Lecture 5
Face Detection and Recognition

Lin ZHANG, PhD
School of Software Engineering
Tongji University
Spring 2015
Any faces contained in the image?
Who are they?

Barack Hussein Obama II, Aug. 04, 1961~

Vladimir Putin, Oct. 07, 1952~
Outline

• Overview

• Face detection
  • Introduction
  • Viola-Jones method

• Face recognition
  • Pre-requisite
  • Principal Component Analysis
  • Eigen-face based approach
  • Sparse representation based approach
  • Collaborative representation based approach
Overview

- Applications of face detection & recognition
Overview

- Applications of face detection & recognition

Hong Kong—Luohu, border control

E-channel
Overview

• Applications of face detection & recognition

National Stadium, Beijing Olympic Games, 2008
Overview

• Applications of face detection & recognition

Check on work attendance
Overview

- Applications of face detection & recognition

Smile detection: embedded in most modern cameras
Overview

• Why is face recognition so difficult?
  • Intra-class variance and inter-class similarity

Images of the same person
Overview

• Why is face recognition so difficult?
  • Intra-class variance and inter-class similarity

Images of twins
Overview

• Different capturing modals

Normal lighting  Infrared lighting  3D

Our focus!

Lin ZHANG, SSE, 2015
Overview

• General architecture
Outline

• Overview

• Face detection
  • Introduction
  • AdaBoost
  • Viola-Jones method

• Face recognition
  • Math pre-requisite
  • Principal Component Analysis
  • Eigen-face based approach
  • Sparse representation based approach
  • Collaborative representation based approach
Introduction

- Identify and locate human faces in an image regardless of their
  - Position
  - Scale
  - Orientation
  - pose (out-of-plane rotation)
  - illumination
Where are the faces, if any?
Introduction

• Why face detection is so important?
  • First step for any fully automatic face recognition system
  • First step in many surveillance systems
Introduction

- Why face detection is so difficult?
  - Pose (out-of-plane rotation)
  - Presence or absence of structural components: beards, mustaches, and glasses
  - Facial expression
  - Occlusion: faces may be partially occluded by other objects
  - Orientation (in-plane rotation)
  - Imaging conditions
Introduction

- Why face detection is so difficult?
Introduction

• Research issues
  • Representation: How to describe a typical face?
  • Scale: How to deal with faces of different sizes?
  • Search strategy: How to spot these faces?
  • Speed: How to speed up the process?
  • Precision: How to locate the faces precisely?
Introduction

• Appearance based methods
  • Train a classifier using positive (and usually negative) examples of faces
  • Representation: different appearance based methods may use different representation schemes
  • Most of the state-of-the-art methods belong to this category

The most successful one: Viola-Jones method!

VJ is based on AdaBoost classifier
Outline

• Overview
• Face detection
  • Introduction
  • AdaBoost
  • Viola-Jones method
• Face recognition
  • Pre-requisite
  • Principal Component Analysis
  • Eigen-face based approach
  • Sparse representation based approach
  • Collaborative representation based approach
AdaBoost (Adaptive Boosting)

- It is a machine learning algorithm\cite{1}
- AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers
- The classifiers it uses can be weak, but as long as their performance is slightly better than random they will improve the final model

\cite{1} Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995
AdaBoost (Adaptive Boosting)

• AdaBoost is an algorithm for constructing a "strong" classifier as a linear combination of simple weak classifiers,

\[ f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \]

• Terminology
  • \( h_t(x) \) is a weak or basis classifier
  • \( H(x) = \text{sgn}(f(x)) \) is the final strong classifier
AdaBoost (algorithm for binary classification)

Given:

- Training set \( (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \), where \( y_i \in \{-1, +1\} \)
- number of iterations \( T \)

Initialize weights for samples \( D_1(i) = 1/m \)

For \( t = 1:T \)

1. find \( h_t = \arg\min_{h_j \in \mathcal{H}} \varepsilon_j, \varepsilon_j = \sum_{i=1}^{m} D_i(i) \left[ h_j(x_i) \neq y_i \right] \)
2. if \( \varepsilon_t \geq 0.5 \), stop;
3. set \( \alpha_t = 0.5 \ln \left( \frac{1-\varepsilon_t}{\varepsilon_t} \right) \)
4. update weights for samples \( D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{\text{Denom}} \)

Outputs the final classifier,

\[
H(x) = \text{sgn} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)
\]
Outline

• Overview

• Face detection
  • Introduction
  • AdaBoost
  • Viola-Jones method

• Face recognition
  • Pre-requisite
  • Principal Component Analysis
  • Eigen-face based approach
  • Sparse representation based approach
  • Collaborative representation based approach
Viola-Jones face detection

- VJ face detector\[^1\]
  - Harr-like features are proposed and computed based on \textit{integral image}
  - A simple and efficient classifier is built by selecting a small number of important features by using AdaBoost
  - Classifiers are combined in a cascade structure which dramatically increases the detection speed

\[^1\] P. Viola and M.J. Jones, “Robust real-time face detection”, IJCV, 2004
• Haar-like features
  • Compute the difference between the sums of pixels within two (or more) rectangular regions
Viola-Jones face detection

• Integral image

  • The integral image at location \((x, y)\) contains the sum of all the pixels above and to the left of \(x, y\), inclusive:

  \[
  ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')
  \]

  where \(i(x, y)\) is the original image

  • By the following recurrence, the integral image can be computed in one pass over the original image

  \[
  s(x, y) = s(x, y - 1) + i(x, y)
  \]

  \[
  ii(x, y) = ii(x - 1, y) + s(x, y)
  \]

  where \(s(x, y)\) is the cumulative row sum, \(s(x, -1) = 0\), and \(ii(-1, y) = 0\)
Viola-Jones face detection

- Haar-like feature can be efficiently computed by using integral image

Actually,

\[ ii(x_1) = A \]
\[ ii(x_2) = A + B \]
\[ ii(x_3) = A + C \]
\[ ii(x_4) = A + B + C + D \]

\[ D = ii(x_4) + ii(x_1) - ii(x_2) - ii(x_3) \]
Viola-Jones face detection

- Haar-like feature can be efficiently computed by using integral image

\[
\begin{align*}
\text{original image } i(x, y) & \\
\text{integral image } ii(x, y) & \\
\end{align*}
\]

How to calculate A-B in integral image?
Viola-Jones face detection

• VJ face detector
  • Main idea
    • Feature selection: select important features
    • Focus of attention: focus on potential regions
    • Use an integral graph for fast feature evaluation
  • Use AdaBoost to learn
    • A set of important features (feature selection); sort them in the order of importance; each feature can be used as a simple (weak) classifier
    • A cascade of classifiers that combine all the weak classifiers to do a difficult task; filter out the regions that most likely do not contain faces
Viola-Jones face detection

- VJ face detector

The first and second features selected by AdaBoost. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.
Viola-Jones face detection

• Rejection cascade
  • Within an image, most sub-images are non-face instances
  • Use smaller and efficient classifiers to reject many negative examples at early stage while detecting almost all the positive instances
  • Simpler classifiers are used to reject the majority of sub-windows
  • More complex classifiers are used at later stage to examine difficult cases
  • Learn the cascade classifier using Adaboost, i.e., learn an ensemble of weak classifiers
• Rejection cascade

Rejection cascade: each node represents a multitree boosted classifier ensemble tuned to rarely miss a true face while rejecting a possibly small fraction of nonfaces.
Viola-Jones face detection

- Implementation
  - VJ face detector has been implemented in OpenCV
  - OpenCV has also provided the training result from a frontal face dataset and the result is contained in “haarcascade_frontalface_alt2.xml”
  - A demo program has been provided on our course website: FaceDetectionEx
Viola-Jones face detection

- Demo time: some examples

original image

VJ face detection result
Viola-Jones face detection

• Demo time: some examples
Viola-Jones face detection

• Summary
  • Three main components
    • Integral image: efficient convolution
    • Use Adaboost for feature selection
    • Use Adaboost to learn the cascade classifier
  • Pros:
    • Fast and fairly robust; runs in real time
  • Cons:
    • Very time consuming in training stage (may take days in training)
    • Requires lots of engineering work
Outline

• Overview
• Face detection
  • Introduction
  • AdaBoost
  • Viola-Jones method
• Face recognition
  • Pre-requisite
  • Principal Component Analysis
  • Eigen-face based approach
  • Sparse representation based approach
  • Collaborative representation based approach
Pre-requisite: 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

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

- In general case

If \( \mathbf{x}_0 \) is a stationary point of \( f(\mathbf{x}) \), \( \mathbf{x} \in \mathbb{R}^{n \times 1} \)

\[
\frac{\partial f}{\partial x_1} \bigg|_{x_0} = 0, \quad \frac{\partial f}{\partial x_2} \bigg|_{x_0} = 0, \ldots, \quad \frac{\partial f}{\partial x_n} \bigg|_{x_0} = 0
\]
Pre-requisite: Lagrange multiplier

- Lagrange multiplier is a strategy for finding the local extremum 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
Pre-requisite: Lagrange multiplier

- Lagrange multiplier is a strategy for finding the local extremum 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: \( F(x; \lambda_1, \ldots, \lambda_m) = f(x) + \sum_{k=1}^{m} \lambda_k g_k(x) \)

\((x_0, \lambda_{10}, \ldots, \lambda_{m0})\) 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!
Pre-requisite: 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 $F(x, y, \lambda)$
Pre-requisite: 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 \( p_0 \).

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

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

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

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

\( (0.5, 0.5) \) is a stationary point of \( f(x, y) \) under constraints
Pre-requisite: matrix differentiation

• Function is a vector and the variable is a scalar

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

Definition

\[ \frac{df}{dt} = \left[ \frac{df_1}{dt}, \frac{df_2}{dt}, ..., \frac{df_n}{dt} \right]^T \]
Pre-requisite: matrix differentiation

- Function is a matrix and the variable is a scalar

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

Definition

\[ \frac{df}{dt} = \begin{bmatrix} \frac{df_{11}(t)}{dt} & \frac{df_{12}(t)}{dt}, \ldots, \frac{df_{1m}(t)}{dt} \\ \frac{df_{21}(t)}{dt} & \frac{df_{22}(t)}{dt}, \ldots, \frac{df_{2m}(t)}{dt} \\ \vdots \\ \frac{df_{n1}(t)}{dt} & \frac{df_{n2}(t)}{dt}, \ldots, \frac{df_{nm}(t)}{dt} \end{bmatrix} = \begin{bmatrix} \frac{df_{ij}(t)}{dt} \end{bmatrix}_{n \times m} \]
Pre-requisite: 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**

\[
\frac{df}{dx} = \left[ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right]^T
\]

In a similar way,

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

\[
\frac{df}{dx} = \left[ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right]
\]
Pre-requisite: matrix differentiation

- Function is a vector and the variable is a vector
  \[ \mathbf{x} = [x_1, x_2, \ldots, x_n]^T, \quad \mathbf{y} = [y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_m(\mathbf{x})]^T \]

Definition

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

• Function is a vector and the variable is a vector

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

In a similar way,

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

- Function is a vector and the variable is a vector

Example:

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

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

- Function is a scalar and the variable is a matrix

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

Definition

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

• Useful results

(1) \( x, a \in \mathbb{R}^{n\times1} \)

Then,

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

How to prove?
Pre-requisite: matrix differentiation

• Useful results

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

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

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

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

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

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

• Overview
• Face detection
  • Introduction
  • Viola-Jones method
• Face recognition
  • Pre-requisite
  • Principal Component Analysis
  • Eigen-face based approach
  • Sparse representation based approach
  • Collaborative representation based approach
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.
- The number of principal components is less than or equal to the number of original variables.
- 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 the data points scatter most?

How to find?

De-correlation!
Principal Component Analysis (PCA)

• Identify the orientation with largest variance

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

$$X = \{x_1, x_2, \ldots, x_n\}, x_i \in \mathbb{R}^{p \times 1}, X \in \mathbb{R}^{p \times n}$$

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

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

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

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

$\alpha$ 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 $\lambda_2$
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

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

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)

• Illustration

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

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

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

Eigen-values = 11.5562, 0.4418

Corresponding eigen-vectors:

\[ \alpha_1 = \begin{pmatrix} 0.6779 \\ 0.7352 \end{pmatrix} \]
\[ \alpha_2 = \begin{pmatrix} -0.7352 \\ 0.6779 \end{pmatrix} \]
Principal Component Analysis (PCA)

- Illustration
Principal Component Analysis (PCA)

• Illustration

Coordinates of the data points in the new coordinate system

\[\text{newC} = \begin{pmatrix} \alpha_1^T \\ \alpha_2^T \end{pmatrix} \mathbf{x} \]

\[\begin{pmatrix} 0.6779 & 0.7352 \\ -0.7352 & 0.6779 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \\ -0.211 & 0.107 & 0.348 & 0.094 & -0.245 & 0.139 & -0.386 & 0.011 & -0.018 & -0.199 \end{pmatrix} \]
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)

• 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

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

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

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

\[ = \begin{pmatrix} 0.6779 & 0.7352 \end{pmatrix} \times \begin{pmatrix} 3.459 & 0.854 & 3.623 & 2.905 & 4.307 & 3.544 & 2.532 & 1.487 & 2.193 & 1.407 \end{pmatrix} \]
Principal Component Analysis (PCA)

• Illustration

Data recovered if only 1 PC used

Original data
Outline

• Overview

• Face detection
  • Introduction
  • Viola-Jones method

• Face recognition
  • Pre-requisite
  • Principal Component Analysis
  • Eigen-face based approach
  • Sparse representation based approach
  • Collaborative representation based approach
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 \( (u_1, u_2, \ldots, u_M) \) define the face space
  • Step 4: compute the representation coefficients of each training image \( x_i \) on the \( M \)-d subspace

\[
\mathbf{r}_i = \begin{pmatrix}
    u_1^T \\
    u_2^T \\
    \vdots \\
    u_M^T
\end{pmatrix}
\begin{pmatrix}
    x_i
\end{pmatrix}
\]
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 \( \frac{1}{n-1}XX^T \in \mathbb{R}^{p \times p} \); very large

Instead, we perform eigen-value decomposition to \( X^TX \in \mathbb{R}^{n \times n} \)

\[ X^TXv_i = \lambda_i v_i \]

Pre-multiply \( X \) on both sides

\[ XX^TXv_i = \lambda_i XX^Tv_i \]

\( Xv_i \) is the eigen-vector of \( XX^T \)
• 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 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

\[ r_i = \begin{pmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_6^T \end{pmatrix} x_i \]
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

\[ t = \begin{pmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_6^T \end{pmatrix} \]

\[ t = 0.52u_1 + 0.17u_2 - 0.01u_3 - 0.39u_4 + 0.67u_5 - 0.29u_6 \]

\[ t = (0.52 \ 0.17 \ -0.01 \ -0.39 \ 0.67 \ 0.29)^T \] is the representation vector of this testing image

Lin ZHANG, SSE, 2015
Eigen-face based face recognition

• Example— testing stage

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

This guy should be Lin!

Lin ZHANG, SSE, 2015
Eigen-face based face recognition

• Example— testing stage

We set threshold = 0.50
This guy does not exist in the dataset!
Outline

• Overview
• Face detection
  • Introduction
  • Viola-Jones method
• Face recognition
  • Pre-requisite
  • Principal Component Analysis
  • Eigen-face based approach
  • Sparse representation based approach
  • Collaborative representation based approach
Sparse representation based approach

- Motivations
  - Signals are sparse in some selected domain
  - It has strong physiological support
Sparse representation based approach

- SR-based face recognition
  - It was proposed in [1]
  - In such a system, the choice of features is no longer crucial
  - It is robust to occlusion and corruption

Sparse representation based approach

• Illustration

\[ y = \alpha_{1,1} v_{1,1} + \alpha_{1,2} v_{1,2} + \alpha_{2,1} v_{2,1} + \alpha_{2,2} v_{2,2} \]

\[ + \alpha_{3,1} v_{3,1} + \alpha_{3,2} v_{3,2} + \alpha_{4,1} v_{4,1} + \alpha_{4,2} v_{4,2} \]

We expect that all the coefficients are zero except \( \alpha_{3,1}, \alpha_{3,2} \)
Sparse representation based approach

• Problem formulation

We define a matrix $A$ for the $n$ training samples of all $k$ object classes

$$A = [A_1, A_2, ..., A_k] = \begin{bmatrix} v_{1,1}, v_{1,2}, ..., v_{k,n_k} \end{bmatrix}$$

Then, the linear representation of a testing sample $y$ can be expressed as

$$y = Ax_0$$

where $x_0 = \begin{bmatrix} 0, ..., 0, \alpha_{i,1}, \alpha_{i,2}, ..., \alpha_{i,n_i}, 0, ..., 0 \end{bmatrix}^T \in \mathbb{R}^n$ is a coefficient vector whose entries are zero except those associated with the $i$th class.
Sparse representation based approach

This motivates us to seek the most sparsest solution to $y = Ax$, solving the following optimization problem:

$$x_0 = \arg \min \|x\|_0, \text{s.t. } \|Ax - y\|_2 \leq \varepsilon \quad (1)$$

where $\|\cdot\|_0$ denotes the $l_0$-norm, which counts the number of non-zero entries in a vector.

However, solving (1) is a NP-hard problem, though some approximation solutions can be found efficiently.

Thus, usually, (1) can be rewritten as a $l_1$-norm minimization problem.
Sparse representation based approach

If the solution $x_0$ is sparse enough, the solution of $l_0$-minimization problem is equal to the solution to the following $l_1$-norm minimization problem:

$$x_0 = \arg \min \| x \|_1, \text{s.t., } \| Ax - y \|_2 \leq \varepsilon \quad (1)$$

The above minimization problem could be solved in polynomial time by standard linear programming methods.

There is an equivalent form for (1)

$$x_0 = \arg \min_x \| y - Ax \|_2^2 + \lambda \| x \|_1 \quad (2)$$

Several different methods for solving $l_1$-norm minimization problem in the literature, such as the $l_1$-magic method (refer to the course website)
Sparse representation based approach

Algorithm

1. **Input**: a matrix of training samples
   \[ A = [A_1, A_2, \ldots, A_k] \in \mathbb{R}^{m \times n} \] for \( k \) classes; \( y \in \mathbb{R}^m \), a test sample; and an error tolerance \( \varepsilon > 0 \)

2. Normalize the columns of \( A \) to have unit \( l_2 \)-norm

3. Solve the \( l_1 \)-minimization problem
   \[ x_0 = \arg \min \|x\|_1, \text{ s.t., } \|Ax - y\|_2 \leq \varepsilon \]

4. Compute the residuals
   \[ r_i(y) = \|y - A \delta_i(x_0)\|_2, \quad i = \{1, \ldots, k\} \]

5. **Output**: identity(\( y \)) = \( \arg \min_i r_i(y) \)

---

For \( x \in \mathbb{R}^n \), \( \delta_i(x) \in \mathbb{R}^n \) is a new vector whose only non-zero entries are the entries in \( x \) that are associated with class \( i \)
Sparse representation based approach

- Illustration

A valid test image. Recognition with $12 \times 10$ downsampled images as features. The test image $y$ belongs to subject 1. The values of the sparse coefficients recovered are plotted on the right together with the two training examples that correspond to the two largest sparse coefficients.
The residuals $r_i(y)$ of a test image of subject 1 with respect to the projected sparse coefficients $\delta_i(x_0)$ by $l_1$-minimization.
Sparse representation based approach

• Summary

• It provides a novel idea for face recognition
• By solving the sparse minimization problem, the “position” of the big coefficients can indicate the category of the examined image
• It is robust to occlusion and partial corruption
Outline

• Overview
• Face detection
  • Introduction
  • Viola-Jones method
• Face recognition
  • Pre-requisite
  • Principal Component Analysis
  • Eigen-face based approach
  • Sparse representation based approach
  • Collaborative representation based approach
Collaborative representation based classification with regularized least square was proposed in [1].

Motivation

- SRC method is based on $l_1$-minimization; however, $l_1$-minimization is time consuming. So, is it really necessary to solve the $l_1$-minimization problem for face recognition?
- Is it $l_1$-minimization or the collaborative representation that makes SRC work?

CRC_RLS

• Key points of CRC_RLS
  • It is the collaborative representation, not the $l_1$-norm minimization that makes the SRC method works well for face recognition
  • Thus, the $l_1$-norm regularization can be relaxed to $l_2$-norm regularization
CRC_RLS

SRC method:

\[ x_0 = \arg \min_x \| y - Ax \|_2^2 + \lambda \| x \|_1 \]  \hspace{1cm} (1)

CRC_RLS:

\[ x_0 = \arg \min_x \| y - Ax \|_2^2 + \lambda \| x \|_2^2 \]  \hspace{1cm} (2)

(1) is not easy to solve; can be solved by iteration methods.

However, (2) has a closed-form solution:

\[ x_0 = \left( A^T A + \lambda E \right)^{-1} A^T y \]

can be pre-computed.

Can you work it out?
Algorithm

1. **Input**: a matrix of training samples
   \[ A = \begin{bmatrix} A_1, A_2, \ldots, A_k \end{bmatrix} \in \mathbb{R}^{m \times n} \text{ for } k \text{ classes}; y \in \mathbb{R}^m, \text{ a test sample}; \]
2. Normalize the columns of \( A \) to have unit \( l_2 \)-norm
3. Pre-compute \( P = \left( A^T A + \lambda E \right)^{-1} A^T \)
4. Code \( y \) over \( A \)
   \[ x_0 = Py \]
5. Compute the residuals \( r_i(y) = \|y - A \hat{\delta}_i(x_0)\|_2, i = \{1, \ldots, k\} \)
6. **Output**: identity(\( y \)) = argmin_i r_i(y)

For \( x \in \mathbb{R}^n, \hat{\delta}_i(x) \in \mathbb{R}^n \) is a new vector whose only non-zero entries are the entries in \( x \) that are associated with class \( i \)
By solving CRC_RLS,

\[ x_0 = [-0.10, -0.04, -0.09, 0.16, 0.68, 0.14, 0.06, 0.17]^T \]

\[ r_1 = \| \mathbf{v}_{1,1} \times (-0.10) + \mathbf{v}_{1,2} \times (-0.04) - y \|_2 = 1.14 \]

\[ r_2 = \| \mathbf{v}_{2,1} \times (-0.09) + \mathbf{v}_{2,2} \times (0.16) - y \|_2 = 0.93 \]

\[ r_3 = \| \mathbf{v}_{3,1} \times (0.68) + \mathbf{v}_{3,2} \times (0.14) - y \|_2 = 0.27 \]

\[ r_4 = \| \mathbf{v}_{4,1} \times (0.06) + \mathbf{v}_{4,2} \times (0.17) - y \|_2 = 0.79 \]
CRC_RLS

- CRC_RLS vs. SRC

The coding coefficients of a query sample
CRC_RLS

- CRC_RLS vs. SRC

<table>
<thead>
<tr>
<th>Method</th>
<th>Recognition rate</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRC($l_1$-ls)</td>
<td>0.979</td>
<td>5.3988 s</td>
</tr>
<tr>
<td>SRC(ALM)</td>
<td>0.979</td>
<td>0.128 s</td>
</tr>
<tr>
<td>SRC(FISTA)</td>
<td>0.914</td>
<td>0.1567 s</td>
</tr>
<tr>
<td>SRC(Homotopy)</td>
<td>0.945</td>
<td>0.0279 s</td>
</tr>
<tr>
<td><strong>CRC_RLS</strong></td>
<td><strong>0.979</strong></td>
<td><strong>0.0033 s</strong></td>
</tr>
</tbody>
</table>

**Speed-up** 8.5 ~ 1636 times

Recognition rate and speed on the Extended Yale B database
Thanks for your attention