题。

本征脸方法是很经典的人脸识别算法，算法相对简单，成功地实现了降维处理，减少了计算量，运行速度快，鲁棒性高。很多地计算机视觉、模式识别，尤其是人脸识别、追踪等，都有应用，或在此基础上延申。

Matlab 中有很多功能强大的函数，尤其是对于矩阵运算相关的，实验过程中应该充分利用这些函数，其基于更基层的 C 语言编写，并进行了很专业的优化。这对个人编写的程序效率会有很大提高，也实现了程序的简洁性。