Linear dimensionality reduction with application to face recognition

Slide credits: Lihi Zelnik-Manor
Today

- Motivation and background for face recognition
- Methods for face recognition
  - Eigenfaces: Principal Component Analysis (PCA)
  - Fisherfaces: Linear Discriminant Analysis (LDA)
Face recognition – why?

- Digital photography
Face recognition – why?

- Digital photography
- Surveillance
Face recognition – why?

- Digital photography
- Surveillance
- Photo albums
Face recognition – why?

- Digital photography
- Surveillance
- Photo albums
- Person tracking
Face recognition – why?

- Digital photography
- Surveillance
- Photo albums
- Person tracking
- Emotions and expressions
Face recognition – why?

- Digital photography
- Surveillance
- Photo albums
- Person tracking
- Emotions and expressions
- Security
- Tele-conferencing
- Etc.
Recognition vs. detection

Yes, there are faces

No localization

Identification

Categorization
Recognition vs. detection

Yes, there is a “John Lennon”

No localization

Identification

Categorization
Recognition vs. detection

Detection

No localization

Identification

Categorization

John Lennon
Recognition vs. detection

Detection

No localization

Identification

Categorization
Recognition vs. detection

- Detection
  - PCA & Eigenfaces [Turk & Pentland 1991]
  - LDA & Fisherfaces [Bullumeur et al, 1997]
  - AdaBoost [Viola & Jones, 2001]

- No localization
  - Identification
  - Categorization
Eigenfaces

Turk & Pentland,
Eigenfaces for Recognition,
An image is a point in high-dimensions
Eigenimages

Database

\[
\begin{pmatrix}
\vdots
\end{pmatrix}
\begin{pmatrix}
\vdots
\end{pmatrix}
= \begin{pmatrix}
\vdots
\end{pmatrix}
\]

Basis spanning face subspace

Query

Search in low-dimensions

Coefficients describing specific faces
Eigenfaces – key idea

1. Reduce dimension using PCA
2. Recognize using Euclidean distance
PCA
Principal Component Analysis

A simple method for dimensionality reduction
Dimensionality reduction
Dimensionality reduction

We can represent the orange points with *only* their $v_1$ coordinates
- since $v_2$ coordinates are all approximately 0
- This makes it much cheaper to store and compare points
- A bigger deal for higher dimensional problems

How do we find $v_1$ and $v_2$?
Consider the variation along direction \( v \) among all of the orange points:

\[
\text{var}(v) = \sum_{\text{orange point } x} \| (x - \bar{x})^T \cdot v \|^2
\]

- What unit vector \( v \) minimizes \( \text{var}(v) \)? \( v_2 = \min_v \{ \text{var}(v) \} \)
- What unit vector \( v \) maximizes \( \text{var}(v) \)? \( v_1 = \max_v \{ \text{var}(v) \} \)
PCA – main idea

Solution:

- \( v_1 \) is eigenvector of \( A \) with largest eigenvalue
- \( v_2 \) is eigenvector of \( A \) with smallest eigenvalue

\[
\begin{align*}
\text{var}(v) &= \sum_x \|(x - \bar{x})^T \cdot v\| \\
&= \sum_x v^T (x - \bar{x})(x - \bar{x})^Tv \\
&= v^T \left[ \sum_x (x - x)(x - x)^T \right] v \\
&= v^T A v \quad \text{where} \quad A = \sum_x (x - \bar{x})(x - \bar{x})^T
\end{align*}
\]
PCA – main idea

\( \overline{x} \) is the mean of the orange points

\[ y = W^T(x - x_0) \]

- Solution:
  - \( v_1 \) is eigenvector of \( A \) with largest eigenvalue
  - \( v_2 \) is eigenvector of \( A \) with smallest eigenvalue
PCA – high dimension

- We seek a linear transformation $W$ such that
  $$y_j = W^T x_j \quad j = 1,2, \ldots N$$

- $x_j = n \times 1$ – original image in high dimension
- $y_j = m \times 1$ – low dimension $m \ll n$
- $W = n \times m$ orthonormal

- **Solution:**
  $$W = \text{Top } m \text{ eigenvectors of } \text{cov}(x)$$
PCA - algorithm

1. Input: $x_j, \ j = 1,2, \ldots N$
2. Compute mean: $\bar{x}$
3. Compute difference from mean $x_j - \bar{x}$
4. Compute covariance matrix $\text{cov}(x)$
5. Compute eigenvectors
6. Build $W$ from the top $m$ eigenvectors
7. Compute low-dimension representation
   \[ y_j = W^T(x_j - \bar{x}) \]

In practice- $\text{cov}(x)$ is very large! Using a manipulation of SVD instead…
Illustration of Eigenfaces
Recognition with eigenfaces

- **Training:**
  1. Align training face images $x_1, \ldots, x_N$
  2. Run PCA to obtain $W, \bar{x}$ and $y_1, \ldots, y_N$
Recognition with eigenfaces

- **Testing:**
  1. Take a query image $x$
  2. Subtract the average face and project onto low-dim subspace $y = W^T(x - \bar{x})$
  3. Compare $y$ with all training images $y_1, \ldots, y_N$
Projection and reconstruction

- Each face can be approximated from its low-dimensional representation:

\[ \hat{x}_j = \bar{x} + W y_j \]
Reconstruction and errors

- Selecting the top $m$ eigenvectors implies information loss
Choosing the dimension $m$

- How many eigenfaces to use?
- Look at the decay of the eigenvalues
  - the eigenvalue tells you the amount of variance “in the direction” of that eigenface
  - ignore eigenfaces with low variance

![Graph showing the decay of eigenvalues](image)
Choosing the dimension $m$

- How many eigenfaces to use?
- Look at $r_k$:

we need 3 eigenvectors to cover 70% of the variability of this dataset
Eigenfaces - summary

- **Pros:**
  - Non-iterative
  - Simple

- **Cons:**
  - Requires face alignment
  - Non-localized
  - PCA is optimal for reconstruction but may not be optimal for discriminating faces
Recognition vs. detection

- Detection
  - No localization
  - Identification
  - Categorization

- Recognition
  - PCA & Eigenfaces
    - [Turk & Pentland 1991]
  - LDA & Fisherfaces
    - [Bullumeur et al, 1997]
  - AdaBoost
    - [Viola & Jones, 2001]
Fisherfaces

P.N. Belhumeur, J. P. Hespanha, D.J. Kriegman; *Eigenfaces vs. Fisherfaces: recognition using class specific linear projection*; IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), 1997
Fisherfaces
FLD: Intro

- Learn a linear classifier \(\{w, b\}\) using a training set.
- Classify the test set according to \(\text{sign}\{w^T x - b\}\)
Fisher Linear Discriminant

- Our goal is to find a linear projection that discriminates between the classes.
Fisher Linear Discriminant

- Our goal is to find a linear projection that discriminates between the classes.
Fisher Linear Discriminant

Maximize the between class scatter
Fisher Linear Discriminant

Minimize the within class scatter
FLD: scatter matrices

\[ N \text{ vectors (features, images): } \begin{bmatrix} x_1, \ldots, x_N \end{bmatrix} \]

\[ K \text{ classes: } \begin{bmatrix} c_1, \ldots, c_K \end{bmatrix} \]

Mean of each class:
\[ \mu_k = \frac{1}{N_k} \sum_{x_i \in c_k} x_i \]

Mean of all vectors:
\[ \mu = \frac{1}{N} \sum_{i=1}^{N} x_i \]

Scatter of class \( k \)
\[ S_k = \sum_{x_i \in c_k} (x_i - \mu_k)(x_i - \mu_k)^T \]

Total within class scatter:
\[ S_W = \sum_{k=1}^{K} S_k \]

Between class scatter
\[ S_B = \sum_{k=1}^{K} N_k \left| (\mu - \mu_k)(\mu - \mu_k)^T \right| \]
FLD: practice (1)

• After projection: 

\[ y_i = W^T x_i \]

• Between class scatter (of y’s): 

\[ S_B(y) = W^T S_B(x)W \]

• Within class scatter (of y’s): 

\[ S_W(y) = W^T S_W(x)W \]
The wanted projection:

\[ W_{\text{optimal}} = \arg \max_{W} \frac{\mid S_B(y) \mid}{\mid S_W(y) \mid} = \arg \max_{W} \frac{\mid W^T S_B(x) W \mid}{\mid W^T S_W(x) W \mid} \]

How is it found?
Fisher Linear Discriminant

- generalized eigen-decomposition:

\[
\frac{|w^T A w|}{|w^T B w|} \rightarrow \text{max} \quad \iff \quad A w = B w D
\]

\[
[V, D] = \text{eig}(A, B) \quad \text{returns diagonal matrix } D \text{ of generalized eigenvalues and full matrix } V \text{ whose columns are the corresponding right eigenvectors, so that } A V = B V D.
\]
The wanted projection:

\[ W_{\text{optimal}} = \arg \max_W \frac{|S_B(y)|}{|S_W(y)|} = \arg \max_W \frac{|W^T S_B(x)W|}{|W^T S_W(x)W|} \]

How is it found?

\[ W_{\text{optimal}} = \nu_{\text{max}} \left( (S_W(x))^{-1} S_B(x) \right) \]
Fisher Linear Discriminant

As a result we get a good separation of the data
FLD: example result

- Face recognition under varying lighting, and expression

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

<table>
<thead>
<tr>
<th>With glasses</th>
<th>Without glasses</th>
<th>3 Lighting conditions</th>
<th>5 expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

48
FLD: example result

![Graph showing error rate vs. number of principal components for Eigenface, Eigenface w/o first three components, and Fisherface (7.3%)](image-url)
Fisherfaces - summary

- **Pros:**
  - Non-iterative
  - Simple

- **Cons:**
  - Requires face alignment
  - Non-localized
  - Limited to linear separation
Recap

- PCA can be used as a linear algorithm for dimensionality reduction.
- Only if the data is distributed according to the linearity assumption.
- Can fulfill various goals such as classification and compression.
- FLD can be used as a more robust algorithm for linear classification.
- More cool algorithms to come next…
End – Linear dimensionality reduction

Now you know how it works