

---

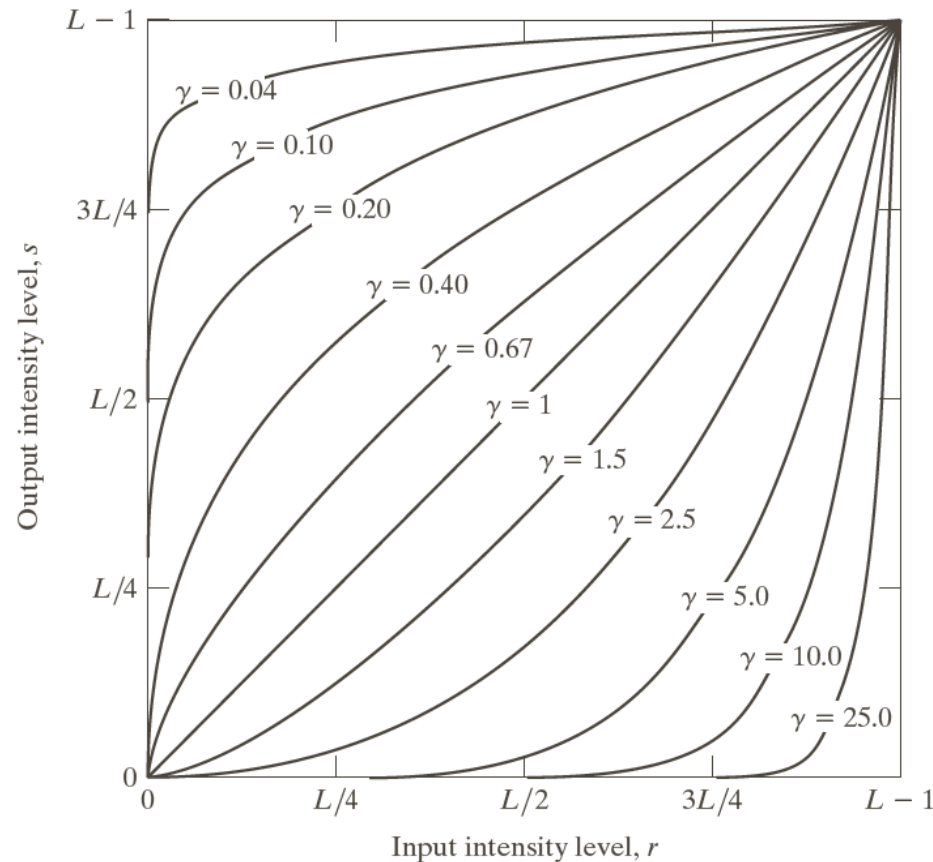
**COMP 435 Spring 2010**

**Lecture 04**

# **Transforms & Subspace Models**

# Re-Cap: Pixel based processing

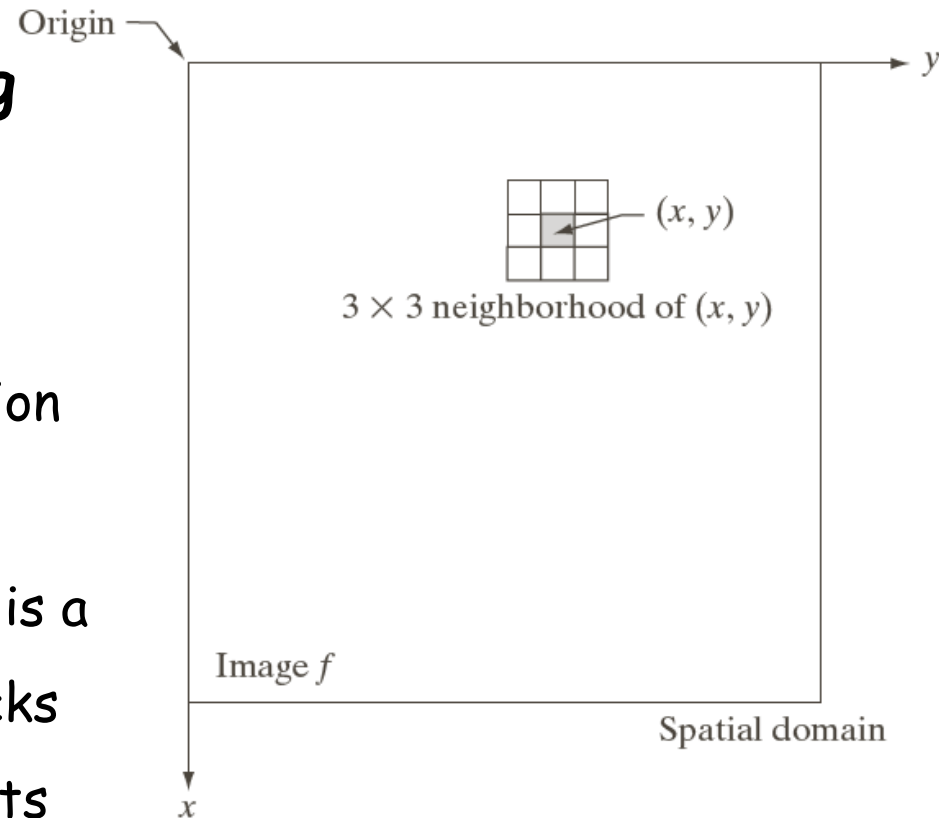
- Pixel based image processing
  - Transfer functions mapping pixel intensity values:  $y=T(x)$
  - Color conversion to intensity:  
 $G=0.3R+0.6G+0.1B$
  - Image Content Adaptive processing: Histogram Equalization
  - Matlab implementations:
    - Built in functions: `imread`, `imshow`, `imhist`, `histeq`, `color2rgb`



$$Y=T(x)$$

# Re-Cap: Block based processing

- **Filter based image processing**
  - A filter mask of size  $m \times m$  is identified
  - Center at input image pixel location  $(x, y)$
  - Output image pixel value at  $(x, y)$  is a weighted sum of input image blocks centered around  $(x, y)$  with weights given by the mask
  - Matlab implementations:
    - » `im2= Imfilter(im, mask)`



**New image pixel value at  $(x, y)$ :  
Weighted sum of  $3 \times 3$  original  
image pixels**

# Re-Cap: Filter based Image processing

- **Smoothing/De-Noising**
  - Average, Gaussian
  - Median Filter (non-linear)
    - » Matlab: `medfilter()`
- **Edge detection**
  - Sobel/Prewitt filters
  - 1<sup>st</sup> order approach: Derivative of Gaussian filter:
    - » smoothing + differentiation, looking for large values
  - 2<sup>nd</sup> order approach: Laplacian of Gaussian filter:
    - » Smoothing + laplacian: look for zero-crossings
- **Matlab implementations for different filter masks:**
  - `fspecial`, `medfilter`

# Transform & Subspace Modeling in Image Processing

- **Lecture 4 Outline**

- Background knowledge
  - » Statistical concepts: mean, variance, co-variance
  - » Linear Algebra: Eigenvectors, Eigenvalues
- Principal Component Analysis - PCA
  - » Mathematical Formulation
  - » Numerical Solution & Matlab Implementation
- Discrete Cosine Transform (DCT )
  - » Formulation
  - » Numerical Solution & Matlab Implementation
- Linear Discriminant Analysis (LDA), aka Fisher Discriminant Analysis
  - » Formulation
  - » Numerical Solution & Matlab Implementation

---

# Statistics & Linear Algebra Background

# Statistics

- Given a set of  $n$ -point data  $\{X_k\}$  in  $\mathbb{R}^d$ :

- The mean is  $E\{x\}$

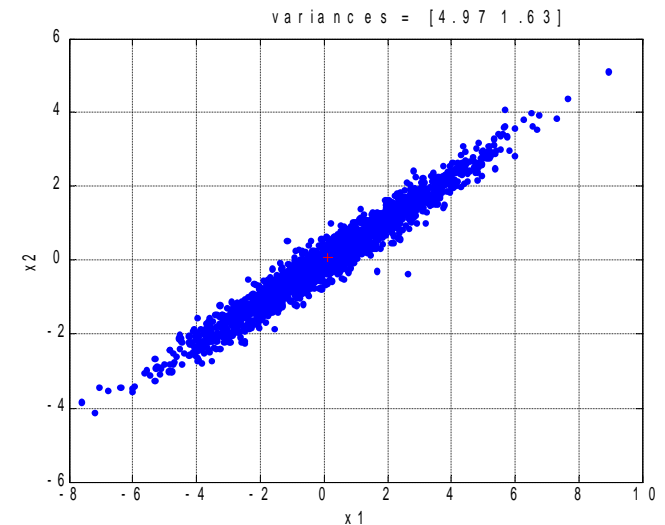
$$\bar{X} = \frac{\sum_{i=1}^n X_i}{n}$$

- The variance is  $\text{Var}\{x\}$

$$\text{var}(X) = \frac{\sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})}{(n - 1)}$$

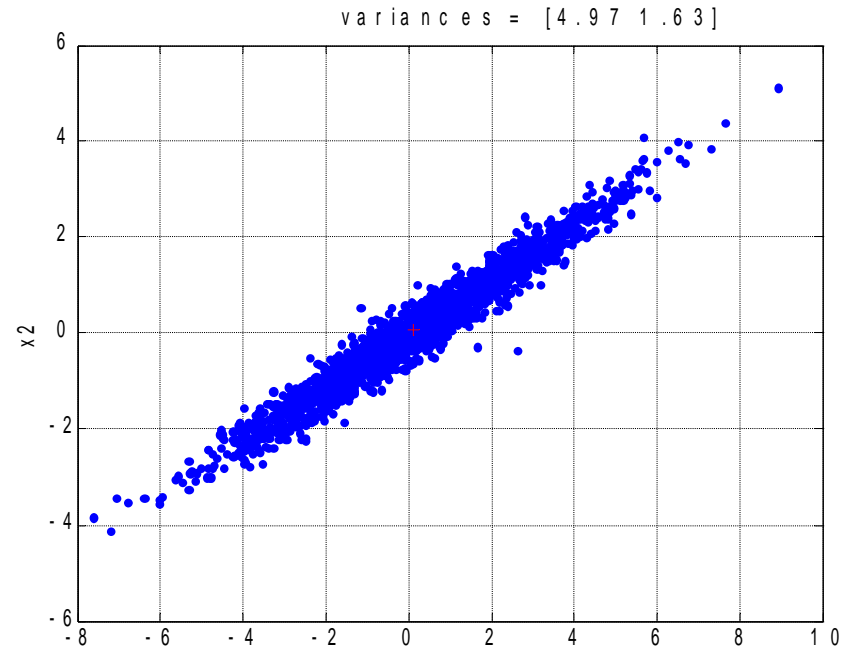
- The co-variance between two data set is

$$\text{cov}(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{(n - 1)}$$



# Statistics

- Matlab implementation
  - The mean is  $R^d$ 
    - `mean(x)` – assuming row vectors
  - The variance is  $R^d$ 
    - `var(x)` – assuming row vectors
  - The co-variance of vector data: `dxd`:
    - `cov(x)`, with element  $\text{cov}(j,k) = \text{cov}(x_j, x_k)$
  - MATLAB:

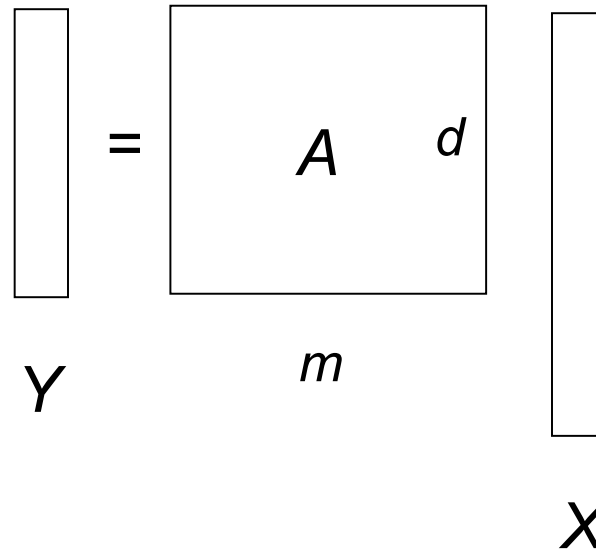


```
% compute mean:
mx = mean(x);
plot(mx(1), mx(2), '+r');
% compute variance:
v1 = var(x(:,1)); v2=var(x(:,2));
str=sprintf('\n variances = [%1.2f %1.2f]', v1, v2); title(str);
% covariance of x: n x 2:
% cv=cov(x) : 2x2 matrix:
% cv(1,1) = var(x1); cv(2,2)=var(x2);
% cv(1,2) = cv(2,1)
cov(x)
```

# Algebra

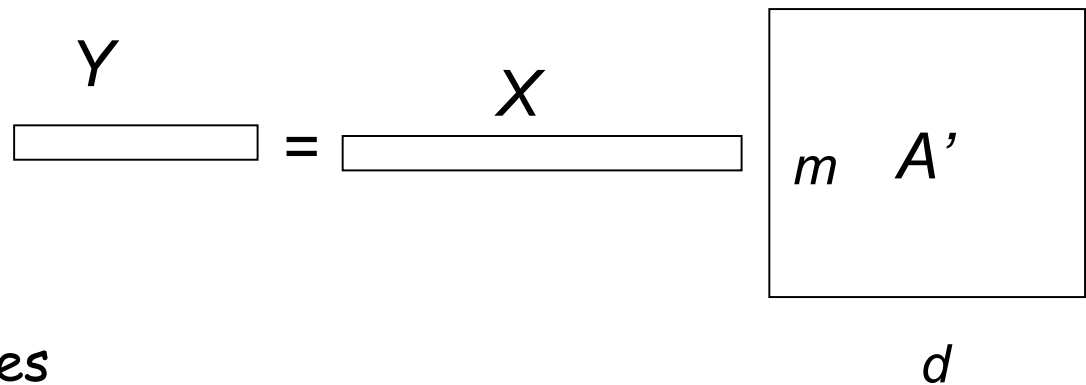
- **Transform A:  $Y=AX$**

- $X$ , col vector in  $\mathbb{R}^d$
- $Y$ , col vector in  $\mathbb{R}^m$
- $A$ , col  $d \times m$  matrix, where row entries are the basis of transform  $A$



- **In matlab**

- $X, Y$ , row vecotrs
- $Y=X*A'$
- $A'$ :  $m \times d$ , where col entries are the basis functions



- Example:  $y=Ax$ ,  $y^T=x^T A^T$ .

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad [y_1, y_2, y_3] = [x_1, x_2, x_3] \times \begin{bmatrix} a_{1,1} & a_{2,1} & a_{3,1} \\ a_{1,2} & a_{2,2} & a_{3,2} \\ a_{1,3} & a_{2,3} & a_{3,3} \end{bmatrix}$$

- Transform A basis:
  - $k^{\text{st}}$  basis:  $A(k,:) = [a_{k,1}, a_{k,2}, a_{k,3}]$ , row vector

# Eigenvectors

- For transform  $Y=AX$ , if exists:

$$y = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \times \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \lambda \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}$$

then

$e=[e^1, e^2, e^3]^T$  is an eigenvector,

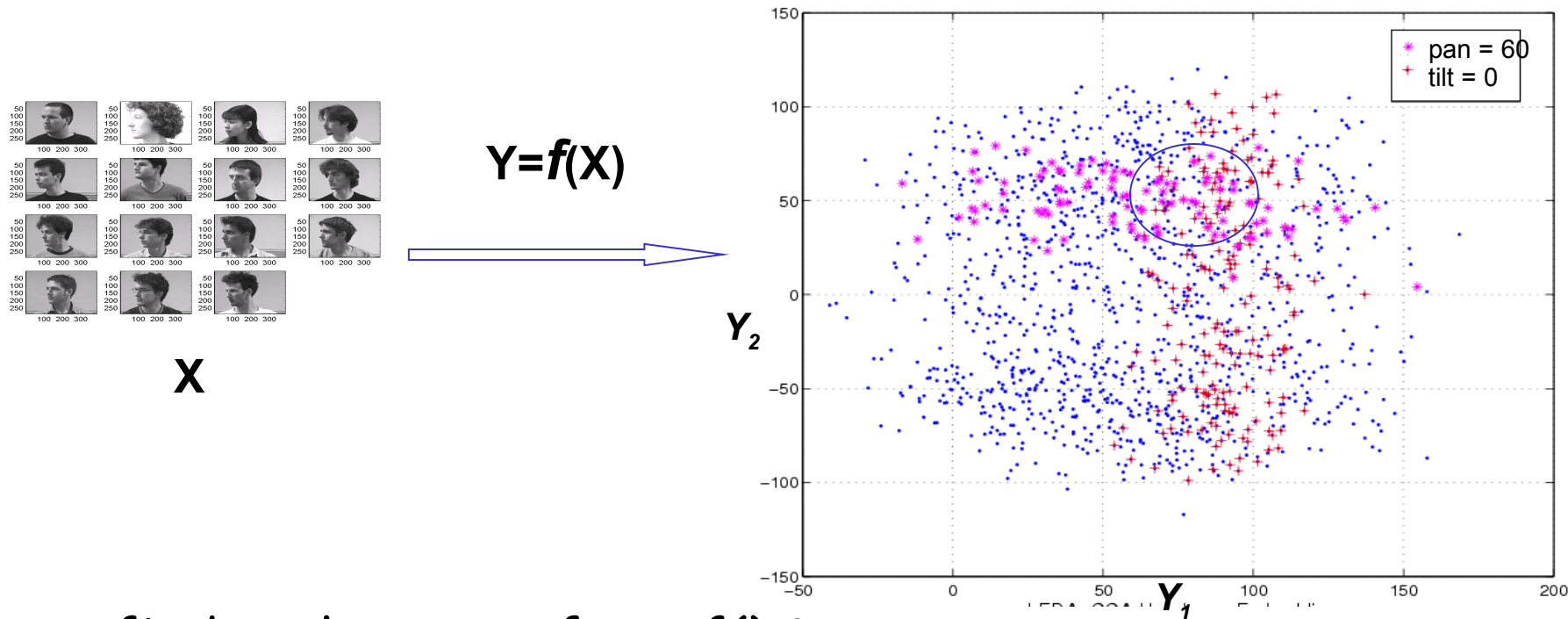
$\lambda$  is the eigenvalue associated with this eigenvector

meaning, for transform  $A$ ,  $e$  is just scaled by the eigenvalue.

# Why Transform ?

- **Appearance modeling**

- Model  $w \times h$  pixel luminance field as a point  $x$  in  $R^{w \times h}$
- Learn a mapping  $f$ , such that samples from different classes are well separated and well-behaving for classification, after the mapping



How to find such a transform  $f()$  ?

---

# Principal Component Analysis

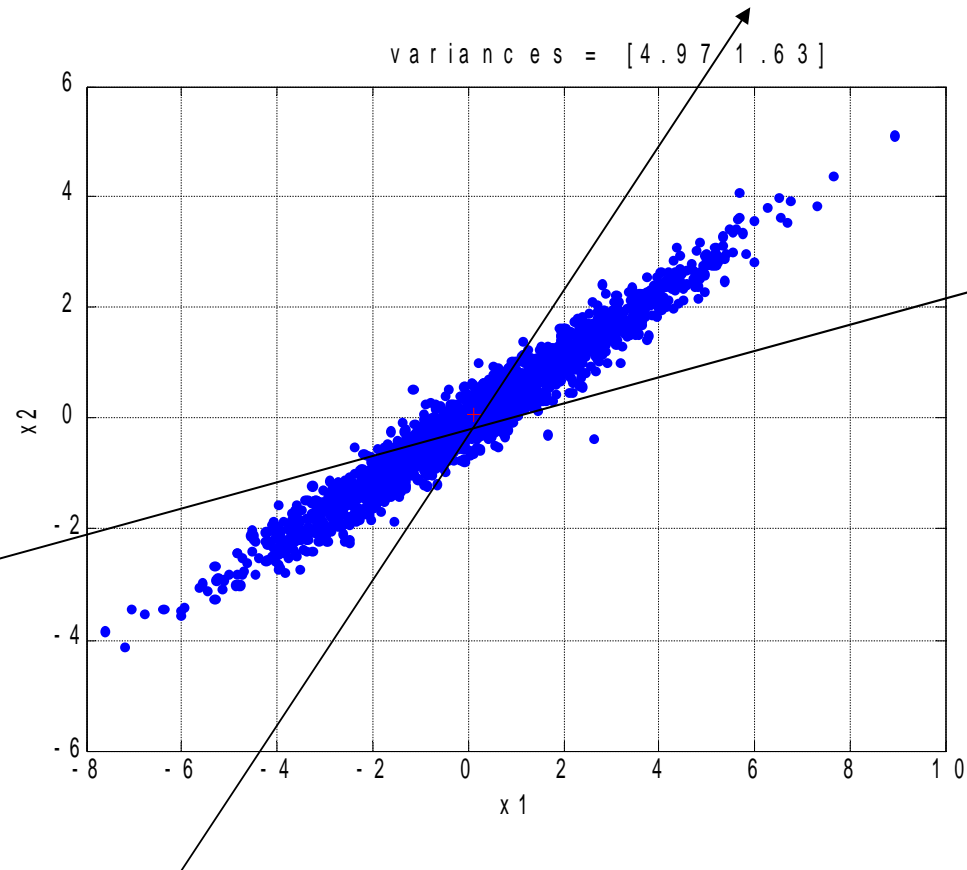
# Outline

---

- PCA is trying to captures the “principle” variations in the data
- It is computed by finding the Eigen vectors of the covariance matrix of the data
- Geometrically, it finds the largest variations directions of the underlying data
- Can be applied in data compression, pattern recognition

# Motivation of PCA

- How to approximate the data ?
  - By a point
  - By a line
  - By a plane ?



# Best Point Approximation

- Find a point  $x_0$  in  $\mathbb{R}^d$ , such that:

$$x_0^* = \arg \min_{x_0} J_0(x_0) \quad s.t.$$
$$J_0(x_0) = \sum_{k=1}^n \|x_k - x_0\|^2$$

*Notation !*

consider the mean is given as:

$$m = \frac{1}{n} \sum_{k=1}^n x_k$$

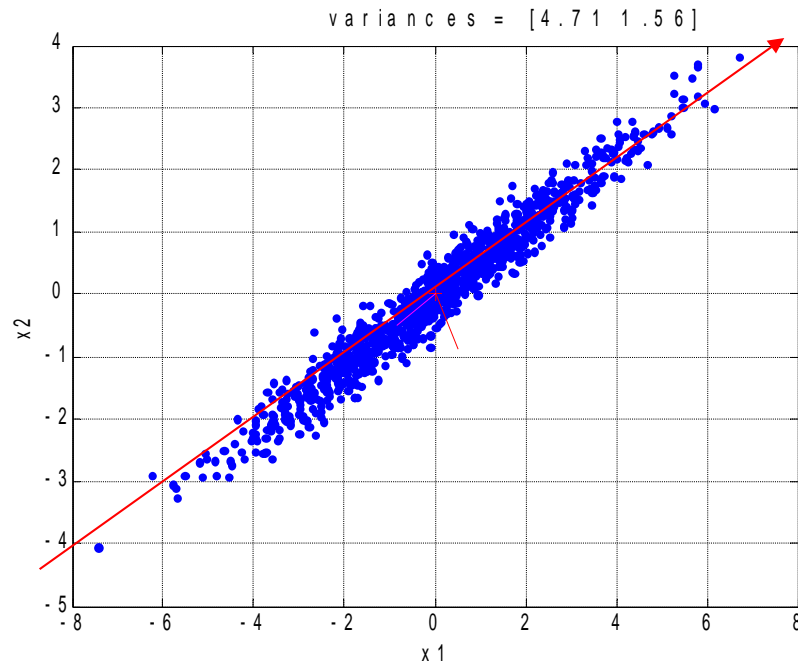
# Best Point Approximation

$$\begin{aligned} J_0(x_0) &= \sum_{k=1}^n \|x_k - x_0\|^2 \\ &= \sum_{k=1}^n \|(x_k - m) - (x_0 - m)\|^2 \\ &= \sum_{k=1}^n \|(x_0 - m)\|^2 - 2 \sum_{k=1}^n (x_0 - m)^T (x_k - m) + \sum_{k=1}^n \|(x_k - m)\|^2 \\ &= \sum_{k=1}^n \|(x_0 - m)\|^2 - 2(x_0 - m)^T \overbrace{\sum_{k=1}^n (x_k - m)}^{=0} + \sum_{k=1}^n \|(x_k - m)\|^2 \\ &= \sum_{k=1}^n \|(x_0 - m)\|^2 + \overbrace{\sum_{k=1}^n \|(x_k - m)\|^2}^{\text{independent}} \end{aligned}$$

- Since the 2<sup>nd</sup> term is independent of  $x_0$ , the minimal value of  $J_0(x_0)$  is found when  $x_0=m$ .
  - i.e., the best point approximate of a set of data points is their mean.

# The best line approximation of data

- How to represent data set by a line ?
  - Find a line going thru the data mean and along the max variation direction of the data



- Assuming zero mean, line is represented as  $y=w^T x$ , where  $w$  is the basis,  $w^T w = \|w\| = 1$ .

# Mathematically...

- Objective function

$$\max_w J(\mathbf{w}) = E\{y^2\} = E\{(\mathbf{w}^T \mathbf{x})^2\}, \quad s.t. \mathbf{w}^T \mathbf{w} = 1$$

- since:

$$J(\mathbf{w}) = E\{(y)^2\} = E\{(\mathbf{w}^T \mathbf{x})^T (\mathbf{w}^T \mathbf{x})\} = E\{(\mathbf{w}^T \mathbf{x}^T \mathbf{x} \mathbf{w})\} = (\mathbf{w}^T \mathbf{S} \mathbf{w})$$

- Equivalent to:

$$\max_w (\mathbf{w}^T \mathbf{S} \mathbf{w}), \quad s.t. \mathbf{w}^T \mathbf{w} = 1$$

# Lagrangian of the problem

- From optimization theory (AMA 450), the lagrangian of the problem is:

$$L(\lambda) = w^T S w + \lambda (1 - w^T w)$$

- KKT condition gives us:

$$\frac{\partial L}{\partial w} = 0, \Rightarrow Sw = \lambda w$$

*Eigen problem*



# Line Solution

- From eigen theory, the max variation is achieved by eigen vectors associated with the largest eigen value (variance after projection)
- So, one line approximation of the data set is given by the eigen vector of the covariance matrix with the largest eigen value !

# Verify this by Matlab

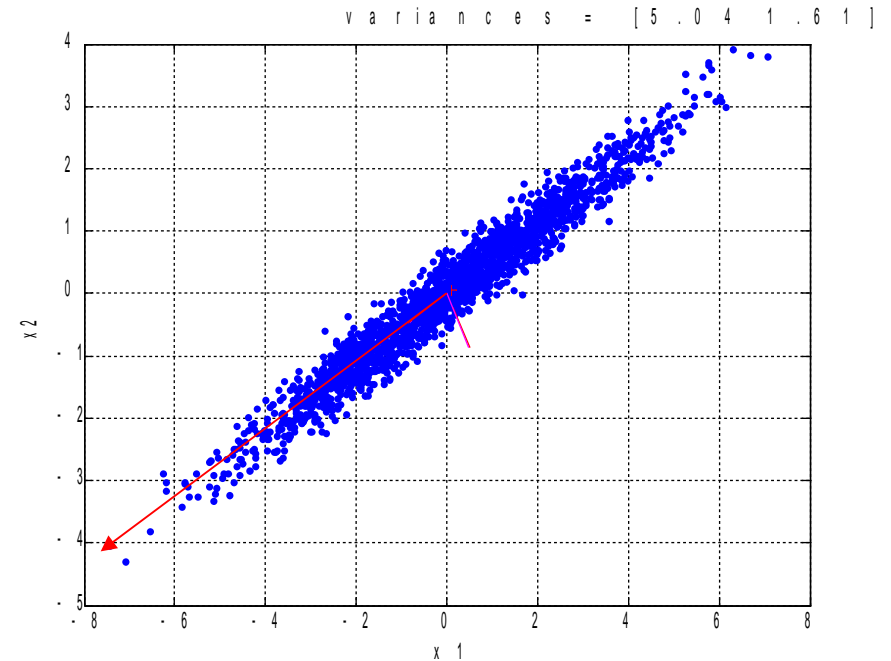
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pca of x
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[A1, L1]=eig(cov(x));
a1 = A1(1,:);
a2 = A1(2,:);
fprintf('\n eigen values:'); fprintf('%1.2f ', diag(L1));
px = [0 a1(1)]; py=[0, a1(2)]; plot(px, py, '-m'); hold on;
px = [0 a2(1)]; py=[0, a2(2)]; plot(px, py, '-r');
```

*Covariance:*

4.8931	2.6984
2.6984	1.5638

*Eigen Vecs:*

0.4873	-0.8732
-0.8732	-0.4873



*Eigen Values:*

0.06	6.40
------	------

# Best Subspace Approximation

- Find a  $m$ -dimension subspace spanned by  $\mathbf{W}=[\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_m]$ 
  - Projection:  $\mathbf{y}=\mathbf{W}\mathbf{x}$
  - Reconstruction back in  $\mathbf{x}$  space:  $\mathbf{P}\mathbf{x}=(\mathbf{W}^T\mathbf{W})\mathbf{x}$
- Objective:
  - Find optimal subspace  $\mathbf{W}$ , such that the reconstruction error is minimized:

$$\underset{\mathbf{W}}{\operatorname{argmin}} J_{MSE}(\mathbf{W}) = E\{\|\mathbf{x} - \mathbf{P}\mathbf{x}\|^2\}, \text{ s.t., } \mathbf{W}^T\mathbf{W} = \mathbf{I}$$

# Mathematically...

- The objective function:

$$J_{MSE}(\mathbf{W}) = E\{\mathbf{x}^T \mathbf{x}\} - E\{\mathbf{x}^T \mathbf{P} \mathbf{x}\}$$

*minimizing*  $J_{MSE}(\mathbf{W}) \longrightarrow$  *maximizing*  $E\{\mathbf{x}^T \mathbf{P} \mathbf{x}\}$

- Equivalent to:

$$\max_{\mathbf{W}} E\{\mathbf{x}^T \mathbf{W} \mathbf{W}^T \mathbf{x}\} \text{ s.t. } \mathbf{W}^T \mathbf{W} = \mathbf{I}$$

The Lagrangian is

$$L(\mathbf{W}, \lambda) = E\{\mathbf{x}^T \mathbf{W} \mathbf{W}^T \mathbf{x}\} + \lambda(\mathbf{I} - \mathbf{W}^T \mathbf{W})$$

The set of KKT conditions gives:

$$\frac{\partial L(\mathbf{W}, \lambda)}{\partial \mathbf{w}_i} = 2E\{\mathbf{x} \mathbf{x}^T\} \mathbf{w}_i - 2\lambda \mathbf{w}_i, \quad \forall i$$

# Gives us the same Eigen Problem solution

Let's denote by  $\mathbf{S} = E\{\mathbf{x}\mathbf{x}^T\}$  (note:  $E\{\mathbf{x}\} = \mathbf{0}$ ).  
The KKT conditions give:

$$\mathbf{S}\mathbf{w}_i = \lambda\mathbf{w}_i, \quad \forall i$$

or in a more concise matrix form:

$$\mathbf{S}\mathbf{W} = \lambda\mathbf{W}$$

- Compute the Covariance  $S$  of the data set
- Find its eigen-vectors, sort by eigen values

# Matlab svd Implementation: $[A, s, L]=\text{princomp}(X)$

Learning the principal components from  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ :

1. calculating  $\mathbf{m} = \frac{1}{N} \sum_{k=1}^N \mathbf{x}_k$
2. centering  $\mathbf{A} = [\mathbf{x}_1 - \mathbf{m}, \dots, \mathbf{x}_N - \mathbf{m}]$
3. calculating  $\mathbf{S} = \sum_{k=1}^N (\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^T = \mathbf{A}\mathbf{A}^T$
4. eigenvalue decomposition

$$\mathbf{S} = \mathbf{U}^T \Sigma \mathbf{U}$$

5. sorting  $\lambda_i$  and  $\mathbf{e}_i$
6. finding the bases

$$\mathbf{W} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_m]$$

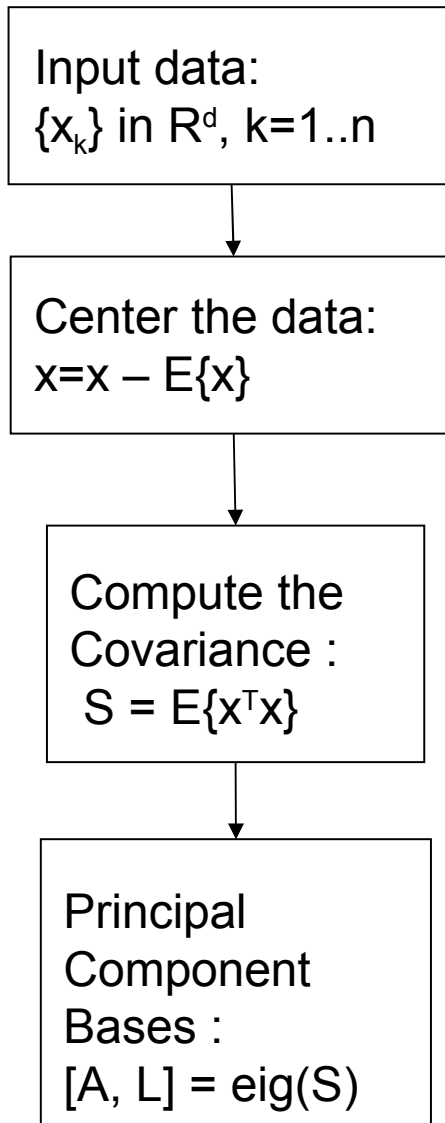
*SVD*  
*In matlab*



**Note:** The components for  $\mathbf{x}$  is

$$\mathbf{y} = \mathbf{W}^T (\mathbf{x} - \mathbf{m}), \text{ where } \mathbf{x} \in \mathbb{R}^n \text{ and } \mathbf{y} \in \mathbb{R}^m$$

# Computation of PCA



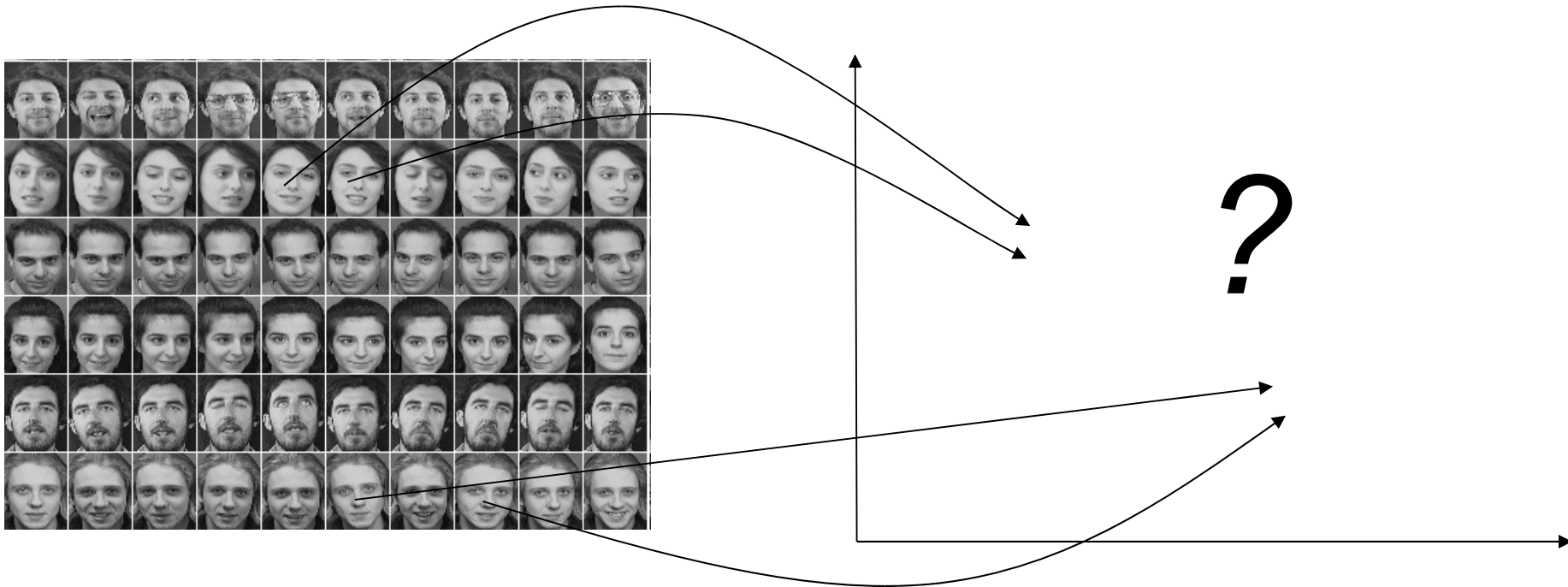
*Input data in row vectors,*  
 $x: n \times d$

```
mx = mean(x);  
x = x - repmat(mx, n, 1);
```

```
S = cov(x);  
[A, L]=eig(S);  
Indx = sort(diag(L));  
%first 2 component:  
a1=A(indx(1), : );  
a2=A(indx(2), : );
```

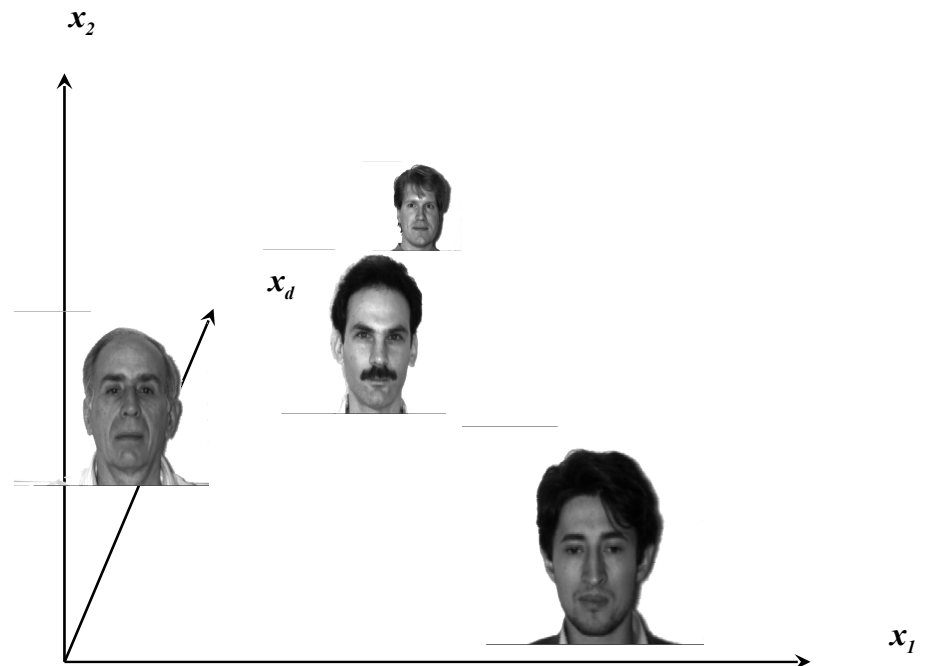
# Applications of PCA

- Face subspace representation
  - An  $w \times h$  pixel face image can be viewed as a vector of  $d=wxh$  dimensions
  - If  $w=h=20$ , then  $d=400$ , dimensionality too high
  - Un-necessary info removal



## PCA for Face Recognition

- A  $M \times N$  (e.g.,  $256 \times 256$ ) pixel image occupies a single point in 65,536-dimensional image space.
- The same type of images occupy a small region of this large image space, i.e. the intrinsic dimension is smaller
- Similarly, different types should occupy different areas of this smaller region.
- We can identify a person by finding the nearest 'known' vector in image space.

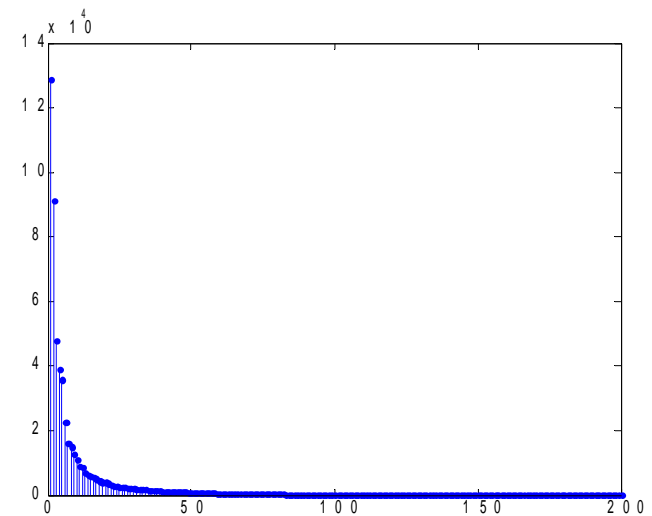
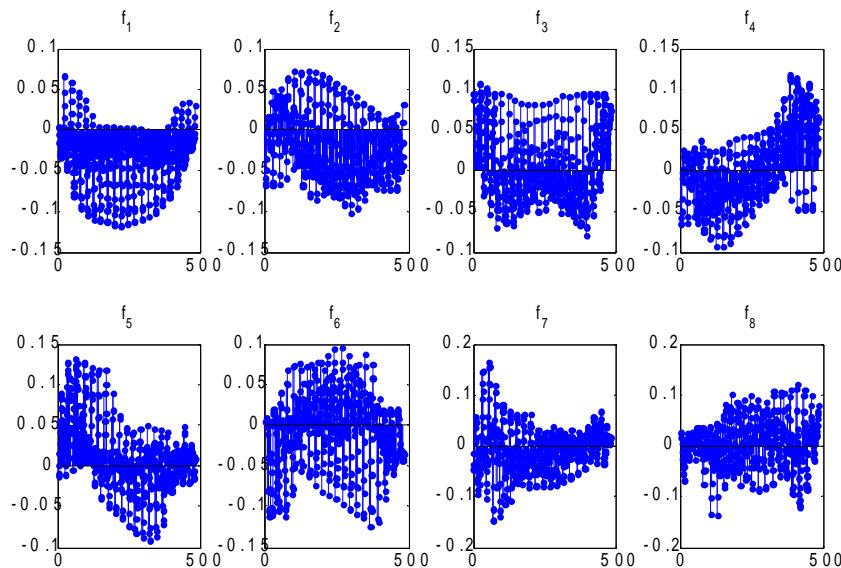


# Compute Eigenface

- **Compute Eigen Faces**
  - All face images of  $W \times H$  pixels are first scaled down to  $w \times h$  pixel icons (e.g,  $w=20$ ,  $h=24$ )
    - » Resize: `icon = imresize(fim, [24, 20])`
    - » Convert to vector: `vec = icon(:)`
  - Stack all face data into an  $n \times d$  matrix
  - Compute its Covariance  $S = \text{cov}(\text{faces})$
  - Find its eigen vectors and eigen values

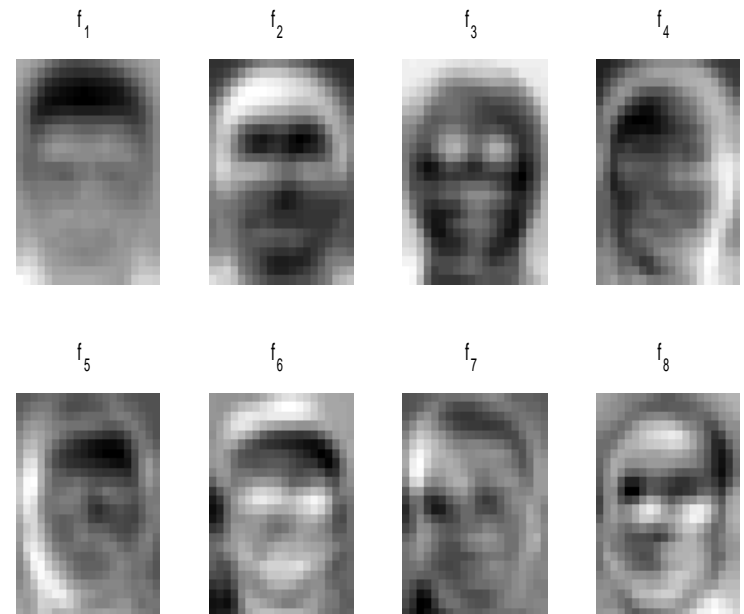
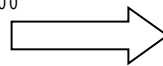
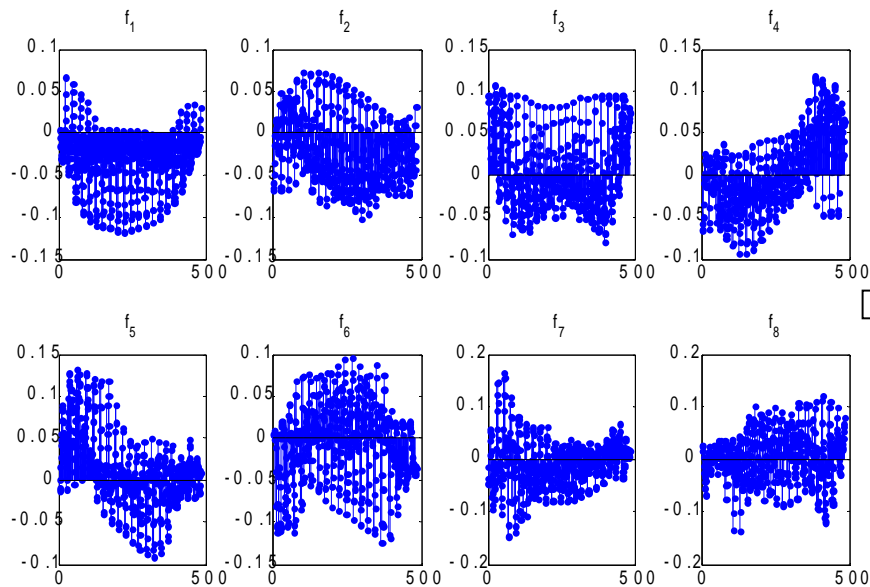
# Eigen Faces

- Eigenfaces and Eigenvalues
  - Original space dimension:  $24 \times 20 = 480$
  - First 8 principal component basis functions
  - First 200 principal dimensions and associated eigenvalues (energy)



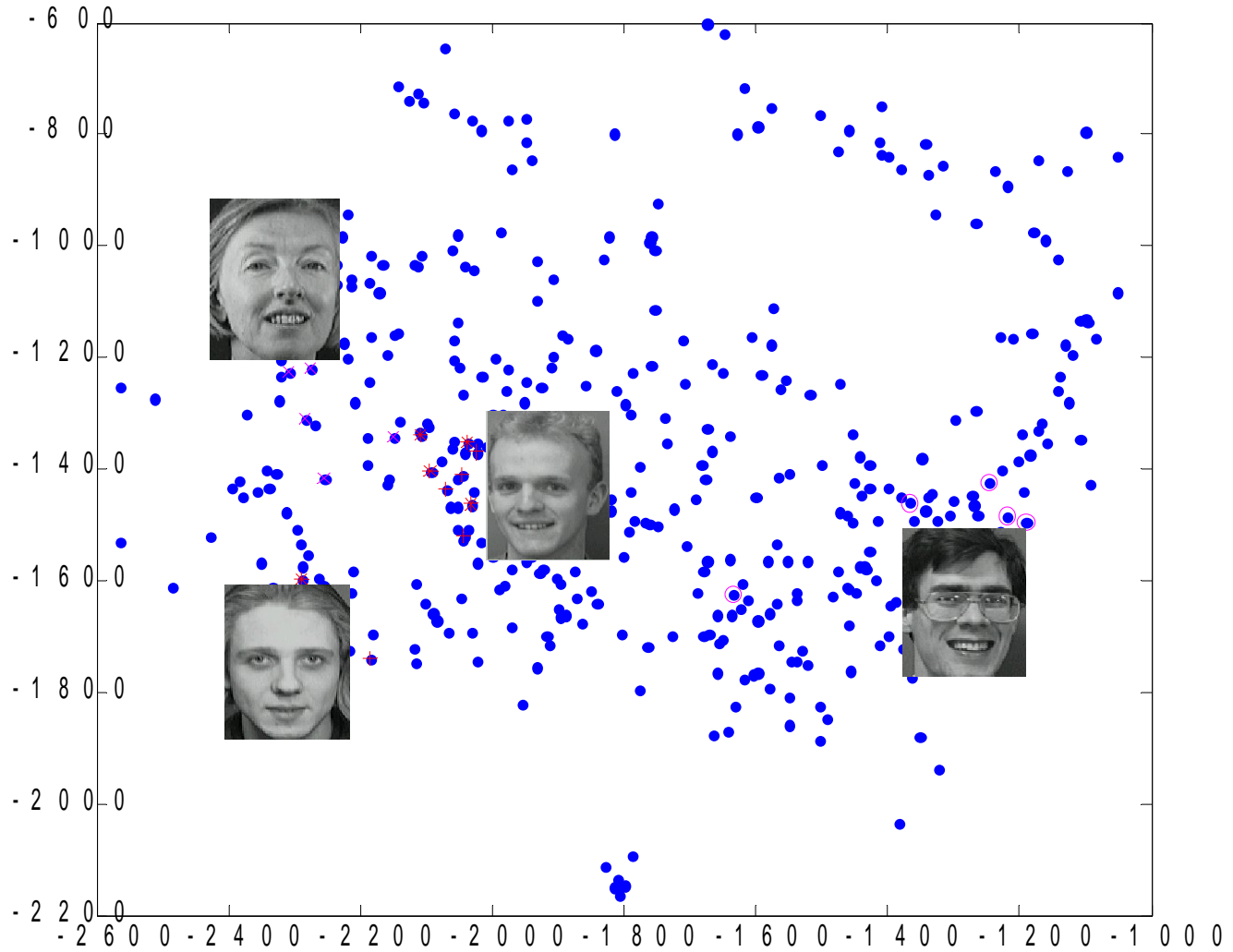
# How to plot Eigenfaces ?

- Eigenface bases functions are 1-d functions:
  - Use `reshape(x, h, w)` to convert vector back to icon images



# Eigenface Projections

- Once we found the eigenface bases, project original face images to the eigenface space:  $x = \text{faces} * A(:, 1:2);$



# PCA summary

- PCA is an signal approximation method
- Find a lower dimensional representation of the original high dimensional data by minimizing MSE re-construction error (point, line, subspace)
- For biometric problems, it is widely used as the first step in modeling
- Draw backs: unsupervised, class labels not considered.
- LDA address this by derive new objective functions

---

# Discrete Cosine Transform

# DCT - Discrete Cosine Transform

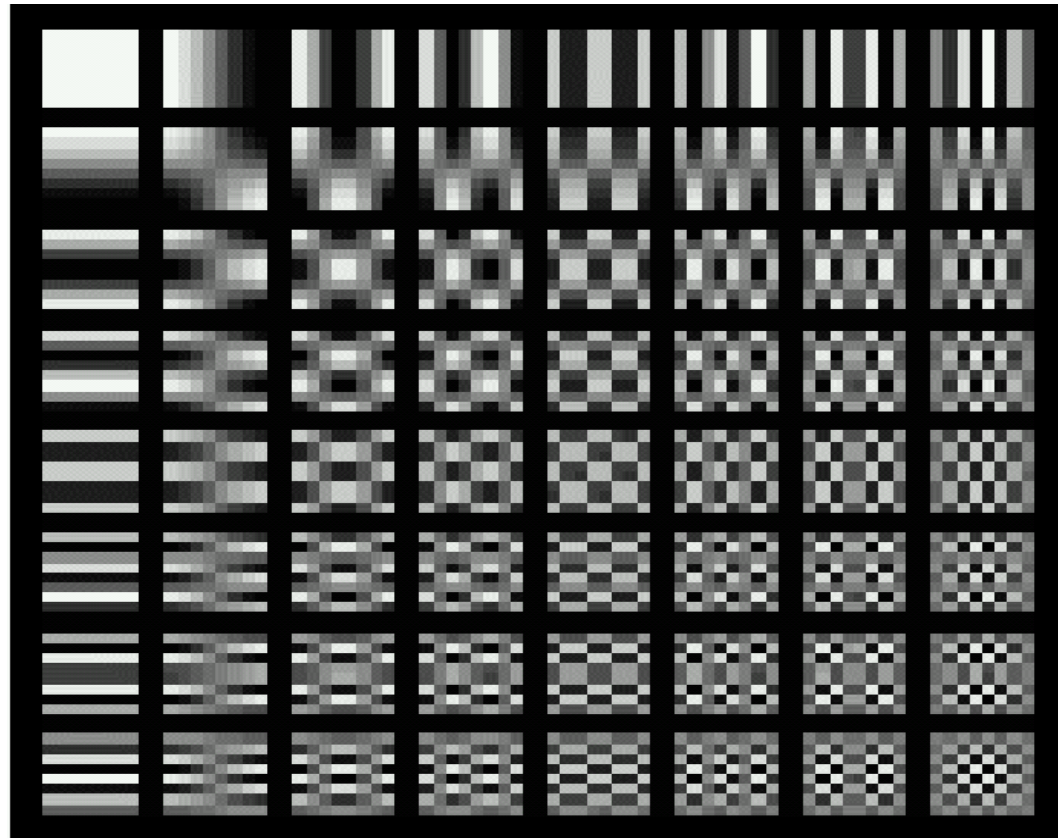
- Another widely used signal representation method
- Can be regarded as an approximation of PCA for a wide class of signals
- An N-point DCT is given by:

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[ \frac{\pi}{N} \left( n + \frac{1}{2} \right) k \right] \quad k = 0, \dots, N - 1.$$

# 2D DCT

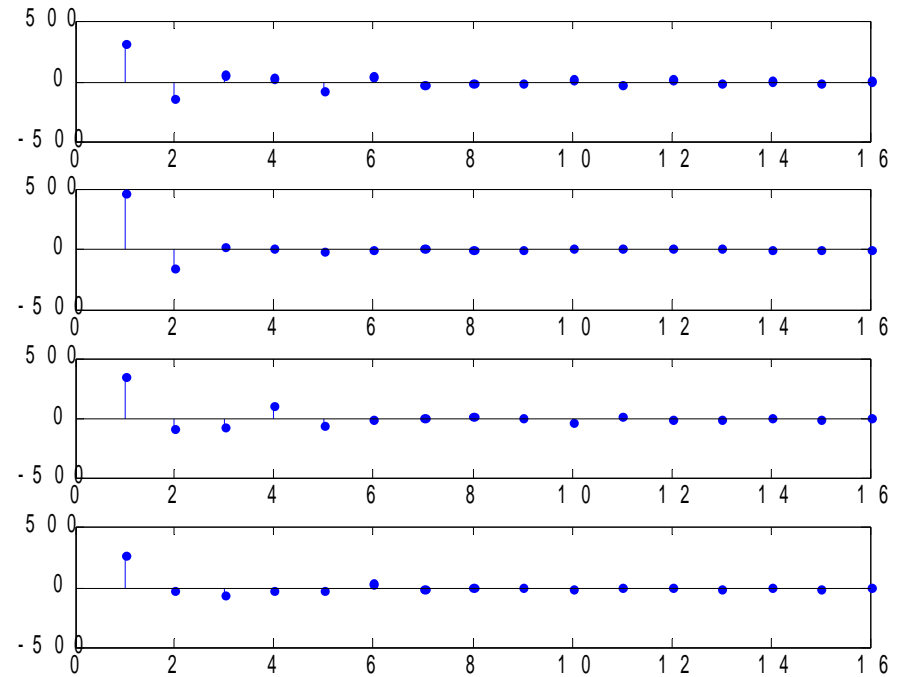
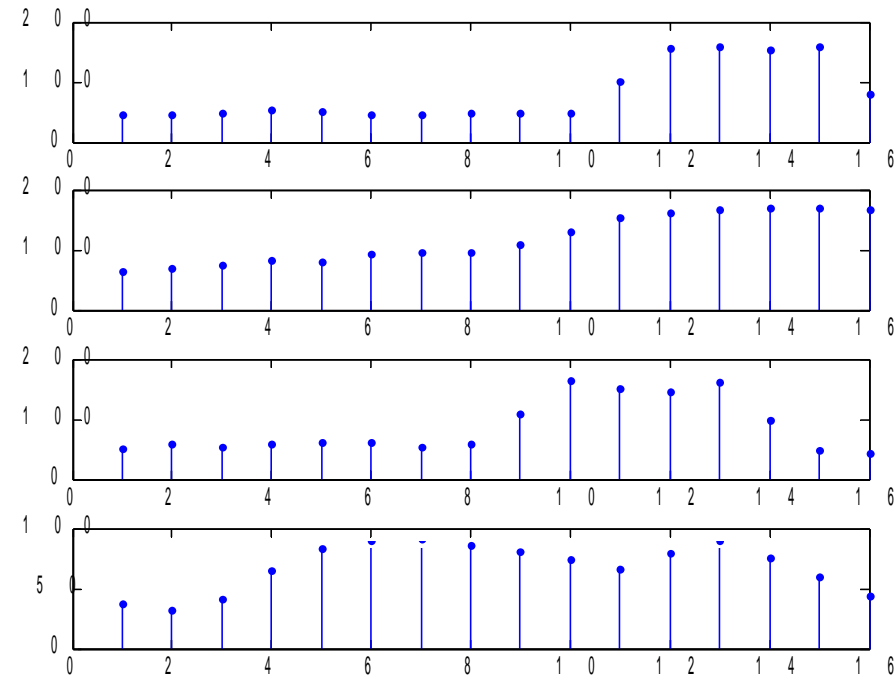
$$X_{k_1, k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1, n_2} \cos \left[ \frac{\pi}{N_1} \left( n_1 + \frac{1}{2} \right) k_1 \right] \cos \left[ \frac{\pi}{N_2} \left( n_2 + \frac{1}{2} \right) k_2 \right]$$

*Basis (8x8):*



# Matlab Implementation

- 1-d DCT:
  - An n-point DCT:  $y = \text{dct}(x, n)$ ;

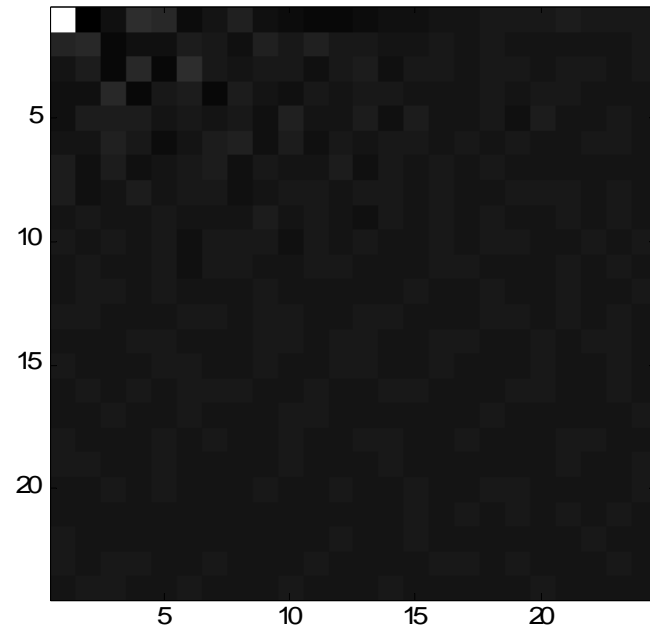
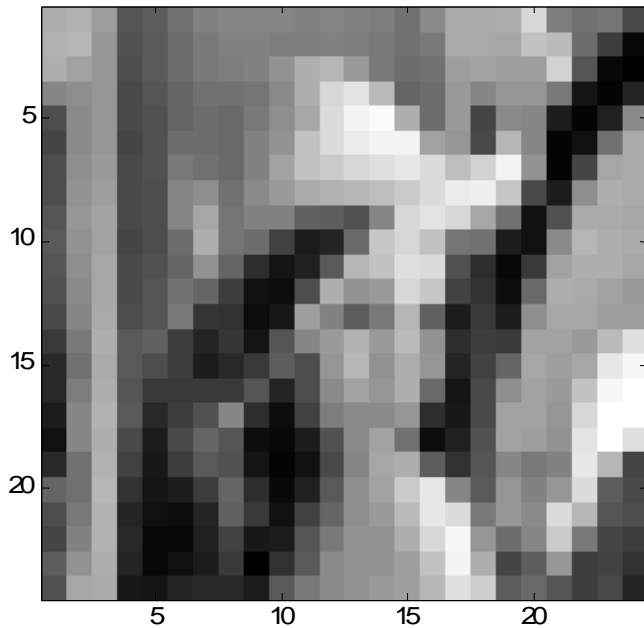


– Notice that small number of coefficients can represent the original signal

– Original can be recovered by  $x_2 = \text{idct}(y, n)$

# Matlab Implementation

- 2-d DCT:
  - For an  $h \times w$  points DCT:  $im2=dct2(im, h, w)$



- Notice that small number of co-efficients can represent the original signal

# DCT summary

---

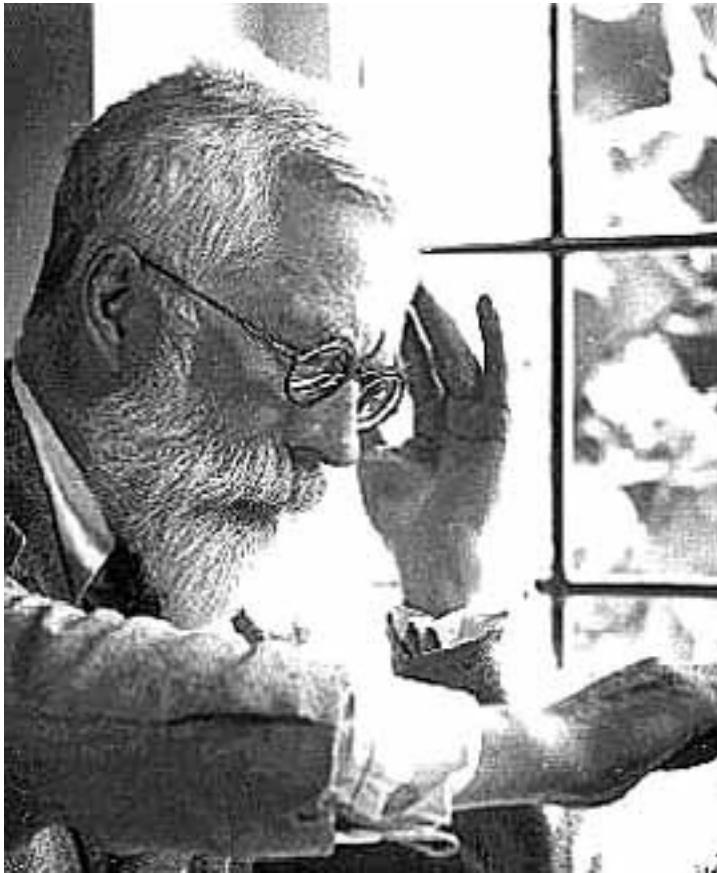
- DCT is an approximation of PCA in signal lower dimensional representation.
- It works well for image/video coding
- But the transform is not computed from the data, there not adaptive as PCA.

---

# Linear Discriminant Analysis

# First Invented By Fisher in 1936

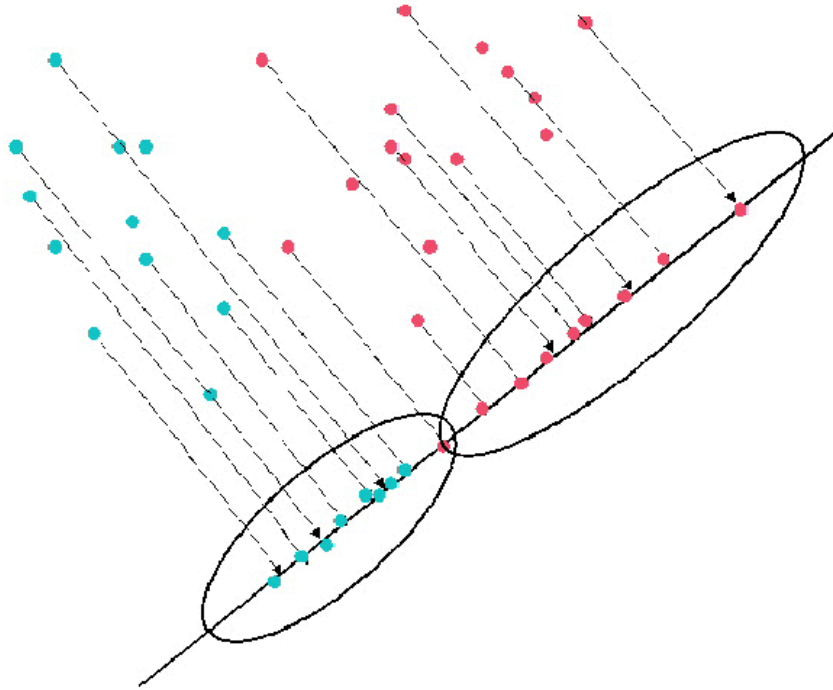
- **Ronald A. Fisher, 1890-1962**



“The elaborate mechanism built on the theory of infinitely large samples is not accurate enough for simple laboratory data. Only by systematically tackling small sample problems on their merits does it seem possible to apply accurate tests to practical data.”

1936

# LDA Formulation



- ▶ Finding an optimal linear projection  $\mathbf{W}$
- ▶ Catches major difference between classes and discount irrelevant factors
- ▶ In the projected discriminative subspace, data are clustered

# Definitions

We have two sets of labeled data:  $\mathcal{D}_1 = \{\mathbf{x}_1, \dots, \mathbf{x}_{n_1}\}$  and  $\mathcal{D}_2 = \{\mathbf{x}_1, \dots, \mathbf{x}_{n_2}\}$ . Let's define some terms:

- ▶ The centers of two classes,  $\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}_i$
- ▶ Data scatter by definition

$$\mathbf{S} = \sum_{\mathbf{x} \in \mathcal{D}} (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T$$

- ▶ **Within-class scatter:**

$$\mathbf{S}_w = \mathbf{S}_1 + \mathbf{S}_2$$

- ▶ **Between-class scatter:**

$$\mathbf{S}_b = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$$

# Fisher Criterion - 2 class case

Input: We have two sets of labeled data:  $\mathcal{D}_1 = \{\mathbf{x}_1, \dots, \mathbf{x}_{n_1}\}$  and  $\mathcal{D}_2 = \{\mathbf{x}_1, \dots, \mathbf{x}_{n_2}\}$ .

Output: We want to find a 1-d linear projection  $\mathbf{w}$  that maximizes the separability between these two classes.

- ▶ Projected data:  $\mathcal{Y}_1 = \mathbf{w}^T \mathcal{D}_1$  and  $\mathcal{Y}_2 = \mathbf{w}^T \mathcal{D}_2$
- ▶ Projected class centers:  $\tilde{m}_i = \mathbf{w}^T \mathbf{m}_i$
- ▶ Projected within-class scatter (it is a scalar in this case)

$$\tilde{\mathbf{S}}_w = \mathbf{w}^T \mathbf{S}_w \mathbf{w} \quad \textit{prove it!}$$

- ▶ Projected between-class scatter (it is a scalar in this case)

$$\tilde{\mathbf{S}}_b = \mathbf{w}^T \mathbf{S}_b \mathbf{w} \quad \textit{prove it!}$$

- ▶ Fisher Linear Discriminant

$$J(\mathbf{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{\mathbf{S}}_1 + \tilde{\mathbf{S}}_2} = \frac{|\tilde{\mathbf{S}}_b|}{|\tilde{\mathbf{S}}_w|} = \frac{\mathbf{w}^T \mathbf{S}_b \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}}$$

# Rayleigh Quotient Theorem

## Theorem

$f(\lambda) = \|\mathbf{Ax} - \lambda\mathbf{Bx}\|_B$  where  $\|\mathbf{z}\|_B \triangleq \mathbf{z}^T \mathbf{B}^{-1} \mathbf{z}$  is minimized by the Rayleigh quotient

$$\lambda = \frac{\mathbf{x}^T \mathbf{Ax}}{\mathbf{x}^T \mathbf{Bx}}$$

## Proof.

$$\begin{aligned} \frac{\partial f(\lambda)}{\partial \lambda} &= (\mathbf{Bx})^T (\mathbf{Bz}) = \mathbf{x}^T \mathbf{B}^T \mathbf{B}^{-1} \mathbf{z} \\ &= \mathbf{x}^T (\mathbf{Ax} - \lambda \mathbf{Bx}) = \mathbf{x}^T \mathbf{Ax} - \lambda \mathbf{x}^T \mathbf{Bx} \end{aligned}$$

# Eigen Solution for LDA

## Theorem

$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_b \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}}$  is maximized when

$$\mathbf{S}_b \mathbf{w} = \lambda \mathbf{S}_w \mathbf{w}$$

## Proof.

Let  $\mathbf{w}^T \mathbf{S}_w \mathbf{w} = c \neq 0$ . We can construct the Lagrangian as

$$L(\mathbf{w}, \lambda) = \mathbf{w}^T \mathbf{S}_b \mathbf{w} - \lambda(\mathbf{w}^T \mathbf{S}_w \mathbf{w} - c)$$

Then KKT is

$$\frac{\partial L(\mathbf{w}, \lambda)}{\partial \mathbf{w}} = \mathbf{S}_b \mathbf{w} - \lambda \mathbf{S}_w \mathbf{w}$$

It is clearly that

$$\mathbf{S}_b \mathbf{w}^* = \lambda \mathbf{S}_w \mathbf{w}^*$$

# Multiple Classes

Now, we have  $c$  number of classes:

- ▶ within-class scatter  $\mathbf{S}_w = \sum_{i=1}^c \mathbf{S}_i$  as before
- ▶ between-class scatter is a bit different from 2-class

$$\mathbf{S}_b \triangleq \sum_{i=1}^c n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$$

- ▶ total scatter

$$\mathbf{S}_t \triangleq \sum_x (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T = \mathbf{S}_w + \mathbf{S}_b$$

- ▶ MDA is to find a subspace with bases  $\mathbf{W}$  that maximizes

$$J(\mathbf{W}) = \frac{|\tilde{\mathbf{S}}_b|}{|\tilde{\mathbf{S}}_w|} = \frac{|\mathbf{W}^T \mathbf{S}_b \mathbf{W}|}{|\mathbf{W}^T \mathbf{S}_w \mathbf{W}|}$$

# Solution by Generalized Eigen Value Decomp

- ▶ The solution is obtained by G-EVD

$$\mathbf{S}_b \mathbf{w}_i = \lambda_i \mathbf{S}_w \mathbf{w}_i$$

where each  $\mathbf{w}_i$  is a generalized eigenvector

- ▶ In practice, what we can do is the following
  - ▶ find the eigenvalues as the root of the characteristic polynomial

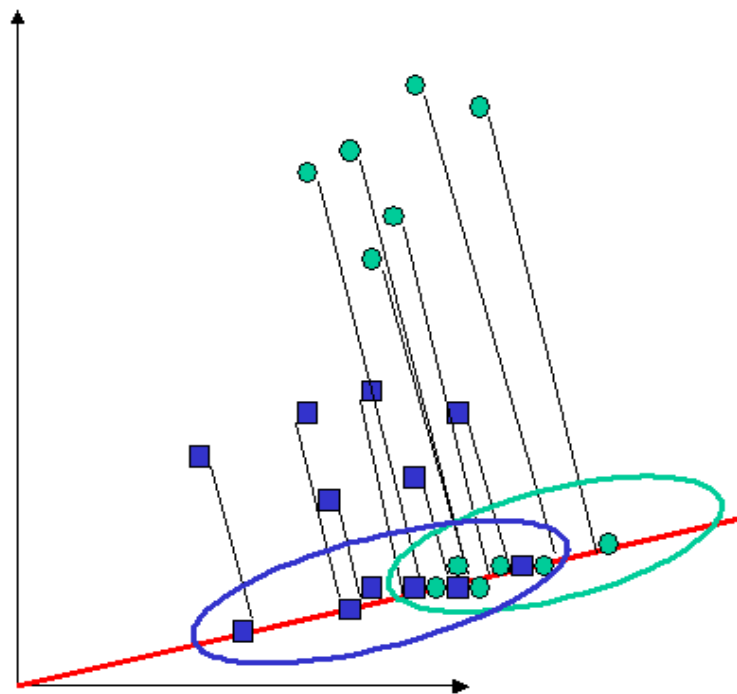
$$|\mathbf{S}_b - \lambda_i \mathbf{S}_w| = 0$$

- ▶ for each  $\lambda_i$ , solve  $\mathbf{w}_i$  from

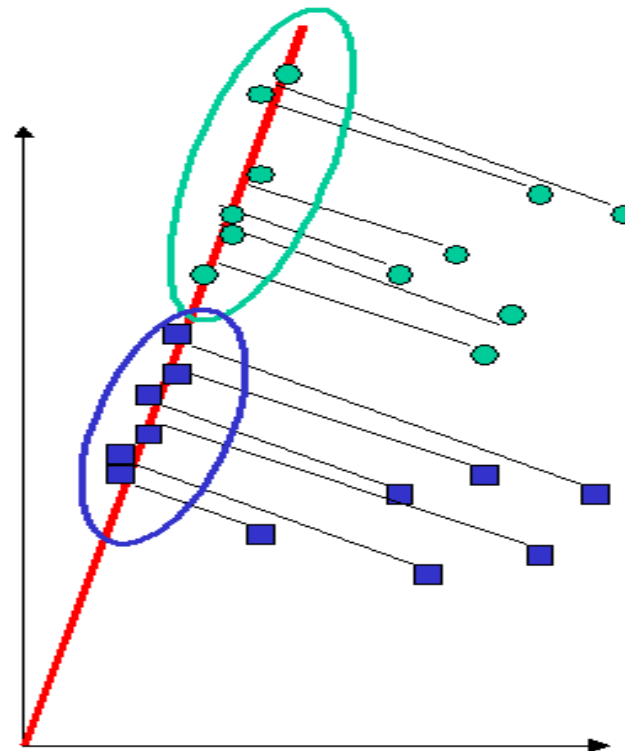
$$(\mathbf{S}_b - \lambda_i \mathbf{S}_w) \mathbf{w}_i = 0$$

*In Matlab, generalized eigen decomp:  
[E, D]=eigs(Sw, Sb)*

# PCA vs LDA first component



*LDA*



*PCA*

# Application of LDA

- **Fisherface in face recognition**
  - Instead of eigenface, it finds the discriminant subspace by including class label info in subspace modeling
  - Compute within class scatter:
  - Compute between class scatter:
  - Solve generalized eigen value problem by  $\text{eig}(S_b, S_w)$ ;

# Matlab Implementation

- Compute Inter/Intra Scatters

```
%function [A, nClass]=myLDA(X, Lb1)
% var
[n kDim]=size(X);
uLabels = sort(unique(Lb1));
nClass = length(uLabels);

% LDA
mX = mean(X);
mm = zeros(kDim, kDim);

% compute between class variance Sb and within class variance Sw
for k=1:nClass
    indx = find(Lb1==uLabels(k));
    mXk = mean(X(indx,:)); % m x kDim
    mm = mm + length(indx)*mXk'*mXk;
end

Sw = X'*X - mm;
Sb = mm - n*mX'*mX;
Sw = (Sw+Sw')/2;
Sb = (Sb+Sb')/2;
```

# Matlab Implementation

- Generalized Eigen Value Decomp:

```
option = struct('disp',0);  
[A, egv]=eigs(Sb, Sw, nClass-1, 'la', option);  
  
for k=1:nClass-1  
    A(:,k) = A(:,k)./norm(A(:,k));  
end
```

# Fisherface Example



- Eigenface is sensitive to lighting changes
- Fisherface only selects features that are helping the recognition.

# Summary

- **Transform based image processing**
  - Operates on the whole image
  - PCA is optimal in signal re-construction
  - DCT is a good approximation of PCA for most signals
  - LDA is a discriminant model, taking class labels into consideration, therefore performs better
- **An unified graph embedding theory for subspace models:**
  - Xiaofei He's PhD thesis.