

Facoltà di Scienze MM. FF. NN.

Università di Verona

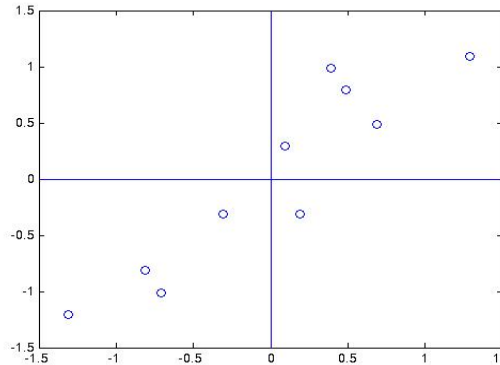
A.A. 2014-15

# **Teoria e Tecniche del Riconoscimento**

*Estrazione delle feature: Principal Component Analysis,  
Eigenfaces*

# Principal Component Analysis

- Il dato = un vettore **N**-dimensionale di valori di pixel
- Supponiamo  $N=2 \rightarrow$  dati = punti 2D



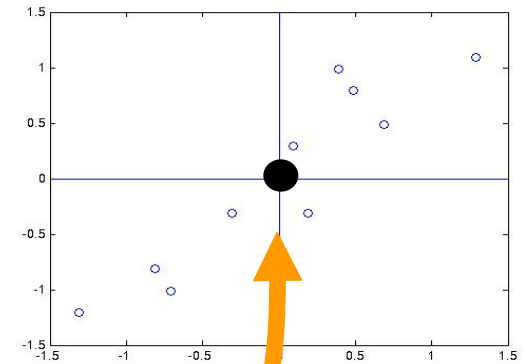
Come posso descrivere in modo **compatto** questi **M** punti?

- SOL 1:** Con un punto (vettore) solo, che minimizzi la distanza media quadratica con tutti i pts

$$\operatorname{argmin}_{\mathbf{x}^{(0)}} J_0(\mathbf{x}^{(0)}) = \sum_{k=1}^M \left\| \mathbf{x}^{(0)} - \mathbf{x}^{(k)} \right\|^2$$

- Soluzione:** il vettore media

$$\mathbf{m} = \frac{1}{M} \sum_{k=1}^M \mathbf{x}^{(k)}$$



**Problema**  
**m poco espressivo!!!**

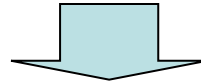
# PCA

- **SOL2: Con una retta di proiezione**, che minimizzi la distanza media quadratica con tutti i pti

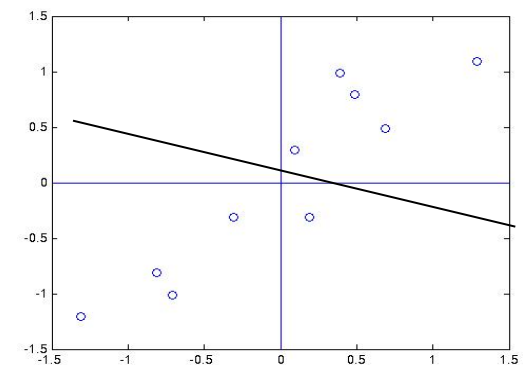
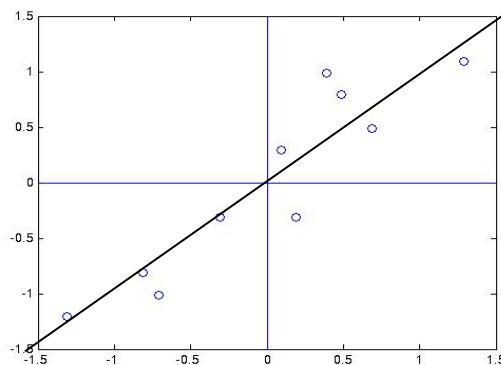
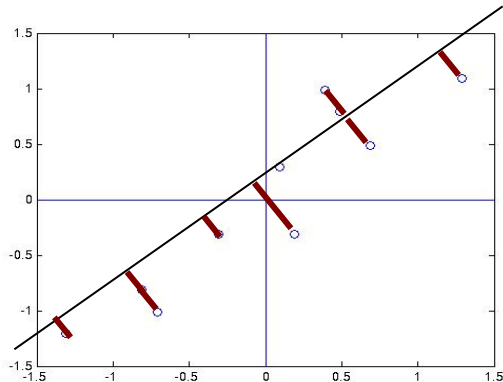
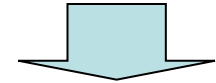
- Supp. una retta di proiezione nello spazio



Alcune vanno bene  
intuitivamente




... altre no.



# PCA

- Per convenienza, imponiamo passaggio dalla media, e specifichiamo quindi i punti della retta come

$$\mathbf{x}^{(k)} = \mathbf{m} + a^{(k)} \mathbf{e}$$
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} + a \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$


- Troviamo quindi i coefficienti  $\{a^{(k)}\}_{k=1\dots M}$  che minimizzano la distanza quadratica

$$\operatorname{argmin}_{\{a^{(k)}\}_{k=1\dots M}} J_1(\{a^{(k)}\}_{k=1\dots M}, \mathbf{e}) = \sum_{k=1}^M \left\| (\mathbf{m} + a^{(k)} \mathbf{e}) - \mathbf{x}^{(k)} \right\|^2$$

$$\dots \rightarrow a^{(k)} = \mathbf{e}^T (\mathbf{x}^{(k)} - \mathbf{m})$$

# PCA

- Sostituendo  $a^{(k)} = \mathbf{e}^T (\mathbf{x}^{(k)} - \mathbf{m})$  in  $J_1(\{a^{(k)}\}_{k=1\dots M}, \mathbf{e})$  otteniamo  $J_1(\mathbf{e})$ , ossia

$$J_1(\mathbf{e}) = -\mathbf{e}^T \mathbf{S} \mathbf{e} + \sum_{k=1}^M \|\mathbf{x}^{(k)} - \mathbf{m}\|^2$$

Scatter matrix  $\approx$  mat. di covarianza

$$\mathbf{S} = \sum_{k=1}^M (\mathbf{x}^{(k)} - \mathbf{m})(\mathbf{x}^{(k)} - \mathbf{m})^T$$

- Minimizzare  $J_1(\mathbf{e})$  significa massimizzare  $\mathbf{e}^T \mathbf{S} \mathbf{e}$  tenuto conto che  $\|\mathbf{e}\| = 1$ , via moltiplicatori di Lagrange; ossia  $u = \mathbf{e}^T \mathbf{S} \mathbf{e} + \lambda(\mathbf{e}^T \mathbf{e} - 1)$

- Minimizzo;

$$\frac{\partial u}{\partial \mathbf{e}} = 2\mathbf{S}\mathbf{e} - 2\lambda\mathbf{e} = 0$$



$$\mathbf{S}\mathbf{e} = \lambda\mathbf{e}$$



A conti fatti:

1.  $\mathbf{e}$  deve essere autovettore;
2. Poiché

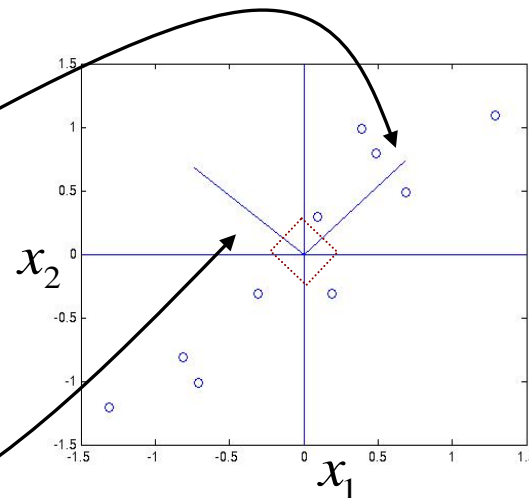
$$\mathbf{e}^T \mathbf{S} \mathbf{e} = \lambda \mathbf{e}^T \mathbf{e} = \lambda$$

$\lambda$  deve essere massimo

# Esempio

- La matrice di covarianza è  $S = \begin{bmatrix} 0.617 & 0.615 \\ 0.615 & 0.717 \end{bmatrix}$
- Otengo 2 autovettori (e autovalori): quale prendo?

$$\mathbf{e}_{(1)} = \begin{bmatrix} -0.735 \\ 0.678 \end{bmatrix} \quad \mathbf{e}_{(2)} = \begin{bmatrix} -0.678 \\ 0.735 \end{bmatrix}$$
$$\lambda_{(1)} = 0.049 \quad \lambda_{(2)} = 1.284$$



# Esempio - osservazioni

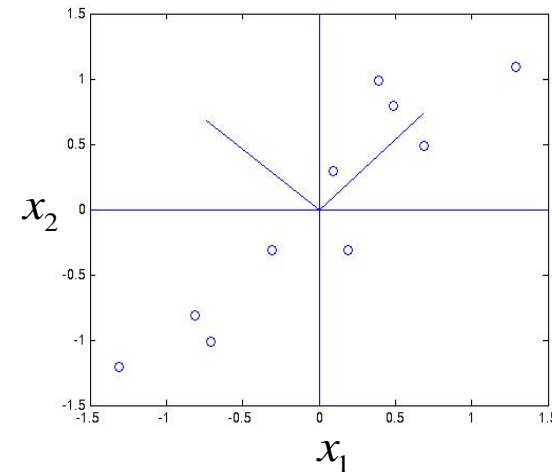
- La matrice di covarianza  $S = \begin{bmatrix} 0.617 & 0.615 \\ 0.615 & 0.717 \end{bmatrix}$  è simmetrica, reale

→ gli autovettori  $\{\mathbf{e}\}$  sono ortogonali, formanti una base per lo spazio N dimensionale

$$\mathbf{e}_1 = \begin{bmatrix} -0.735 \\ 0.678 \end{bmatrix} \quad \mathbf{e}_2 = \begin{bmatrix} -0.678 \\ 0.735 \end{bmatrix}$$

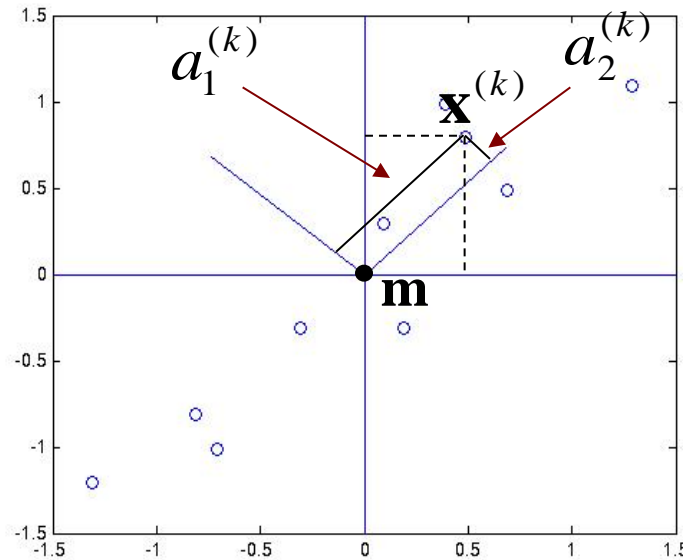
$$\lambda_1 = 0.049 \quad \lambda_2 = 1.284$$

→ gli autovalori più grandi si hanno nelle direzioni di massima dispersione dei dati



# Esempio - osservazioni

- I coefficienti  $a_i^{(k)}$ , per  $i=1,\dots,N$ , sono le componenti del punto  $\mathbf{x}^{(k)}$  per la base trovata



- Quindi un punto può essere scritto come 
$$\mathbf{x}^{(k)} = \mathbf{m} + \sum_{i=1}^D a_i^{(k)} \mathbf{e}_i$$



# Da PCA ad eigenfaces

- Dato un gruppo di punti, ho trovato la base che descrive lo scattering dei punti
- Ogni punto puo' essere mappato tramite i componenti della base (numero componenti = numero dimensioni N)
- Usare meno componenti (**le componenti principali**) permette di proiettare i punti in un sottospazio altamente informativo → riduzione della dimensionalità dei punti
- Ora passiamo alle facce (...punti!) → dataset di M facce, N dimensioni



$$\mathbf{x}^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_N^{(1)} \end{bmatrix}$$



$$\mathbf{x}^{(2)} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ \vdots \\ x_N^{(2)} \end{bmatrix}$$

...






$$\mathbf{x}^{(M)} = \begin{bmatrix} x_1^{(M)} \\ x_2^{(M)} \\ \vdots \\ x_N^{(M)} \end{bmatrix}$$

N.B.:  
di solito,  
 **$M \ll N$**

# Da PCA ad eigenfaces (2)

- Preprocessing -
  - Le immagini devono contenere esclusivamente facce (no outliers)
  - Le immagini devono essere ragionevolmente prive di rumore
  - Le facce devono avere la stessa dimensione → riscalamento
  - Stesse condizioni di illuminazione, o compensazione
  - Non ci deve essere sfondo → ritaglia le facce, catturale su sfondo neutro (alternativa: background subtraction)
  - In maniera automatica, utilizzo di un metodo di face detection

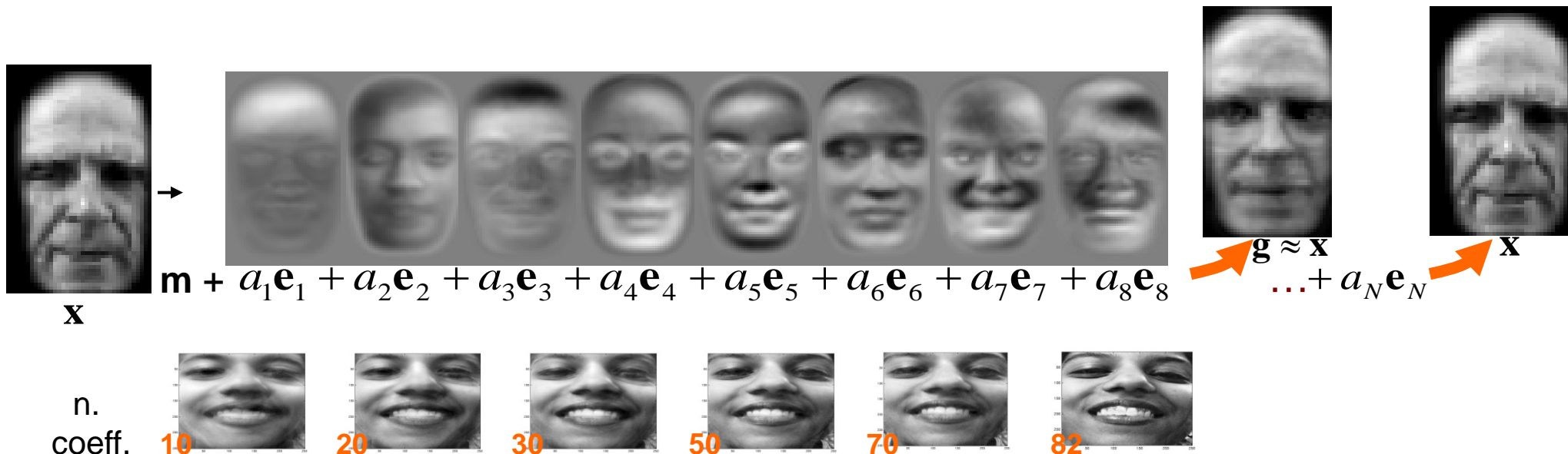

$$\mathbf{x}^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_N^{(1)} \end{bmatrix}$$

$$\mathbf{x}^{(2)} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ \vdots \\ x_N^{(2)} \end{bmatrix} \quad \dots$$

$$\mathbf{x}^{(M)} = \begin{bmatrix} x_1^{(M)} \\ x_2^{(M)} \\ \vdots \\ x_N^{(M)} \end{bmatrix}$$

# Eigenfaces - proiezioni

- Dopo aver ricavato gli autovettori  $\{\mathbf{e}\}$  (**eigenfaces**) dalla matrice di covarianza  $S$ , calcolo le componenti  $\{a\}$  (o **proiezione**) per la faccia  $\mathbf{x}$

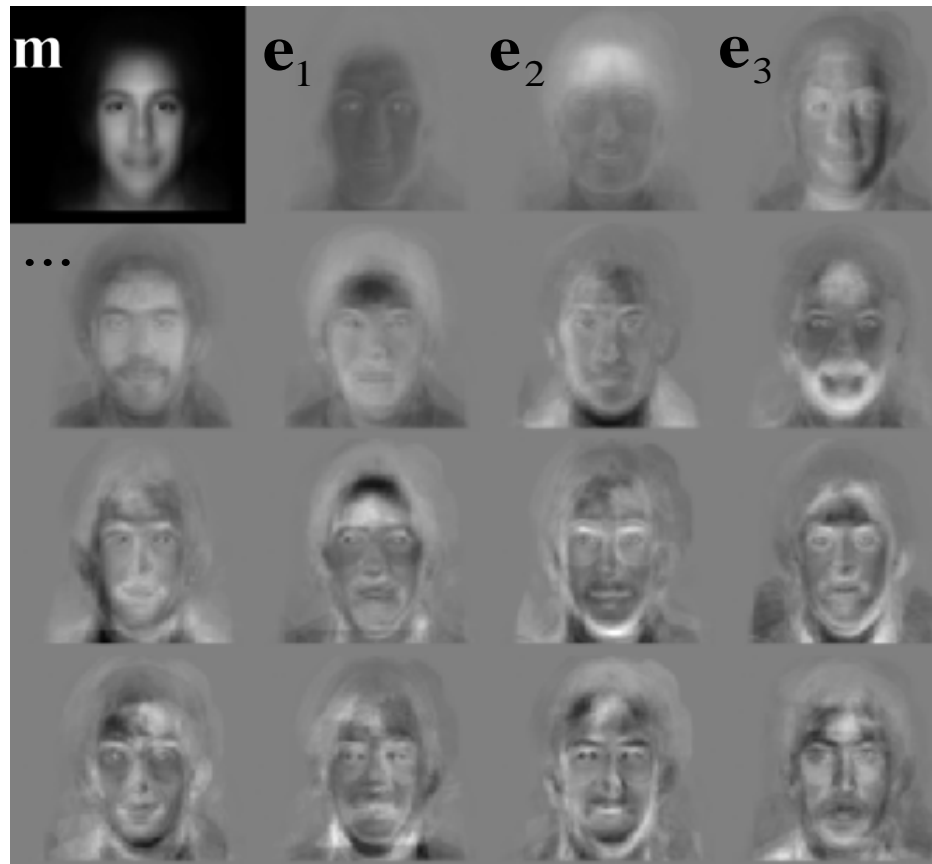
$$\underbrace{\langle \mathbf{e}'_1 (\mathbf{x} - \mathbf{m}), \mathbf{e}'_1 (\mathbf{x} - \mathbf{m}) \rangle}_{a_1} \underbrace{\langle \mathbf{e}'_2 (\mathbf{x} - \mathbf{m}), \mathbf{e}'_2 (\mathbf{x} - \mathbf{m}) \rangle}_{a_2} \dots \underbrace{\langle \mathbf{e}'_N (\mathbf{x} - \mathbf{m}), \mathbf{e}'_N (\mathbf{x} - \mathbf{m}) \rangle}_{a_N} \text{ ed ho } \mathbf{x} = \mathbf{m} + a_1 \mathbf{e}_1 + a_2 \mathbf{e}_2 + \dots + a_N \mathbf{e}_N$$

- Le **ricostruzioni**,  $\mathbf{g}$ , possono avvenire anche nei sottospazi descritti dalle eigenfaces con eigenvalues più grandi – riduzione dimensionalità



# Eigenfaces

- Autovettori  $\{\mathbf{e}\}$  (**eigenfaces**): in pratica?



# Eigenfaces - problema

- Sia  $\mathbf{A} = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(M)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(M)} \\ \vdots & \vdots & \vdots & \vdots \\ x_N^{(1)} & x_N^{(2)} & \dots & x_N^{(M)} \end{bmatrix}$  pertanto  $\mathbf{S} = \mathbf{A}\mathbf{A}'$   
 $N \times M$   $N \times N$
- Con un'img 256 x 256 ho  $\mathbf{S}$  : problemi di overflow!  
 $65536 \times 65536$
- Trick: calcolo gli autovettori di  $\mathbf{A}'\mathbf{A}$ , ossia  $\{\tilde{\mathbf{e}}\}$ , tenuto conto  
 $M \times M$   
 che di solito si usano 20-30 eigenfaces e che

$$\mathbf{A}'\mathbf{A}\tilde{\mathbf{e}} = \lambda\tilde{\mathbf{e}} \rightarrow \mathbf{A}\mathbf{A}'\mathbf{A}\tilde{\mathbf{e}} = \lambda\mathbf{A}\tilde{\mathbf{e}}$$

$$\rightarrow \mathbf{S}\mathbf{A}\tilde{\mathbf{e}} = \lambda\mathbf{A}\tilde{\mathbf{e}} \quad \text{quindi} \quad \mathbf{e} = \mathbf{A}\tilde{\mathbf{e}}$$

# Eigenfaces - note

- Gli  $M$  autovalori di  $\mathbf{A}'\mathbf{A}$  corrispondono agli  $M$  autovalori più grandi di  $\mathbf{S}$  (così come i corrispondenti autovettori)

# Eigenfaces - algoritmo

1. Sottraggo ad ogni immagine  $\mathbf{x}^{(k)}$ ,  $k=1\dots M$ , la media  $\mathbf{m} = \frac{1}{M} \sum_{k=1}^M \mathbf{x}^{(k)}$  ottenendo i vettori  $\{\tilde{\mathbf{x}}^{(k)}\}$  da cui costruisco la matrice  $\mathbf{A}_{N \times M}$
2. Calcolo M autovettori  $\{\tilde{\mathbf{e}}_i\}_{i=1,\dots,M}$  di  $\mathbf{A}'\mathbf{A}_{M \times M}$  formando la matrice  $\mathbf{V}_{M \times M}$
3. Ricavo gli M autovettori più grandi di  $\mathbf{S} = \mathbf{A}\mathbf{A}'$ , ossia  $\mathbf{e}_i = \mathbf{A}\tilde{\mathbf{e}}_i$   
o in forma matriciale  $\mathbf{U}_{N \times M} = \mathbf{A}_{N \times M} \mathbf{V}_{M \times M}$
4. Ricavo i corrispondenti componenti (o coeff. di proiezione)

$$a_i^{(k)} = \mathbf{e}_i' \left( \tilde{\mathbf{x}}^{(k)} \right)$$

o in forma matriciale

$$\mathbf{\omega}_{M \times 1}^{(k)} = \mathbf{U}_{M \times N}' \left( \tilde{\mathbf{x}}_{N \times 1}^{(k)} \right)$$

# Proprietà chiave della rappresentazione ad eigenfaces

Date

- 2 immagini  $\mathbf{x}_1, \mathbf{x}_2$  usate per costruire l'eigenspace
- $\mathbf{w}_1$  è la proiezione nell'eigenspace dell'img  $\mathbf{x}_1$
- $\mathbf{w}_2$  è la proiezione nell'eigenspace dell'img  $\mathbf{x}_2$

allora,

$$\| \mathbf{w}_2 - \mathbf{w}_1 \| \approx \| \mathbf{x}_2 - \mathbf{x}_1 \|$$

ossia, la distanza nell'eigenspace è approssimativamente uguale alla distanza tra due immagini.



# Riconoscimento con le eigenfaces

TRAINING

1. Analizza il database d'immagini etichettato (<volti, identità>)
  - a) Esegui PCA — calcola le eigenfaces, formo  $\mathbf{U}$
  - b) Calcola i K coefficienti per ogni immagine  $\mathbf{x}^{(k)}$   $k=1, \dots, M$
  - c) Calcola le M proiezioni nello spazio delle eigenfaces  $\omega^{(k)}$   $k=1, \dots, M$
  - d) Calcolo la soglia

$$\theta = \max \left\{ \left\| \omega^{(j)} - \omega^{(k)} \right\| \right\} \text{ per } j, k = 1, \dots, M$$

RICONOSCIMENTO

2. Data una nuova img (da riconoscere)  $\mathbf{x}$ ,
  - a) Ne sottraggo la media del dataset di training  $\mathbf{m}$ , ottengo  $\tilde{\mathbf{x}}$
  - b) Proietto, ossia calcolo le K componenti
  - c) Calcolo il set di distanze

$$\tilde{\mathbf{x}} \rightarrow \boldsymbol{\omega} = [a_1, a_2, \dots, a_K]' \quad \text{ossia} \quad \boldsymbol{\omega} = \mathbf{U}' \tilde{\mathbf{x}}$$

$$\left( \varepsilon^{(k)} \right)^2 = \left\| \boldsymbol{\omega} - \omega^{(k)} \right\|^2 \text{ per } k = 1, \dots, M$$

# Riconoscimento con le eigenfaces

3. Ricostruisco la faccia usando eigenfaces e componenti

$$\tilde{\mathbf{g}} = \sum_{i=1}^K a_i \mathbf{e}_i \quad \text{oppure} \quad \tilde{\mathbf{g}} = \mathbf{U}\boldsymbol{\omega}$$

4. Calcolo la distanza tra la faccia di partenza incognita e la ricostruzione

$$\xi^2 = \|\tilde{\mathbf{g}} - \tilde{\mathbf{x}}\|^2$$

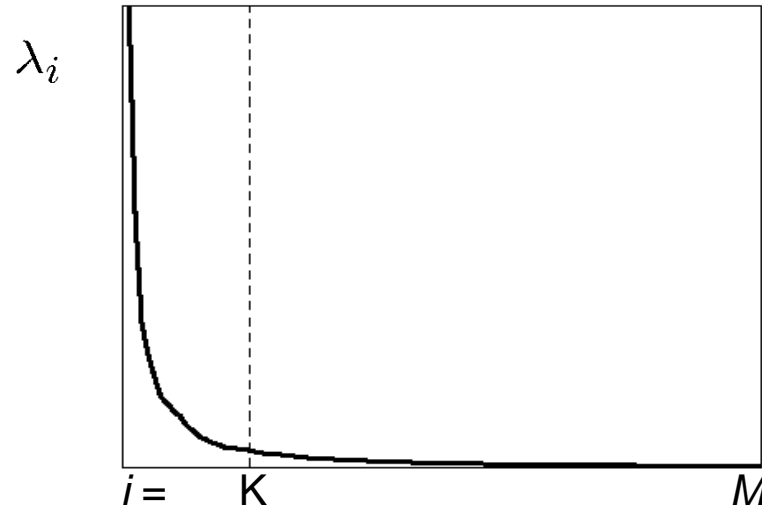
5. Se

- $\xi \geq \theta$  non è una faccia
- $\xi < \theta$  e  $\varepsilon^{(k)} \geq \theta, (k = 1, \dots, M)$  è una nuova faccia
- $\xi < \theta$  e  $\min \{\varepsilon^{(k)}\} < \theta$  è una faccia conosciuta, la  $k_{\text{best}}$ -esima, dove

$$k_{\text{best}} = \underset{k}{\operatorname{argmin}} \{ \varepsilon^{(k)} \}$$

# Dettagli pratici

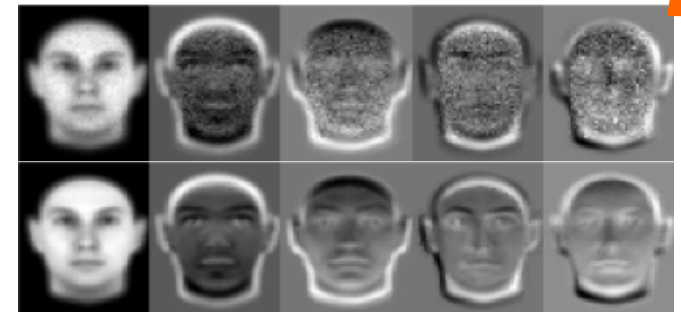
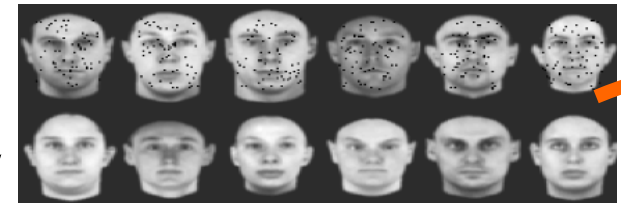
- Quanti eigenvector usare?
- Controlla il decadimento degli eigenvalues  $\{\lambda_i\}$
- Dati “buoni” ossia trattabili hanno poche dimensioni ad alta varianza
- Nel caso in cui tutti gli  $N$  autovalori sono stati calcolati per un dataset  $N$ -dimensionale vale



$$\sigma_{\text{covered}} = \frac{\sum_{i=1}^K \lambda_i}{\sum_{j=1}^N \lambda_j}$$

# Problemi eigenfaces

- **Illuminazione**: stessi soggetti con differente illuminazione risultano lontani nello spazio delle eigenfaces
- **Pose differenti**: le componenti di un volto frontale non servono per un riconoscimento con profili
- **Allineamento differente**
- **Espressione facciale differente**
- **Outlier**
  - Sample outliers = non facce
  - Intra-sample outliers = facce affette da rumore



# Problemi eigenfaces (2)

- Funzionale solo per la rappresentazione di dati appartenenti ad un'unica classe
- Non separano classi differenti

