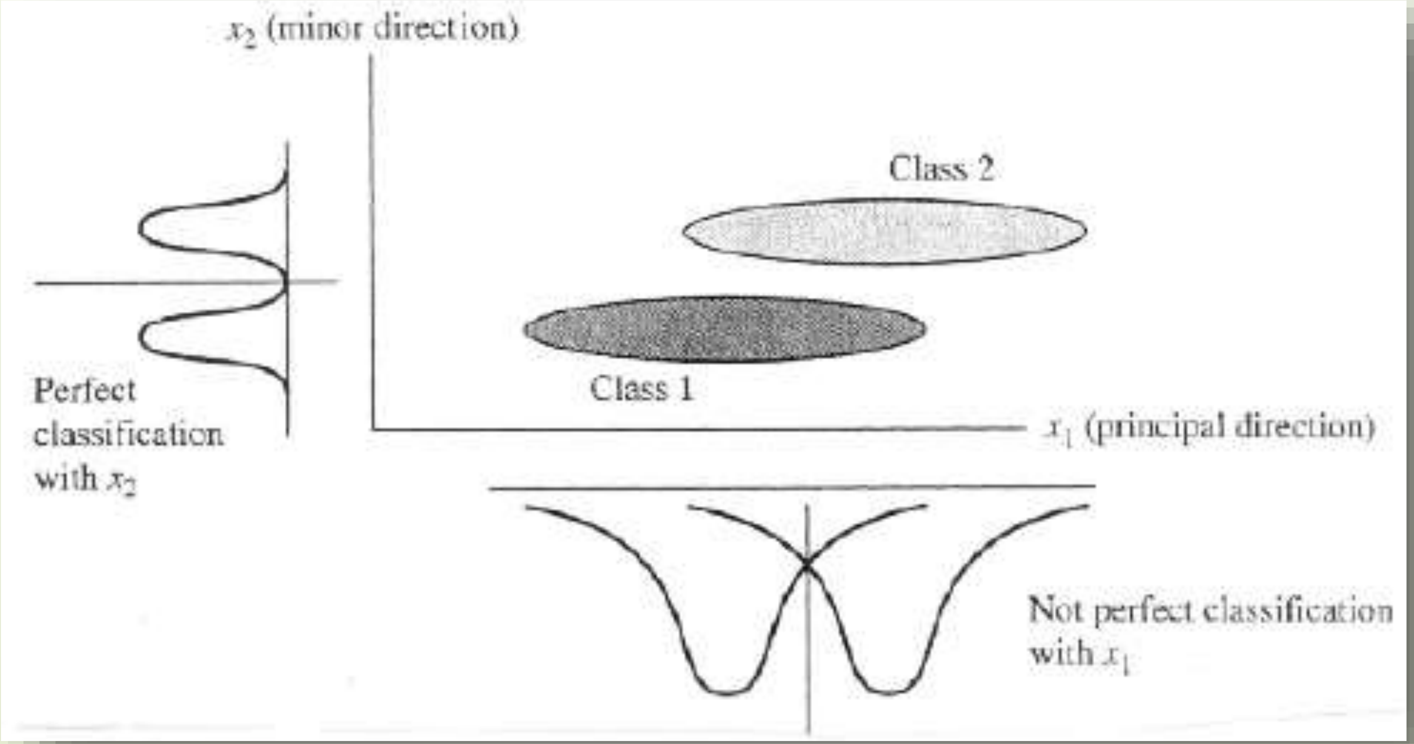
Linear Discriminant Analysis

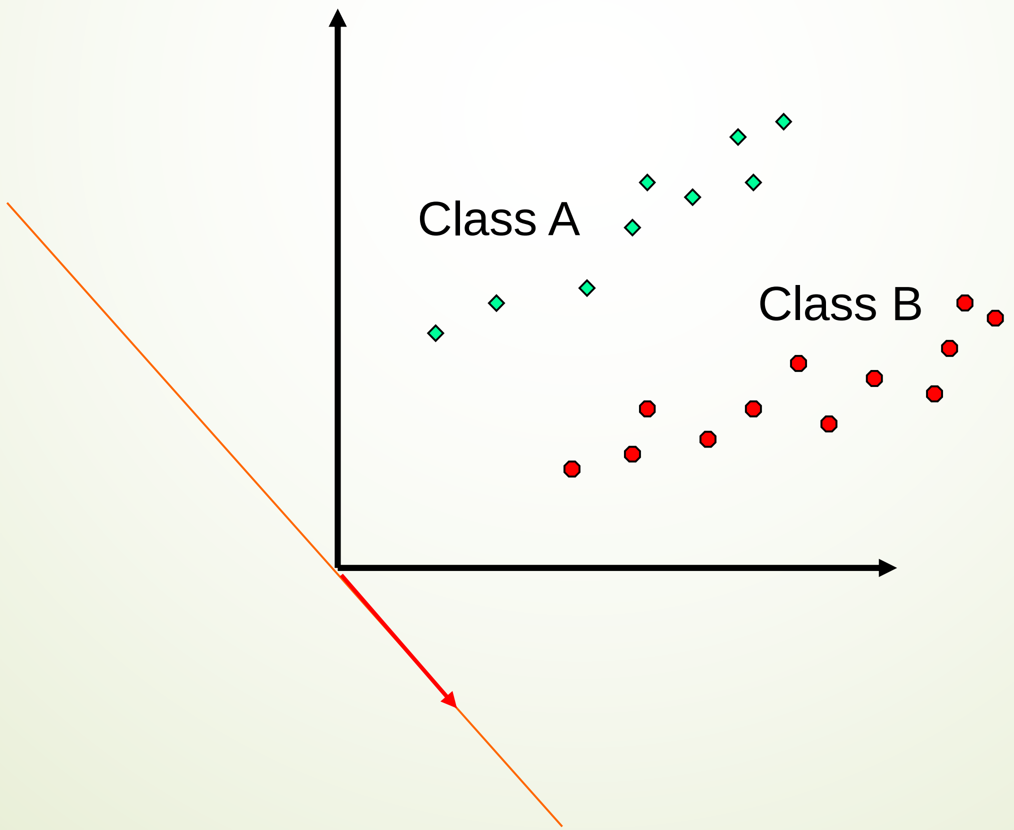
- LDA comes with concept of class.
- PCA don't use concept of classes.
- LDA is an enhancement to PCA
- Class in face recognition means a specific person, and elements of class are his/her face images.
- Suppose there two class, then class 1 will have images of 1st person and class 2 will have images of 2nd person.



- Multiple classes and PCA
  - Suppose there are *C* classes in the training data.
  - PCA is based on the sample covariance which characterizes the scatter of the entire data set, *irrespective of class-membership*.
  - The projection axes chosen by PCA might not provide good discrimination power.

- What is the goal of LDA?
  - Perform dimensionality reduction while saving as much of the class discriminatory information as possible.
  - Search to find directions along which the classes are best separated.
  - Takes into consideration the scatter *within-classes* but also the scatter *between-classes*.

- LDA maximizes the between-class scatter
- LDA minimizes the within-class scatter



Algorithm

- Assumptions
  - Square images with Width=Height=N
  - M is the number of images in the database
  - P is the number of persons in the database

- The database

$$= \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_{N^2} \end{pmatrix}$$

$$= \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_{N^2} \end{pmatrix}$$

Fisherfaces, the algorithm

- We compute the average of all faces

$$\vec{m} = \frac{1}{M}\begin{pmatrix} a_1 & + b_1 & + \cdots & + h_1 \\ a_2 & + b_2 & + \cdots & + h_2 \\ \vdots & \vdots & & \vdots \\ a_{N^2} & + b_{N^2} & + \cdots & + h_{N^2} \end{pmatrix}, \quad \textit{where } M = 8$$

Fisherfaces, the algorithm

- Compute the average face of each person

$$\vec{x} = \frac{1}{2}\begin{pmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \vdots \\ a_{N^2} + b_{N^2} \end{pmatrix}, \quad \vec{y} = \frac{1}{2}\begin{pmatrix} c_1 + d_1 \\ c_2 + d_2 \\ \vdots \\ c_{N^2} + d_{N^2} \end{pmatrix},$$

$$\vec{z} = \frac{1}{2}\begin{pmatrix} e_1 + f_1 \\ e_2 + f_2 \\ \vdots \\ e_{N^2} + f_{N^2} \end{pmatrix}, \quad \vec{w} = \frac{1}{2}\begin{pmatrix} g_1 + h_1 \\ g_2 + h_2 \\ \vdots \\ g_{N^2} + h_{N^2} \end{pmatrix}$$

Fisherfaces, the algorithm

- And subtract them from the training faces

$$\vec{a}_m = \begin{pmatrix} a_1 - x_1 \\ a_2 - x_2 \\ \vdots \\ a_{N^2} - x_{N^2} \end{pmatrix}, \quad \vec{b}_m = \begin{pmatrix} b_1 - x_1 \\ b_2 - x_2 \\ \vdots \\ b_{N^2} - x_{N^2} \end{pmatrix}, \quad \vec{c}_m = \begin{pmatrix} c_1 - y_1 \\ c_2 - y_2 \\ \vdots \\ c_{N^2} - y_{N^2} \end{pmatrix}, \quad \vec{d}_m = \begin{pmatrix} d_1 - y_1 \\ d_2 - y_2 \\ \vdots \\ d_{N^2} - y_{N^2} \end{pmatrix},$$

$$\vec{e}_m = \begin{pmatrix} e_1 - z_1 \\ e_2 - z_2 \\ \vdots \\ e_{N^2} - z_{N^2} \end{pmatrix}, \quad \vec{f}_m = \begin{pmatrix} f_1 - z_1 \\ f_2 - z_2 \\ \vdots \\ f_{N^2} - z_{N^2} \end{pmatrix}, \quad \vec{g}_m = \begin{pmatrix} g_1 - w_1 \\ g_2 - w_2 \\ \vdots \\ g_{N^2} - w_{N^2} \end{pmatrix}, \quad \vec{h}_m = \begin{pmatrix} h_1 - w_1 \\ h_2 - w_2 \\ \vdots \\ h_{N^2} - w_{N^2} \end{pmatrix}$$

Fisherfaces, the algorithm

- We build scatter matrices S_1, S_2, S_3, S_4

$$S_1 = \left(\vec{a}_m \vec{a}_m^{\mathrm{T}} + \vec{b}_m \vec{b}_m^{\mathrm{T}}\right),\ S_2 = \left(\vec{c}_m \vec{c}_m^{\mathrm{T}} + \vec{d}_m \vec{d}_m^{\mathrm{T}}\right),$$
$$S_3 = \left(\vec{e}_m \vec{e}_m^{\mathrm{T}} + \vec{f}_m \vec{f}_m^{\mathrm{T}}\right),\ S_4 = \left(\vec{g}_m \vec{g}_m^{\mathrm{T}} + \vec{h}_m \vec{h}_m^{\mathrm{T}}\right)$$

- And the within-class scatter matrix S_W

$$S_W = S_1 + S_2 + S_3 + S_4$$

Fisherfaces, the algorithm

- The between-class scatter matrix

$$S_B = 2(\vec{x} - \vec{m})(\vec{x} - \vec{m})^{\mathrm{T}} + 2(\vec{y} - \vec{m})(\vec{y} - \vec{m})^{\mathrm{T}} + 2(\vec{z} - \vec{m})(\vec{z} - \vec{m})^{\mathrm{T}} + 2(\vec{w} - \vec{m})(\vec{w} - \vec{m})^{\mathrm{T}}$$

- We are searching the matrix W maximizing

$$J(W) = \frac{\left|W^{\mathrm{T}} S_B W\right|}{\left|W^{\mathrm{T}} S_W W\right|}$$

Fisherfaces, the algorithm

- Columns of W are eigenvectors of $S_W^{-1} S_B$
  - We have to compute the inverse of S_W
  - We have to multiply the matrices
  - We have to compute the eigenvectors

Recognition

- Project faces onto the LDA-space
- To classify the face
  - Project it onto the LDA-space
  - Run a nearest-neighbor classifier
  - Nearest is our answer