

EE 4211 Computer Vision

Lecture 10: Face recognition

Semester A, 2020-2021

Schedules

Week	Date	Topics
1	Sep. 4	Introduction/Imaging
2	Sep. 11	Image enhancement in spatial domain
3	Sep. 18	Image enhancement in frequency domain (HW1 out)
4	Sep. 25	Morphological processing
5	Oct. 2	Image restoration(HW1 due)
6	Oct. 9	Image restoration
7	Oct. 16	Midterm (no tutorials this week)
8	Oct. 23	Edge detection (HW2 out, illustrate the project)
9	Oct. 30	Image segmentation (HW2 due)
10	Nov. 6	Face recognition with PCA, LDA (tutorial on deep learning framework)
11	Nov. 13	Face recognition based on deep learning Image segmentation based on deep learning (tutorial on coding)
12	Nov. 20	Object detection with traditional methods (Quiz) Object detection based on deep learning
13	Nov. 27	Project presentation and summary
14	Dec. 4	Review and Summary

Midterm

Ⓜ Average Score

58%

Ⓢ High Score

85%

Ⓣ Low Score

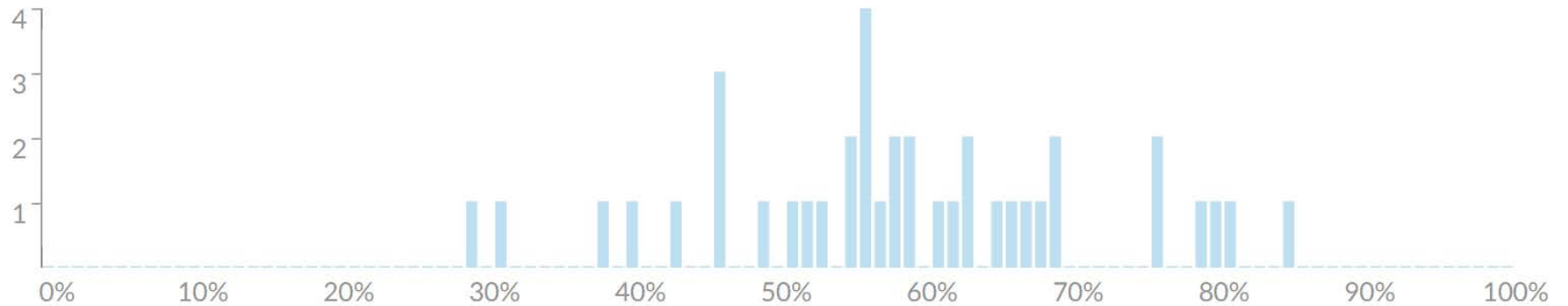
29%

Ⓢ Standard Deviation

12.98

⌚ Average Time

02:11:43



Quiz

- Code
- Open book and open notes
- 12:00 PM-13:00 PM on Nov. 20 (close at 13:10 PM, 10 mins for submission)
- Submit online through canvas, no late submission is allowed
- Please open the camera
- Practice with 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?



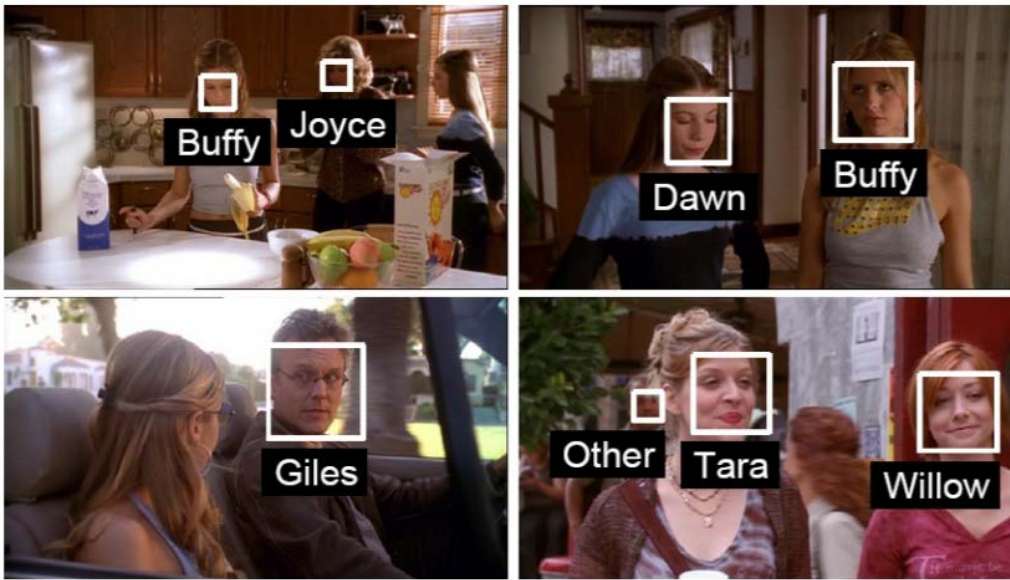
Face Recognition

- Digital photography
- Surveillance
- Album organization

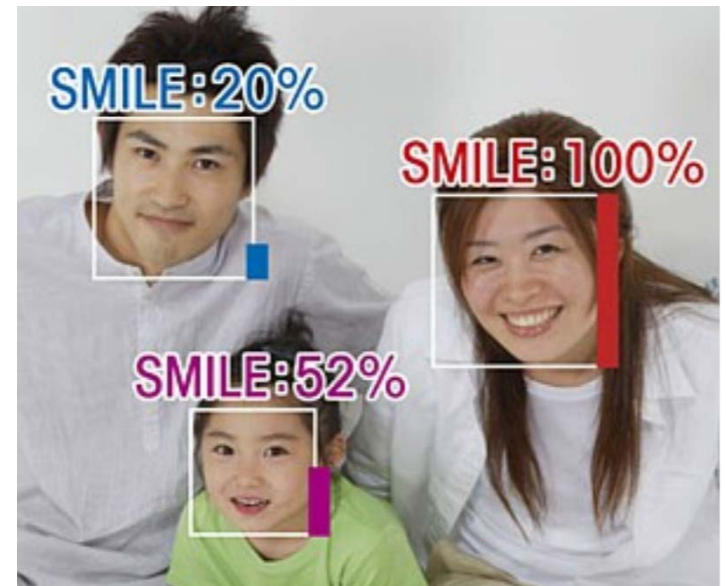


Face Recognition

Person tracking/id.

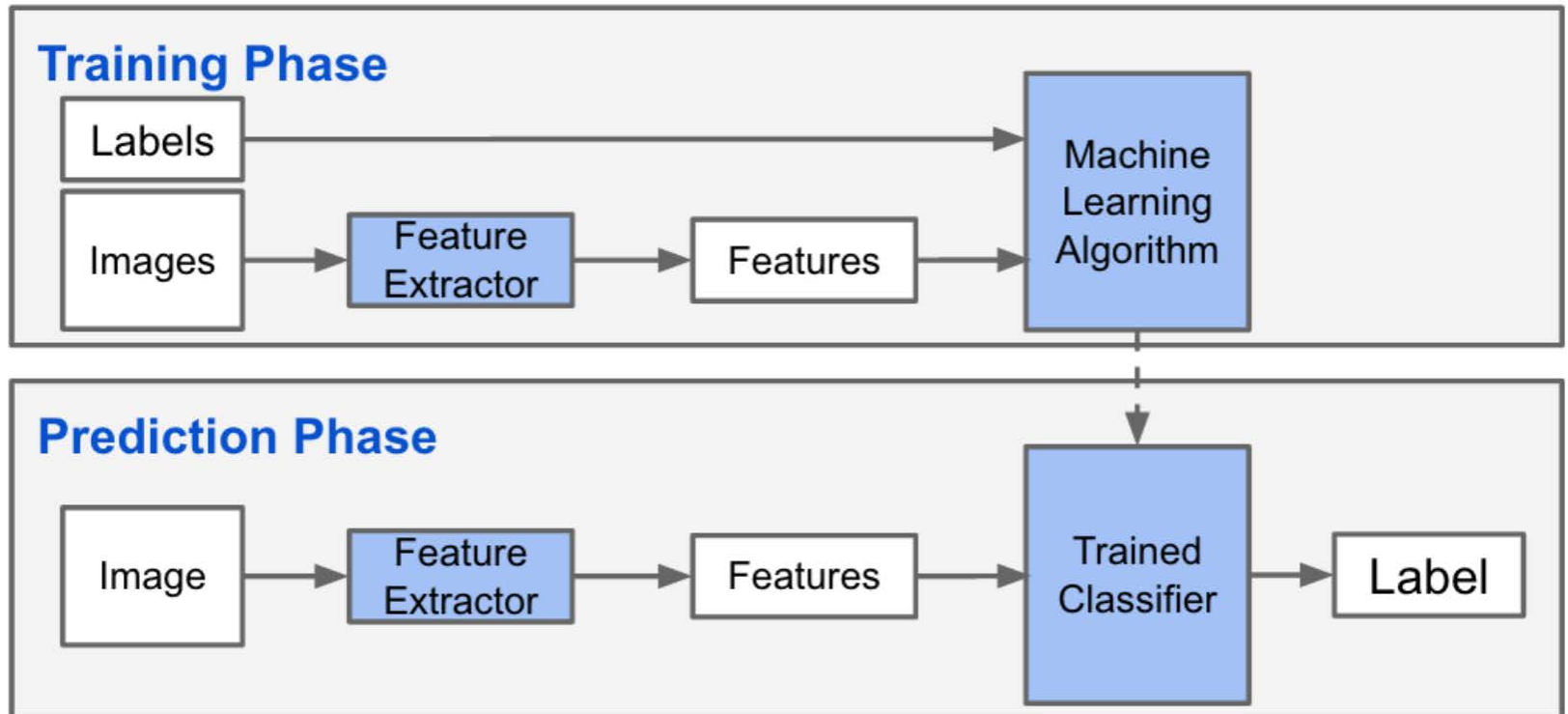


Emotions and expressions



Face Recognition

- Supervised learning in traditional machine learning



Face Recognition

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

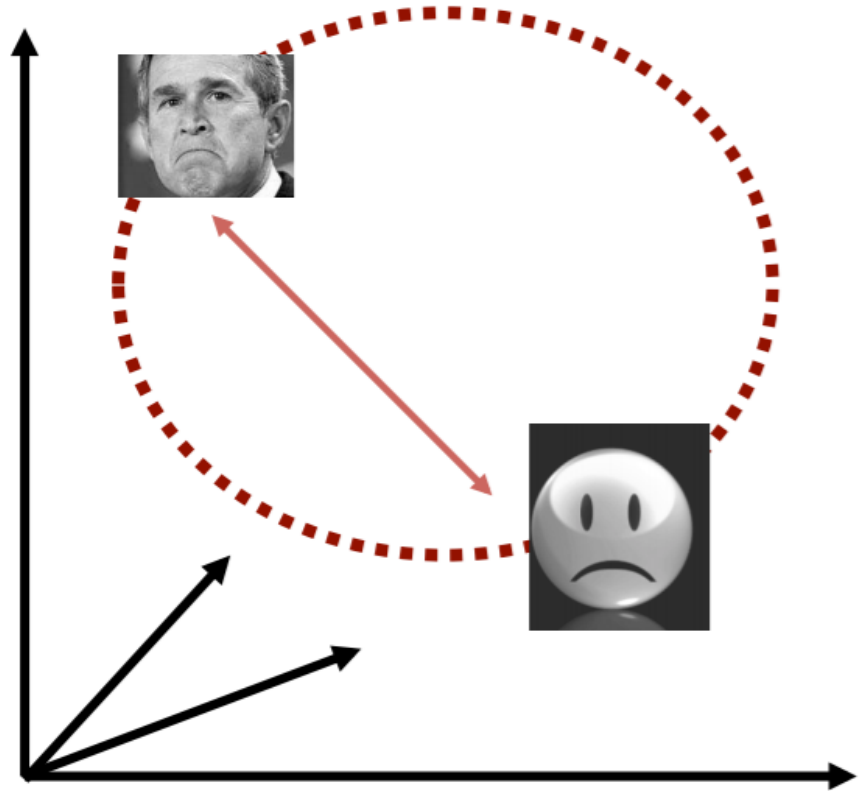


Traditional Machine Learning Flow



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. 100x100 image = 10,000 dim



Key Idea

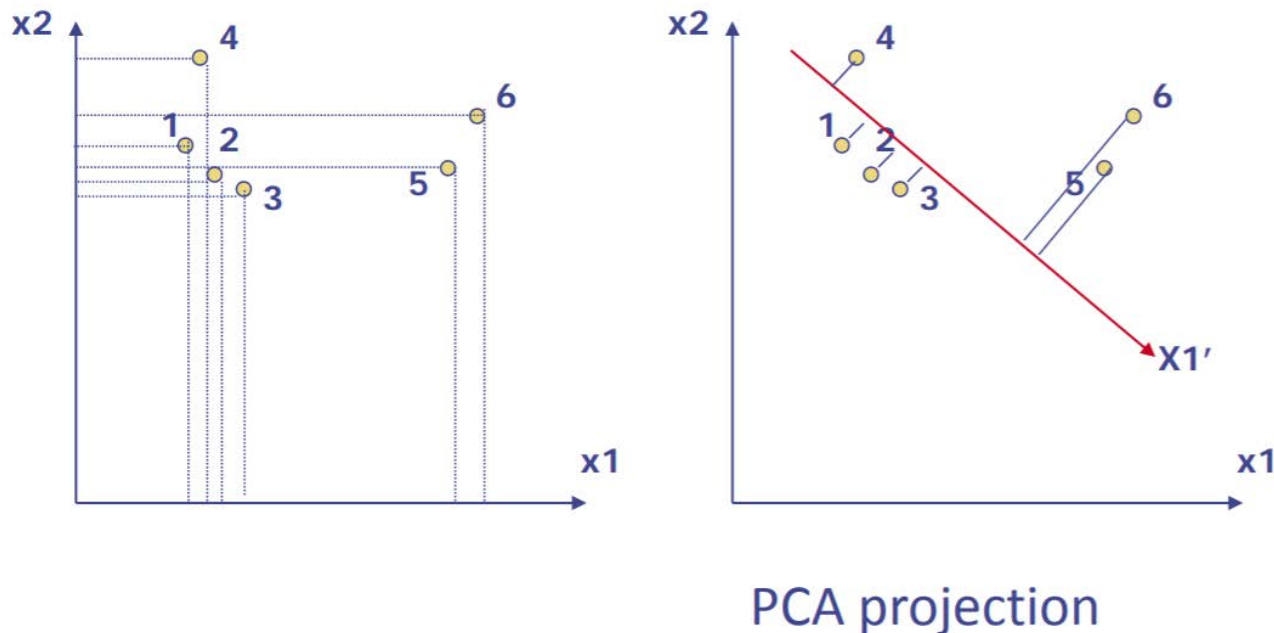
- So, compress them to a low-dimensional **subspace** that captures key appearance characteristics of the visual Degree of Freedoms.
- USE **PCA for estimating the sub-space** (dimensionality reduction)
- Compare two faces by projecting the images into the subspace and measuring the EUCLIDEAN distance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

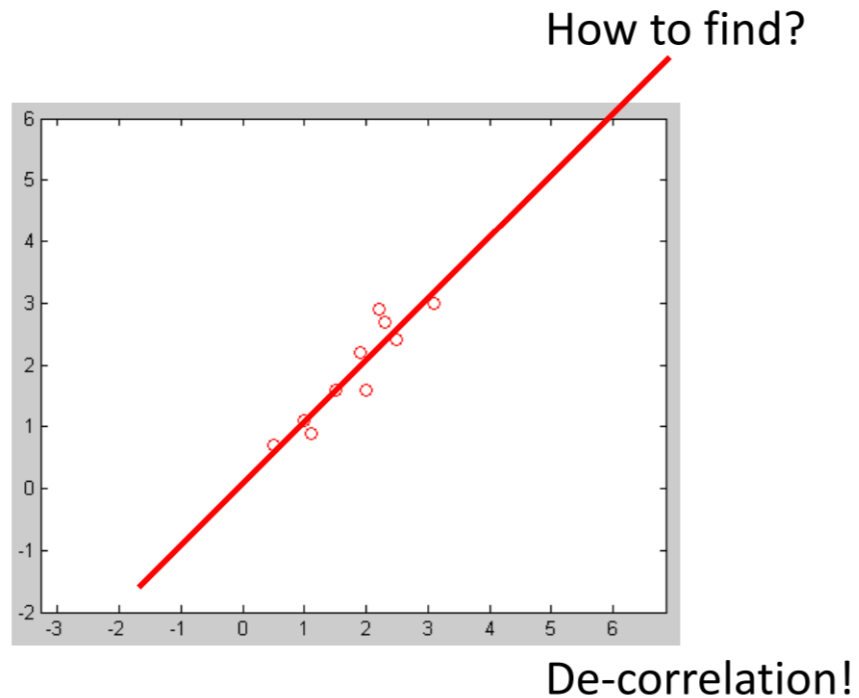
- PCA: converts a set of observations of possibly **correlated variables** into a set of values of linearly **uncorrelated variables** called principal components
- Computes n-dim subspace such that the projection of the data points onto the subspace has the **largest variance** among all n-dim subspaces.



PCA

■ 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

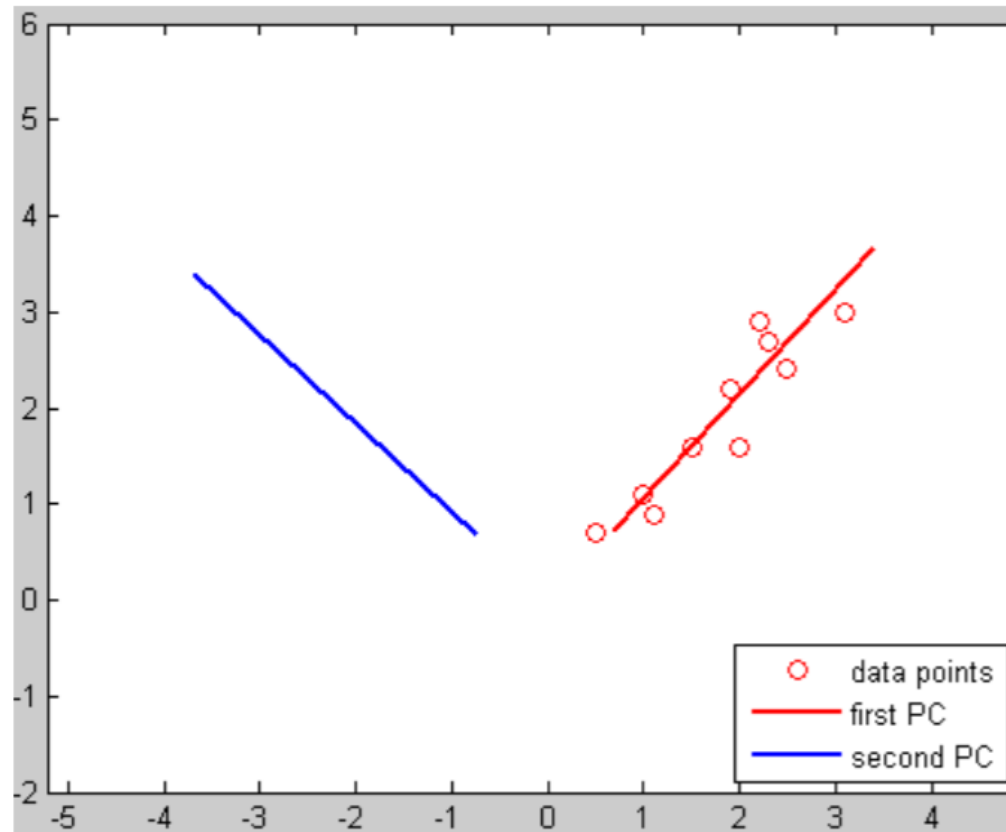
■ Illustration

Coordinates of the data points in the new coordinate system

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

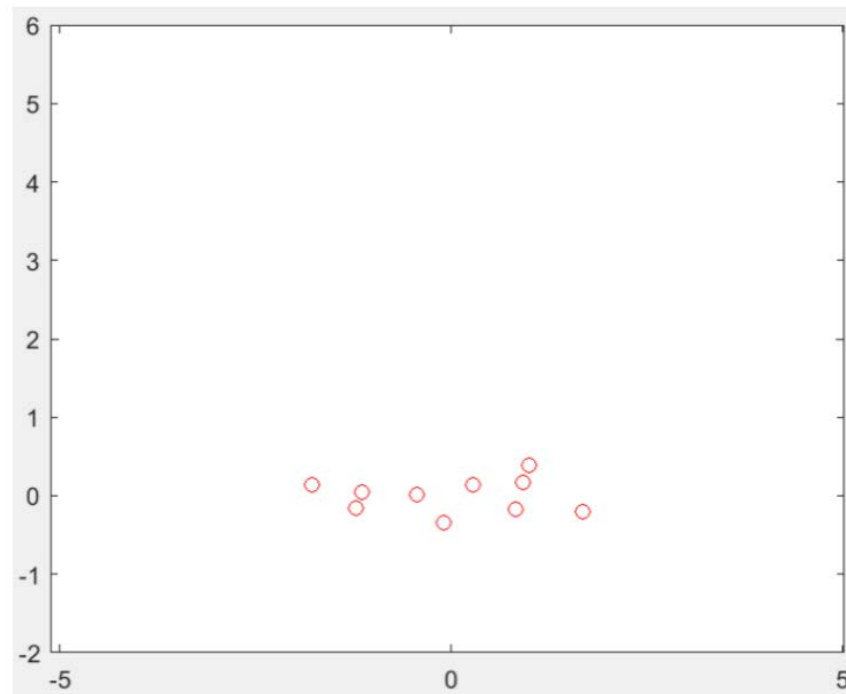
PCA

- 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

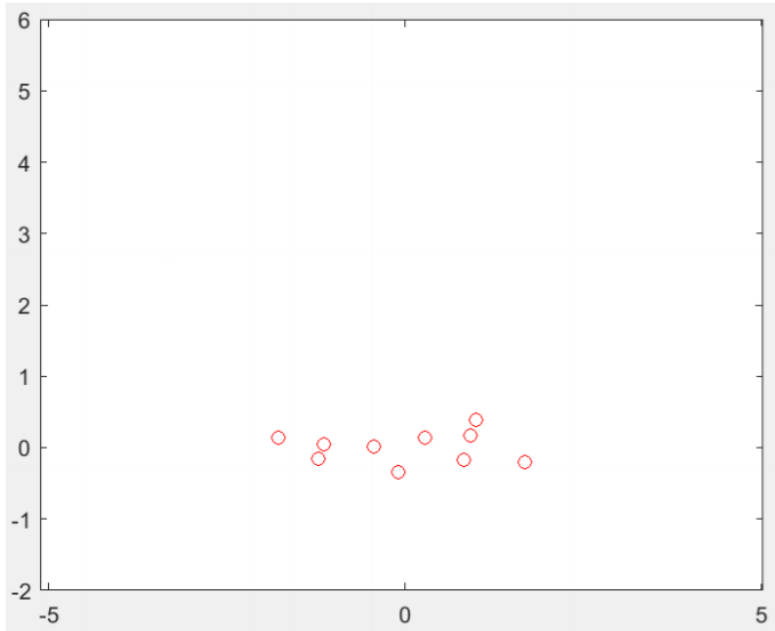
Coordinates of the data points in the new coordinate system

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

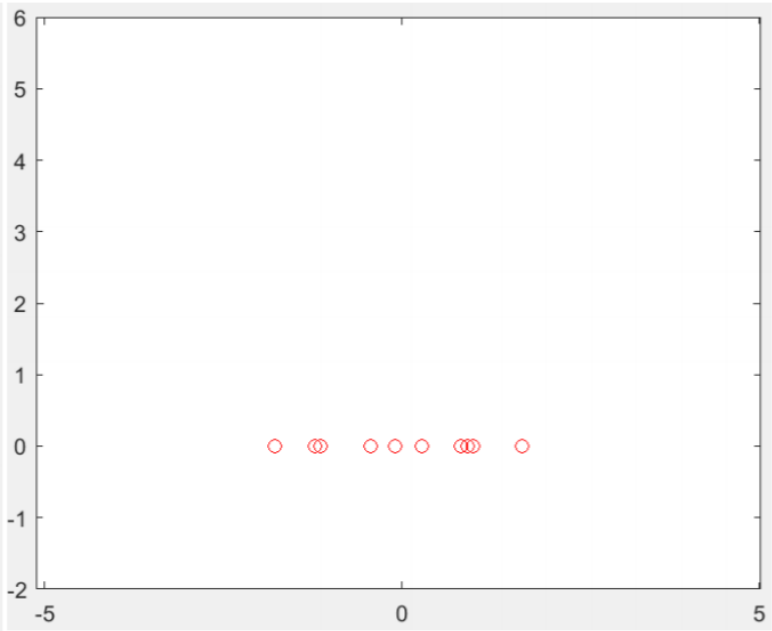
If only the first PC (corresponds to the largest eigen-value) is remained

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

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

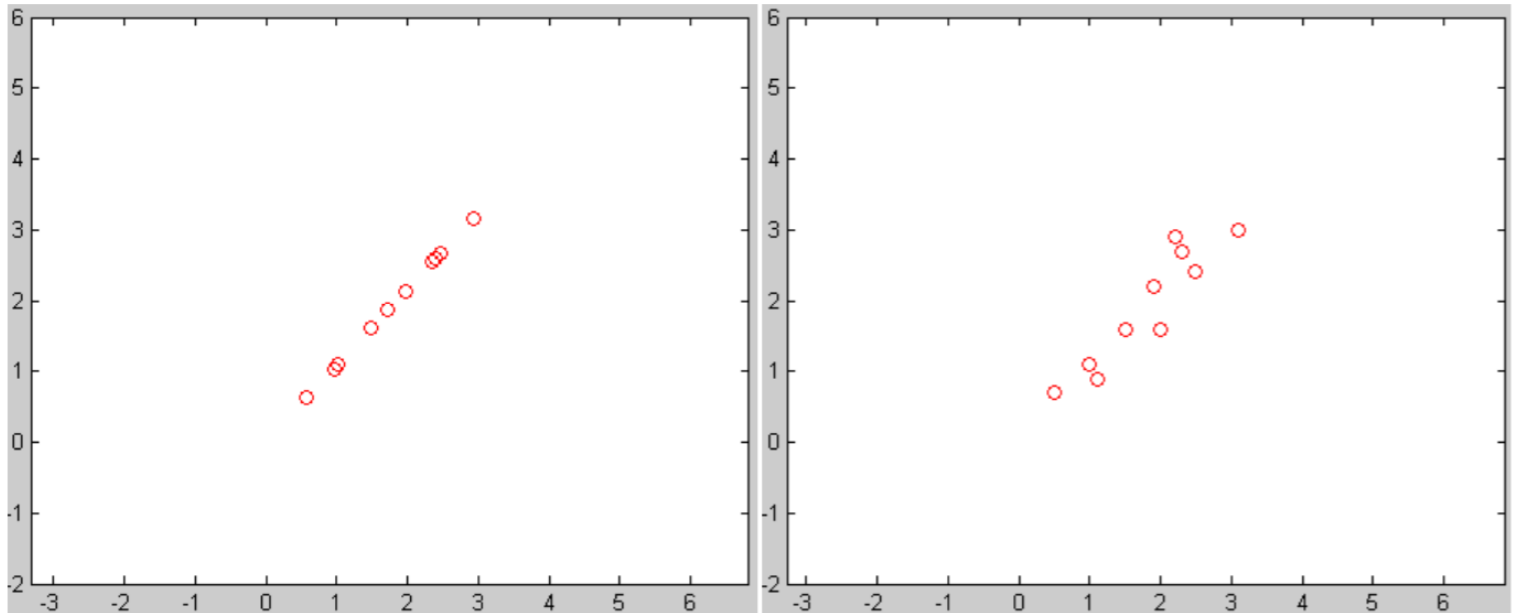
$$\begin{aligned} newC &= (0.6779 \quad 0.7352) \mathbf{X} \\ &= (3.459 \quad 0.854 \quad 3.623 \quad 2.905 \quad 4.307 \quad 3.544 \quad 2.532 \quad 1.487 \quad 2.193 \quad 1.407) \end{aligned}$$

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

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

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

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 $\rightarrow$ 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.

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

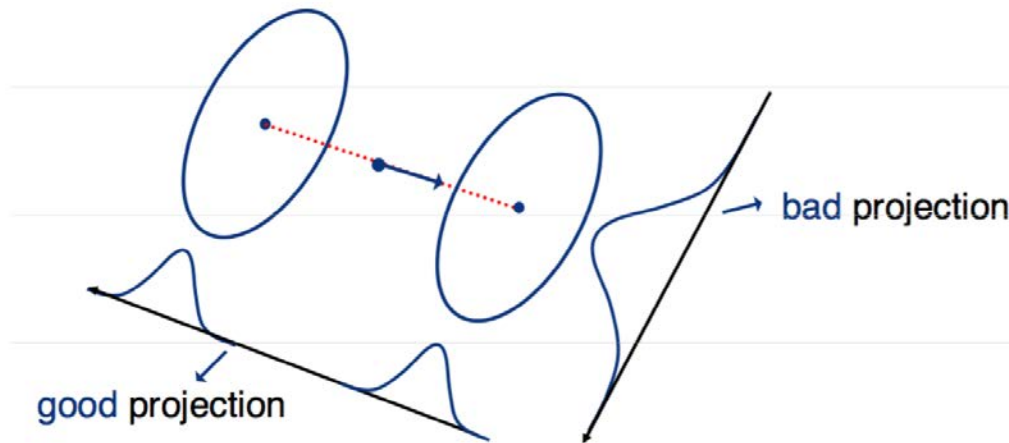
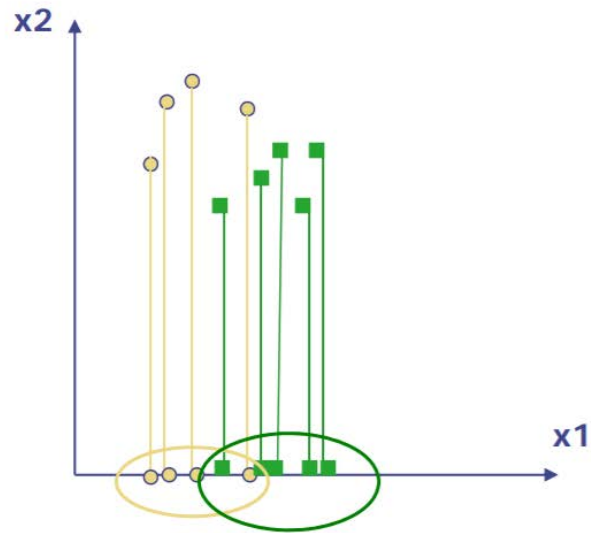
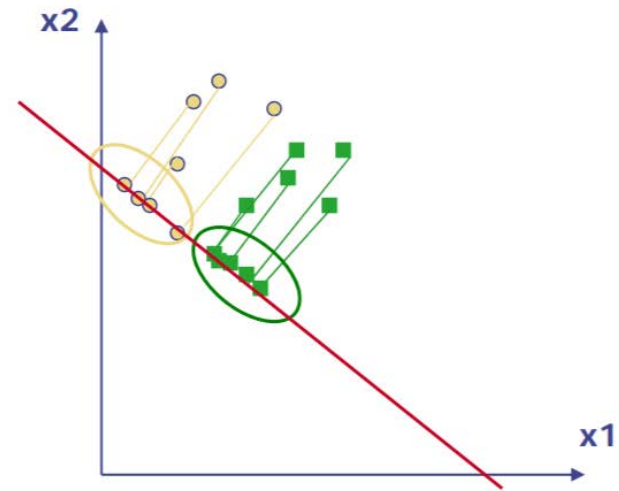


Illustration of the Projection

- 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:

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

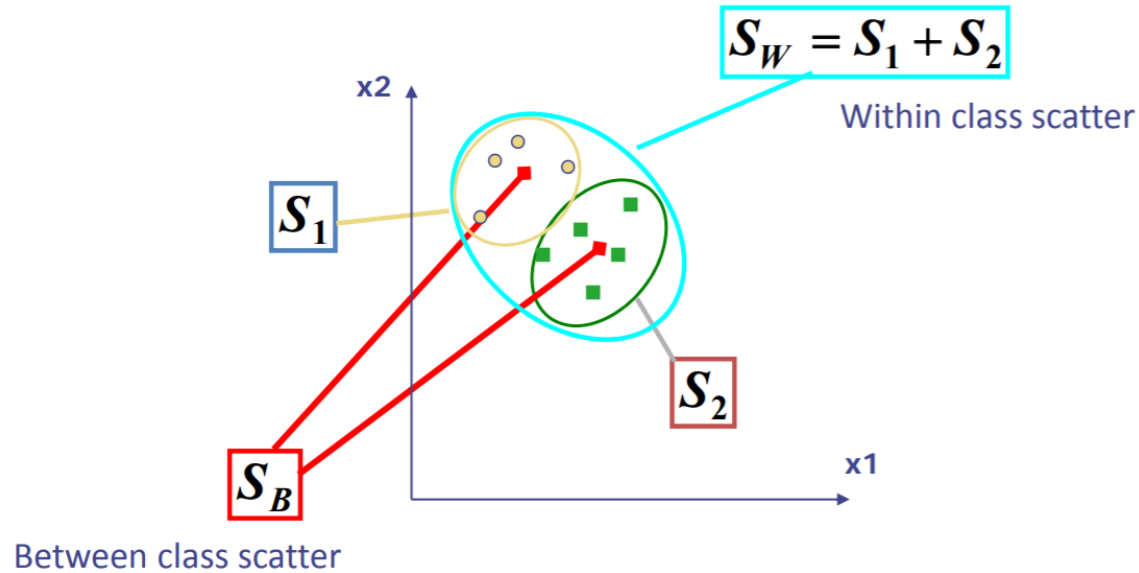
- Between class scatter:

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

- Total scatter:

$$S_T = S_W + S_B$$

LDA



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

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

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

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):

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

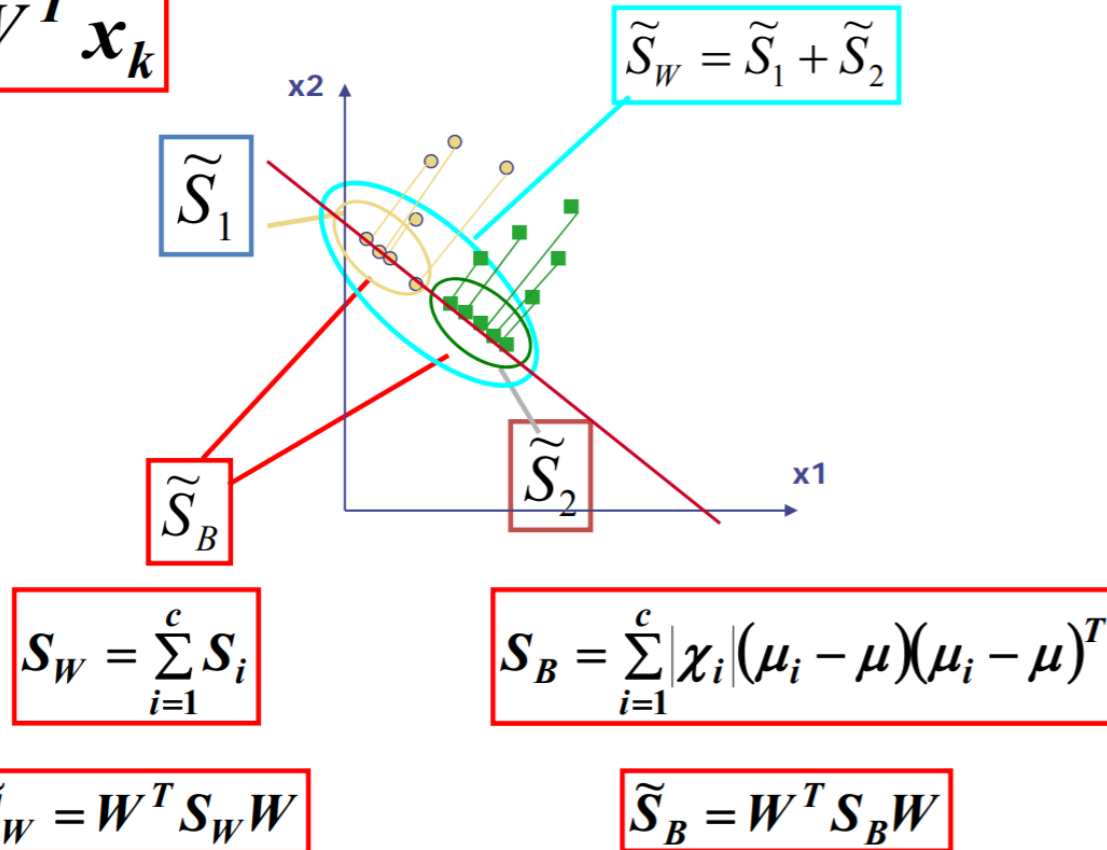
$$(\tilde{\mu}_1 - \tilde{\mu}_2)^2 = (w^T \mu_1 - w^T \mu_2)^2 = w^T \underbrace{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T}_{S_B} w = w^T S_B w$$

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

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

LDA

$$\mathbf{y}_k = \mathbf{W}^T \mathbf{x}_k$$



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

$$S_W^{-1} (\mu_1 - \mu_0) (\mu_1 - \mu_0)^T w = \lambda w$$

- Noting that $(\mu_1 - \mu_0)^T w = \alpha$ is a scalar this can be written as

$$S_W^{-1} (\mu_1 - \mu_0) = \frac{\lambda}{\alpha} w$$

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

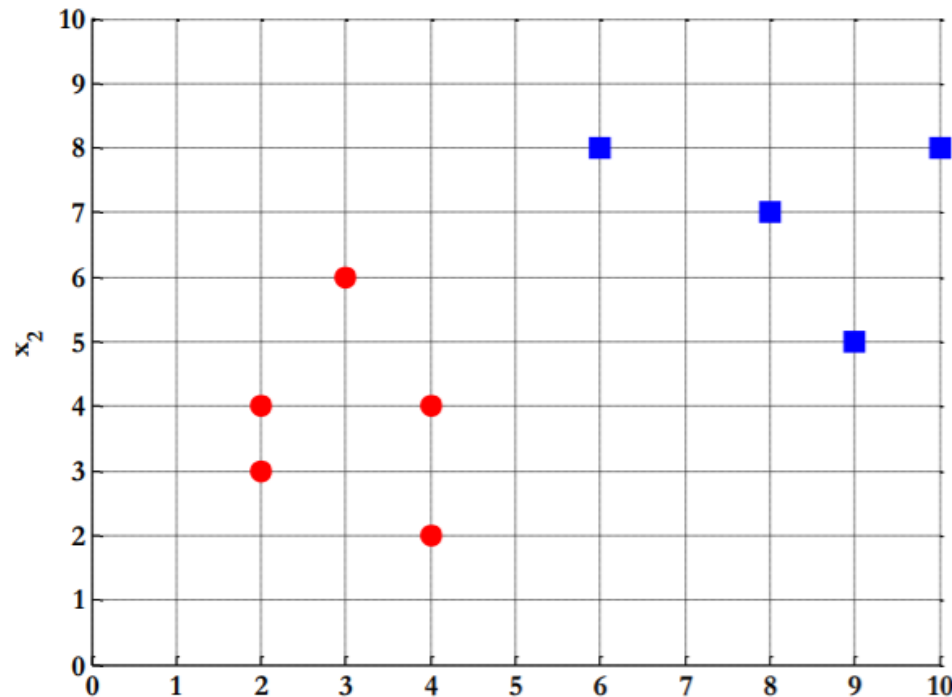
$$w^* = S_W^{-1} (\mu_1 - \mu_0) = (\Sigma_1 + \Sigma_0)^{-1} (\mu_1 - \mu_0)$$

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:

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

```
% 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

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

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;

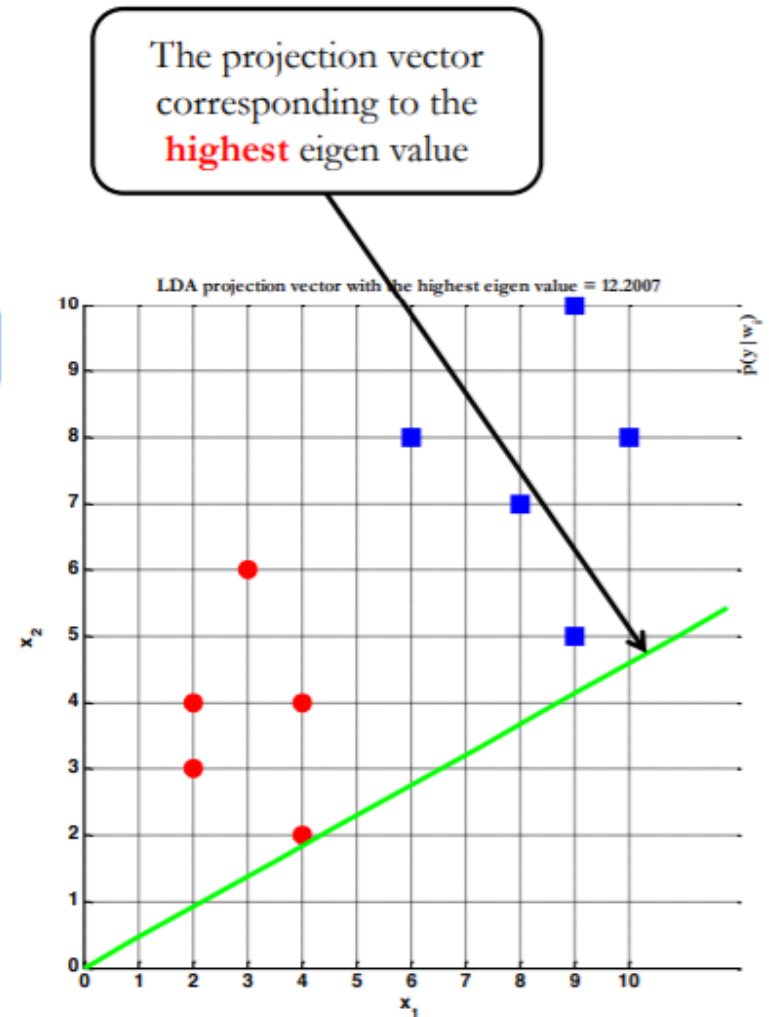
% getting the projection vector
[V,D] = eig(invSw_by_SB)

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

LDA example

- Or directly;

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

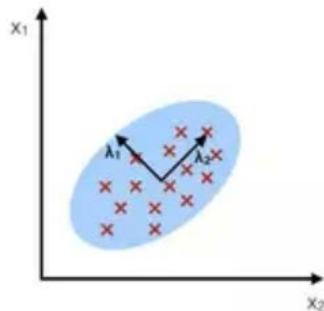


PCA vs. LDA

- 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**.

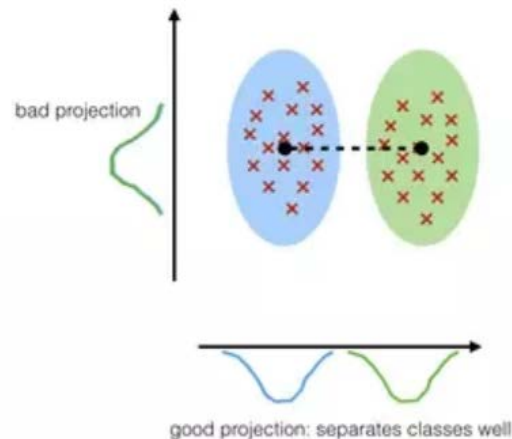
PCA:

component axes that maximize the variance



LDA:

maximizing the component axes for class-separation



Q1

- Write a MATLAB program as following instructions:
- input a gray-scale image '1.tif'.
- apply canny filter on the image
- define a threshold as the 10% of the maximum of the gradient magnitude. Then for each pixel, if the gradient magnitude is smaller than the threshold, change the gradient magnitude to zero. Show corresponding image.
- Utilize the Hough transform to detect the two lines and process edge linking.
- Note: Utilize subplot to plot 4 images, including original image, the processed gradient magnitude, hough transform with largest two points, detected two lines in the original images.
- Note: you need to provide sufficient comments on the codes.

```

%% 1
close all;
clear all;
clc;
I = imread('1.tif');
figure;
subplot(1,4,1);
imshow(I); title('Original'); %(1 point)

BW = edge(I, 'canny', 0.1); %canny detection;%(3 points)
subplot(1,4,2);
imshow(BW); title('Gradient magnitude');

[H,T,R] = hough(BW); %H hough tranform matrix,I,R angle and radius %(2 points)
subplot(1,4,3);
imshow(H, [], 'XData', T, 'YData', R, 'InitialMagnification', 'fit'); %hough transform
xlabel('\theta'), ylabel('\rho'); title('Hough transform');
axis on, axis square, hold on;
P = houghpeaks(H, 2); %(2 points)
x = T(P(:,2));
y = R(P(:,1));
plot(x,y, 's', 'color', 'white');
lines=houghlines(BW,T,R,P);
subplot(1,4,4);
imshow(I), title('Image After Hough transform'); hold on;
for k = 1:length(lines)
xy = [lines(k).point1; lines(k).point2];
plot(xy(:,1),xy(:,2), 'LineWidth', 2, 'Color', 'green');
plot(xy(1,1),xy(1,2), 'x', 'LineWidth', 2, 'Color', 'yellow');
plot(xy(2,1),xy(2,2), 'x', 'LineWidth', 2, 'Color', 'red');
end % rest for 2 points

```

Q2

- input a gray-scale image '2.tif'
- Add pepper noise to the image, choose one of suitable mean filters and one of the suitable Order-Statistics filter to remove noise. Use subplot to plot 4 images, including original image, noise image, image after the first kind of filter, image after the second kind of filter
- Add salt and pepper noise to the image, separate the image into six parts, apply ostu threshold on each part of the image. Use subplot to plot 3 images, including original image, image with the noise, the image after the processing.
- Note: you need to provide sufficient comments on the codes.

```

clc;
clear;
close all;
I = im2double(imread('2.tif'));
[m n]=size(I);

%% Pepper noise % 2 points by adding pepper noise
p_pepper = 0.1;
tmp_mask = rand(m,n);
pepper_idx = find(tmp_mask <= p_pepper);
J1 = I;
J1(pepper_idx) = 0;

figure; % 2 points by showing images
subplot(1,4,1); imshow(I); title('Original image');
subplot(1,4,2); imshow(J1); title('image +pepper noise');

J2 = Contraharmonic_Filter(J1, [3,3], 1.5); % 4 points by choosing the suitable filters;a
subplot(1,4,3); imshow(J2); title('Contraharmonic mean filter with Q=1.5')

J3 = ordfilt2(J1,9,ones(3,3), 'symmetric');
subplot(1,4,4); imshow(J3); title('3x3 max filter');

%% salt and pepper noise
J = imnoise(I,'salt & pepper'); % 2 points by adding salt and pepper noise

block_funtion = @(block_struct) (block_struct.data >= graythresh(block_struct.data));
G = blockproc(J,[m/2 n/3],block_funtion); % 4 points by separating images into six blocks
figure;
subplot(1,3,1);imshow(I); % 2 points by the right demostration
subplot(1,3,2);imshow(J);
subplot(1,3,3);imshow(G);

```

Q3

- input a gray-scale image 'lena.bmp'
- separate this image into four parts: left up, right up, left down, and right down.
- For each part, we apply butterworth low pass filtering, laplacian filtering, ideal high pass filtering, and ideal low pass filtering.
- Use subplot to plot two images, including original image and the image after the processing
- Note: you need to provide sufficient comments on the codes.

```

I=imread('lena.bmp');% 1 point by loading images
[m,n]=size(I);% 1 point by separating images into four parts
I_1=I(1:m/2,1:n/2);
I_2=I(1:m/2,n/2+1:n);
I_3=I(m/2+1:m,1:n/2);
I_4=I(m/2+1:m,n/2+1:n);
% imshow(I);
% subplot(1,4,1);imshow(I_1);
% subplot(1,4,2);imshow(I_2);
% subplot(1,4,3);imshow(I_3);
% subplot(1,4,4);imshow(I_4);

%% BTW low pass filtering % 1.5 points by each accurate filters; 1.5*4=6 points
ff1=imbtwflpf(I_1,20);
out1=imfreqfilt(I_1,ff1);

% laplacian filtering
ff2 = fspecial('laplacian', 0);
out= imfilter(I_2, ff2);
out2 = imsubtract(I_2,out);

% ideal high pass filtering
ff3=imidealflpf(I_3,20);
ff2=1-ff3;
out3=imfreqfilt(I_3,ff2);

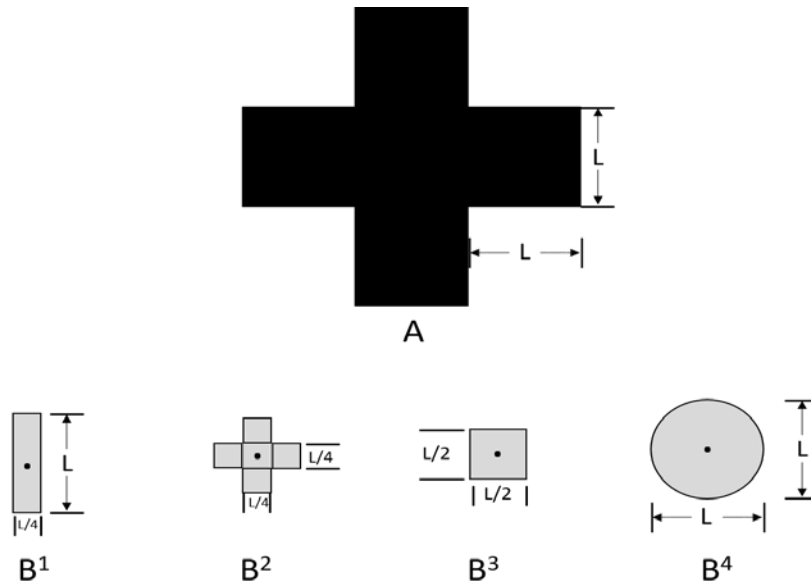
% ideal low pass filtering
ff4=imidealflpf(I_4,20);
out4=imfreqfilt(I_4,ff4);

% Processed image % 2 points by combining them together as the full image
I_2=zeros(m,n);
I2(1:m/2,1:n/2)=out1;
I2(1:m/2,n/2+1:n)=out2;
I2(m/2+1:m,1:n/2)=out3;
I2(m/2+1:m,n/2+1:n)=out4;

```

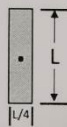
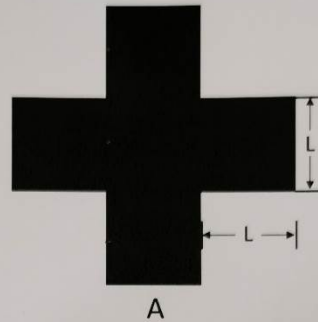
Q4

- Let A denote the set shown shaded in the following figure. Refer to the structuring elements shown (the black dots denote the origin). Sketch the result of the following morphological operations.
- $Y1 = (A \ominus B^4) \oplus B^2$ where $\ominus$ denotes the morphological erosion operator and $\oplus$ denotes the morphological dilation operator; (8 marks)
- $Y2 = (A \ominus B^1) \oplus B^3$. (7 marks)

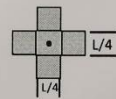


011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100

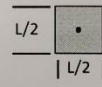
(b) $Y_2 = (A \ominus B^1) \oplus B^3$. (7 marks)



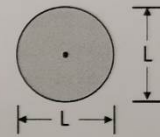
B^1



B^2

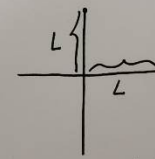


B^3

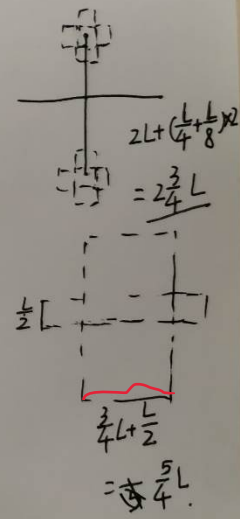


B^4

(a) $A \ominus B^4$

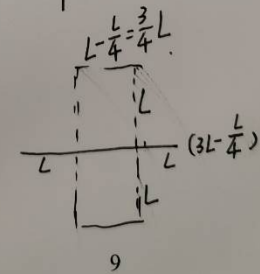


$(A \ominus B^4) \oplus B^2$



Solution:

(b) $A \ominus B^1$



$(A \ominus B^1) \oplus B^3$