Dimensionality reduction

Machine Learning and Deep Learning

Matteo Boschini, Lorenzo Bonicelli, Aniello Panariello, Emanuele Frascaroli

November 15th, 2022

University of Modena and Reggio Emilia

Agenda

Principal Component Analysis

Eigenfaces

Principal Component Analysis

- Linear dimensionality reduction model
  - Subspace projection is linear
  - Reconstruction is linear
- Projects data in a new space subject to:
  - the direction exhibiting highest variance in feature space is projected on the first axis, the one exhibiting the second highest variance on the second axis, and so on.
  - axis of the new space are orthogonal (covariance is zero).

PCA: algorithm

- Arrange your data in a $n \times d$ matrix X, where n is the number of samples and d is data dimensionality
- Compute the mean μ (d-dimensional vector) of all samples

$\mu_{\text{PIXEL } 1,1} = \dfrac{\Sigma \text{ TUTTI PIXEL 1 IMG.}}{\text{N° PIXEL 1}}$

. . . .

$\mu_{\text{PIXEL } nn} = \text{''} \; |_{1=nn}$

- Compute convariance matrix

$$\Sigma = (X - \mu)^T (X - \mu)$$

→ SI ESTRAGGONO AUTOVETTORI ORDINATI DAI CORRISP. AUTOVALORI

- Pick the first m eigenvectors of Σ (ordered by decreasing eigenvalues), where m is the dimensionality you want your data to be projected to
- Arrange such eigenvectors in a $d \times m$ matrix E → MATRICE DI TRASFORMAZIONE
- Compute the projected samples as $P = X \cdot E$ (PROIEZIONE)

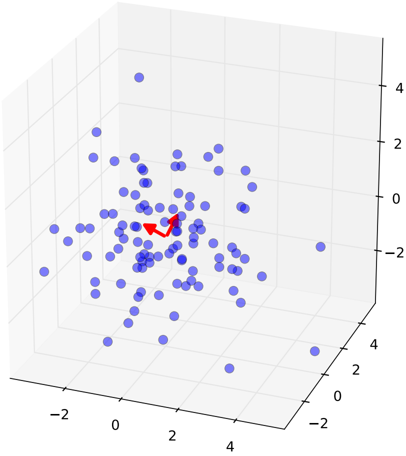
DIMENSIONE RIDOTTA

$P_{dim} = (n \times d)(d \times k) = n \times k$

- You can compute the reconstruction as $\tilde{X} = P \cdot E^T$ (RICOSTRUZIONE)

$\tilde{X}_{dim} = (n \times k)(d \times n) = n \times n$ (CI SARÀ PERDITA DI INFORMAZIONE)

PCA: plotting components

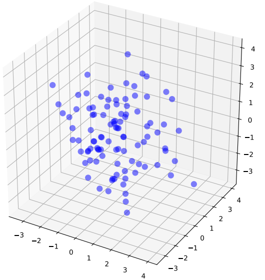


PCA: projecting and reconstructing (2D)

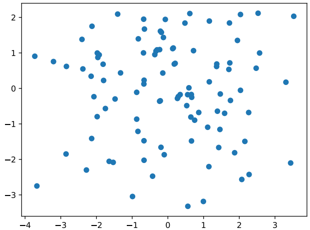
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

C'È UN MINIMO DI PERDITA.

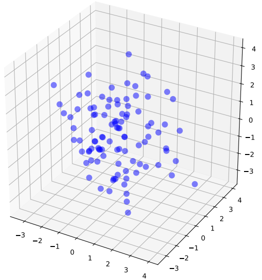
original data



projection



reconstruction

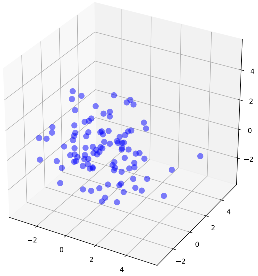


PCA: projecting and reconstructing (1D)

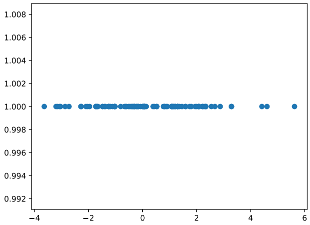
3D RIDUCO A CASO
1-DIM

PERDO MOLTA
INFORMAZIONE.

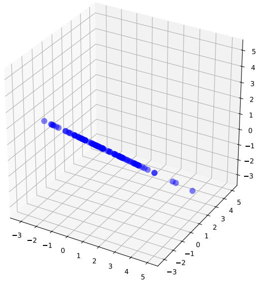
original data



projection



reconstruction

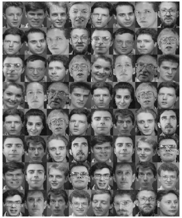


Eigenfaces

Eigenfaces

Famous algorithm for face recognition. Training is as simple as:

- load faces and annotations from the Olivetti dataset (datasets.get_faces_dataset takes care of loading and flattening images)



- Select a number of principal components and fit a PCA on training faces

To classify a test image:

- Project the image in the reduced spaces built in the training phase
- Perform **nearest neighbor classification**:
  - Roughly speaking, choose the class of the nearest training example (in the reduced space)

Eigenfaces: a magic trick to compute eigenvectors

Each Olivetti image is 112×92. Once flattened, is a vector of 10304 pixels: (TANTE FEATURES)

- The convariance matrix is 10304×10304 → ALTO COSTO DI COMPUTAZ.
- Computing eigenvectors and eigenvalues is a pain
- Instead, compute the covariance matrix of transposed X:

$$\Sigma = (X - \mu)(X - \mu)^T$$

(invece che DIM > n° features · n° ... spazio dim = n° el. × n° campi)

- Once selected the principal components $\tilde{E}$ of this weirdo space, you can compute the original eigenvectors just like:

$$E = X^T \cdot \tilde{E}$$

- Normalize the retrieved eigenvectors to have unit lenght:

$$E_i \leftarrow \frac{E_i}{\sqrt{\lambda_i}} \qquad i = 1, 2, \ldots, m$$

where λ_i is the eigenvalue corresponding to the eigenvector E_i

Perché autovalori Ei sono normalizzati rispetto la radice del corrispettivo autovalore.
Pertìchè Ei /sqrt(Gamma_i) é sempre <0 e <1 ?

In particolare la matrice di covarianza $\Sigma = \dim(d \times d)$, $d = n°$ features

stimare 10000 autovalori e autovettori è costoso.

Uso questo trick, invece di $\Sigma_{d\times d} = (n\times d)^T @ \, n\times d$

$= d\times n @ d\times n = d\times d$

Ottengo invece $\Sigma^T = \tilde{\Sigma}$ che è una $n\times n = 200 \times 200$ ~ questa scompo.

$$\tilde{S} = n\cdot d @ n\cdot d^T = n\cdot d @ d\cdot n = n\times n$$

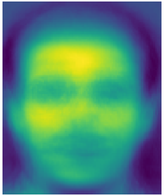
Poi da $\tilde{E} = \dim(n\times c)$, $c = n°$ dimensioni PCA.

$\tilde{S} \to B$ con $E = \underbrace{d\times n}_{X^T} @ \underbrace{n\times c}_{\tilde{E}} \to d\times c$

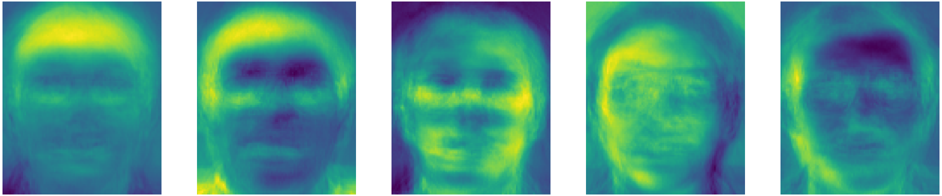
$$\Rightarrow P = n\times d @ d\times c = n\times c$$

Eigenfaces: some plots

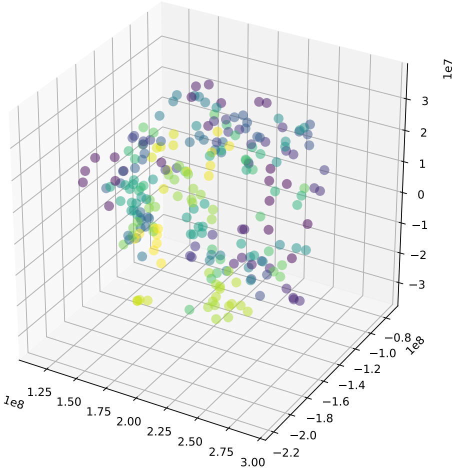
- Mean face:



- Eigenvectors:



Eigenfaces: face space



Eigenfaces: how many dimensions?