

# Face Recognition

Lecture-14

# Face Recognition



# Simple Approach

- Recognize faces (mug shots) using gray levels (appearance).
- Each image is mapped to a long vector of gray levels.
- Several views of each person are collected in the database during training.
- During recognition a vector corresponding to an unknown face is compared with all vectors in the database.
- The face from database, which is closest to the unknown face is declared as a recognized face.

# Problems and Solution

- Problems :
  - Dimensionality of each face vector will be very large (250,000 for a 512X512 image!)
  - Raw gray levels are sensitive to noise, and lighting conditions.
- Solution:
  - Reduce dimensionality of face space by finding principal components (eigen vectors) to span the face space
  - Only a few most significant eigen vectors can be used to represent a face, thus reducing the dimensionality

# Eigen Vectors and Eigen Values

The eigen vector,  $x$ , of a matrix  $A$  is a special vector, with the following property

$$Ax = \lambda x \quad \text{Where } \lambda \text{ is called eigen value}$$

To find eigen values of a matrix  $A$  first find the roots of:

$$\det(A - \lambda I) = 0$$

Then solve the following linear system for each eigen value to find corresponding eigen vector

$$(A - \lambda I)x = 0$$

# Example

$$A = \begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix}$$

Eigen Values

$$\lambda_1 = 7, \lambda_2 = 3, \lambda_3 = -1$$

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 4 \\ 4 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Eigen Vectors

# Eigen Values

$$\det(A - \lambda I) = 0$$

$$\det\left(\begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix} -1-\lambda & 2 & 0 \\ 0 & 3-\lambda & 4 \\ 0 & 0 & 7-\lambda \end{bmatrix}\right) = 0$$

$$(-1-\lambda)((3-\lambda)(7-\lambda) - 0) = 0$$

$$(-1-\lambda)(3-\lambda)(7-\lambda) = 0$$

$$\lambda = -1, \quad \lambda = 3, \quad \lambda = 7$$

# Eigen Vectors

$$\lambda = -1$$

$$(A - \lambda I)x = 0$$

$$\begin{pmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 4 & 4 \\ 0 & 0 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$0 + 2x_2 + 0 = 0$$

$$0 + 4x_2 + 4x_3 = 0$$

$$0 + 0 + 8x_3 = 0$$

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 0$$

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$



# Face Recognition

Collect all gray levels in a long vector  $u$ :

$$u = (I(1,1), \dots, I(1,N), I(2,1), \dots, I(2,N), \dots, I(M,1), \dots, I(M,N))^T$$

Collect  $n$  samples (views) of each of  $p$  persons in matrix  $A$  ( $MN \times pn$ ):

$$A = [u_1^1, \dots, u_n^1, u_1^2, \dots, u_n^2, \dots, u_1^p, \dots, u_n^p]$$

Form a correlation matrix  $L$  ( $MN \times MN$ ):

$$L = AA^T$$

Compute eigen vectors,  $\phi_1, \phi_2, \phi_3, \dots, \phi_{n_1}$ , of  $L$ , which form a bases for whole face space

# Face Recognition

Each face,  $u$ , can now be represented as a linear combination of eigen vectors

$$u = \sum_{i=1}^{n_1} a_i \phi_i$$

Eigen vectors for a symmetric matrix are orthonormal:

$$\phi_i^T \cdot \phi_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

# Face Recognition

$$u_x^T \cdot \phi_i = \left( \sum_{i=1}^n a_i \phi_i \right)^T \cdot \phi_i$$

$$= (a_1 \phi_1^T + a_2 \phi_2^T + \dots + a_i \phi_i^T + \dots + a_n \phi_n^T) \phi_i$$

$$u_x^T \cdot \phi_i = (a_1 \phi_1^T \cdot \phi_i + a_2 \phi_2^T \cdot \phi_i + \dots + a_i \phi_i^T \cdot \phi_i + \dots + a_n \phi_n^T \cdot \phi_i)$$

$$u_x^T \cdot \phi_i = a_i$$

Therefore:  $a_i = u_x^T \cdot \phi_i$

# Face Recognition

$L$  is a large matrix, computing eigen vectors of a large matrix is time consuming. Therefore compute eigen vectors of a smaller matrix,  $C$ :

$$C = A^T A$$

Let  $\alpha_i$  be eigen vectors of  $C$ , then  $A\alpha_i$  are the eigen vectors of  $L$ :

$$C\alpha_i = \lambda_i\alpha_i$$

$$A^T A\alpha_i = \lambda_i\alpha_i$$

$$AA^T(A\alpha_i) = \lambda_i(A\alpha_i)$$

$$L(A\alpha_i) = \lambda_i(A\alpha_i)$$

# Training

- Create  $A$  matrix from training images
- Compute  $C$  matrix from  $A$ .
- Compute eigenvectors of  $C$ .
- Compute eigenvectors of  $L$  from eigenvectors of  $C$ .
- Select few most significant eigenvectors of  $L$  for face recognition.
- Compute coefficient vectors corresponding to each training image.
- For each person, coefficients will form a cluster, compute the mean of cluster.

# Recognition

- Create a vector  $u$  for the image to be recognized.
- Compute coefficient vector for this  $u$ .
- Decide which person this image belongs to, based on the distance from the cluster mean for each person.

```
load faces.mat
C=A'*A;
[vectorC,valueC]=eig(C);
ss=diag(valueC);
[ss,iii]=sort(-ss);
vectorC=vectorC(:,iii);
vectorL=A*vectorC(:,1:5);
Coeff=A'*vectorL;
for i=1:30
    model(i, :)=mean(coeff((5*(i-1)+1):5*I,:));
end
while (1)
    imagename=input('Enter the filename of the image to
Recognize(0 stop):');
    if (imagename <1)
        break;
    end;
    imageco=A(:,imagename)'*vectorL;
    disp ('');
    disp ('The coefficients for this image are:');
```

```
mess1=sprintf('%.2f %.2f %.2f %.2f %.2f',
imageco(1),imageco(2),imageco(3),imageco(4),
imageco(5));
disp(mess1);
top=1;
for I=2:30
    if (norm(model(i,:)-imageco,1)<norm(model
(top, : )-imageco,1))
        top=i
    end
end
mess1=sprintf('The image input was a image of person
number %d',top);
disp(mess1);
end
```



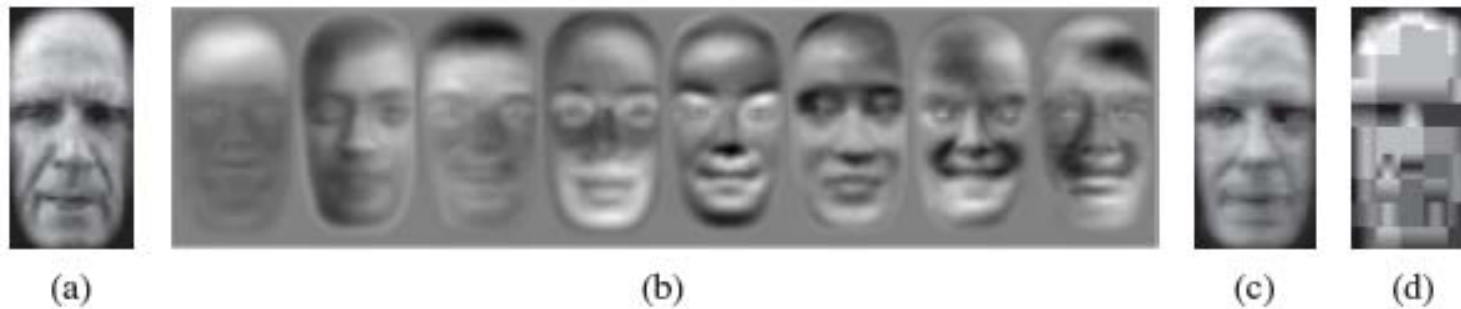
# 14.2 Face Recognition

Szeliski's book

Kirby and Sirovich (1990)

Face image can be compressed:

$$\tilde{x} = m + \sum_{i=0}^{M-1} a_i u_i$$



(c) PCA  
Reconstruction  
(85 bytes)

(d) JPEG  
Reconstruction  
(530 bytes)

Scatter or Co-variance matrix:  $C = \frac{1}{N} \sum (x_j - m)(x_j - m)^T$

Eigen Decomposition:  $C = UAU^T \frac{1}{N} \sum_{j=0}^{N-1} \lambda_i u_i u_i^T$

Any arbitrary vector  $x$  can be represented:  $a_i = (x - m) \cdot u_i$

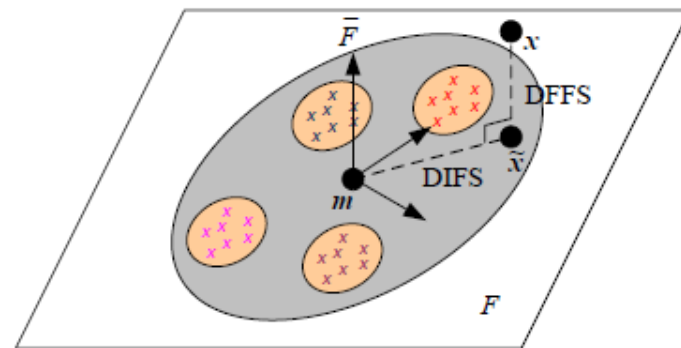
$$\tilde{x} = m + \sum_{i=0}^{M-1} a_i u_i$$

The distance of a projected face DIFS (Distance in face space)

$$DIFS = \|\tilde{x} - m\| = \sqrt{\sum_{i=0}^{M-1} a_i^2}$$

The distance between two faces

$$DIFS = \|\tilde{x} - \tilde{y}\| = \sqrt{\sum_{i=0}^{M-1} (a_i - b_i)^2}$$



We are not utilizing the eigen value information,  
compute Mahalonobis distance

$$DIFS' = \|\tilde{\mathbf{x}} - \mathbf{m}\|_{C^{-1}} = \sqrt{\sum_{i=0}^{M-1} a_i^2 / \lambda_i^2}$$

Pre-scale the eigen vectors by eigenvalues:

$$\hat{U} = UA^{-1/2}$$

Euclidean  $\mathbf{d}(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}.$

Mahalonobis  $d(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = \sqrt{(\vec{\mathbf{x}} - \vec{\mathbf{y}})^T S^{-1} (\vec{\mathbf{x}} - \vec{\mathbf{y}})}.$

# Problems in Face Recognition

- Within class variation
- Between class variation

# Images Taken under Different Illuminations



Note the wide range of illumination variation, which can be more dramatic than inter-personal variations

# LDA (Linear Discriminative Analysis) Fisher Faces

Slides credit:

*Introduction to Pattern Analysis*  
*Ricardo Gutierrez-Osuna*  
*Texas A&M University*

[http://courses.cs.tamu.edu/rgutier/cs790\\_w02/16.pdf](http://courses.cs.tamu.edu/rgutier/cs790_w02/16.pdf)

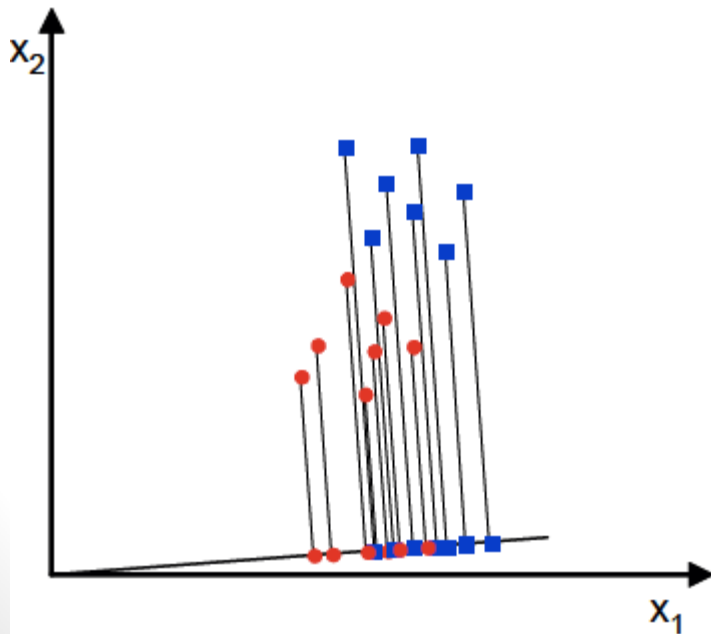
# LDA

D-dimensional samples  $\{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$

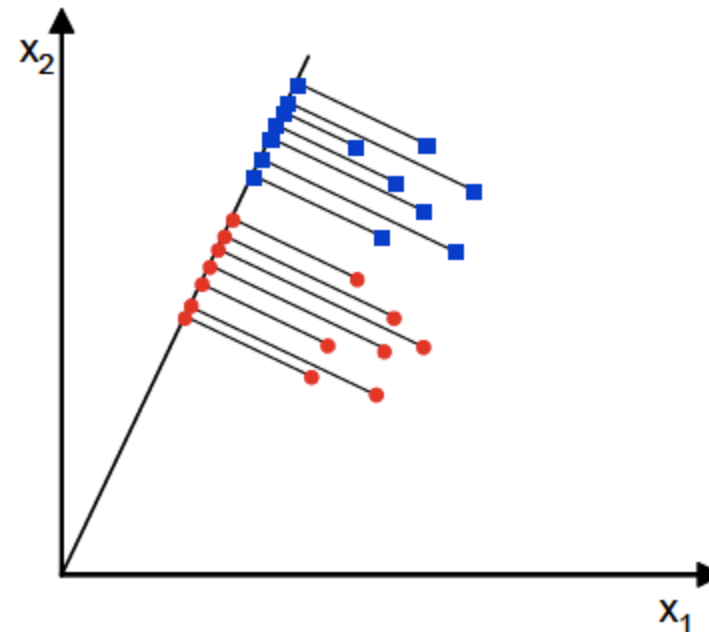
$N_1$  : Class - 1

$N_2$  : Class - 2

Project  $X$  onto a line



$$y = w^T x$$





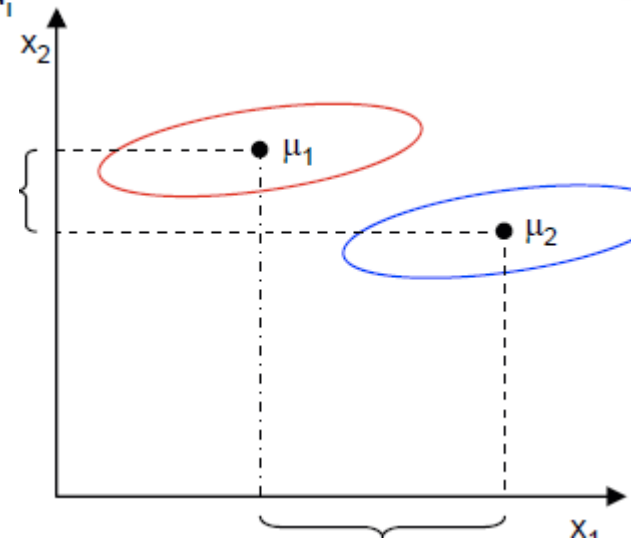
Find a measure of separation?

Distance between projected means

$$\mu_i = \frac{1}{N_i} \sum_{x \in \omega_i} x$$

$$\tilde{\mu}_i = \frac{1}{N_i} \sum_{y \in \omega_i} y = \frac{1}{N_i} \sum_{x \in \omega_i} w^T x = w^T \mu_i$$

$$J(w) = |\tilde{\mu}_1 - \tilde{\mu}_2| = |w^T (\mu_1 - \mu_2)|$$



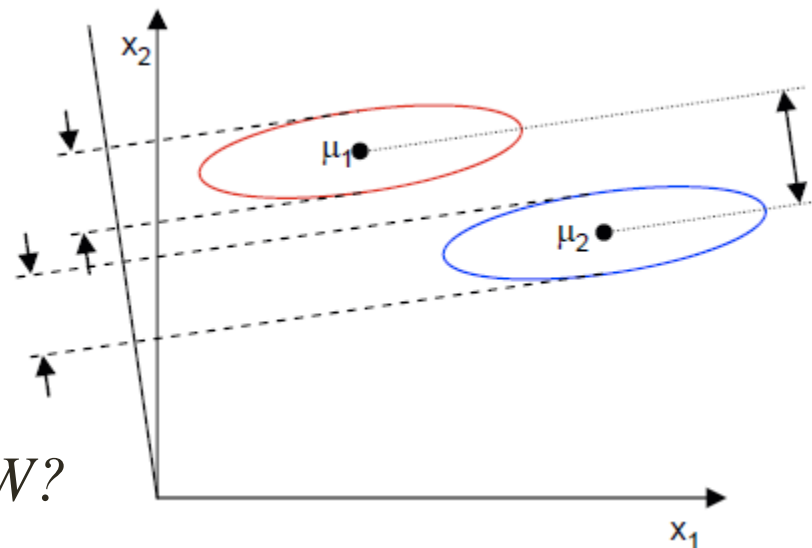
It is not good measure, since it does not consider standard deviation within a class

Maximize a function that represents the difference between the means normalized by a measure of the within-class scatter

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2$$

$(\tilde{s}_1^2 + \tilde{s}_2^2)$  is called the within-class scatter

$$J(W) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$



How to find optimum  $W$ ?

# Find Scatter Matrices

$$S_i = \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T \quad S_1 + S_2 = S_W$$

where  $S_W$  is called the within-class scatter matrix

Scatter Matrices for the projection

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2 = \sum_{x \in \omega_i} (w^T x - w^T \mu_i)^2 = \sum_{x \in \omega_i} w^T (x - \mu_i)(x - \mu_i)^T w = w^T S_i w$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = w^T S_W w$$

$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T \mu_1 - w^T \mu_2)^2 = w^T \underbrace{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}_{S_B} w = w^T S_B w$$

$S_B$  is called the between-class scatter

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

$$w^* = \operatorname{argmax}_w \left\{ \frac{w^T S_B w}{w^T S_W w} \right\}$$

# Find Optimum $W$

$$\frac{d}{dW}[J(W)] = \frac{d}{dW} \left[ \frac{W^T S_B W}{W^T S_W W} \right] = 0$$

$$\frac{d}{dW}[J(W)] = \frac{(W^T S_W W)2S_B W - (W^T S_B W)2S_W W}{(W^T S_W W)^2} = 0$$

$$\frac{d}{dW}[J(W)] = \frac{(W^T S_W W)2S_B W - (W^T S_B W)2S_W W}{(W^T S_W W)} = 0$$

$$\frac{d}{dW}[J(W)] = \frac{(W^T S_W W)}{(W^T S_W W)} S_B W - \frac{(W^T S_B W)}{(W^T S_W W)} S_W W = 0$$

$$\frac{d}{dW}[J(W)] = S_B W - JS_W W = 0$$

Eigen Value problem

$$\frac{d}{dW}[J(W)] = S_W^{-1} S_B W - JW = 0$$

$$S_W^{-1} S_B W = JW$$

# Example

- $X_1=(x_1,x_2)=\{(4,1),(2,4),(2,3),(3,6),(4,4)\}$
- $X_2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$

$$S_1 = \begin{bmatrix} 0.80 & -0.40 \\ -0.40 & 2.60 \end{bmatrix}; S_2 = \begin{bmatrix} 1.84 & -0.04 \\ -0.04 & 2.64 \end{bmatrix}$$

$$\mu_1 = [3.00 \quad 3.60]; \quad \mu_2 = [8.40 \quad 7.60]$$

$$S_B = \begin{bmatrix} 29.16 & 21.60 \\ 21.60 & 16.00 \end{bmatrix}; S_W = \begin{bmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{bmatrix}$$

$$S_W^{-1}S_B v = \lambda v \Rightarrow |S_W^{-1}S_B - \lambda I| = 0 \Rightarrow \begin{vmatrix} 11.89 - \lambda & 8.81 \\ 5.08 & 3.76 - \lambda \end{vmatrix} = 0 \Rightarrow \lambda = 15.65$$

$$\begin{bmatrix} 11.89 & 8.81 \\ 5.08 & 3.76 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 15.65 \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Rightarrow \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0.91 \\ 0.39 \end{bmatrix}$$

