Lecture 17: Face Recognition

Dr. Juan Carlos Niebles
Stanford AI Lab

Professor Fei-Fei Li
Stanford Vision Lab
What we will learn today

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


“Faces” in the brain

Courtesy of Johannes M. Zanker
“Faces” in the brain

Kanwisher, et al. 1997
Face Recognition

• Digital photography
Face Recognition

- Digital photography
- Surveillance

Matching with Database
- Name: Alireza,
  Date: 25 My 2007 15:45
  Place: Main corridor

Detecting....

Name: Unknown
Date: 25 My 2007 15:45
Place: Main corridor
Face Recognition

- Digital photography
- Surveillance
- Album organization
Face Recognition

- Digital photography
- Surveillance
- Album organization
- Person tracking/id.
Face Recognition

- Digital photography
- Surveillance
- Album organization
- Person tracking/id.
- Emotions and expressions
Face Recognition

• Digital photography
• Surveillance
• Album organization
• Person tracking/id.
• Emotions and expressions
• Security/warfare
• Tele-conferencing
• Etc.
The Space of Faces

- An image is a point in a high dimensional space
  - If represented in grayscale intensity, an $N \times M$ image is a point in $\mathbb{R}^{NM}$
  - E.g. 100x100 image = 10,000 dim
The Space of Faces

- An image is a point in a high dimensional space
  - If represented in grayscale intensity, an N x M image is a point in $\mathbb{R}^{NM}$
  - E.g. 100x100 image = 10,000 dim

- However, relatively few high dimensional vectors correspond to valid face images

- We want to effectively model the subspace of face images
• Compute n-dim subspace such that the projection of the data points onto the subspace has the largest variance among all n-dim subspaces.
• Maximize the scatter of the training images in face space
Key Idea

- 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.
What we will learn today

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


PCA Formulation

• Basic idea:
  – If the data lives in a subspace, it is going to look very flat when viewed from the full space, e.g.
  
  \[ \text{1D subspace in 2D} \]

  – This means that if we fit a Gaussian to the data the equiprobability contours are going to be highly skewed ellipsoids.

Slide inspired by N. Vasconcelos
PCA Formulation

- If $x$ is Gaussian with covariance $\Sigma$, the equiprobability contours are the ellipses whose
  
  - Principal components $\phi_i$ are the eigenvectors of $\Sigma$
  - Principal lengths $\lambda_i$ are the eigenvalues of $\Sigma$

- by computing the eigenvalues we know the data is
  
  - Not flat if $\lambda_1 \approx \lambda_2$
  - Flat if $\lambda_1 >> \lambda_2$
Given sample $\mathcal{D} = \{x_1, \ldots, x_n\}$, $x_i \in \mathcal{R}^d$

- compute sample mean: $\hat{\mu} = \frac{1}{n} \sum_i (x_i)$

- compute sample covariance: $\hat{\Sigma} = \frac{1}{n} \sum_i (x_i - \hat{\mu})(x_i - \hat{\mu})^T$

- compute eigenvalues and eigenvectors of $\hat{\Sigma}$

$$\hat{\Sigma} = \Phi \Lambda \Phi^T, \quad \Lambda = diag(\sigma_1^2, \ldots, \sigma_n^2) \quad \Phi^T \Phi = I$$

- order eigenvalues $\sigma_1^2 > \ldots > \sigma_n^2$

- if, for a certain $k$, $\sigma_k << \sigma_1$ eliminate the eigenvalues and eigenvectors above $k$. 

Slide inspired by N. Vasconcelos
Given principal components $\phi_i, i \in 1, \ldots, k$ and a test sample $T = \{t_1, \ldots, t_n\}, t_i \in \mathbb{R}^d$

- subtract mean to each point $t_i' = t_i - \mu$

- project onto eigenvector space $y_i = At_i'$ where

\[
A = \begin{bmatrix}
\phi_1^T \\
\vdots \\
\phi_k^T
\end{bmatrix}
\]

- use $T' = \{y_1, \ldots, y_n\}$ to estimate class conditional densities and do all further processing on $y$. 

Slide inspired by N. Vasconcelos
PCA by SVD

• An alternative manner to compute the principal components, based on singular value decomposition

• Quick reminder: SVD
  – Any real n x m matrix (n>m) can be decomposed as

\[ A = M \Sigma N^T \]

  – Where M is an (n x m) column orthonormal matrix of left singular vectors (columns of M)
  – \( \Sigma \) is an (m x m) diagonal matrix of singular values
  – \( N^T \) is an (m x m) row orthonormal matrix of right singular vectors (columns of N)

\[ M^T M = I \quad N^T N = I \]

Slide inspired by N. Vasconcelos
PCA by SVD

- To relate this to PCA, we consider the data matrix

$$X = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}$$

- The sample mean is

$$\mu = \frac{1}{n} \sum_i x_i = \frac{1}{n} \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \frac{1}{n} X 1$$
PCA by SVD

- Center the data by subtracting the mean to each column of $X$
- The centered data matrix is

$$X_c = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} - \begin{bmatrix} \mu & \cdots & \mu \end{bmatrix}$$

$$= X - \mu 1^T = X - \frac{1}{n} X 11^T = X \left( I - \frac{1}{n} 11^T \right)$$

Slide inspired by N. Vasconcelos
PCA by SVD

- The sample covariance matrix is

\[
\Sigma = \frac{1}{n} \sum_{i} (x_i - \mu)(x_i - \mu)^T = \frac{1}{n} \sum_{i} x_i^c (x_i^c)^T
\]

where \(x_i^c\) is the \(i^{th}\) column of \(X_c\)

- This can be written as

\[
\Sigma = \frac{1}{n} \begin{bmatrix} x_1^c & \ldots & x_n^c \end{bmatrix} \begin{bmatrix} - & x_1^c & - \\ \vdots & \vdots & \vdots \\ - & x_n^c & - \end{bmatrix} = \frac{1}{n} X_c X_c^T
\]
PCA by SVD

- The matrix

\[ X_c^T = \begin{bmatrix} - & X_1^c & - \\ & \vdots & \\ - & X_n^c & - \end{bmatrix} \]

is real (n x d). Assuming n>d it has SVD decomposition

\[ X_c^T = MN^T \]

\[ M^TM = I \quad N^TN = I \]

and

\[ \Sigma = \frac{1}{n} X_cX_c^T = \frac{1}{n} N\Pi\Pi^TN^T = \frac{1}{n} \Pi\Pi^2N^T \]
PCA by SVD

Note that $N$ is $(d \times d)$ and orthonormal, and $\Pi^2$ is diagonal. This is just the eigenvalue decomposition of $\Sigma$

It follows that

- The eigenvectors of $\Sigma$ are the columns of $N$
- The eigenvalues of $\Sigma$ are

$$\lambda_i = \frac{1}{n} \pi_i$$

This gives an alternative algorithm for PCA
In summary, computation of PCA by SVD

1. Given $X$ with one example per column
   - Create the centered data matrix
     \[
     X_c^T = \left( I - \frac{1}{n} \textbf{1}\textbf{1}^T \right) X^T
     \]
   - Compute its SVD
     \[
     X_c^T = \text{M}\text{P}\text{IN}^T
     \]
   - Principal components are columns of $N$, eigenvalues are
     \[
     \lambda_i = \frac{1}{n} \pi_i^2
     \]
Rule of thumb for finding the number of PCA components

- A natural measure is to pick the eigenvectors that explain p% of the data variability
  - Can be done by plotting the ratio $r_k$ as a function of $k$

- E.g. we need 3 eigenvectors to cover 70% of the variability of this dataset

$\sum_{i=1}^{k} \lambda_i^2$ \over $\sum_{i=1}^{n} \lambda_i^2$
What we will learn today

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


Eigenfaces: key idea

• Assume that most face images lie on a low-dimensional subspace determined by the first $k$ ($k<<d$) directions of maximum variance
• Use PCA to determine the vectors or “eigenfaces” that span that subspace
• Represent all face images in the dataset as linear combinations of eigenfaces

Eigenface algorithm

• Training

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

2. Compute average face $\mu = \frac{1}{N} \sum x_i$

3. Compute the difference image (the centered data matrix)

$$X_c = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} - \begin{bmatrix} \mu & \cdots & \mu \end{bmatrix}$$

$$= X - \mu 1^T = X - \frac{1}{n} X 11^T = X \left( I - \frac{1}{n} 11^T \right)$$

Note that each image is formulated into a long vector!
Eigenface algorithm

4. Compute the covariance matrix

\[
\Sigma = \frac{1}{n} \begin{bmatrix} x_1^c & \cdots & x_n^c \\ \vdots & \ddots & \vdots \\ x_1^c & \cdots & x_n^c \end{bmatrix} \begin{bmatrix} - & - & - \\ - & - & - \\ - & - & - \end{bmatrix} = \frac{1}{n} X_c X_c^T
\]

5. Compute the eigenvectors of the covariance matrix \( \Sigma \)

6. Compute each training image \( x_i \) 's projections as

\[
x_i \rightarrow (x_i^c \cdot \phi_1, x_i^c \cdot \phi_2, \ldots, x_i^c \cdot \phi_K) \equiv (a_1, a_2, \ldots, a_K)
\]

7. Visualize the estimated training face \( x_i \)

\[
x_i \approx \mu + a_1 \phi_1 + a_2 \phi_2 + \ldots + a_K \phi_K
\]
6. Compute each training image $x_i$’s projections as

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

7. Visualize the estimated training face $x_i$

$$x_i \approx \mu + a_1 \phi_1 + a_2 \phi_2 + \ldots + a_K \phi_K$$
Eigenface algorithm

• Testing
  1. Take query image \( t \)
  2. Project into eigenface space and compute projection

\[
t \rightarrow ((t - \mu) \cdot \phi_1, (t - \mu) \cdot \phi_2, ..., (t - \mu) \cdot \phi_K) \equiv (w_1, w_2, ..., w_K)
\]

3. 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)
Visualization of eigenfaces

Eigenfaces look somewhat like generic faces.
Reconstruction and Errors

- Only selecting the top $K$ eigenfaces $\rightarrow$ reduces the dimensionality.
- Fewer eigenfaces result in more information loss, and hence less discrimination between faces.
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...

  • See supplementary materials for “Linear Discriminative Analysis”, aka “Fisherfaces”
What we will learn today

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


Fischer’s Linear Discriminant Analysis

- Goal: find the best separation between two classes

Slide inspired by N. Vasconcelos
Basic intuition: PCA vs. LDA
Linear Discriminant Analysis (LDA)

• We have two classes such that

\[ E_{X|Y}[X | Y = i] = \mu_i \]

\[ E_{X|Y}[(X - \mu_i)(X - \mu_i)^T | Y = i] = \Sigma_i \]

• We want to find the line \( z \) that best separates them

\[ z = w^T x \]

• One possibility would be to maximize

\[
\left( E_{Z|Y}[Z | Y = 1] - E_{Z|Y}[Z | Y = 0] \right)^2 = \\
\left( E_{X|Y}[w^T x | Y = 1] - E_{X|Y}[w^T x | Y = 0] \right)^2 = (w^T [\mu_1 - \mu_0])^2
\]

Slide inspired by N. Vasconcelos
Linear Discriminant Analysis (LDA)

• However, this difference

\[ (w^T [\mu_1 - \mu_0])^2 \]

can be arbitrarily large by simply scaling \( w \)

• We are only interested in the direction, not the magnitude

• Need some type of normalization

• Fisher suggested

\[
\max_w \frac{\text{between class scatter}}{\text{within class scatter}} = \max_w \frac{(E_{Z|Y}[Z|Y=1] - E_{Z|Y}[Z|Y=0])^2}{\text{var}[Z|Y=1] + \text{var}[Z|Y=0]}
\]
Linear Discriminant Analysis (LDA)

• We have already seen that

\[
\left( E_{z|y}[z \mid y=1] - E_{z|y}[z \mid y=0] \right)^2 = \left( w^T [\mu_1 - \mu_0] \right)^2
= w^T [\mu_1 - \mu_0] [\mu_1 - \mu_0]^T w
\]

• also

\[
\text{var}[z \mid y=i] = E_{z|y} \left\{ \left( z - E_{z|y}[z \mid y=i] \right)^2 \mid y = i \right\}
= E_{z|y} \left\{ \left( w^T [x - \mu_i] \right)^2 \mid y = i \right\}
= E_{z|y} \left\{ w^T [x - \mu_i] [x - \mu_i]^T w \mid y = i \right\}
= w^T \Sigma_i w
\]
Linear Discriminant Analysis (LDA)

• And

\[ J(w) = \frac{(E_{Z|Y}[Z|Y = 1] - E_{Z|Y}[Z|Y = 0])^2}{\text{var}[Z|Y = 1] + \text{var}[Z|Y = 0]} \]

\[ = \frac{w^T (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w}{w^T (\Sigma_1 + \Sigma_0)w} \]

• which can be written as

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

\[ S_B = (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T \]

\[ S_W = (\Sigma_1 + \Sigma_0) \]

between class scatter

within class scatter

Slide inspired by N. Vasconcelos
Visualization

\[ S_W = S_1 + S_2 \]

Within class scatter

Between class scatter

Fei-Fei Li
Linear Discriminant Analysis (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.

\[ \max_w w^T S_B w \text{ subject to } w^T S_W w = K \]

– 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 \( \lambda \)

Slide inspired by N. Vasconcelos
Linear Discriminant Analysis (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

Slide inspired by N. Vasconcelos
Linear Discriminant Analysis (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 \( w \)
  \[ w^* = S_w^{-1} (\mu_1 - \mu_0) = (\Sigma_1 + \Sigma_0)^{-1} (\mu_1 - \mu_0) \]
PCA vs. LDA

- Eigenfaces exploit the max scatter of the training images in face space.
- Fisherfaces attempt to maximise the between class scatter, while minimising the within class scatter.
Results: Eigenface vs. Fisherface (1)

- Input: 160 images of 16 people
- Train: 159 images
- Test: 1 image

- Variation in Facial Expression, Eyewear, and Lighting

With glasses  |  Without glasses  |  3 Lighting conditions  |  5 expressions

![Images showing variations in facial expression, eyewear, and lighting conditions.](image-url)
Eigenface vs. Fisherface (2)

![Graph showing error rate vs. number of principal components for Eigenface and Fisherface methods. The graph compares the performance of Eigenface and Fisherface in terms of error rate and number of principal components. Eigenface shows a significant decrease in error rate as the number of principal components increases, whereas Fisherface maintains a lower and more stable error rate. The graph indicates that Fisherface performs better than Eigenface, especially when the number of principal components is low.]
What we have learned today

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