

EE 4211 Computer Vision

Lecture 9: Face recognition

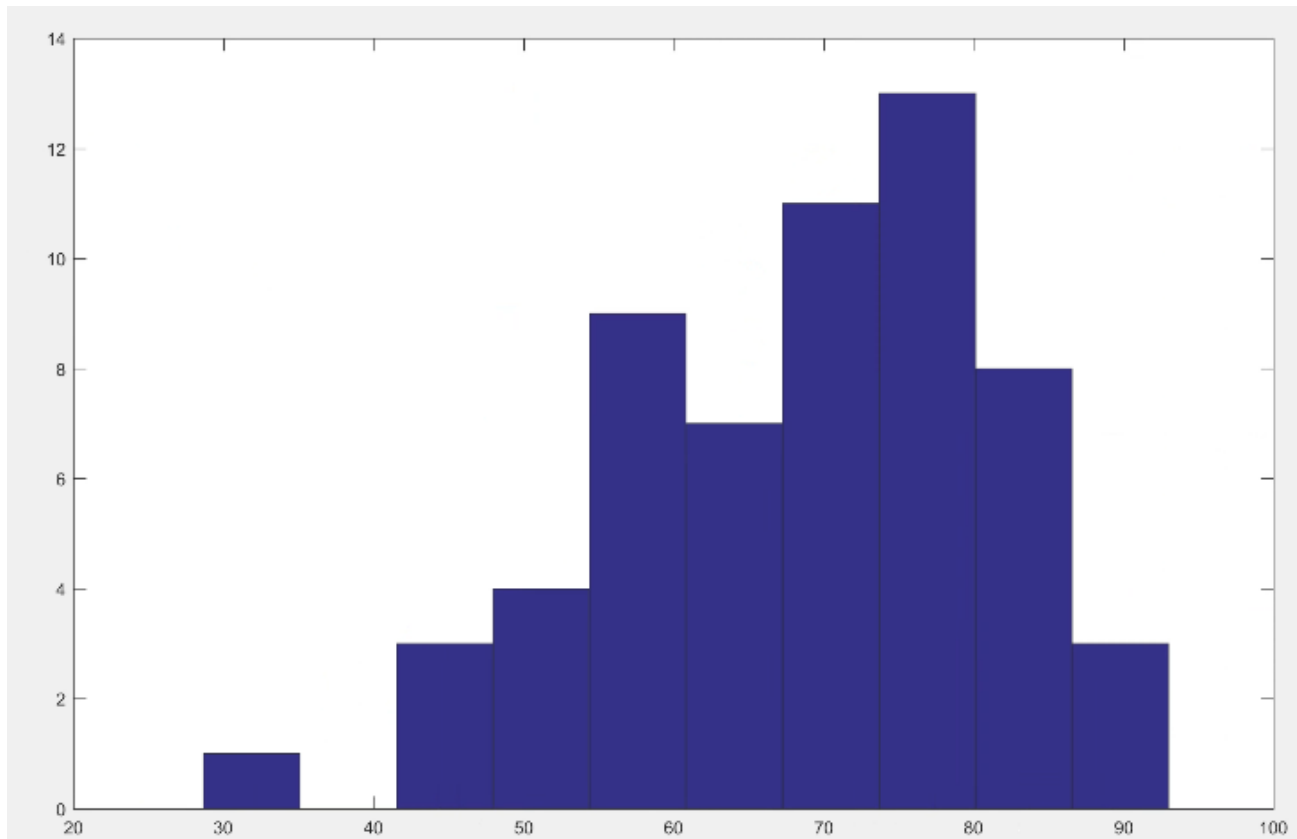
Semester B, 2021-2022

Schedules

Week	Date	Topics
1	Jan. 11 (face to face)	Introduction/Imaging
2	Jan. 18 (online)	Image enhancement in spatial domain
3	Jan. 25 (online)	Image enhancement in frequency domain (HW1 out)
4	Feb. 8 (online)	Morphological processing
5	Feb. 15 (online)	Image restoration (HW1 due)
6	Feb. 22 (online)	Midterm (no tutorials this week)
7	Mar. 1 (online)	Edge detection (HW2 out)
8	Mar. 8 (online)	Image segmentation
9	Mar. 15 (online)	Face recognition with PCA, LDA (tutorial on segmentation) (HW2 due)(will publish project group information today.)
10	Mar. 22 (online)	Face recognition based on deep learning (Quiz on two code questions, 1 hour) Image segmentation based on deep learning
11	Mar. 29 (online)	Object detection with traditional methods Object detection based on deep learning (tutorial on detection)
12	Apr. 5	Events / Public Holidays
13	Apr. 12 (online)	Invited project presentation and Summary

Midterm

- Mean: 69.22



Quiz

- Code
- Open book and open notes
- 13:45-14:45 on Mar. 22 (close at 15:00 PM, 15 mins for submission)
- Submit online through canvas/assignment, no late submission is allowed
- Open two cameras since you have to do that in the final exam. You can use this quiz to get familiar about this arrangement

Outline

- Introduction to face recognition
- Principal Component Analysis (PCA)
- The Eigenfaces Algorithm
- Linear Discriminant Analysis (LDA)

Overview

- Face recognition problem
 - Given a still image, identify one or more persons in this image using a stored database of facial images



Who is she?



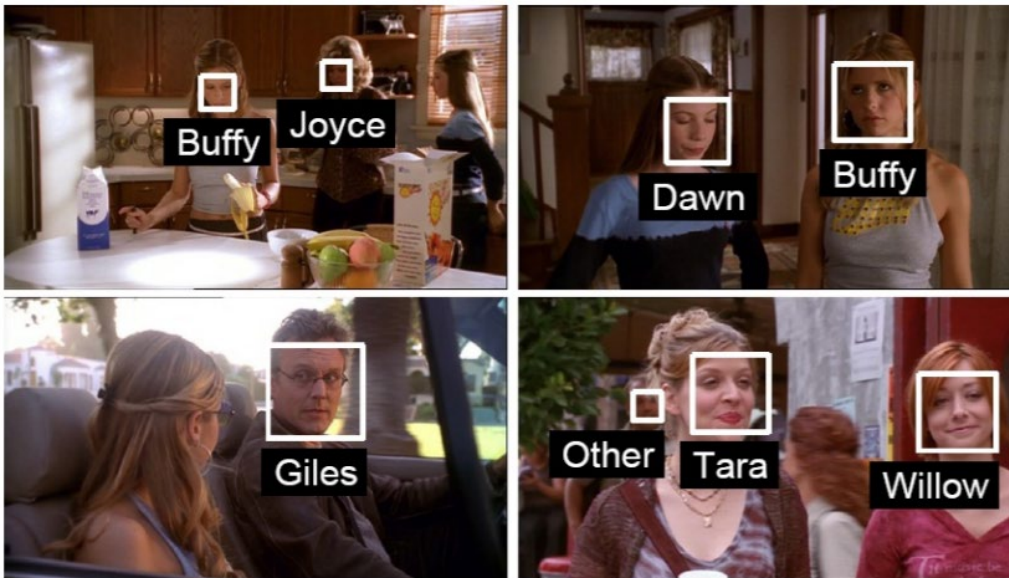
Face Recognition

- Digital photography
- Surveillance
- Album organization

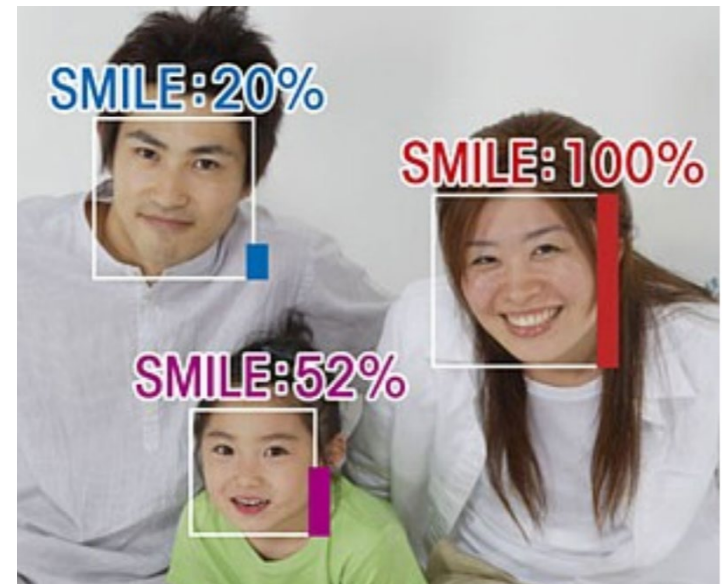


Face Recognition

Person tracking/id.

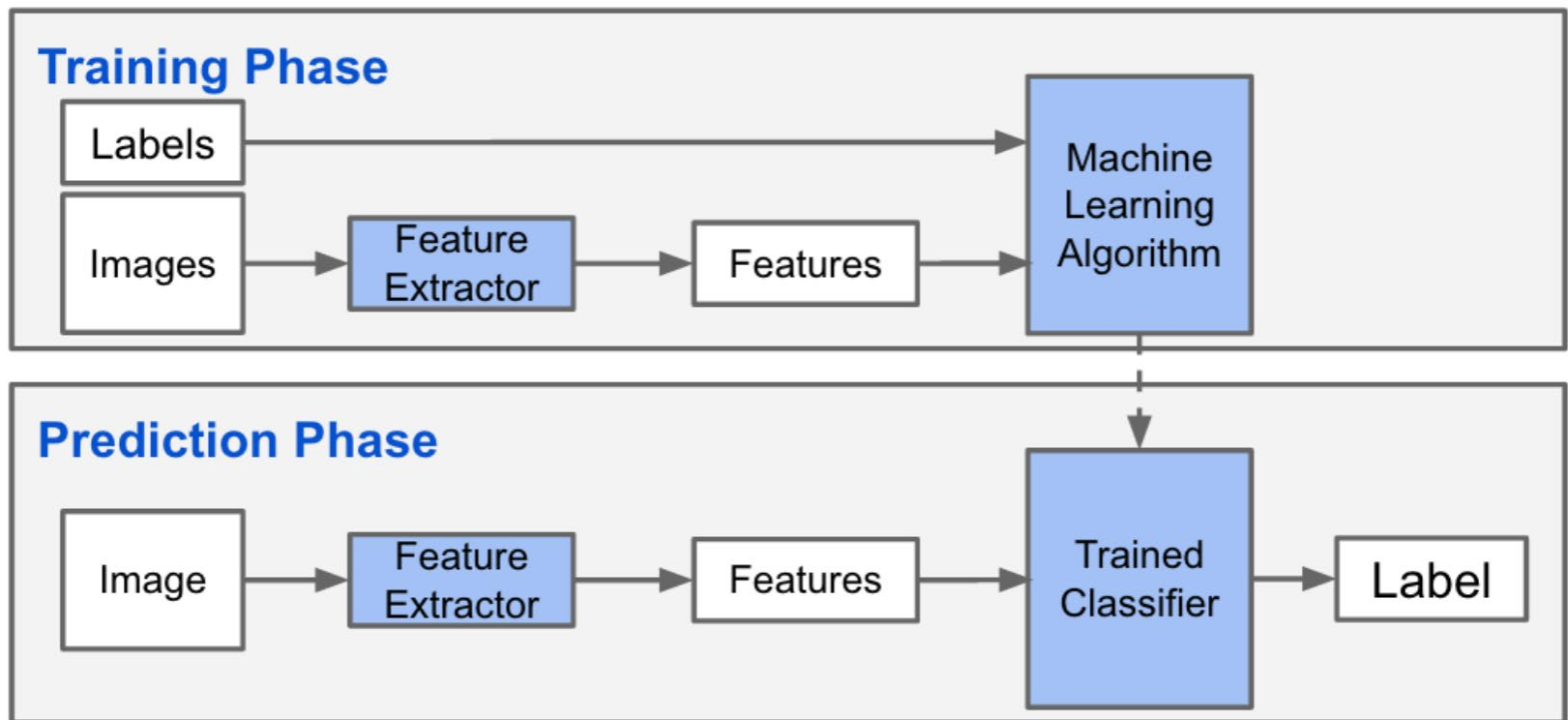


Emotions and expressions



Face Recognition

- Supervised learning in traditional machine learning
- Sometimes, we directly calculate the distance among the testing images and training images to assign labels.



Face Recognition

- Deep learning based methods
- Jointly optimize the feature extraction and training procedure
- Achieve better performance compared with the previous machine learning methods

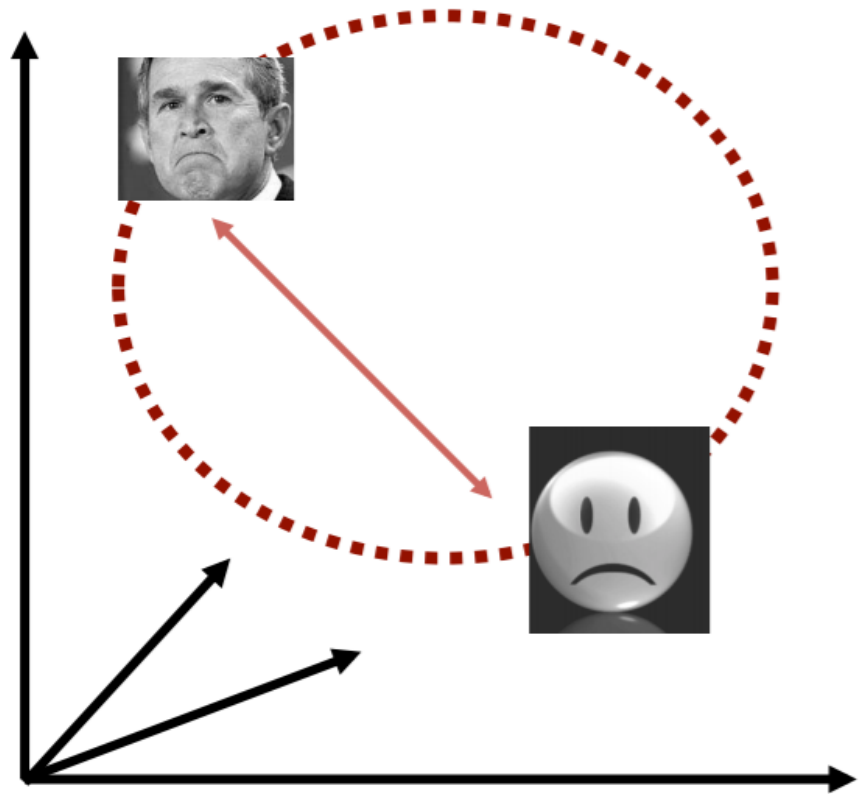


Traditional Machine Learning Flow



The Space of Faces

- Not a machine learning based method
- An image is a point in a high dimensional space
- If represented in grayscale intensity, an $N \times M$ image is a point in R^{NM}
- E.g. 100×100 image = 10,000 dim



Key Idea

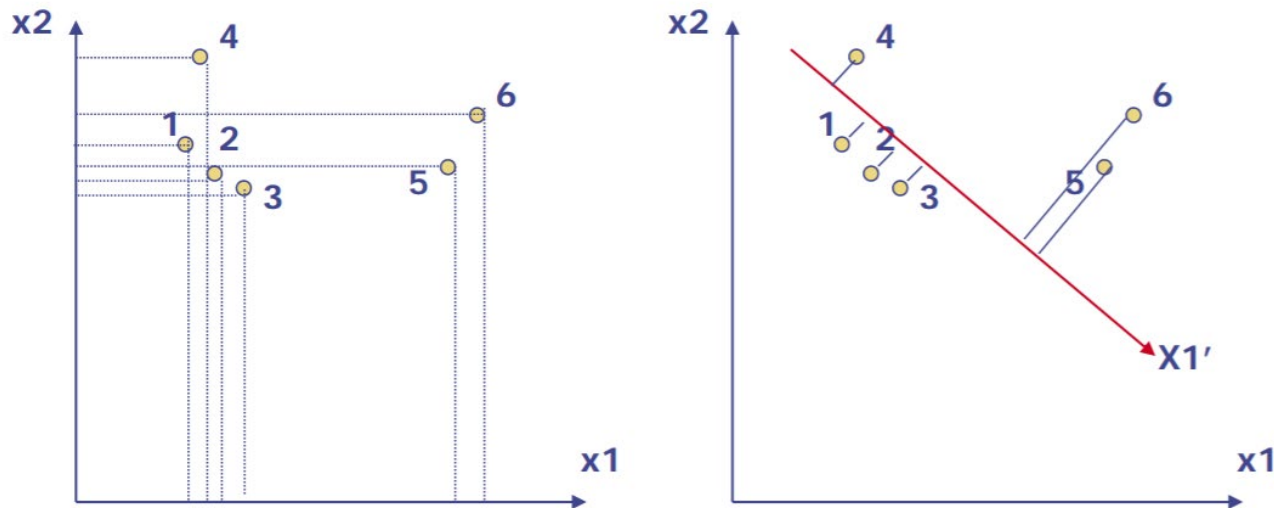
- So, compress them to a low-dimensional **subspace** that captures key appearance characteristics of the visual Degree of Freedoms.
- USE **PCA for estimating the sub-space** (dimensionality reduction)
- Compare two faces by projecting the images into the subspace and measuring the EUCLIDEAN distance between them.

Outline

- Introduction to face recognition
- Principal Component Analysis (PCA)
- The Eigenfaces Algorithm
- Linear Discriminant Analysis (LDA)

USE PCA for estimating the sub-space

- PCA: converts a set of observations of possibly **correlated variables** into a set of values of linearly **uncorrelated variables** called principal components
- Computes n-dim subspace such that the projection of the data points onto the subspace has the **largest variance** among all n-dim subspaces.

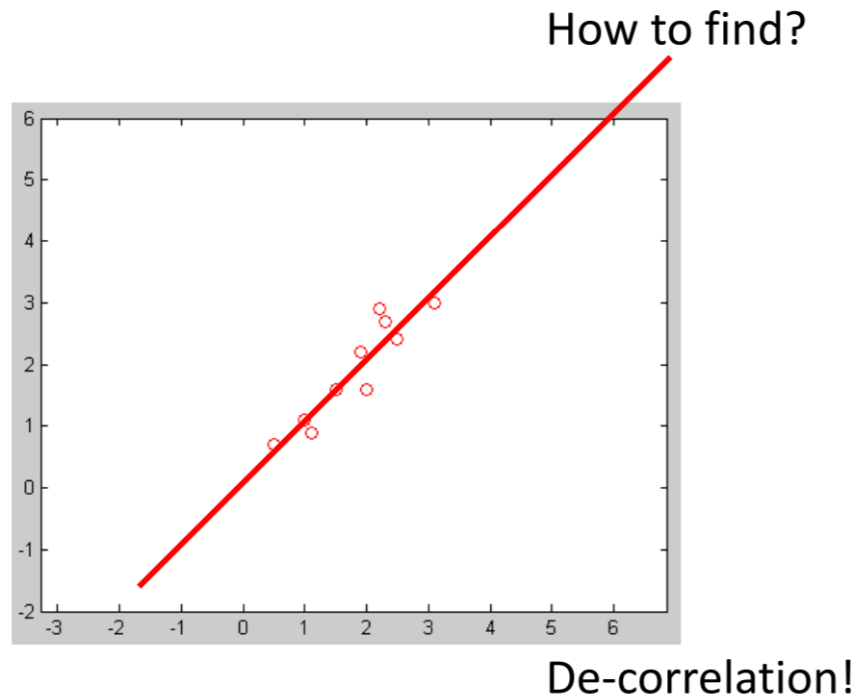


PCA projection

PCA

■ Illustration

x , y
(2.5, 2.4)
(0.5, 0.7)
(2.2, 2.9)
(1.9, 2.2)
(3.1, 3.0)
(2.3, 2.7)
(2.0, 1.6)
(1.0, 1.1)
(1.5, 1.6)
(1.1, 0.9)



Principal Component Analysis (PCA)

- **Purpose:** Identify the orientation with largest variance
- Suppose $\mathbf{X}$ contains n data points, and each data point is p -dimensional, that is

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}, \mathbf{x}_i \in \mathbb{R}^{p \times 1}, \mathbf{X} \in \mathbb{R}^{p \times n}$$

- Now, we want to find such a unit vector

$$\alpha_1 = \arg \max_{\alpha} \left(\text{var} \left(\alpha^T \mathbf{X} \right) \right), \alpha \in \mathbb{R}^{p \times 1}$$

PCA

- Identify the orientation with largest variance

$$\begin{aligned}\text{var}(\alpha^T \mathbf{X}) &= \frac{1}{n-1} \sum_{i=1}^n (\alpha^T \mathbf{x}_i - \alpha^T \mu)^2 = \frac{1}{n-1} \sum_{i=1}^n \alpha^T (\mathbf{x}_i - \mu) (\mathbf{x}_i - \mu)^T \alpha \\ &= \alpha^T C \alpha\end{aligned}\quad (\text{Note that: } \alpha^T (\mathbf{x}_i - \mu) = (\mathbf{x}_i - \mu)^T \alpha)$$

$$\text{where } \mu = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$$

$$\text{and } C = \frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \mu) (\mathbf{x}_i - \mu)^T \text{ is the covariance matrix}$$

PCA

- Identify the orientation with largest variance

Since α is unit, $\alpha^T \alpha = 1$

Based on Lagrange multiplier method, we need to,

$$\arg \max_{\alpha} \left(\alpha^T C \alpha - \lambda (\alpha^T \alpha - 1) \right)$$

$$0 = \frac{d \left(\alpha^T C \alpha - \lambda (\alpha^T \alpha - 1) \right)}{d \alpha} = 2C\alpha - 2\lambda\alpha \rightarrow C\alpha = \lambda\alpha$$

α is C 's eigen-vector

Since,

$$\max \left(\text{var} \left(\alpha^T \mathbf{X} \right) \right) = \max \left(\alpha^T C \alpha \right) = \max \left(\alpha^T \lambda \alpha \right) = \max \left(\lambda \right)$$

Thus,

PCA

- Identify the orientation with largest variance
- Thus, α_1 should be the eigen-vector of C corresponding to the largest eigen-value of C
- Results: the eigen-vectors of C forms a set of orthogonal basis and they are referred as **Principal Components** of the original data X
- You can consider PCs as a set of orthogonal coordinates. Under such a coordinate system, variables are not correlated.

PCA

- Express data in PCs

Suppose $\{\alpha_1, \alpha_2, \dots, \alpha_p\}$ are PCs derived from $\mathbf{X}$, $\mathbf{X} \in \mathbb{R}^{p \times n}$

Then, a data point $\mathbf{x}_i \in \mathbb{R}^{p \times 1}$ can be linearly represented by $\{\alpha_1, \alpha_2, \dots, \alpha_p\}$, and the representation coefficients are

$$\mathbf{c}_i = \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \\ \vdots \\ \alpha_p^T \end{pmatrix} \mathbf{x}_i$$

Actually, $\mathbf{c}_i$ is the coordinates of $\mathbf{x}_i$ in the new coordinate system spanned by $\{\alpha_1, \alpha_2, \dots, \alpha_p\}$

PCA

■ Illustration

x, y

(2.5, 2.4)

(0.5, 0.7)

(2.2, 2.9)

(1.9, 2.2)

(3.1, 3.0)

(2.3, 2.7)

(2.0, 1.6)

(1.0, 1.1)

(1.5, 1.6)

(1.1, 0.9)

$$\mathbf{X} = \begin{pmatrix} 2.5 & 0.5 & 2.2 & 1.9 & 3.1 & 2.3 & 2.0 & 1.0 & 1.5 & 1.1 \\ 2.4 & 0.7 & 2.9 & 2.2 & 3.0 & 2.7 & 1.6 & 1.1 & 1.6 & 0.9 \end{pmatrix}$$

$$\text{cov}(\mathbf{X}) = \begin{pmatrix} 5.549 & 5.539 \\ 5.539 & 6.449 \end{pmatrix}$$

Eigen-values = 11.5562, 0.4418

Corresponding eigen-vectors:

$$\alpha_1 = \begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$$

$$\alpha_2 = \begin{pmatrix} -0.7352 \\ 0.6779 \end{pmatrix}$$

PCA

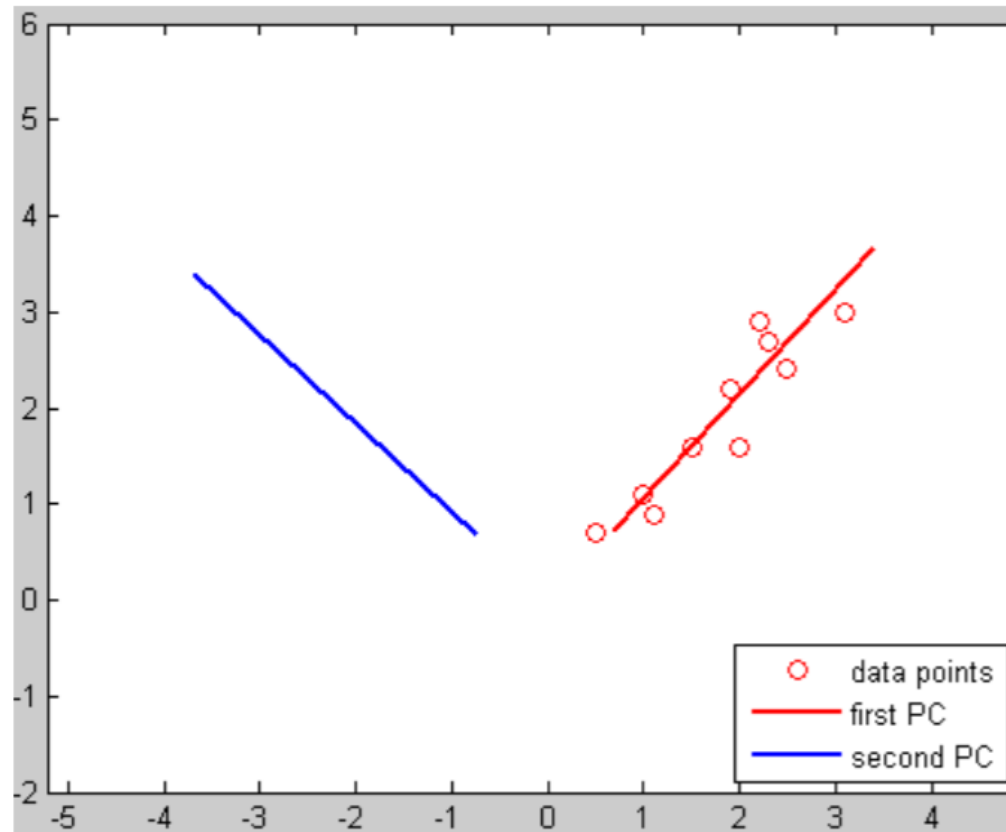
■ Illustration

Coordinates of the data points in the new coordinate system

$$\begin{aligned} newC &= \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \end{pmatrix} \mathbf{X} \\ &= \begin{pmatrix} 0.6779 & 0.7352 \\ -0.7352 & 0.6779 \end{pmatrix} \mathbf{X} \\ &= \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \\ -0.211 & 0.107 & 0.348 & 0.094 & -0.245 & 0.139 & -0.386 & 0.011 & -0.018 & -0.199 \end{pmatrix} \end{aligned}$$

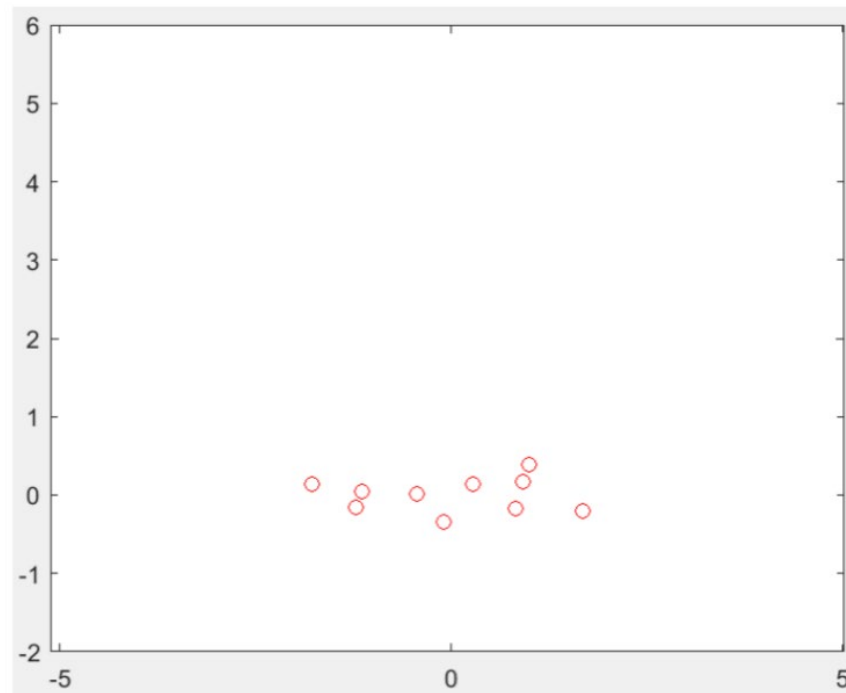
PCA

■ Illustration



PCA

- Illustration
- Coordinates of the data points in the new coordinate system
Draw *newC* on the plot
- In such a new system, two variables are linearly independent!



PCA

- Data dimension reduction with PCA

Suppose $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^{p \times 1}, \{\alpha_i\}_{i=1}^p, \alpha_i \in \mathbb{R}^{p \times 1}$ are the PCs

If all of $\{\alpha_i\}_{i=1}^p$ are used, $\mathbf{c}_i = \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \\ \vdots \\ \alpha_p^T \end{pmatrix} \mathbf{x}_i$ is still p -dimensional

If only $\{\alpha_i\}_{i=1}^m, m < p$ are used, $\mathbf{c}_i$ will be m -dimensional

That is, the dimension of the data is reduced!

PCA

■ Illustration

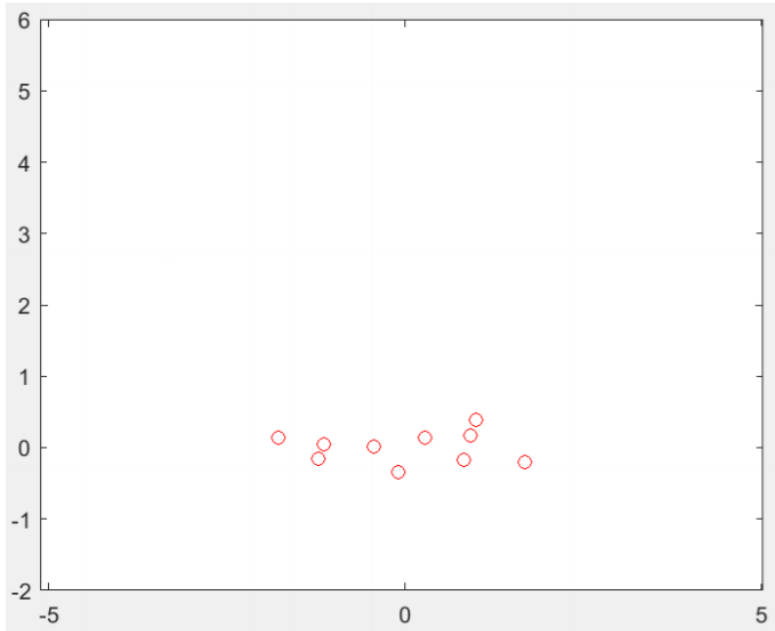
Coordinates of the data points in the new coordinate system

$$newC = \begin{pmatrix} 0.6779 & 0.7352 \\ -0.7352 & 0.6779 \end{pmatrix} \mathbf{X}$$

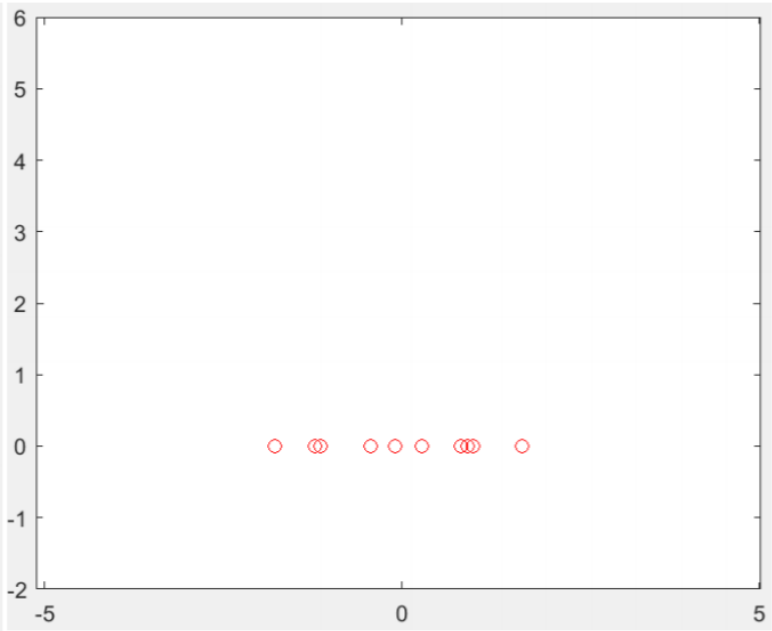
If only the first PC (corresponds to the largest eigen-value) is remained

$$\begin{aligned} newC &= (0.6779 \quad 0.7352) \mathbf{X} \\ &= (3.459 \quad 0.854 \quad 3.623 \quad 2.905 \quad 4.307 \quad 3.544 \quad 2.532 \quad 1.487 \quad 2.193 \quad 1.407) \end{aligned}$$

PCA



All PCs are used



Only 1 PC is used

Dimension reduction!

PCA

- If only the first PC (corresponds to the largest eigen-value) is remained

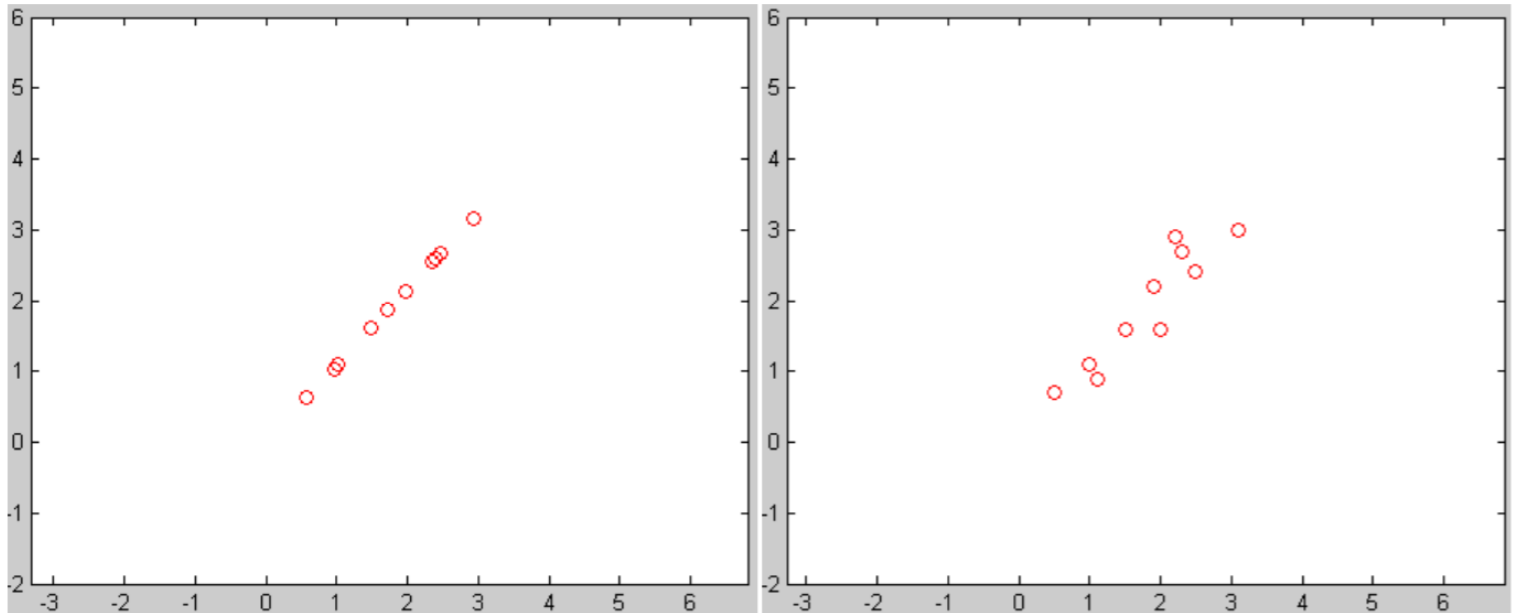
$$\begin{aligned} newC &= (0.6779 \quad 0.7352) \mathbf{X} \\ &= (3.459 \quad 0.854 \quad 3.623 \quad 2.905 \quad 4.307 \quad 3.544 \quad 2.532 \quad 1.487 \quad 2.193 \quad 1.407) \end{aligned}$$

How to recover $newC$ to the original space? Easy

$$\begin{aligned} &(0.6779 \quad 0.7352)^T newC \\ &= \begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix} (3.459 \quad 0.854 \quad 3.623 \quad 2.905 \quad 4.307 \quad 3.544 \quad 2.532 \quad 1.487 \quad 2.193 \quad 1.407) \end{aligned}$$

PCA

■ Illustration



Data recovered if only 1 PC used

Original data

PCA example

- Assume the data pairs (1,1), (1,2), (3,2), (4,2), and (6,3)
- $\mu_X = 15/5 = 3$ and $\mu_Y = 10/5 = 2$
- The mean-centered data becomes (-2,-1), (-2,0), (0,0), (1,0), and (3,1)
- The data matrix $\mathbf{X} = \begin{bmatrix} -2 & -2 & 0 & 1 & 3 \\ -1 & 0 & 0 & 0 & 1 \end{bmatrix}$
- The covariance matrix $\frac{1}{5}\mathbf{X}\mathbf{X}^T = \frac{1}{5} \begin{bmatrix} 18 & 5 \\ 5 & 2 \end{bmatrix}$
- The characteristic polynomial is $\frac{1}{5} \det \begin{vmatrix} 18 - \lambda & 5 \\ 5 & 2 - \lambda \end{vmatrix}$

PCA example

- The characteristic polynomial is $\frac{1}{5} \det \begin{vmatrix} 18 - \lambda & 5 \\ 5 & 2 - \lambda \end{vmatrix}$
 $= \frac{1}{5} ((18 - \lambda)(2 - \lambda) - 25) = \frac{1}{5}(\lambda^2 - 20\lambda + 11)$
- When setting it to zero we can omit the factor $\frac{1}{5}$
- We get $\lambda = \frac{20 \pm \sqrt{400 - 44}}{2}$, $\lambda_1 \approx 19.43$ and $\lambda_2 \approx 0.57$
as the eigenvalues of the covariance matrix
- We always choose the eigenvalues to be in decreasing order: $\lambda_1 > \lambda_2 > \dots$

PCA example

- The first eigenvalue $\lambda_1 \approx 19.43$ corresponds to an eigenvector $(1, 0.29)$ or anything parallel to it
- The second eigenvalue $\lambda_2 \approx 0.57$ corresponds to an eigenvector $(-0.29, 1)$ or anything parallel to it

PCA example (1)

- If you have forgotten steps required for finding eigenvalues and eigenvectors, here is an example

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

$$\det(A - \lambda I) = \begin{vmatrix} 3 - \lambda & 1 \\ 1 & 3 - \lambda \end{vmatrix} = (3 - \lambda)^2 - 1 = 0$$
$$\lambda_1 = 2, \quad \lambda_2 = 4$$



$$(A - \lambda_1 I)x = \begin{bmatrix} 3 - 2 & 1 \\ 1 & 3 - 2 \end{bmatrix} x = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} x = 0$$

$$x = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$(A - \lambda_2 I)x = \begin{bmatrix} 3 - 4 & 1 \\ 1 & 3 - 4 \end{bmatrix} x = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} x = 0$$

$$x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

PCA example (2)

For the give data $\{(-1,-2); (-1,0); (0,0); (2,1); (0,1)\}$; please illustrate the steps to calculate the following (handwritten):

- (a) The covariance matrix;
- (b) Eigen values;
- (c) Eigen vectors;
- (d) Apply PCA on the dataset and represent the data after the dimension reduction.
(Please illustrate the calculation steps one by one).

PCA example (2)

Solutions:

$$X = \begin{bmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{bmatrix};$$

$$(a) \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \tilde{X} = X - \mu = \begin{bmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\text{cov}(X) = \frac{1}{n-1} \tilde{X} \tilde{X}^T = \frac{1}{4} \begin{bmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & -2 \\ -1 & 0 \\ 0 & 0 \\ 2 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1.5 & 1 \\ 1 & 1.5 \end{bmatrix}$$

$$(b) \det \begin{vmatrix} 1.5 - \lambda & 1 \\ 1 & 1.5 - \lambda \end{vmatrix} = 0 \Rightarrow (1.5 - \lambda)^2 - 1 = 0$$

$$\lambda^2 - 3\lambda + \frac{5}{4} = 0; \Rightarrow \lambda_1 = 2.5; \lambda_2 = 0.5$$

$$(c) \text{when } \lambda_1 = 2.5; \vec{v}_1 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

$$\begin{bmatrix} 1.5 - 2.5 & 1 \\ 1 & 1.5 - 2.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\Rightarrow \vec{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow \vec{v}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\text{when } \lambda_2 = 0.5; \vec{v}_2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

$$\begin{bmatrix} 1.5 - 0.5 & 1 \\ 1 & 1.5 - 0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\Rightarrow \vec{v}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \Rightarrow \vec{v}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$(d) \text{after the reduction new } X = \frac{1}{\sqrt{2}} [1, 1] \times \begin{bmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{3}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & \frac{3}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

PCA codes

rand: returns an M-by-N random matrix

subplot: breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle.

```
clc;
close all;
clear all;
numdata=20;

%% step 1, generating a dataset
x1=rand(numdata/2,1);
y1=rand(numdata/2,1);
x2=3*rand(numdata/2,1)+3;
y2=3*rand(numdata/2,1)+3;
x=[x1;x2];
y=[y1;y2];
subplot(4,1,1);
plot(x,y, 'o');
title('Original Data');

%% step 2, finding a mean and subtracting
xmean=mean(x);
ymean=mean(y);
xnew=x-xmean*ones(numdata,1);
ynew=y-ymean*ones(numdata,1);
subplot(4,1,2);
plot(xnew,ynew, 'o');
title('mean is deducted')
```

PCA codes

cov: For matrices, where each row is an observation, and each column a variable, *cov*(X) is the covariance matrix.

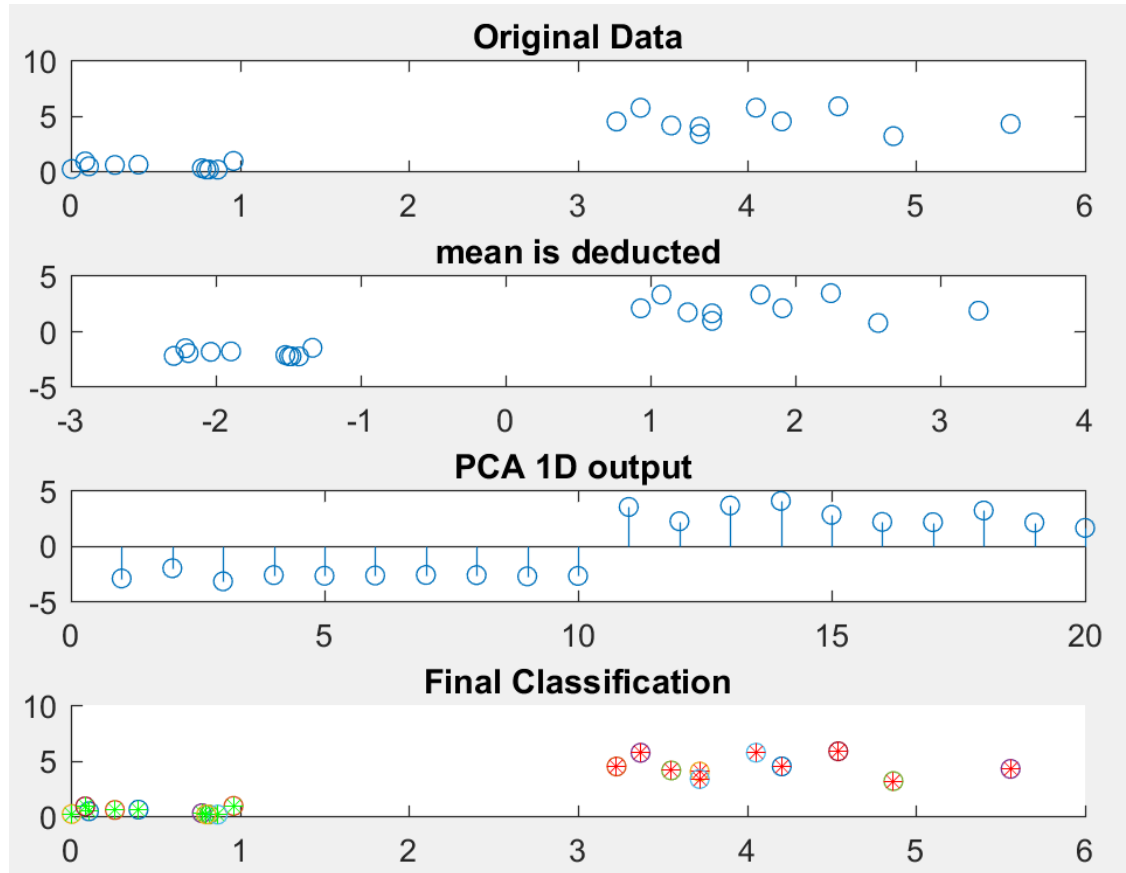
```
%% step 3, covariance matrix
covariancematrix=cov(xnew,ynew);
%% step 4, Finding Eigenvectors
[V,D] = eig(covariancematrix);
D=diag(D);
maxeigval=V(:,find(D==max(D)));
```

eig: produces a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors

```
%% step 5, Deriving the new data set
%finding the projection onto the eigenvectors
finaldata=maxeigval'*[xnew,ynew]';
subplot(4,1,3);
stem(finaldata, 'DisplayName', 'finaldata', 'YDataSource', 'finaldata');
title('PCA 1D output ')
%we do a classification now
subplot(4,1,4);
title('Final Classification')
hold on
for i=1:size(finaldata,2)
    if finaldata(i)>=0
        plot(x(i),y(i),'o')
        plot(x(i),y(i),'r*')

    else
        plot(x(i),y(i),'o')
        plot(x(i),y(i),'g*')
    end
end
end
```

PCA codes



Outline

- Introduction to face recognition
- Principal Component Analysis (PCA)
- **The Eigenfaces Algorithm**
- Linear Discriminant Analysis (LDA)

Eigen-face based face recognition

- Key ideas
 - Images in the original space are highly correlated
 - So, compress them to a low-dimensional subspace that captures key appearance characteristics of the visual DOFs
 - Use PCA for estimating the sub-space (dimensionality reduction)
 - Compare two faces by projecting the images into the subspace and measuring the Euclidean distance between them

Eigenface algorithm

■ Training

1. Align training images $x_1, x_2, \dots, x_N$



Note that each image is formulated into a long vector!

2. Compute average face $\mu = \frac{1}{N} \sum x_i$
3. Compute the difference image (the centered data matrix)

$$\begin{aligned} X_c &= \begin{bmatrix} | & & | \\ x_1 & \dots & x_n \\ | & & | \end{bmatrix} - \begin{bmatrix} | & & | \\ \mu & \dots & \mu \\ | & & | \end{bmatrix} \\ &= X - \mu 1^T = X - \frac{1}{n} X 1 1^T = X \left(I - \frac{1}{n} 1 1^T \right) \end{aligned}$$

Eigenface algorithm

■ Training

4. Compute the covariance matrix

$$\Sigma = \frac{1}{n} \begin{bmatrix} | & & | \\ x_1^c & \dots & x_n^c \\ | & & | \end{bmatrix} \begin{bmatrix} - & x_1^c & - \\ \vdots & & \\ - & x_n^c & - \end{bmatrix} = \frac{1}{n} X_c X_c^T$$

5. Compute the eigenvectors of the covariance matrix Σ
6. Compute each training image x_i 's projections as

$$x_i \rightarrow (x_i^c \cdot \phi_1, x_i^c \cdot \phi_2, \dots, x_i^c \cdot \phi_K) \equiv (a_1, a_2, \dots, a_K)$$

7. Visualize the estimated training face x_i

$$x_i \approx \mu + a_1 \phi_1 + a_2 \phi_2 + \dots + a_K \phi_K$$

Eigenface algorithm

■ Training



6. Compute each training image x_i 's projections as

$$x_i \rightarrow (x_i^c \cdot \phi_1, x_i^c \cdot \phi_2, \dots, x_i^c \cdot \phi_K) \equiv (a_1, a_2, \dots, a_K)$$

7. Visualize the estimated training face x_i

$$x_i \approx \mu + a_1\phi_1 + a_2\phi_2 + \dots + a_K\phi_K$$

Eigenface algorithm

- Testing
- 1. Take query image t
- 2. Project t into eigenface space and compute projection

$$t \rightarrow ((t - \mu) \cdot \phi_1, (t - \mu) \cdot \phi_2, \dots, (t - \mu) \cdot \phi_K) \equiv (w_1, w_2, \dots, w_K)$$

- Compare projection w with all N training projections
 - Simple comparison metric: Euclidean
 - Simple decision: K-Nearest Neighbor

(note: this “K” refers to the k-NN algorithm, is different from the previous K’s referring to the # of principal components)

Eigenface algorithm

- Eigenfaces look somewhat like generic faces



Reconstruction and Errors

- Only selecting the top P eigenfaces -> reduces the dimensionality.
- Fewer eigenfaces result in more information loss, and hence less discrimination between faces.

$P = 4$



$P = 200$



$P = 400$



Summary for Eigenface

- Pros

- Non-iterative, globally optimal solution

- Limitations

- PCA projection is optimal for reconstruction from a low dimensional basis, but may NOT be optimal for discrimination.

Code for face recognition

- Please run the uploaded codes “face recogniton”, and consider what’s the meaning of the function `slmetric_pw`, how did this function evaluate the performance.
- Please noted that this code is just for you to extend your understanding of PCA and LDA for face recognition. It will not be included in the quiz

Outline

- Introduction to face recognition
- Principal Component Analysis (PCA)
- The Eigenfaces Algorithm
- Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA)

- Fisher's Linear Discriminant (FLD)
- Eigenfaces **exploit the max scatter of the training images** in face space
- Fisherfaces attempt to **maximize the between class scatter, while minimizing the within class scatter.**
- Goal: find the best separation between two classes

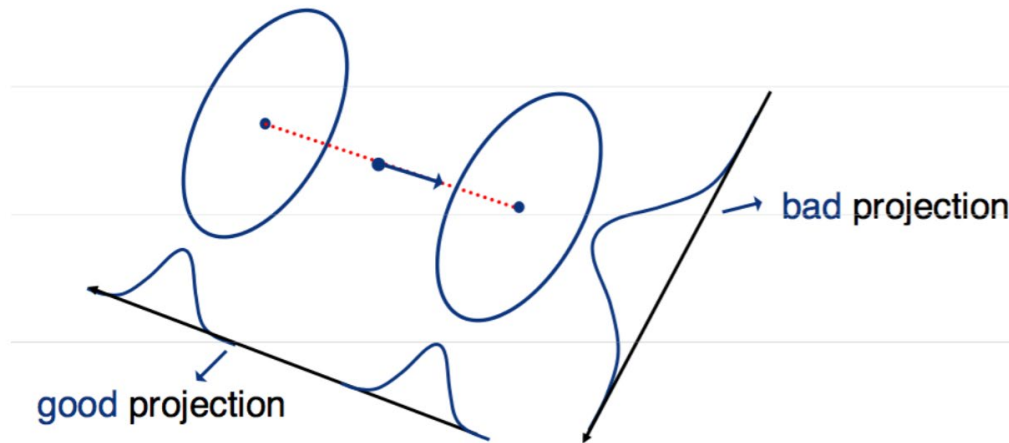
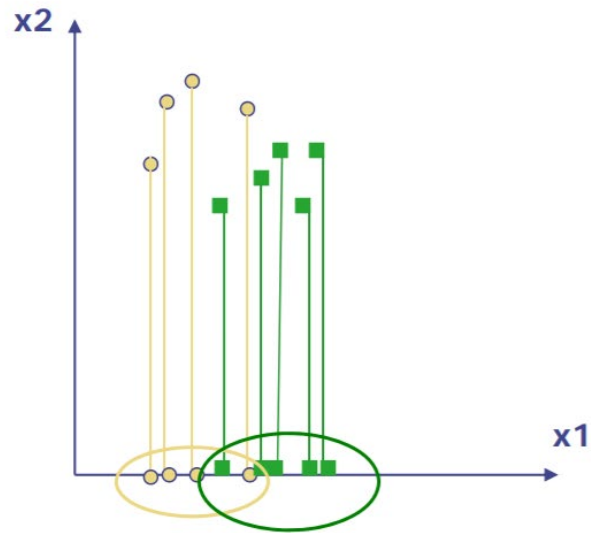
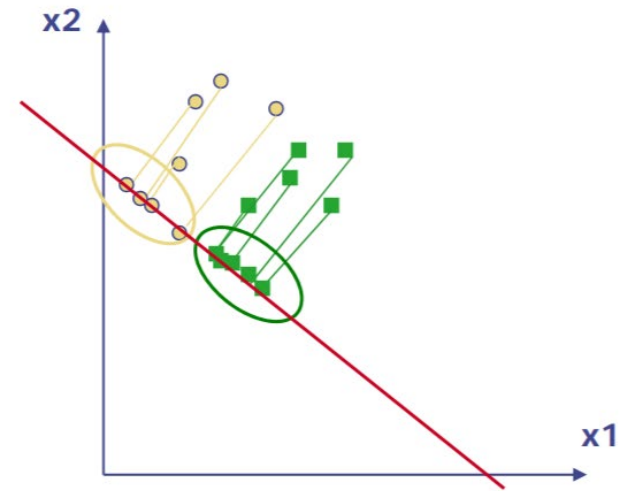


Illustration of the Projection

- Using two classes as example:



Poor Projection



Good

Variables

- N Sample images $\{x_1, \dots, x_N\}$
- c classes $\{\chi_1, \dots, \chi_c\}$
- Average of each class $\mu_i = \frac{1}{N_i} \sum_{x_k \in \chi_i} x_k$
- Total average: $\mu = \frac{1}{N} \sum_{k=1}^N x_k$

Scatters

- Scatter of class i:

$$S_i = \sum_{x_k \in \mathcal{X}_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

- Within class scatter:

$$S_W = \sum_{i=1}^c S_i$$

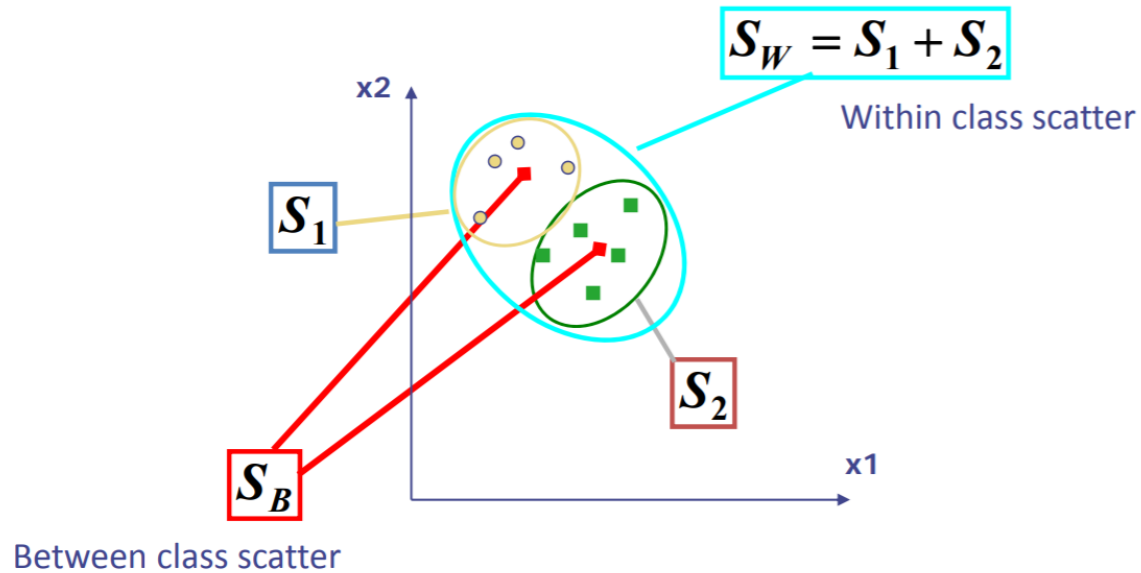
- Between class scatter:

$$S_B = \sum_{i=1}^c |\mathcal{X}_i| (\mu_i - \mu)(\mu_i - \mu)^T$$

- Total scatter:

$$S_T = S_W + S_B$$

LDA



$$S_i = \sum_{x_k \in \mathcal{X}_i} (x_k - \mu_i)(x_k - \mu_i)^T \quad S_W = \sum_{i=1}^c S_i \quad S_B = \sum_{i=1}^c |\mathcal{X}_i| (\mu_i - \mu)(\mu_i - \mu)^T$$

Mathematical Formulation (1)

- After projection:

$$y_k = W^T x_k$$

- Between class scatter (of y's):

$$\tilde{S}_B = W^T S_B W$$

- Within class scatter (of y's):

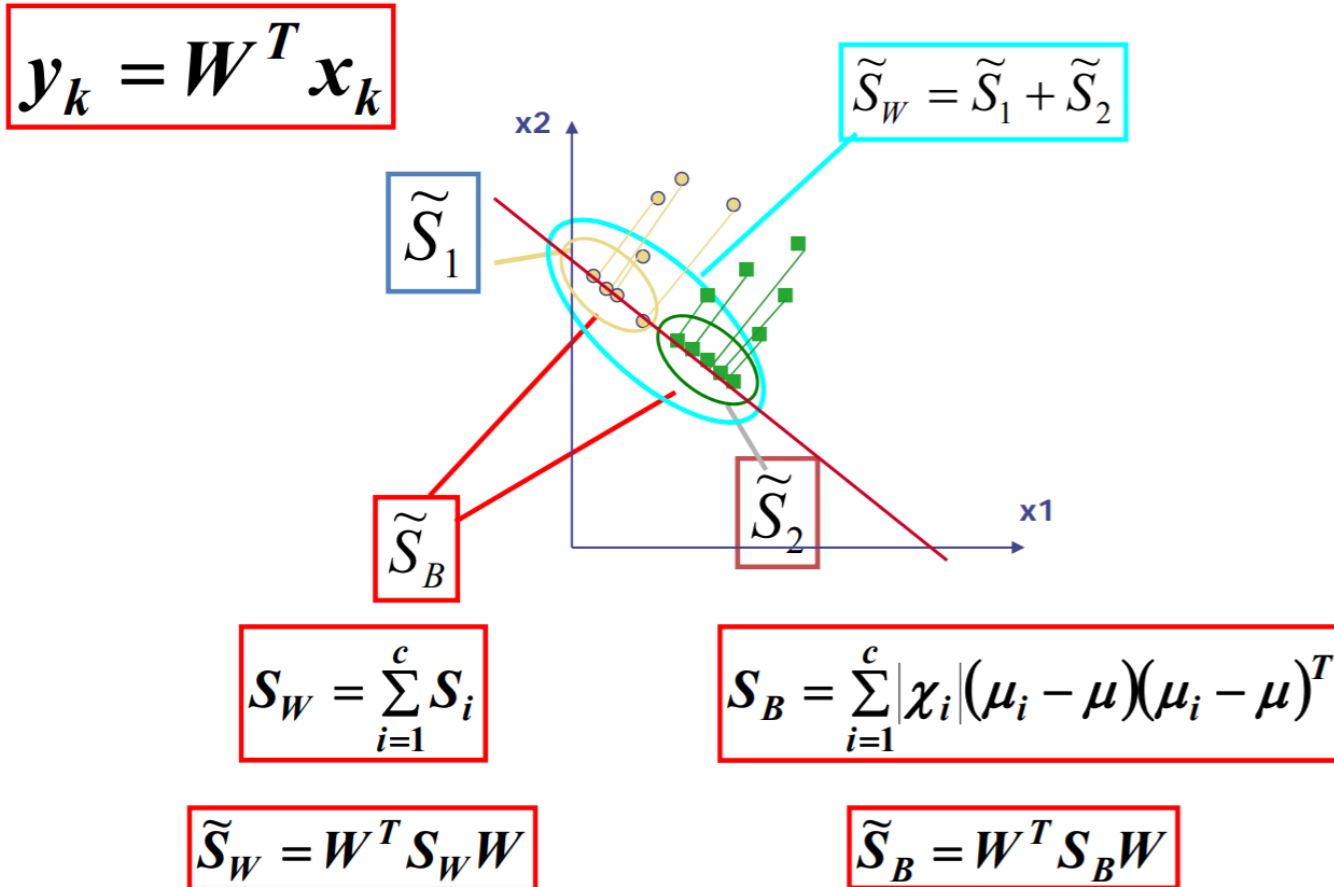
$$\tilde{S}_W = W^T S_W W$$

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

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

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

LDA



LDA

- Maximizing the ratio

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

- Is equivalent to maximizing the numerator while keeping the denominator constant, i.e.

$$\max_w w^T S_B w \quad \text{subject to} \quad w^T S_W w = K$$

- And can be accomplished using Lagrange multipliers, where we define the Lagrangian as

$$L = w^T S_B w - \lambda(w^T S_W w - K)$$

- And maximize with respect to both w and λ

LDA

- Setting the gradient of

$$L = w^T (S_B - \lambda S_W) w + \lambda K$$

- With respect to w to zeros we get

$$\nabla_w L = 2(S_B - \lambda S_W) w = 0$$

- or

$$S_B w = \lambda S_W w$$

- This is a generalized eigenvalue problem
- The solution is easy when $S_w^{-1} = (\Sigma_1 + \Sigma_0)^{-1}$ exists

LDA

- In this case

$$S_W^{-1} S_B w = \lambda w$$

- And using the definition of S_B

$$S_W^{-1} (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T w = \lambda w$$

- Noting that $(\mu_2 - \mu_1)^T w = \alpha$ is a scalar this can be written as

$$S_W^{-1} (\mu_2 - \mu_1) = \frac{\lambda}{\alpha} w$$

- and since we don't care about the magnitude of w

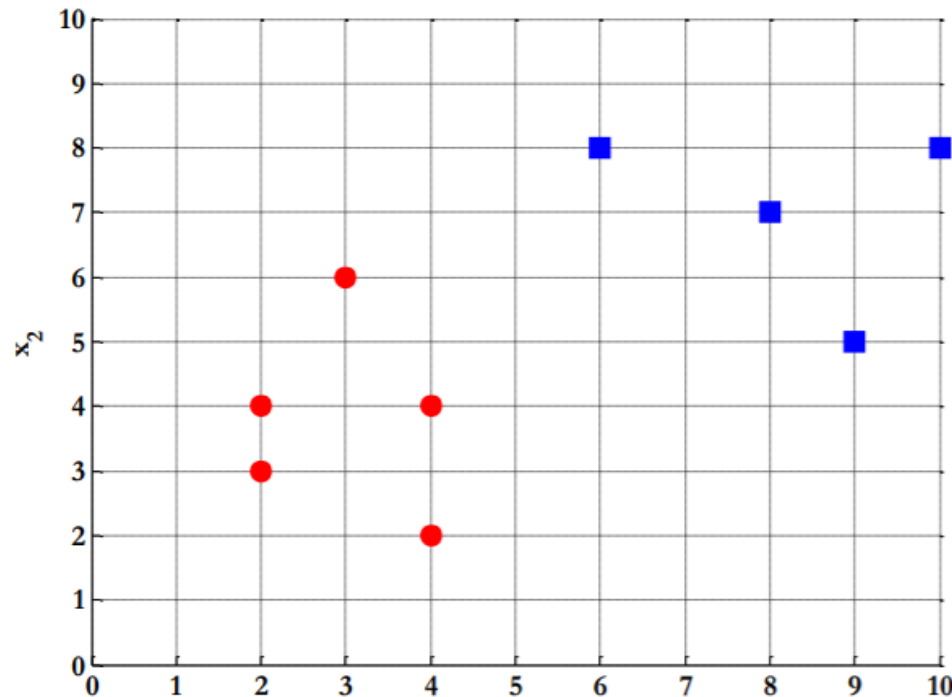
$$w^* = S_W^{-1} (\mu_2 - \mu_1) = (\Sigma_2 + \Sigma_1)^{-1} (\mu_2 - \mu_1)$$

LDA example

- Compute the Linear Discriminant projection for the following two dimensional dataset

Samples for class ω_1 : $\mathbf{X}_1=(x_1,x_2)=\{(4,2),(2,4),(2,3),(3,6),(4,4)\}$

Sample for class ω_2 : $\mathbf{X}_2=(x_1,x_2)=\{(9,10),(6,8),(9,5),(8,7),(10,8)\}$



```
% samples for class 1
X1 = [4,2;
      2,4;
      2,3;
      3,6;
      4,4];

% samples for class 2
X2 = [9,10;
      6,8;
      9,5;
      8,7;
      10,8];
```

LDA example

- The classes mean are:

$$\mu_1 = \frac{1}{N_1} \sum_{x \in \omega_1} x = \frac{1}{5} \left[\begin{pmatrix} 4 \\ 2 \end{pmatrix} + \begin{pmatrix} 2 \\ 4 \end{pmatrix} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 6 \end{pmatrix} + \begin{pmatrix} 4 \\ 4 \end{pmatrix} \right] = \begin{pmatrix} 3 \\ 3.8 \end{pmatrix}$$

$$\mu_2 = \frac{1}{N_2} \sum_{x \in \omega_2} x = \frac{1}{5} \left[\begin{pmatrix} 9 \\ 10 \end{pmatrix} + \begin{pmatrix} 6 \\ 8 \end{pmatrix} + \begin{pmatrix} 9 \\ 5 \end{pmatrix} + \begin{pmatrix} 8 \\ 7 \end{pmatrix} + \begin{pmatrix} 10 \\ 8 \end{pmatrix} \right] = \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix}$$

```
% class means  
Mu1 = mean(X1) ' ;  
Mu2 = mean(X2) ' ;
```

LDA example

- Covariance matrix of the first class:

$$\begin{aligned} S_1 &= \sum_{x \in \omega_1} (x - \mu_1)(x - \mu_1)^T = \left[\begin{pmatrix} 4 \\ 2 \end{pmatrix} - \begin{pmatrix} 3 \\ 3.8 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 2 \\ 4 \end{pmatrix} - \begin{pmatrix} 3 \\ 3.8 \end{pmatrix} \right]^2 \\ &\quad + \left[\begin{pmatrix} 2 \\ 3 \end{pmatrix} - \begin{pmatrix} 3 \\ 3.8 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 3 \\ 6 \end{pmatrix} - \begin{pmatrix} 3 \\ 3.8 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 4 \\ 4 \end{pmatrix} - \begin{pmatrix} 3 \\ 3.8 \end{pmatrix} \right]^2 \\ &= \begin{pmatrix} 1 & -0.25 \\ -0.25 & 2.2 \end{pmatrix} \end{aligned}$$

```
% covariance matrix of the first class  
S1 = cov(X1);
```

LDA example

- Covariance matrix of the second class:

$$\begin{aligned} S_2 &= \sum_{x \in \omega_2} (x - \mu_2)(x - \mu_2)^T = \left[\begin{pmatrix} 9 \\ 10 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 6 \\ 8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right]^2 \\ &\quad + \left[\begin{pmatrix} 9 \\ 5 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 8 \\ 7 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right]^2 + \left[\begin{pmatrix} 10 \\ 8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right]^2 \\ &= \begin{pmatrix} 2.3 & -0.05 \\ -0.05 & 3.3 \end{pmatrix} \end{aligned}$$

```
% covariance matrix of the first class  
S2 = cov(X2);
```

LDA example

- Within-class scatter matrix:

$$S_w = S_1 + S_2 = \begin{pmatrix} 1 & -0.25 \\ -0.25 & 2.2 \end{pmatrix} + \begin{pmatrix} 2.3 & -0.05 \\ -0.05 & 3.3 \end{pmatrix} \\ = \begin{pmatrix} 3.3 & -0.3 \\ -0.3 & 5.5 \end{pmatrix}$$

```
% within-class scatter matrix  
Sw = S1 + S2 ;
```

LDA example

- Between-class scatter matrix:

$$\begin{aligned} S_B &= (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \\ &= \left[\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right] \left[\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right]^T \\ &= \begin{pmatrix} -5.4 \\ -3.8 \end{pmatrix} \begin{pmatrix} -5.4 & -3.8 \end{pmatrix} \\ &= \begin{pmatrix} 29.16 & 20.52 \\ 20.52 & 14.44 \end{pmatrix} \end{aligned}$$

```
% between-class scatter matrix  
SB = (Mu1-Mu2) * (Mu1-Mu2) ' ;
```

LDA example

- The LDA projection is then obtained as the solution of the generalized eigen value problem

$$S_W^{-1}S_B w = \lambda w$$

$$\Rightarrow |S_W^{-1}S_B - \lambda I| = 0$$

$$\Rightarrow \left| \begin{pmatrix} 3.3 & -0.3 \\ -0.3 & 5.5 \end{pmatrix}^{-1} \begin{pmatrix} 29.16 & 20.52 \\ 20.52 & 14.44 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right| = 0$$

$$\Rightarrow \left| \begin{pmatrix} 0.3045 & 0.0166 \\ 0.0166 & 0.1827 \end{pmatrix} \begin{pmatrix} 29.16 & 20.52 \\ 20.52 & 14.44 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right| = 0$$

$$\Rightarrow \left| \begin{pmatrix} 9.2213 - \lambda & 6.489 \\ 4.2339 & 2.9794 - \lambda \end{pmatrix} \right|$$

$$= (9.2213 - \lambda)(2.9794 - \lambda) - 6.489 \times 4.2339 = 0$$

$$\Rightarrow \lambda^2 - 12.2007\lambda = 0 \Rightarrow \lambda(\lambda - 12.2007) = 0$$

$$\Rightarrow \lambda_1 = 0, \lambda_2 = 12.2007$$

LDA example

- Hence

$$\begin{pmatrix} 9.2213 & 6.489 \\ 4.2339 & 2.9794 \end{pmatrix} w_1 = 0 \underbrace{\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}}_{\lambda_1}$$

and

$$\begin{pmatrix} 9.2213 & 6.489 \\ 4.2339 & 2.9794 \end{pmatrix} w_2 = \underbrace{12.2007}_{\lambda_2} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

Thus;

$$w_1 = \begin{pmatrix} -0.5755 \\ 0.8178 \end{pmatrix} \quad \text{and} \quad w_2 = \begin{pmatrix} 0.9088 \\ 0.4173 \end{pmatrix} = w^*$$

```
% computing the LDA projection
invSw = inv(Sw);

invSw_by_SB = invSw * SB;

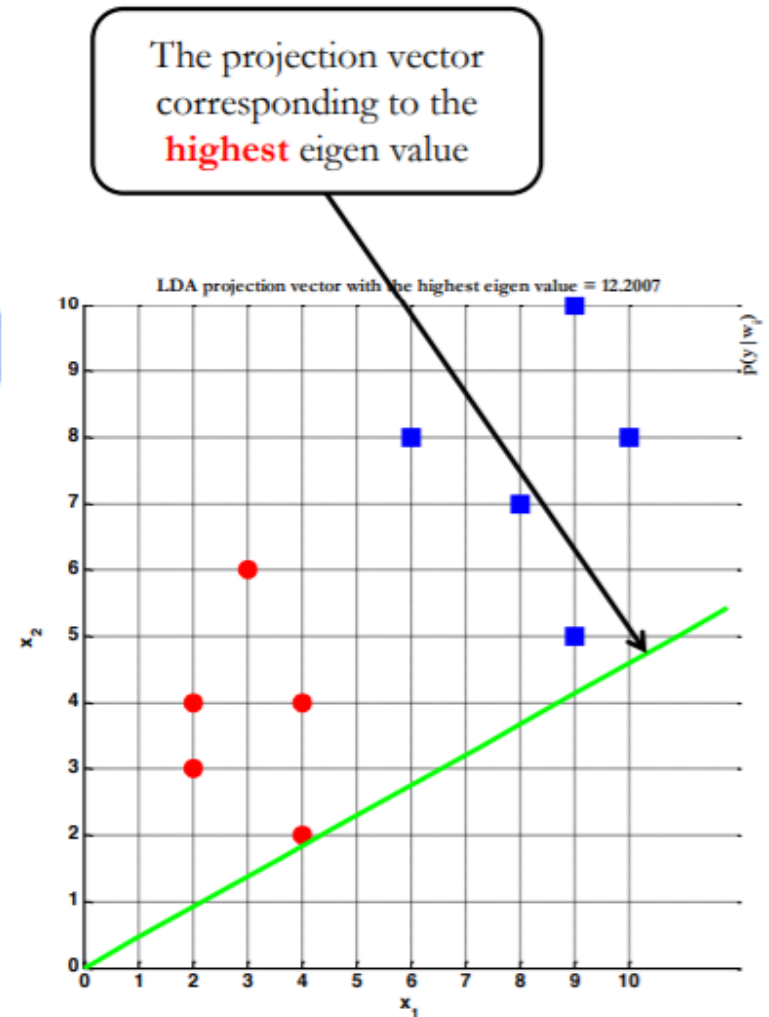
% getting the projection vector
[V,D] = eig(invSw_by_SB)

% the projection vector
W = V(:,1);
```

LDA example

- Or directly;

$$\begin{aligned} w^* &= S_W^{-1}(\mu_1 - \mu_2) = \begin{pmatrix} 3.3 & -0.3 \\ -0.3 & 5.5 \end{pmatrix}^{-1} \left[\begin{pmatrix} 3 \\ 3.8 \end{pmatrix} - \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix} \right] \\ &= \begin{pmatrix} 0.3045 & 0.0166 \\ 0.0166 & 0.1827 \end{pmatrix} \begin{pmatrix} -5.4 \\ -3.8 \end{pmatrix} \\ &= \begin{pmatrix} 0.9088 \\ 0.4173 \end{pmatrix} \end{aligned}$$

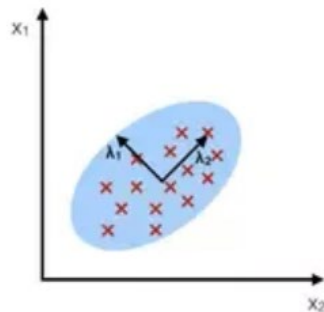


PCA vs. LDA

- Eigenfaces exploit the **max scatter** of the training images in face space
- Fisherfaces attempt to **maximize the between class scatter**, while **minimizing the within class scatter**.

PCA:

component axes that maximize the variance



LDA:

maximizing the component axes for class-separation

