Lecture 3
Principal Component Analysis

Lin ZHANG, PhD
School of Software Engineering
Tongji University
Fall 2020
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

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

Definition

\[ \frac{df}{dt} = \left[ \frac{df_1(t)}{dt}, \frac{df_2(t)}{dt}, \ldots, \frac{df_n(t)}{dt} \right]^T \]
Matrix differentiation

- Function is a matrix and the variable is a scalar

\[
f(t) = \begin{bmatrix}
    f_{11}(t) & f_{12}(t) & \cdots & f_{1m}(t) \\
    f_{21}(t) & f_{22}(t) & \cdots & f_{2m}(t) \\
    \vdots & \vdots & \ddots & \vdots \\
    f_{n1}(t) & f_{n2}(t) & \cdots & f_{nm}(t)
\end{bmatrix} = \begin{bmatrix} f_{ij}(t) \end{bmatrix}_{n \times m}
\]

Definition

\[
\frac{df}{dt} = \begin{bmatrix}
    \frac{df_{11}(t)}{dt} & \frac{df_{12}(t)}{dt} & \cdots & \frac{df_{1m}(t)}{dt} \\
    \frac{df_{21}(t)}{dt} & \frac{df_{22}(t)}{dt} & \cdots & \frac{df_{2m}(t)}{dt} \\
    \vdots & \vdots & \ddots & \vdots \\
    \frac{df_{n1}(t)}{dt} & \frac{df_{n2}(t)}{dt} & \cdots & \frac{df_{nm}(t)}{dt}
\end{bmatrix} = \begin{bmatrix} \frac{df_{ij}(t)}{dt} \end{bmatrix}_{n \times m}
\]
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

\[ \frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1}, & \frac{\partial f}{\partial x_2}, & \ldots, & \frac{\partial f}{\partial x_n} \end{bmatrix}^T \]

In a similar way,

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

\[ \frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1}, & \frac{\partial f}{\partial x_2}, & \ldots, & \frac{\partial f}{\partial x_n} \end{bmatrix} \]
Matrix differentiation

- Function is a vector and the variable is a vector

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

Definition

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

- Function is a vector and the variable is a vector

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

In a similar way,

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

- Function is a vector and the variable is a vector

Example:

\[
\begin{bmatrix}
y_1(x) \\
y_2(x)
\end{bmatrix},
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix},
\begin{align*}
y_1(x) &= x_1^2 - x_2, \\
y_2(x) &= x_3^2 + 3x_2
\end{align*}
\]

\[
\frac{dy^T}{dx} = \begin{bmatrix}
\frac{\partial y_1(x)}{\partial x_1} & \frac{\partial y_2(x)}{\partial x_1} \\
\frac{\partial y_1(x)}{\partial x_2} & \frac{\partial y_2(x)}{\partial x_2} \\
\frac{\partial y_1(x)}{\partial x_3} & \frac{\partial y_2(x)}{\partial x_3}
\end{bmatrix} = \begin{bmatrix}
2x_1 & 0 \\
-1 & 3 \\
0 & 2x_3
\end{bmatrix}
\]
Matrix differentiation

- Function is a scalar and the variable is a matrix

\[ f(X), X \in \mathbb{R}^{m \times n} \]

**Definition**

\[
\frac{df(X)}{dX} = \begin{bmatrix}
\frac{\partial f}{\partial x_{11}} & \frac{\partial f}{\partial x_{12}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\
\frac{\partial f}{\partial x_{m1}} & \frac{\partial f}{\partial x_{m2}} & \cdots & \frac{\partial f}{\partial x_{mn}}
\end{bmatrix}
\]
Matrix differentiation

- Useful results

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

Then,

\[ \frac{d a^T x}{dx} = a, \quad \frac{d x^T a}{dx} = a \]

How to prove?
Matrix differentiation

• Useful results

(2) \( A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^{n \times 1} \) Then, \( \frac{dA x}{dX^T} = A \)

(3) \( A \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^{n \times 1} \) Then, \( \frac{d x^T A^T}{dX} = A^T \)

(4) \( A \in \mathbb{R}^{n \times n}, x \in \mathbb{R}^{n \times 1} \) Then, \( \frac{d x^T A x}{dX} = (A + A^T)x \)

(5) \( X \in \mathbb{R}^{m \times n}, a \in \mathbb{R}^{m \times 1}, b \in \mathbb{R}^{n \times 1} \) Then, \( \frac{d a^T X b}{dX} = ab^T \)

(6) \( X \in \mathbb{R}^{n \times m}, a \in \mathbb{R}^{m \times 1}, b \in \mathbb{R}^{n \times 1} \) Then, \( \frac{d a^T X^T b}{dX} = ba^T \)

(7) \( x \in \mathbb{R}^{n \times 1} \) Then, \( \frac{dx^T x}{dx} = 2x \)
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

  \[
  \frac{df}{dx} \bigg|_{x_0} = 0
  \]

- **Two-variables function**

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

  \[
  \frac{\partial f}{\partial x} \bigg|_{(x_0, y_0)} = 0, \quad \frac{\partial f}{\partial y} \bigg|_{(x_0, y_0)} = 0
  \]
Lagrange multiplier

• In general case

If \( f(x) \), \( x \in \mathbb{R}^{n \times 1} \) achieves a local extremum at \( x_0 \) and it is differential at \( x_0 \), then \( x_0 \) is a stationary point of \( f(x) \), i.e.,

\[
\frac{\partial f}{\partial x_1} \bigg|_{x_0} = 0, \quad \frac{\partial f}{\partial x_2} \bigg|_{x_0} = 0, \ldots, \quad \frac{\partial f}{\partial x_n} \bigg|_{x_0} = 0
\]

Or in other words,

\[
\nabla f(x) \bigg|_{x=x_0} = 0
\]
Lagrange multiplier

- Lagrange multiplier is a strategy for finding the stationary point of a function subject to equality constraints.

Problem: find stationary points for \( y = f(x), \ x \in \mathbb{R}^{n \times 1} \)
under \( m \) constraints \( g_k(x) = 0, k = 1, 2, \ldots, m \)

Solution:

\[ F(x; \lambda_1, \ldots, \lambda_m) = f(x) + \sum_{k=1}^{m} \lambda_k g_k(x) \]

If \( (x_0, \lambda_{10}, \lambda_{20}, \ldots, \lambda_{m0}) \) is a stationary point of \( F \), then,
\( x_0 \) is a stationary point of \( f(x) \) with constraints.

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

- Lagrange multiplier is a strategy for finding the stationary point of a function subject to equality constraints.

Problem: find stationary 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!
Lagrange multiplier

• Example

Problem: for a given point $p_0 = (1, 0)$, among all the points lying on the line $y = x$, identify the one having the least distance to $p_0$.

The distance is

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

Now we want to find the stationary point of $f(x, y)$ under the constraint $g(x, y) = y - x = 0$

According to Lagrange multiplier method, construct another function

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

Find the stationary point for $F(x, y, \lambda)$
Lagrange multiplier

• Example

Problem: for a given point $p_0 = (1, 0)$, among all the points lying on the line $y = x$, identify the one having the least distance to $p_0$.

![Graph showing a line and a point labeled $p_0$.]

\[
\begin{align*}
\frac{\partial F}{\partial x} &= 0 \\
\frac{\partial F}{\partial y} &= 0 \\
\frac{\partial F}{\partial \lambda} &= 0
\end{align*}
\]

\[
\begin{align*}
2(x - 1) + \lambda &= 0 \\
2y - \lambda &= 0 \\
x - y &= 0
\end{align*}
\]

\[
\begin{align*}
x &= 0.5 \\
y &= 0.5 \\
\lambda &= 1
\end{align*}
\]

$(0.5, 0.5, 1)$ is a stationary point of $F(x, y, \lambda)$

$(0.5, 0.5)$ is a stationary point 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

\[ x, y \]
(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)

Along which orientation do the data points scatter most?

How to find?
De-correlation!

Lin ZHANG, SSE, 2020
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

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

Now, we want to find such a unit vector \( \alpha_1 \),

\[
\alpha_1 = \operatorname{arg\,max}_{\alpha} \left( \operatorname{var}(\alpha^T \mathbf{X}) \right), \alpha \in \mathbb{R}^{p \times 1}
\]
Principal Component Analysis (PCA)

• Identify the orientation with largest variance

\[
\text{var}(\alpha^T x) = \frac{1}{n-1} \sum_{i=1}^{n} (\alpha^T x_i - \alpha^T \mu)^2 = \frac{1}{n-1} \sum_{i=1}^{n} \alpha^T (x_i - \mu)(x_i - \mu)^T \alpha
\]

\[= \alpha^T C \alpha\]

(Note that: \(\alpha^T (x_i - \mu) = (x_i - \mu)^T \alpha\))

where \(\mu = \frac{1}{n} \sum_{i=1}^{n} x_i\)

and \(C = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T\) is the covariance matrix.
Principal Component Analysis (PCA)

• Identify the orientation with largest variance

Since \( \alpha \) is unit, \( \alpha^T \alpha = 1 \)

Based on Lagrange multiplier method, we need to,

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

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

\( \alpha \) is \( C \)'s eigen-vector

Since,

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

Thus,
Principal Component Analysis (PCA)

- Identify the orientation with largest variance

Thus, $\alpha_1$ should be the eigen-vector of $C$ corresponding to the largest eigen-value of $C$

What is another orientation $\alpha_2$, orthogonal to $\alpha_1$, and along which the data can have the second largest variation?

Answer: it is the eigen-vector associated to the second largest eigen-value $\lambda_2$ of $C$ and such a variance is $\lambda_2$

Assignment!
Principal Component Analysis (PCA)

• Identify the orientation with largest variance

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.
Principal Component Analysis (PCA)

• Express data in PCs

Suppose \( \{\alpha_1, \alpha_2, ..., \alpha_p\} \) are PCs derived from \( X, X \in \mathbb{R}^{p \times n} \).

Then, a data point \( x_i \in \mathbb{R}^{p \times 1} \) can be linearly represented by \( \{\alpha_1, \alpha_2, ..., \alpha_p\} \), and the representation coefficients are

\[
c_i = \begin{pmatrix}
\alpha_1^T \\
\alpha_2^T \\
\vdots \\
\alpha_p^T 
\end{pmatrix} x_i
\]

Actually, \( c_i \) is the coordinates of \( x_i \) in the new coordinate system spanned by \( \{\alpha_1, \alpha_2, ..., \alpha_p\} \).
Principal Component Analysis (PCA)

• Summary

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

Suppose each of its feature has zero-mean

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

$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p \end{bmatrix}$ 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

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

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

\[
cov(X) = \begin{pmatrix}
5.549 & 5.539 \\
5.539 & 6.449
\end{pmatrix}
\]

Eigen-values = 11.5562, 0.4418

\[
\alpha_1 = \begin{pmatrix}
0.6779 \\
0.7352
\end{pmatrix}
\]

\[
\alpha_2 = \begin{pmatrix}
-0.7352 \\
0.6779
\end{pmatrix}
\]
Principal Component Analysis (PCA)

- Illustration
Principal Component Analysis (PCA)

• Illustration

Coordinates of the data points in the new coordinate system

\[
newC = \begin{pmatrix}
\alpha_1^T \\
\alpha_2^T \\
\end{pmatrix} X \\
\begin{pmatrix}
0.6779 & 0.7352 \\
-0.7352 & 0.6779 \\
\end{pmatrix} X \\
= \begin{pmatrix}
-0.211 & 0.107 & 0.348 & 0.094 & -0.245 & 0.139 & -0.386 & 0.011 & -0.018 & -0.199 \\
\end{pmatrix}
\]
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 $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^{p \times 1}, \{\mathbf{\alpha}_i\}_{i=1}^p, \mathbf{\alpha}_i \in \mathbb{R}^{p \times 1}$ are the PCs

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

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

That is, the dimension of the data is reduced!
Principal Component Analysis (PCA)

• Data dimension reduction with PCA

Suppose \( X = \{x_i\}_{i=1}^{n}, x_i \in \mathbb{R}^{p \times 1} \)

\[
\text{cov}(X) \equiv U \Sigma U^T
\]

\[ U = \begin{bmatrix} u_1, u_2, \ldots, u_m, \ldots, u_p \end{bmatrix} \] spans a new space

For dimension reduction, only \( u_1 \sim u_m \) are used,

\[ U_m = \begin{bmatrix} u_1, u_2, \ldots, u_m \end{bmatrix} \in \mathbb{R}^{p \times m} \]

Data in \( U_m \),

\[ X_{dr} = (U_m)^T X \in \mathbb{R}^{m \times n} \]
Principal Component Analysis (PCA)

- Recovering the dimension-reduction data

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

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

\[
X_{recover} = U \begin{bmatrix}
X_{dr1}, X_{dr2}, \ldots, X_{drn} \\
0 & 0 & 0 \\
\vdots & & \\
0 & 0 & 0
\end{bmatrix}
\]

\[
= U_m X_{dr}
\]
Principal Component Analysis (PCA)

• Illustration

Coordinates of the data points in the new coordinate system

\[ newC = \begin{pmatrix} 0.6779 & 0.7352 \\ -0.7352 & 0.6779 \end{pmatrix} X \]

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

\[ newC = \begin{pmatrix} 0.6779 & 0.7352 \end{pmatrix} X \]

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

\[ newC = \begin{pmatrix} 0.6779 & 0.7352 \end{pmatrix} X \]

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

How to recover \( newC \) to the original space? Easy

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

Eigen-face based face recognition

• Training period
  • Step 1: prepare images \( \{ x_i \} \) 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 \( (u_1, u_2, ..., u_M) \) define the face space
  • Step 4: compute the representation coefficients of each training image \( x_i \) on the \( M \)-d subspace

\[
\mathbf{r}_i = \begin{pmatrix}
    u_{1}^T \\
    u_{2}^T \\
    \vdots \\
    u_{M}^T
\end{pmatrix}
\mathbf{x}_i
\]
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 $100 \times 100$, the covariance matrix $C$ is $10000 \times 10000$

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

The corresponding covariance matrix is

$$
X^T X \in \mathbb{R}^{p \times p}; \text{ very large}
$$

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

$$
X^T X v_i = \lambda_i v_i
$$

Pre-multiply $X$ on both sides

$$
XX^T X v_i = \lambda_i X v_i
$$

$X v_i$ is the eigen-vector of $XX^T$
Eigen-face based face recognition

• Example— training stage

4 classes, 8 samples altogether
Vectorize the 8 images, and stack them into a data matrix $X$
Compute the eigen-faces (PCs) based on $X$
In this example, we retain the first 6 eigen-faces to span the subspace
Eigen-face based face recognition

• Example— training stage

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

\[ u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5 \quad u_6 \]

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

\[
\mathbf{r}_i = \begin{pmatrix}
    u_1^T \\
    u_2^T \\
    \vdots \\
    u_6^T
\end{pmatrix}
\mathbf{x}_i
\]
Eigen-face based face recognition

• Example— training stage

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

\[ = 0.33u_1 - 0.74u_2 + 0.07u_3 - 0.24u_4 + 0.28u_5 + 0.43u_6 \]

\((x_7)\)

\(r_7=(0.33 \ -0.74 \ 0.07 \ -0.24 \ 0.28 \ 0.43)^T\) is the representation vector of the 7th training image
Eigen-face based face recognition

• Example— testing stage

A new image comes, project it to the learned sub-space

\[ t \text{=} \begin{pmatrix} u^T_1 \\ u^T_2 \\ \vdots \\ u^T_6 \end{pmatrix} \text{testI} = 0.52u_1 + 0.17u_2 - 0.01u_3 - 0.39u_4 + 0.67u_5 - 0.29u_6 \]

\[ t \text{=} (0.52 \ 0.17 \ -0.01 \ -0.39 \ 0.67 \ 0.29)^T \] 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

\[ l_2 \text{-norm based distance metric} \]

This guy does not exist in the dataset!

We set threshold = 0.50