Eigenfaces

Face Recognition Using Principal Components Analysis

M. Turk, A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, 3(1), pp. 71-86, 1991.

Slides : George Bebis, UNR

Principal Component Analysis (PCA)

- Pattern recognition in high-dimensional spaces
  - Problems arise when performing recognition in a high-dimensional space (curse of dimensionality).
  - Significant improvements can be achieved by first mapping the data into a *lower-dimensional sub*-space.

$$x = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_N \end{bmatrix} --> reduce\ dimensionality --> y = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix} \quad (K << N)$$

  - The goal of PCA is to reduce the dimensionality of the data while retaining as much information (but no redundancy) as possible in the original dataset.

Principal Component Analysis (PCA)

- Dimensionality reduction
  - PCA allows us to compute a linear transformation that maps data from a high dimensional space to a lower dimensional sub-space.

$$
\begin{aligned}
b_1 &= t_{11}a_1 + t_{12}a_2 + \ldots + t_{1n}a_N \\
b_2 &= t_{21}a_1 + t_{22}a_2 + \ldots + t_{2n}a_N \\
&\ldots \\
b_K &= t_{K1}a_1 + t_{K2}a_2 + \ldots + t_{KN}a_N
\end{aligned}
$$

$$
\text{or } y = Tx \text{ where } T = \begin{bmatrix} t_{11} & t_{12} & \ldots & t_{1N} \\ t_{21} & t_{22} & \ldots & t_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ t_{K1} & t_{K2} & \ldots & t_{KN} \end{bmatrix}
$$

Principal Component Analysis (PCA)

- Lower dimensionality basis
  - Approximate vectors by finding a basis in an appropriate lower dimensional space.

(1) Higher-dimensional space representation:

$$x = a_1 v_1 + a_2 v_2 + \cdots + a_N v_N$$

$v_1, v_2, ..., v_N$ is a basis of the N-dimensional space

(2) Lower-dimensional space representation:

$$\hat{x} = b_1 u_1 + b_2 u_2 + \cdots + b_K u_K$$

$u_1, u_2, ..., u_K$ is a basis of the K-dimensional space

- *Note:* if both bases have the same size ($N = K$), then $x = \hat{x}$)

Principal Component Analysis (PCA)

- Information loss
  - Dimensionality reduction implies information loss !!
  - Want to preserve as much information as possible, that is:

$$\textbf{minimize} \left\| x - \hat{x} \right\| \text{(error)}$$

- How to determine the best lower dimensional sub-space?

The best low-dimensional space can be determined by the "best" eigenvectors of the covariance matrix of x (i.e., the eigenvectors corresponding to the "largest" eigen-values -- also called "principal components").

Principal Component Analysis (PCA)

- Methodology
  - Suppose x_1, x_2, ..., x_M are N x 1 vectors

Step 1: $\bar{x} = \frac{1}{M}\sum_{i=1}^{M} x_i$

Step 2: subtract the mean: $\Phi_i = x_i - \bar{x}$

Step 3: form the matrix $A = [\Phi_1\ \Phi_2 \cdots \Phi_M]$ ($N \times M$ matrix), then compute:

$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^T = \frac{1}{M} AA^T$$

(sample **covariance** matrix, $N \times N$, characterizes the *scatter* of the data)

Step 4: compute the eigenvalues of C: $\lambda_1 > \lambda_2 > \cdots > \lambda_N$

Step 5: compute the eigenvectors of C: $u_1, u_2, \ldots, u_N$

Principal Component Analysis (PCA)

- Methodology – cont.

- Since C is symmetric, $u_1, u_2, \ldots, u_N$ form a basis, (i.e., any vector x or actually $(x - \bar{x})$, can be written as a linear combination of the eigenvectors):

$$x - \bar{x} = b_1 u_1 + b_2 u_2 + \cdots + b_N u_N = \sum_{i=1}^{N} b_i u_i$$

Step 6: (**dimensionality reduction step**) keep only the terms corresponding to the K largest eigenvalues:

$$\hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i \text{ where } K << N$$

- The representation of $\hat{x} - \bar{x}$ into the basis $u_1, u_2, \ldots, u_K$ is thus

$$\begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_K \end{bmatrix}$$

Principal Component Analysis (PCA)

- Linear transformation implied by PCA
  - The linear transformation $R^N \rightarrow R^K$ that performs the dimensionality reduction is:

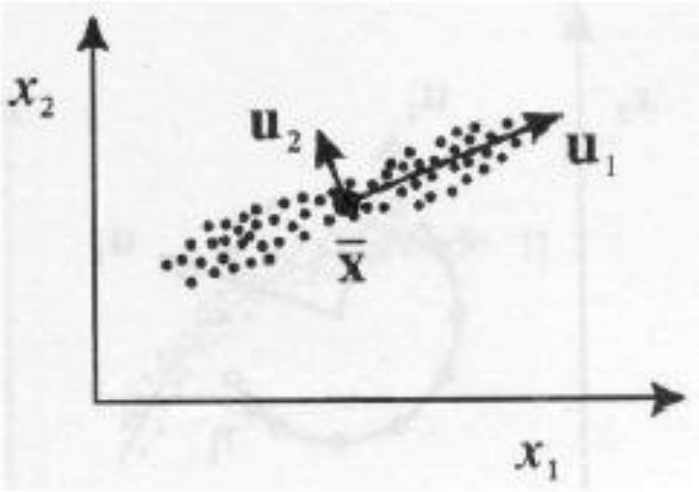
$$\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix} = \begin{bmatrix} u_1^T \\ u_2^T \\ \dots \\ u_K^T \end{bmatrix} (x - \bar{x}) = U^T (x - \bar{x})$$

(i.e., simply computing coefficients of linear expansion)

The above expression assumes that u_i has unit length (i.e., normalized)

Principal Component Analysis (PCA)

- Geometric interpretation
  - PCA projects the data along the directions where the data varies the most.
  - These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues.
  - The magnitude of the Eigen values corresponds to the variance of the data along the eigenvector directions.



Principal Component Analysis (PCA)

- How to choose the principal components?
  - To choose *K*, use the following criterion:

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{N} \lambda_i} > Threshold \quad \text{(e.g., 0.9 or 0.95)}$$

  - In this case, we say that we “preserve” 90% or 95% of the information in our data.
  - If K=N, then we “preserve” 100% of the information in our data.

Principal Component Analysis (PCA)

- Error due to dimensionality reduction
  - The original vector *x* can be reconstructed using its principal components:

$$\hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i \text{ or } \hat{x} = \sum_{i=1}^{K} b_i u_i + \bar{x}$$

  - It can be shown that the low-dimensional basis based on principal components minimizes the reconstruction error:

$$e = \|x - \hat{x}\|$$

  - It can be shown that the error is equal to:

$$e = 1/2 \sum_{i=K+1}^{N} \lambda_i$$

Principal Component Analysis (PCA)

- Standardization
  - The principal components are dependent on the *units* used to measure the original variables as well as on the *range* of values they assume.
  - We should always standardize the data prior to using PCA.
  - A common standardization method is to transform all the data to have zero mean and unit standard deviation:

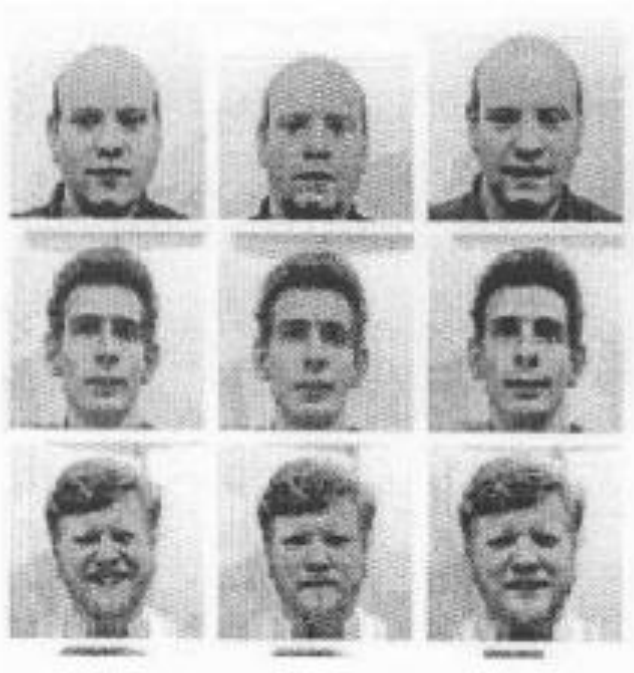
$$\frac{x_i - \mu}{\sigma} \quad (\mu \text{ and } \sigma \text{ are the mean and standard deviation of } x_i\text{'s})$$

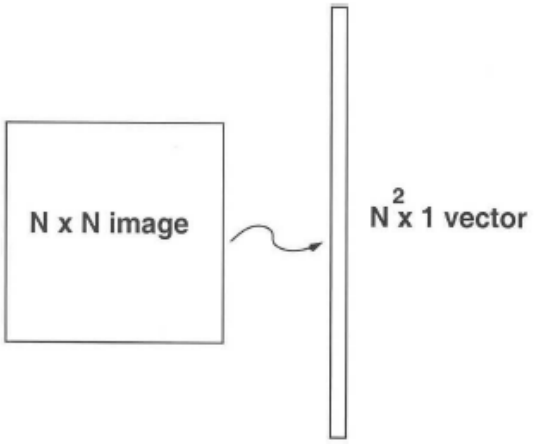
Application to Faces

- Computation low-dimensional basis (i.e.,eigenfaces):

Step 1: obtain face images $I_1, I_2, ..., I_M$ (training faces)

(**very important:** the face images must be *centered* and of the same *size*)





Step 2: represent every image I_i as a vector Γ_i

Application to Faces

- Computation of the eigenfaces – cont.

Step 3: compute the average face vector Ψ:

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$$

Step 4: subtract the mean face:

$$\Phi_i = \Gamma_i - \Psi$$

Step 5: compute the covariance matrix C:

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = \frac{1}{M}AA^T \quad (N^2 \text{x} N^2 \text{ matrix})$$

$$\text{where } A = [\Phi_1\ \Phi_2 \cdots \Phi_M] \quad (N^2 \text{x} M \text{ matrix})$$

Application to Faces

- Computation of the eigenfaces – cont.

Step 6: compute the eigenvectors u_i of AA^T ⟹ $AA^T u_i = \lambda_i u_i$

The matrix AA^T is very large --> not practical !!

IMPORTANT!

Step 6.1: consider the matrix $A^T A$ (M x M matrix)

Step 6.2: compute the eigenvectors v_i of $A^T A$

$$A^T A v_i = \mu_i v_i$$

What is the relationship between u_i and v_i?

$A^T A v_i = \mu_i v_i \Rightarrow AA^T A v_i = \mu_i A v_i \Rightarrow$ $u_i = A v_i$ *and* $\lambda_i = \mu_i$

$CAv_i = \mu_i Av_i$ or $Cu_i = \mu_i u_i$ where $u_i = Av_i$

Thus, AA^T and $A^T A$ have the same eigenvalues and their eigenvectors are related as follows: $u_i = Av_i$!!

Application to Faces

- Computation of the eigenfaces – cont.

Note 1: AA^T can have up to N^2 eigenvalues and eigenvectors.

Note 2: $A^T A$ can have up to M eigenvalues and eigenvectors.

Note 3: The M eigenvalues of $A^T A$ (along with their corresponding eigenvectors) correspond to the M *largest* eigenvalues of AA^T (along with their corresponding eigenvectors).

Step 6.3: compute the M best eigenvectors of AA^T: $u_i = Av_i$

(**important:** normalize u_i such that $\|u_i\| = 1$)

Step 7: keep only K eigenvectors (corresponding to the K largest eigenvalues)

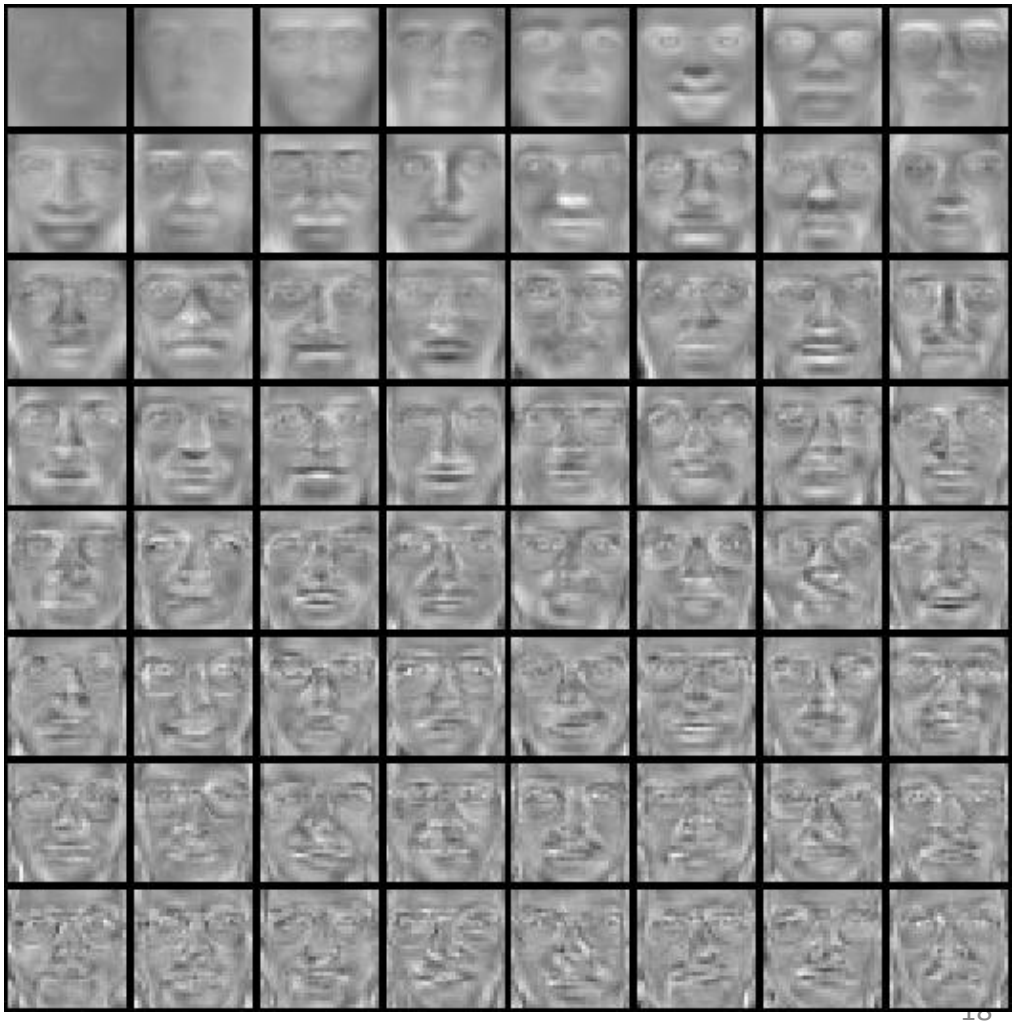
Eigenfaces example

Training images

Eigenfaces example

Mean: μ

Top eigenvectors: $u_1,\ldots u_k$



Application to Faces

- Representing faces onto this basis

- Each face (minus the mean) Φ_i in the training set can be represented as a linear combination of the best K eigenvectors:

$$\hat{\Phi}_i - mean = \sum_{j=1}^{K} w_j u_j, \quad (w_j = u_j^T \Phi_i)$$

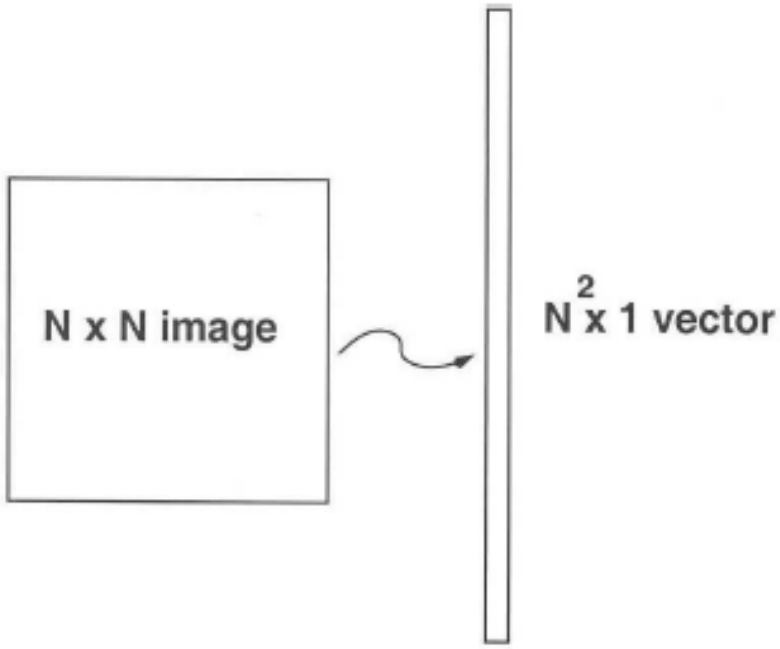
(we call the u_j's *eigenfaces*)

Face reconstruction:

Eigenfaces

- Case Study: Eigenfaces for Face Detection/Recognition
  - M. Turk, A. Pentland, "Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

- Face Recognition
  - The simplest approach is to think of it as a template matching problem
  - Problems arise when performing recognition in a high-dimensional space.
  - Significant improvements can be achieved by first mapping the data into a *lower dimensionality* space.



Eigenfaces

- Face Recognition Using Eigenfaces

- Given an unknown face image Γ (centered and of the same size like the training faces) follow these steps:

Step 1: normalize Γ: $\Phi = \Gamma - \Psi$

Step 2: project on the eigenspace

$$\hat{\Phi} = \sum_{i=1}^{K} w_i u_i \quad (w_i = u_i^T \Phi) \qquad (where \| u_i \| = 1)$$

Step 3: represent Φ as: $\Omega = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_K \end{bmatrix}$

Step 4: find $e_r = \min_l \| \Omega - \Omega^l \|$ where $\| \Omega - \Omega^l \| = \sum_{i=1}^{K} (w_i - w_i^l)^2$

Step 5: if $e_r < T_r$, then Γ is recognized as face l from the training set.

Eigenfaces

- Face Recognition Using Eigenfaces – cont.
  - The distance e_r is called *distance within face space (difs)*
  - The *Euclidean distance* can be used to compute e_r, however, the *Mahalanobis distance* has shown *to* work better:

$$\|\Omega - \Omega^k\| = \sum_{i=1}^{K} \frac{1}{\lambda_i} (w_i - w_i^k)^2$$

(variations along all axes are treated as equally signifi cant)

Face detection and recognition

Detection

Recognition

"Sally"

Eigenfaces

- Face Detection Using Eigenfaces

- Given an unknown image Γ

Step 1: compute $\Phi = \Gamma - \Psi$

Step 2: compute $\hat{\Phi} = \sum_{i=1}^{K} w_i u_i \quad (w_i = u_i^T \Phi)$ $\quad (where \| u_i \| = 1)$

Step 3: compute $e_d = \|\Phi - \hat{\Phi}\|$

Step 4: if $e_d < T_d$, then Γ is a face.

The distance e_d is called *distance from face space (dffs)*