Lecture 3
Principal Component Analysis

Lin ZHANG, PhD
School of Software Engineering
Tongji University
Fall 2023

Content

- Matrix Differentiation
- Lagrange Multiplier
- Principal Component Analysis
- Eigen-face based face classification

Matrix differentiation

- Function is a vector and the variable is a scalar

$$\mathbf{f}(t) = \left[f_1(t), f_2(t), \ldots, f_n(t) \right]^T$$

Definition

$$\frac{d\mathbf{f}}{dt} = \left[\frac{df_1(t)}{dt}, \frac{df_2(t)}{dt}, \ldots, \frac{df_n(t)}{dt} \right]^T$$

Matrix differentiation

- Function is a matrix and the variable is a scalar

$$F(t)=\begin{bmatrix} f_{11}(t)\ f_{12}(t),\ldots,f_{1m}(t) \\ f_{21}(t)\ f_{22}(t),\ldots,f_{2m}(t) \\ \vdots \\ f_{n1}(t)\ f_{n2}(t),\ldots,f_{nm}(t) \end{bmatrix}=\left[f_{ij}(t)\right]_{n\times m}$$

Definition

$$\frac{dF}{dt}=\begin{bmatrix} \frac{df_{11}(t)}{dt}\ \frac{df_{12}(t)}{dt},\ldots,\frac{df_{1m}(t)}{dt} \\ \frac{df_{21}(t)}{dt}\ \frac{df_{22}(t)}{dt},\ldots,\frac{df_{2m}(t)}{dt} \\ \vdots \\ \frac{df_{n1}(t)}{dt}\ \frac{df_{n2}(t)}{dt},\ldots,\frac{df_{nm}(t)}{dt} \end{bmatrix}=\left[\frac{df_{ij}(t)}{dt}\right]_{n\times m}$$

Matrix differentiation

- Function is a scalar and the variable is a vector

$$f(\mathbf{x}), \mathbf{x} = (x_1, x_2, \ldots, x_n)^T$$

Definition

$$\frac{df}{d\mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^T$$

In a similar way,

$$f(\mathbf{x}), \mathbf{x} = (x_1, x_2, \ldots, x_n)$$

$$\frac{df}{d\mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]$$

Matrix differentiation

- Function is a vector and the variable is a vector

$$\mathbf{x}=\left[x_1,x_2,\ldots,x_n\right]^T,\mathbf{y}=\left[y_1(\mathbf{x}),y_2(\mathbf{x}),\ldots,y_m(\mathbf{x})\right]^T$$

Definition

$$\frac{d\mathbf{y}}{d\mathbf{x}^T}=\begin{bmatrix}\frac{\partial y_1(\mathbf{x})}{\partial x_1},\frac{\partial y_1(\mathbf{x})}{\partial x_2},\ldots,\frac{\partial y_1(\mathbf{x})}{\partial x_n}\\ \frac{\partial y_2(\mathbf{x})}{\partial x_1},\frac{\partial y_2(\mathbf{x})}{\partial x_2},\ldots,\frac{\partial y_2(\mathbf{x})}{\partial x_n}\\ \vdots\\ \frac{\partial y_m(\mathbf{x})}{\partial x_1},\frac{\partial y_m(\mathbf{x})}{\partial x_2},\ldots,\frac{\partial y_m(\mathbf{x})}{\partial x_n}\end{bmatrix}_{m\times n}$$

Matrix differentiation

- Function is a vector and the variable is a vector

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T, \mathbf{y} = [y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_m(\mathbf{x})]^T$$

In a similar way,

$$\frac{d\mathbf{y}^T}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial y_1(\mathbf{x})}{\partial x_1}, \frac{\partial y_2(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial y_m(\mathbf{x})}{\partial x_1} \\ \frac{\partial y_1(\mathbf{x})}{\partial x_2}, \frac{\partial y_2(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial y_m(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial y_1(\mathbf{x})}{\partial x_n}, \frac{\partial y_2(\mathbf{x})}{\partial x_n}, \ldots, \frac{\partial y_m(\mathbf{x})}{\partial x_n} \end{bmatrix}_{n \times m}$$

Matrix differentiation

- Function is a vector and the variable is a vector

  Example:

$$\mathbf{y}=\begin{bmatrix} y_1(\mathbf{x}) \\ y_2(\mathbf{x}) \end{bmatrix}, \mathbf{x}=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, y_1(\mathbf{x})=x_1^2-x_2, y_2(\mathbf{x})=x_3^2+3x_2$$

$$\frac{d\mathbf{y}^T}{d\mathbf{x}}=\begin{bmatrix} \frac{\partial y_1(\mathbf{x})}{\partial x_1} & \frac{\partial y_2(\mathbf{x})}{\partial x_1} \\ \frac{\partial y_1(\mathbf{x})}{\partial x_2} & \frac{\partial y_2(\mathbf{x})}{\partial x_2} \\ \frac{\partial y_1(\mathbf{x})}{\partial x_3} & \frac{\partial y_2(\mathbf{x})}{\partial x_3} \end{bmatrix}=\begin{bmatrix} 2x_1 & 0 \\ -1 & 3 \\ 0 & 2x_3 \end{bmatrix}$$

Matrix differentiation

- Function is a scalar and the variable is a matrix

$$f(\mathbf{X}), \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & & \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

Definition

$$\frac{df(\mathbf{X})}{d\mathbf{X}} = \begin{bmatrix} \frac{\partial f}{\partial x_{11}} & \frac{\partial f}{\partial x_{12}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\ \cdots & & & \\ \frac{\partial f}{\partial x_{m1}} & \frac{\partial f}{\partial x_{m2}} & \cdots & \frac{\partial f}{\partial x_{mn}} \end{bmatrix}$$

Matrix differentiation

- Useful results

(1) $\mathbf{x}, \mathbf{a} \in \mathbb{R}^{n\times 1}$

Then,

$$\frac{d\mathbf{a}^T\mathbf{x}}{d\mathbf{x}} = \mathbf{a}, \frac{d\mathbf{x}^T\mathbf{a}}{d\mathbf{x}} = \mathbf{a}$$

How to prove?

Matrix differentiation

- Useful results

(2) $\mathbf{x} \in \mathbb{R}^{n\times 1}$ Then, $\dfrac{d\mathbf{x}^T\mathbf{x}}{d\mathbf{x}} = 2\mathbf{x}$

(3) $\mathbf{y}(\mathbf{x}) \in \mathbb{R}^{m\times 1}$, $\mathbf{x} \in \mathbb{R}^{n\times 1}$, $\dfrac{d\mathbf{y}^T(\mathbf{x})}{d\mathbf{x}} = \left(\dfrac{d\mathbf{y}(\mathbf{x})}{d\mathbf{x}^T}\right)^T$

(4) $A \in \mathbb{R}^{m\times n}, \mathbf{x} \in \mathbb{R}^{n\times 1}$ Then, $\dfrac{dA\mathbf{x}}{d\mathbf{x}^T} = A$

(5) $A \in \mathbb{R}^{m\times n}, \mathbf{x} \in \mathbb{R}^{n\times 1}$ Then, $\dfrac{d\mathbf{x}^T A^T}{d\mathbf{x}} = A^T$

(6) $A \in \mathbb{R}^{n\times n}, \mathbf{x} \in \mathbb{R}^{n\times 1}$ Then, $\dfrac{d\mathbf{x}^T A\mathbf{x}}{d\mathbf{x}} = (A + A^T)\mathbf{x}$

(7) $\mathbf{X} \in \mathbb{R}^{m\times n}, \mathbf{a} \in \mathbb{R}^{m\times 1}, \mathbf{b} \in \mathbb{R}^{n\times 1}$ Then, $\dfrac{d\mathbf{a}^T\mathbf{X}\mathbf{b}}{d\mathbf{X}} = \mathbf{a}\mathbf{b}^T$

Matrix differentiation

- Useful results

(8) $\mathbf{X} \in \mathbb{R}^{n \times m}, \mathbf{a} \in \mathbb{R}^{m \times 1}, \mathbf{b} \in \mathbb{R}^{n \times 1}$ Then, $\dfrac{d\mathbf{a}^T \mathbf{X}^T \mathbf{b}}{d\mathbf{X}} = \mathbf{b}\mathbf{a}^T$

(9) $\mathbf{X} \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times m}$ Then, $\dfrac{d\left(tr\mathbf{X}B\right)}{d\mathbf{X}} = B^T$

(10) $\mathbf{X} \in \mathbb{R}^{n \times n}, \mathbf{X}$ is invertible, $\dfrac{d\left|\mathbf{X}\right|}{d\mathbf{X}} = \left|\mathbf{X}\right|\left(\mathbf{X}^{-1}\right)^T$

Content

- Matrix Differentiation
- Lagrange Multiplier
- Principal Component Analysis
- Eigen-face based face classification

Lagrange multiplier

- Single-variable function

  $f(x)$ is differentiable in (a, b). At $x_0 \in (a,b)$, $f(x)$ achieves an extremum

  $$\rightarrow \quad \frac{df}{dx}\Big|_{x_0} = 0$$

- Two-variables function

  $f(x, y)$ is differentiable in its domain. At (x_0, y_0), $f(x, y)$ achieves an extremum

  $$\rightarrow \quad \frac{\partial f}{\partial x}\Big|_{(x_0,y_0)} = 0, \frac{\partial f}{\partial y}\Big|_{(x_0,y_0)} = 0$$

Lagrange multiplier

- In general case

  If $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^{n\times 1}$ achieves a local extremum at $\mathbf{x}_0$ and it is derivable at $\mathbf{x}_0$, then $\mathbf{x}_0$ is a stationary point of $f(\mathbf{x})$, i.g.,

$$\frac{\partial f}{\partial x_1}\Big|_{\mathbf{x}_0} = 0, \frac{\partial f}{\partial x_2}\Big|_{\mathbf{x}_0} = 0, \ldots, \frac{\partial f}{\partial x_n}\Big|_{\mathbf{x}_0} = 0$$

  Or in other words,

$$\nabla f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_0} = \mathbf{0}$$

Lagrange multiplier

- Lagrange multiplier is a strategy for finding **all the possible** extremum points of a function subject to equality constraints

Problem: find **all the possible** extremum points for $y = f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^{n\times 1}$

under m constraints $g_k(\mathbf{x}) = 0, k = 1, 2, \ldots, m$

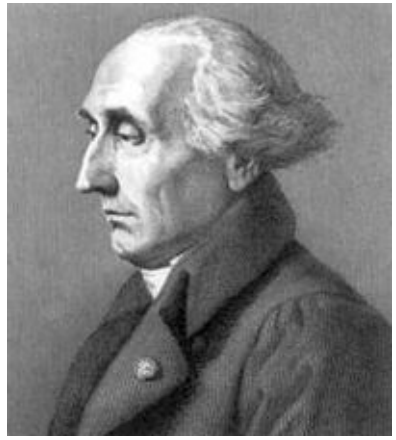
Solution: $F(\mathbf{x}; \lambda_1, \ldots, \lambda_m) = f(\mathbf{x}) + \sum_{k=1}^{m} \lambda_k g_k(\mathbf{x})$

If $\mathbf{x}_0$ is an extremum point of $f(\mathbf{x})$ under constraints

$\exists \lambda_{10}, \lambda_{20} \ldots, \lambda_{m0}$, making $(\mathbf{x}_0, \lambda_{10}, \lambda_{20} \ldots, \lambda_{m0})$

a stationary point of F

Thus, by identifying the stationary points of F, we can get all the possible extremum points of $f(\mathbf{x})$ under equality constraints



Joseph-Louis Lagrange
Jan. 25, 1736~Apr.10, 1813

Lagrange multiplier

- Lagrange multiplier is a strategy for finding **all the possible** extremum points of a function subject to equality constraints

Problem: find **all the possible** extremum points for $y = f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^{n\times 1}$

under m constraints $g_k(\mathbf{x}) = 0, k = 1, 2, \ldots, m$

Solution: $F(\mathbf{x}; \lambda_1, \ldots, \lambda_m) = f(\mathbf{x}) + \sum_{k=1}^{m} \lambda_k g_k(\mathbf{x})$

$(\mathbf{x}_0, \lambda_{10}, \ldots, \lambda_{m0})$ is a stationary point of F →

$$\frac{\partial F}{\partial x_1} = 0, \frac{\partial F}{\partial x_2} = 0, \ldots, \frac{\partial F}{\partial x_n} = 0, \frac{\partial F}{\partial \lambda_1} = 0, \frac{\partial F}{\partial \lambda_2} = 0, \ldots, \frac{\partial F}{\partial \lambda_m} = 0$$

at that point

$n + m$ equations!

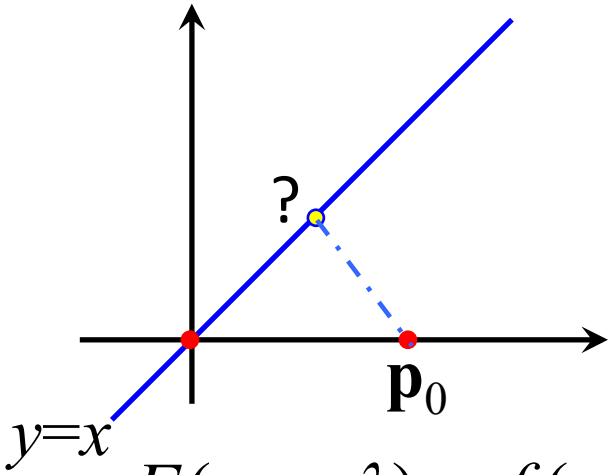
↓

$\mathbf{x}_0$ is a possible extremum point of $f(\mathbf{x})$ under equality constraints

Lagrange multiplier

- Example

Problem: for a given point $\mathbf{p}_0 = (1, 0)$, among all the points lying on the line $y=x$, identify the one having the least distance to $\mathbf{p}_0$.



The distance is

$$f(x,y) = (x-1)^2 + (y-0)^2$$

Now we want to find the global minimizer of $f(x, y)$ under the constraint

$$g(x,y) = y - x = 0$$

According to Lagrange multiplier method, construct the Lagrange function

$$F(x,y,\lambda) = f(x,y) + \lambda g(x,y) = (x-1)^2 + y^2 + \lambda(y-x)$$

Find the stationary point of $F(x,y,\lambda)$

Lagrange multiplier

- Example

$$\begin{cases} \dfrac{\partial F}{\partial x} = 0 \\ \dfrac{\partial F}{\partial y} = 0 \\ \dfrac{\partial F}{\partial \lambda} = 0 \end{cases} \rightarrow \begin{cases} 2(x-1)+\lambda = 0 \\ 2y-\lambda = 0 \\ x-y = 0 \end{cases} \rightarrow \begin{cases} x = 0.5 \\ y = 0.5 \\ \lambda = 1 \end{cases}$$

Thus, $(0.5, 0.5, 1)$ is the only stationary point of $F(x, y, \lambda)$

$(0.5,0.5)$ is the only possible extremum point of $f(x,y)$ under constraints

The global minimizer of $f(x,y)$ under constraints exists

$(0.5,0.5)$ is the global minimizer of $f(x,y)$ under constraints

Content

- Matrix Differentiation
- Lagrange Multiplier
- Principal Component Analysis
- Eigen-face based face classification

Principal Component Analysis (PCA)

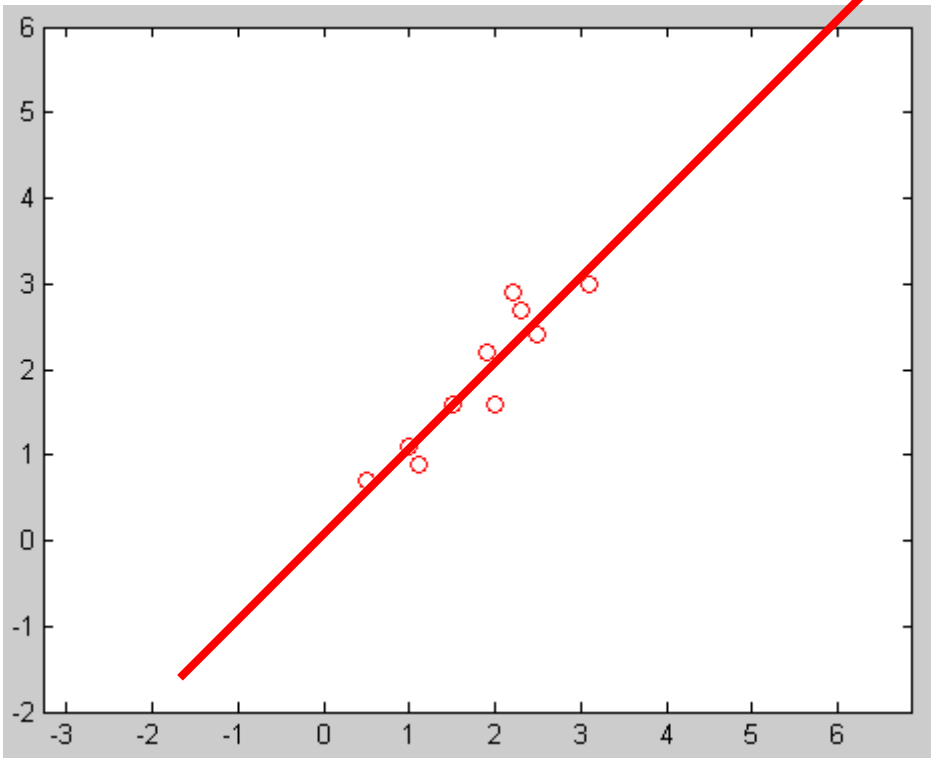
- PCA: converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components
- This transformation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (i.e., uncorrelated with) the preceding components

Principal Component Analysis (PCA)

- Illustration

How to find?

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)



De-correlation!

Along which orientation the data points scatter most?

Principal Component Analysis (PCA)

- Identify the orientation with largest variance

Suppose $\mathbf{X}$ contains n data points, and each data point is p-dimensional, that is

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

Now, we want to find such a unit vector $\mathbf{u}_1$,

$$\mathbf{u}_1 = \arg\max_{\mathbf{u}} \left(\mathrm{var}\left(\mathbf{u}^T \mathbf{X} \right) \right), \mathbf{u} \in \mathbb{R}^{p\times 1}$$

Principal Component Analysis (PCA)

- Identify the orientation with largest variance

$$\mathrm{var}\left(\mathbf{u}^T\mathbf{X}\right) = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{u}^T\mathbf{x}_i - \mathbf{u}^T\boldsymbol{\mu})^2 = \frac{1}{n-1}\sum_{i=1}^{n}\mathbf{u}^T(\mathbf{x}_i-\boldsymbol{\mu})(\mathbf{x}_i-\boldsymbol{\mu})^T\mathbf{u}$$

$$= \mathbf{u}^T\mathbf{C}\mathbf{u}$$

(Note that: $\mathbf{u}^T(\mathbf{x}_i-\boldsymbol{\mu}) = (\mathbf{x}_i-\boldsymbol{\mu})^T\mathbf{u}$)

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

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

Please verify that C is positive semidefinite. (Assignment)

Principal Component Analysis (PCA)

- Identify the orientation with largest variance

Our problem is,

$$\mathbf{u}_1 = \arg\max_{\mathbf{u}} \mathbf{u}^T\mathbf{C}\mathbf{u}, \text{ subject to } \mathbf{u}^T\mathbf{u} = 1$$

The Lagrange function is $F(\mathbf{u},\lambda) = \mathbf{u}^T\mathbf{C}\mathbf{u} - \lambda\left(\mathbf{u}^T\mathbf{u} - 1\right)$

Solve $\nabla F(\mathbf{u},\lambda) = \mathbf{0}$:

$$\mathbf{0} = \frac{\partial\left(\mathbf{u}^T\mathbf{C}\mathbf{u} - \lambda\left(\mathbf{u}^T\mathbf{u} - 1\right)\right)}{\partial\mathbf{u}} = 2\mathbf{C}\mathbf{u} - 2\lambda\mathbf{u} \rightarrow \mathbf{C}\mathbf{u} = \lambda\mathbf{u}$$

$\mathbf{u}$ is $\mathbf{C}$'s eigen-vector

Since,

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

Thus,

Principal Component Analysis (PCA)

- Identify the orientation with largest variance

Thus, $\mathbf{u}_1$ should be the eigen-vector of $\mathbf{C}$ corresponding to the largest eigen-value of $\mathbf{C}$

What is another orientation $\mathbf{u}_2$, orthogonal to $\mathbf{u}_1$, and along which the data can have the second largest variation?

Answer: it is the eigen-vector associated to the second largest eigen-value λ_2 of $\mathbf{C}$ and such a variance is λ_2

Assignment!

In some books, they say to get principal components, SVD is used instead of the eigen-value decomposition. Do you know why?

Principal Component Analysis (PCA)

- Identify the orientation with largest variance

Results: the eigen-vectors of $\mathbf{C}$ forms a set of orthogonal basis and they are referred as **Principal Components** of the original data $\mathbf{X}$

You can consider PCs as a set of orthogonal coordinates. Under such a coordinate system, variables are not correlated.

Principal Component Analysis (PCA)

- Express data in PCs

Suppose $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_p\}$ 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 $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_p\}$, and the representation coefficients are

$$\mathbf{c}_i = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_p^T \end{pmatrix} \mathbf{x}_i$$

Actually, $\mathbf{c}_i$ is the coordinates of $\mathbf{x}_i$ in the new coordinate system spanned by $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_p\}$

Principal Component Analysis (PCA)

- Summary

$\mathbf{X} \in \mathbb{R}^{p \times n}$ is a data matrix, and each column is a data sample

Suppose $\overline{\mathbf{X}}$ is sample-mean subtracted version of $\mathbf{X}$

$$\mathbf{C} = \mathrm{cov}(\mathbf{X}) = \frac{1}{n-1}\overline{\mathbf{X}}\overline{\mathbf{X}}^T \equiv \mathbf{U}\Sigma\mathbf{U}^T$$

$\mathbf{U} = \left[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p\right]$ spans a new space

Data in new space is represented as $\mathbf{X}' = \mathbf{U}^T\mathbf{X}$

In new space, dimensions of data are not correlated

Principal Component Analysis (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)

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

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

Eigen-values = $11.5562, 0.4418$

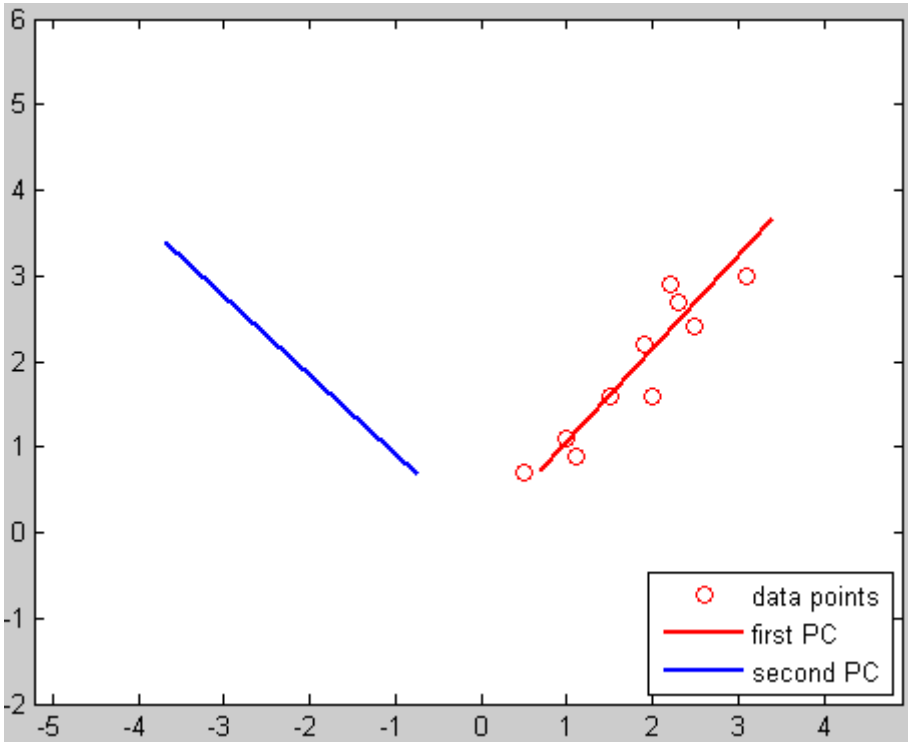
Corresponding eigen-vectors:

$$\mathbf{u}_1 = \begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix}$$

$$\mathbf{u}_2 = \begin{pmatrix} -0.7352 \\ 0.6779 \end{pmatrix}$$

Principal Component Analysis (PCA)

- Illustration



Principal Component Analysis (PCA)

- Illustration

Coordinates of the data points in the new coordinate system

$$newC = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \end{pmatrix} \mathbf{X}$$

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

$$= \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \\ -0.211 & 0.107 & 0.348 & 0.094 & -0.245 & 0.139 & -0.386 & 0.011 & -0.018 & -0.199 \end{pmatrix}$$

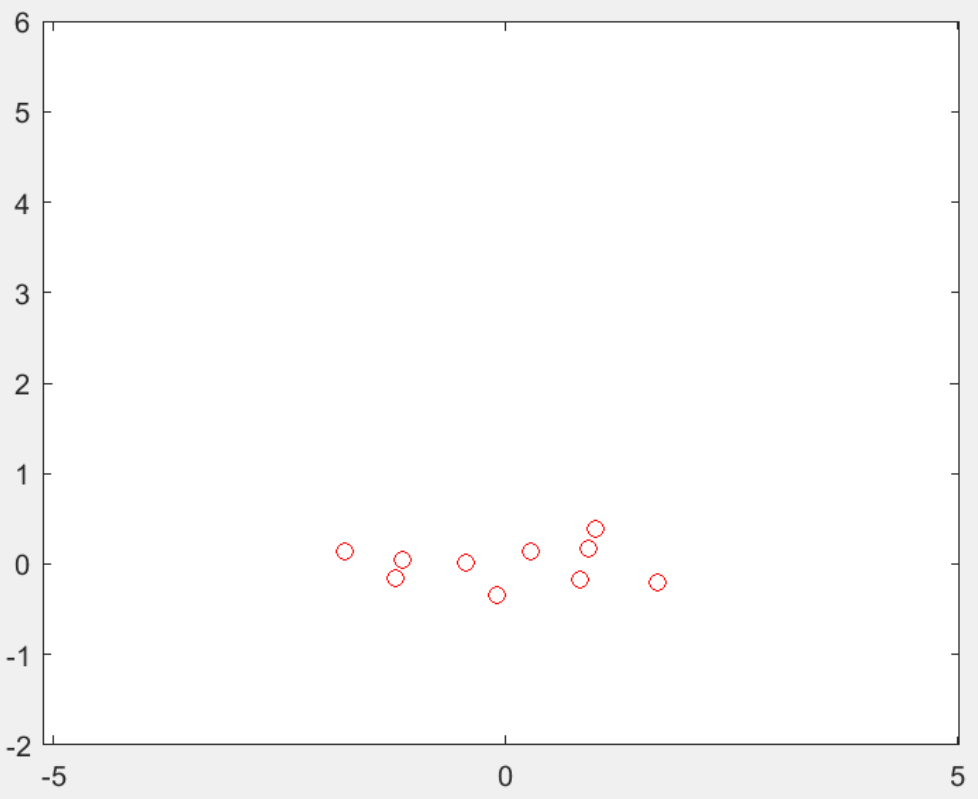
Principal Component Analysis (PCA)

- 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!



Principal Component Analysis (PCA)

- Data dimension reduction with PCA

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

If all of $\{\mathbf{u}_i\}_{i=1}^{p}$ are used, $\mathbf{c}_i = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_p^T \end{pmatrix} \mathbf{x}_i$ is still p-dimensional

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

That is, the dimension of the data is reduced!

Principal Component Analysis (PCA)

- Data dimension reduction with PCA

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

$$\mathbf{C} = \mathrm{cov}(\mathbf{X}) \equiv \mathbf{U}\Sigma\mathbf{U}^T$$

$\mathbf{U} = \left[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m, \ldots, \mathbf{u}_p\right]$ spans a new space

For dimension reduction, only $\mathbf{u}_1 \sim \mathbf{u}_m$ are used,

$$\mathbf{U}_m = \left[\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\right] \in \mathbb{R}^{p\times m}$$

Data in $\mathbf{U}_m$,

$$\mathbf{X}_{dr} = \left(\mathbf{U}_m\right)^T \mathbf{X} \in \mathbb{R}^{m\times n}$$

Principal Component Analysis (PCA)

- Recovering the dimension-reduction data

Suppose $\mathbf{X}_{dr} \in \mathbb{R}^{m \times n}$ are low-dimensional representation of the signals $\mathbf{X} \in \mathbb{R}^{p \times n}$

How to recover $\mathbf{X}_{dr} \in \mathbb{R}^{m \times n}$ to the original p-d space?

$$\mathbf{X}_{recover} = \mathbf{U} \begin{bmatrix} \mathbf{x}_{dr1}, \mathbf{x}_{dr2}, \ldots, \mathbf{x}_{drn} \\ 0 \quad 0 \quad 0 \\ \vdots \\ 0 \quad 0 \quad 0 \end{bmatrix} \Big\} p - m$$

$$= \mathbf{U}_m \mathbf{X}_{dr}$$

Principal Component Analysis (PCA)

- Illustration

Coordinates of the data points in the new coordinate system

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

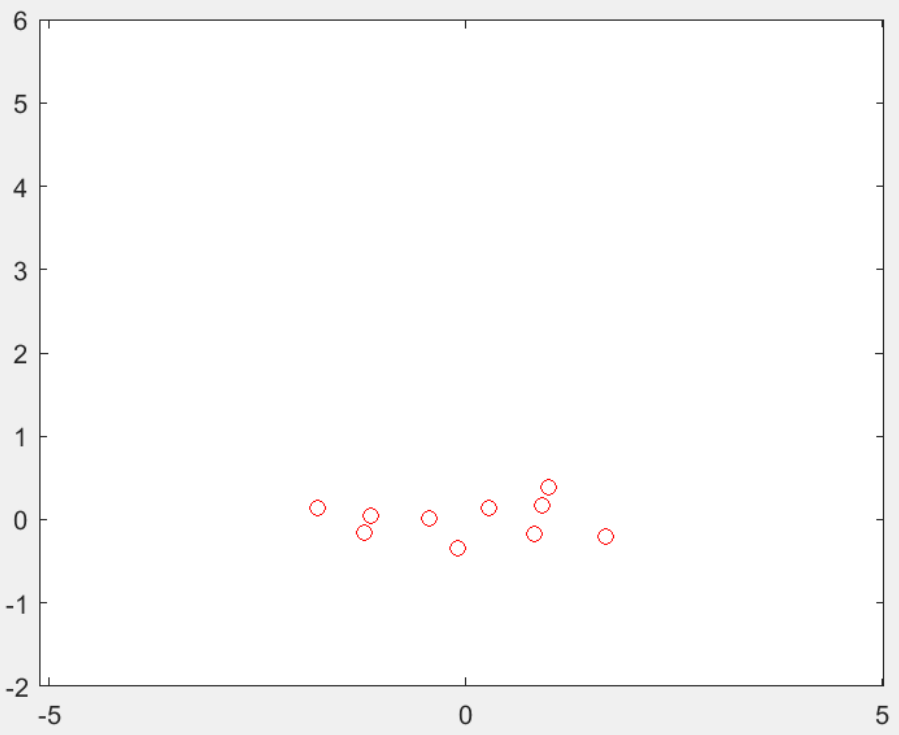
If only the first PC (corresponds to the largest eigen-value) is remained

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

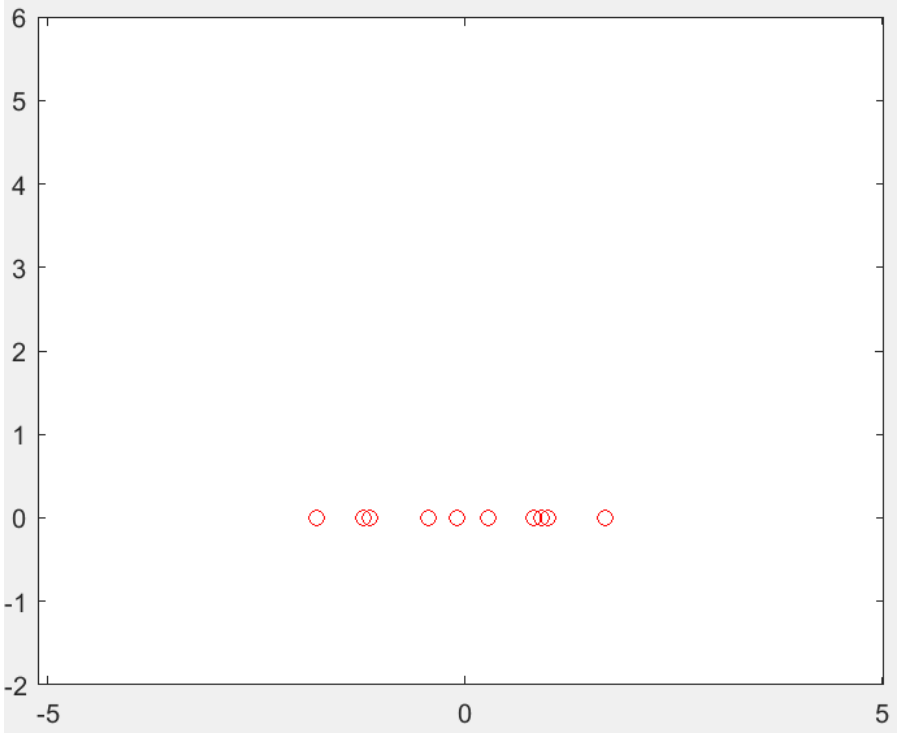
$$= \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \end{pmatrix}$$

Principal Component Analysis (PCA)

- Illustration



All PCs are used



Only 1 PC is used

Dimension reduction!

Principal Component Analysis (PCA)

- Illustration

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

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

$$= \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \end{pmatrix}$$

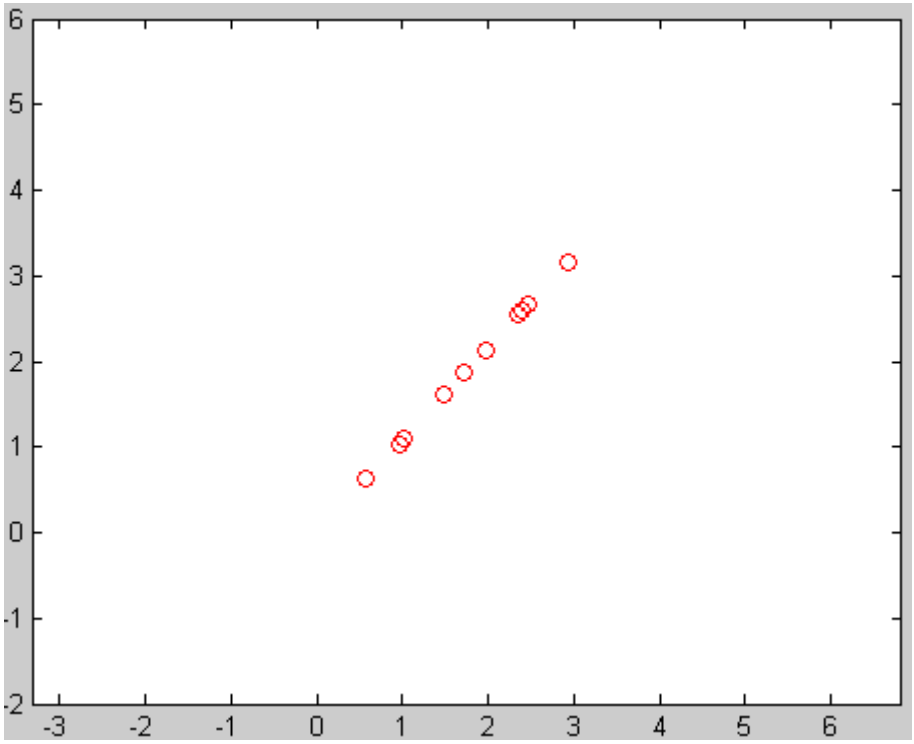
How to recover $newC$ to the original space? Easy

$$\begin{pmatrix} 0.6779 & 0.7352 \end{pmatrix}^T newC$$

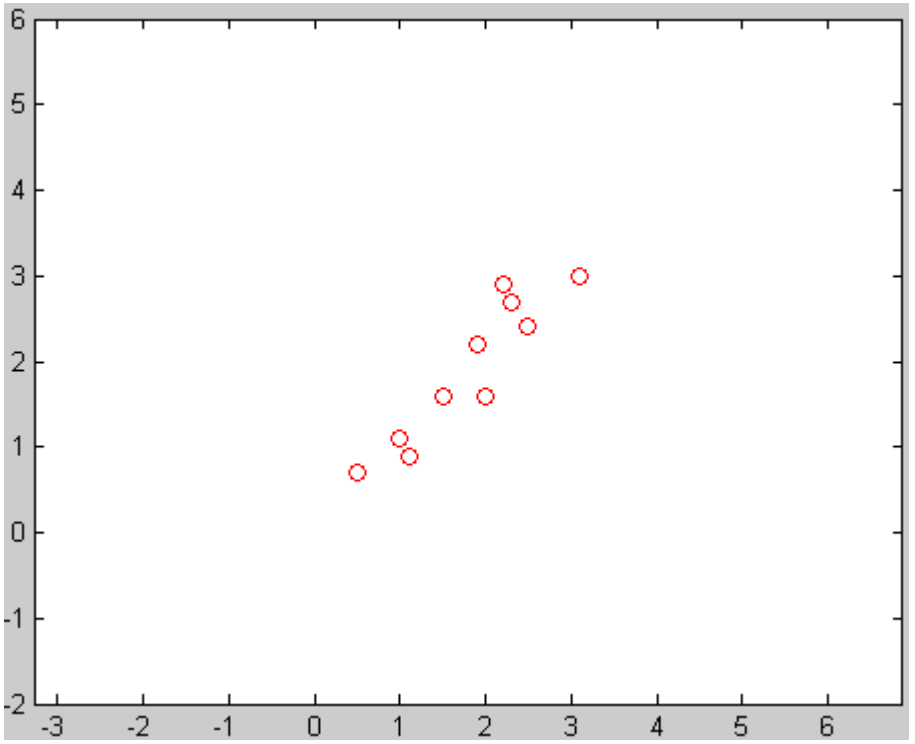
$$= \begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix} \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \end{pmatrix}$$

Principal Component Analysis (PCA)

- Illustration



Data recovered if only 1 PC used



Original data

Content

- Matrix Differentiation
- Lagrange Multiplier
- Principal Component Analysis
- Eigen-face based face classification

Eigen-face based face recognition

- Proposed in [1]
- 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

[1] M. Turk and A. Pentland, Eigenfaces for recognition, Journal of Cognitive Neuroscience' 91

Eigen-face based face recognition

- Training period
  - Step 1: prepare images $\{\mathbf{x}_i\}$ for the training set
  - Step 2: compute the mean image and covariance matrix
  - Step 3: compute the eigen-faces (eigen-vectors) from the covariance matrix and only keep M eigen-faces corresponding to the largest eigenvalues; these M eigen-faces $(\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_M)$ define the face space
  - Step 4: compute the representation coefficients of each training image $\mathbf{x}_i$ on the M-d subspace

$$\mathbf{r}_i = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_M^T \end{pmatrix} \mathbf{x}_i$$

Eigen-face based face recognition

- Testing period
  - Step 1: project the test image onto the M-d subspace to get the representation coefficients
  - Step 2: classify the coefficient pattern as either a known person or as unknown (usually Euclidean distance is used here)

Eigen-face based face recognition

- One technique to perform eigen-value decomposition to a large matrix

If each image is 100×100, the covariance matrix C is 10000×10000

It is formidable to perform PCA for a so large matrix

However the rank of the covariance matrix is limited by the number of training examples: if there are n training examples, there will be at most n-1 eigenvectors with non-zero eigenvalues.

Usually, the number of training examples is much smaller than the dimensionality of the images.

Eigen-face based face recognition

- One technique to perform eigen-value decomposition to a large matrix

Principal components can be computed more easily as follows,

Let $\mathbf{X} \in \mathbb{R}^{p \times n}$ be the matrix of preprocessed n training examples, where each column (p-d) contains one mean-subtracted image; $\left(p \gg n \right)$

The corresponding covariance matrix is $\frac{1}{n-1}\mathbf{X}\mathbf{X}^T \in \mathbb{R}^{p \times p}$; very large

Instead, we perform eigen-value decomposition to $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{n \times n}$

$$\mathbf{X}^T\mathbf{X}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

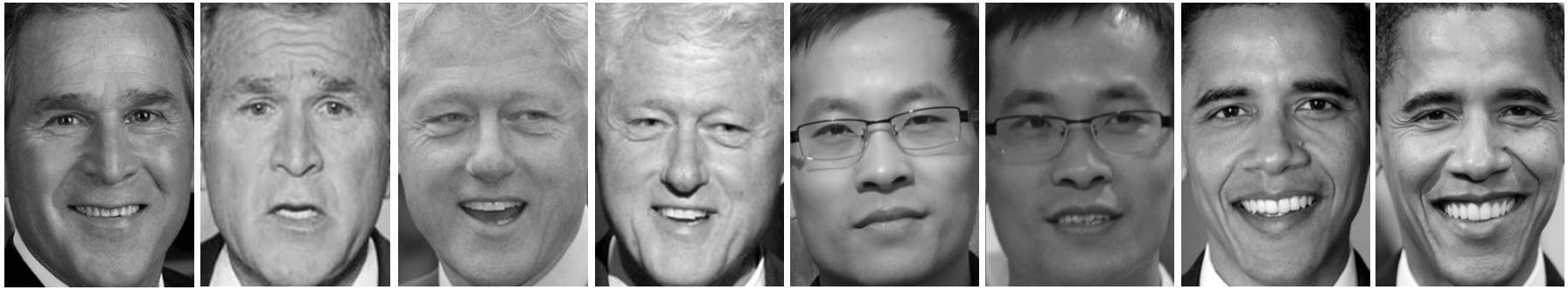
Pre-multiply $\mathbf{X}$ on both sides

$$\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{v}_i = \lambda_i \mathbf{X}\mathbf{v}_i$$

$\mathbf{X}\mathbf{v}_i$ is the eigen-vector of $\mathbf{X}\mathbf{X}^T$

Eigen-face based face recognition

- Example— training stage



4 classes, 8 samples altogether

Vectorize the 8 images, and stack them into a data matrix $\mathbf{X}$

Compute the eigen-faces (PCs) based on $\mathbf{X}$

In this example, we retain the first 6 eigen-faces to span the sub-space

Eigen-face based face recognition

- Example— training stage

If reshaping in the matrix form, 6 eigen-faces appear as follows

Ghost face!

$\mathbf{u}_1$ $\mathbf{u}_2$ $\mathbf{u}_3$ $\mathbf{u}_4$ $\mathbf{u}_5$ $\mathbf{u}_6$

Then, each training face is projected to the learned sub-space

$$\mathbf{r}_i = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_6^T \end{pmatrix} \mathbf{x}_i$$

Eigen-face based face recognition

- Example— training stage

If reshaping in the matrix form, 6 eigen-faces appear as follows

$(\mathbf{x}_7)$

$$= 0.33\mathbf{u}_1 - 0.74\mathbf{u}_2 + 0.07\mathbf{u}_3 - 0.24\mathbf{u}_4 + 0.28\mathbf{u}_5 + 0.43\mathbf{u}_6$$

$\mathbf{r}_7=(0.33\ \ -0.74\ \ 0.07\ \ -0.24\ \ 0.28\ \ 0.43)^T$ is the representation vector of the 7th training image

Eigen-face based face recognition

- Example— testing stage

$\mathbf{u}_1$ $\mathbf{u}_2$ $\mathbf{u}_3$ $\mathbf{u}_4$ $\mathbf{u}_5$ $\mathbf{u}_6$

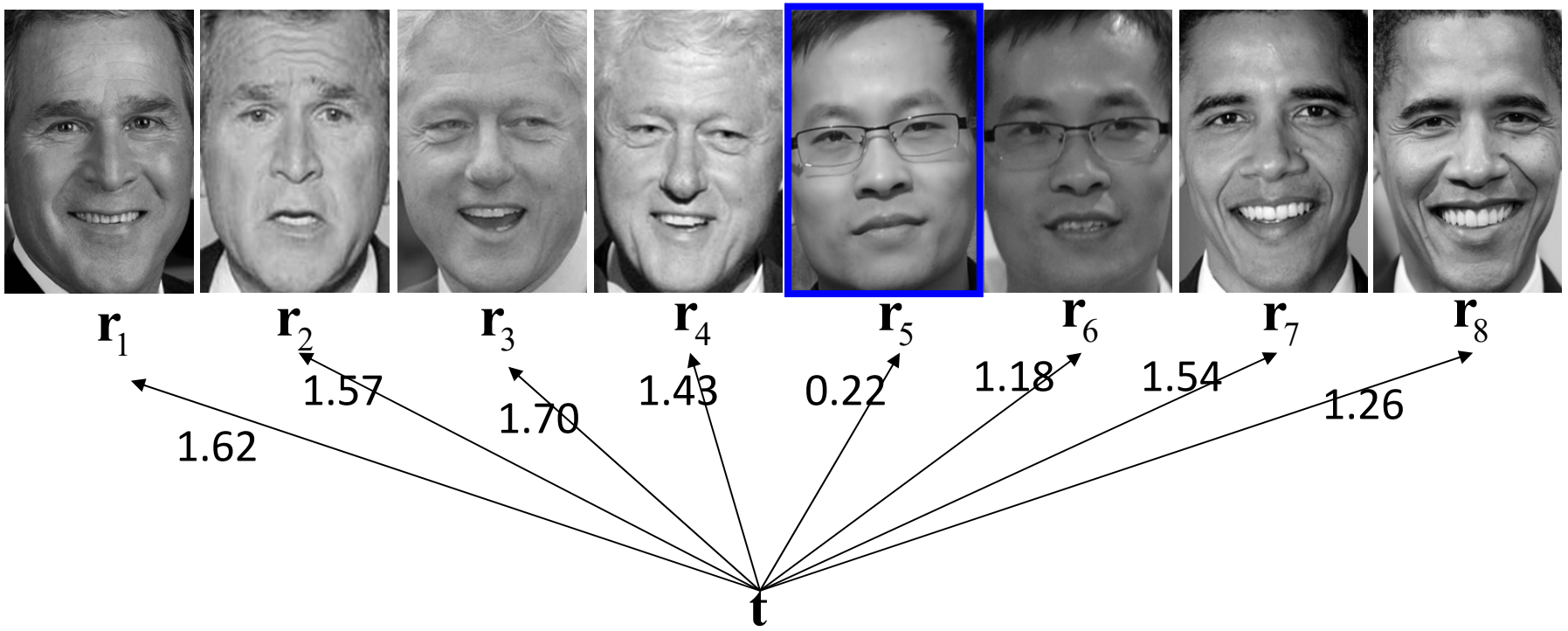
A new image comes, project it to the learned sub-space $\mathbf{t} = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_6^T \end{pmatrix} testI$

$= 0.52\mathbf{u}_1 + 0.17\mathbf{u}_2 - 0.01\mathbf{u}_3 - 0.39\mathbf{u}_4 + 0.67\mathbf{u}_5 - 0.29\mathbf{u}_6$

$\mathbf{t}$=(0.52 0.17 -0.01 –0.39 0.67 0.29)^T is the representation vector of this testing image

Eigen-face based face recognition

- Example— testing stage

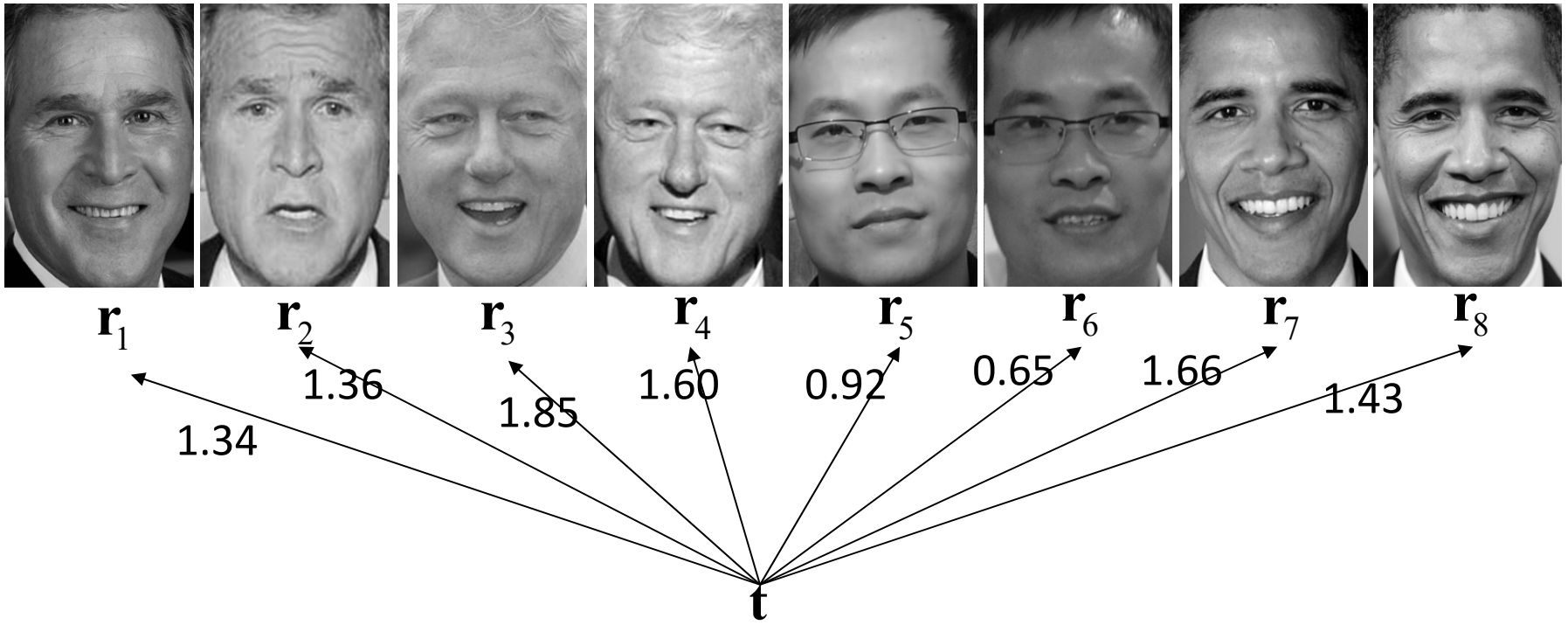


l_2-norm based distance metric

This guy should be Lin!

Eigen-face based face recognition

- Example— testing stage



l_2-norm based distance metric

We set threshold = 0.50

This guy does not exist in the dataset!


Q+A