

Fundamentals of machine learning

Course 4: Basics of dimension reduction

Mathurin Massias & Titouan Vayer

email: mathurin.massias@inria.fr, titouan.vayer@inria.fr

February 5, 2025



Table of contents

Why dimension reduction ?

Principal component analysis

The principle of PCA

Singular value decomposition

Minimum distortion embedding

General setting

Random projections

Non linear methods

Autoencoders

Why dimension reduction ?

- ▶ Visualize high-dimensional data (e.g. in 2D).
- ▶ Interpret the data.
- ▶ Compress the data: advantages for storage and robustness.
- ▶ Reveal the “structure of the data”.
- ▶ Avoid the curse of dimensionality (course 2).

Learning rates & excess risk

Question

Once we have selected a model how many samples n do we need to train it ?

Learning rates & excess risk

Question

Once we have selected a model how many samples n do we need to train it ?

The answer is statistical

- ▶ Can I bound (w.h.p. or in expectation)

$$\text{excess risk} = \text{estimation error} + \text{approximation error} \leq h(n)$$

- ▶ Ideally the function $h(n) \xrightarrow[n \rightarrow +\infty]{} 0$ fast: it is called the *learning rate*

Learning rates & excess risk

Question

Once we have selected a model how many samples n do we need to train it ?

The answer is statistical

- ▶ Can I bound (w.h.p. or in expectation)

$$\text{excess risk} = \text{estimation error} + \text{approximation error} \leq h(n)$$

- ▶ Ideally the function $h(n) \underset{n \rightarrow +\infty}{\rightarrow} 0$ fast: it is called the *learning rate*

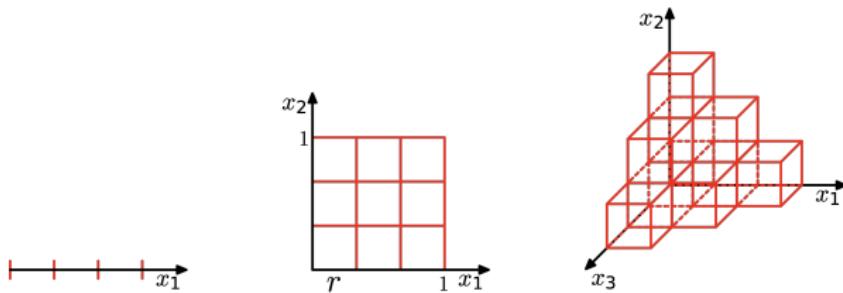
Curse of dimensionality

- ▶ If the best function f^* is only Lipschitz-continuous:

$$\mathbb{E}[\text{excess risk}] \lesssim n^{-1/d}$$

- ▶ This rate **is unavoidable** without further knowledge on f^*
- ▶ Slow rates: exponentially many samples are needed !

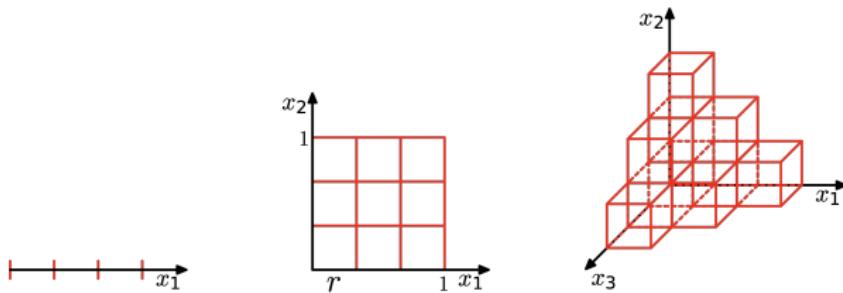
Curse of dimensionality



Think about interpolation: goal find $f : [0, 1]^d \rightarrow \mathbb{R}$

- ▶ You have n values of this function $f(x_i)$ at *sampled locations* x_i
- ▶ To find f you want to interpolate between the $f(x_i)$'s
- ▶ If f is not regular, a good approximation requires precise covering of $[0, 1]^d$ (small meshes)

Curse of dimensionality



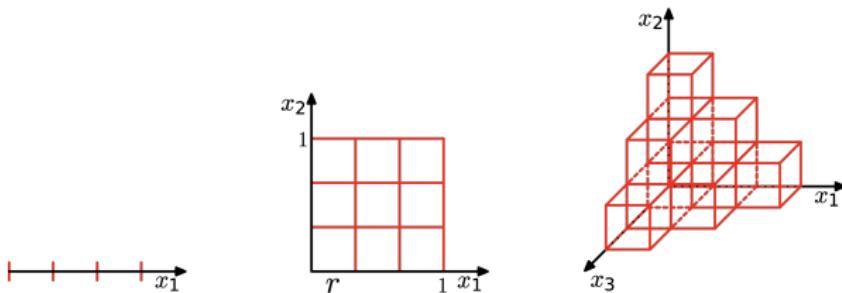
Think about interpolation: goal find $f : [0, 1]^d \rightarrow \mathbb{R}$

- ▶ You have n values of this function $f(x_i)$ at *sampled locations* x_i
- ▶ To find f you want to interpolate between the $f(x_i)$'s
- ▶ If f is not regular, a good approximation requires precise covering of $[0, 1]^d$ (small meshes)

Problem: points are isolated in high dimension

- ▶ Vol. of a hypercube with edge length $r < 1$ is r^d : quickly $\downarrow 0$ as $d \rightarrow \infty$
- ▶ To compensate you need a number of sample which **grows exponentially with d**

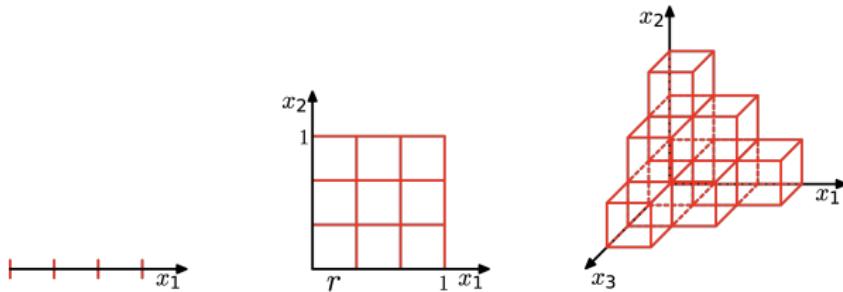
Curse of dimensionality



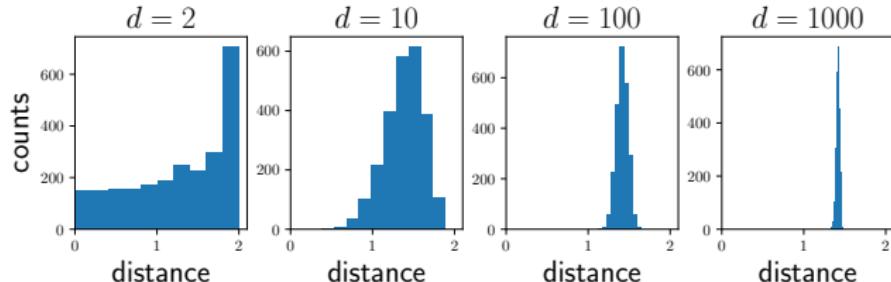
Main ideas

- ▶ Without prior on f^* the required number of samples n to estimate f^* is **exponential in d** .
- ▶ In high dim it is easy to overfit a model.
- ▶ The notion of nearest neighbors vanishes in high dim.
- ▶ Three remedies: use simple models (linear), dimensionality reduction or prior on f^*

Curse of dimensionality

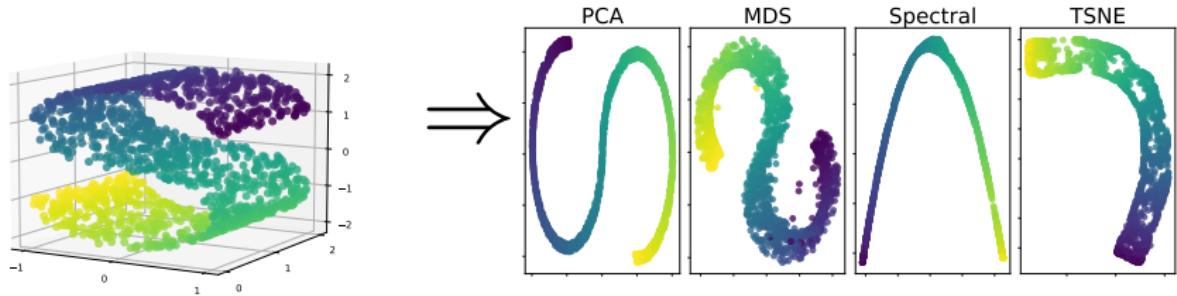


- ▶ Number of points needed to cover the hypercube cube $[0, 1]^d$ with precision r : $\left(\frac{2}{r}\right)^d$
- ▶ Distances between points become meaningless. Pairwise distances between 50 random points of \mathbb{R}^d :



The big picture

Original dataset



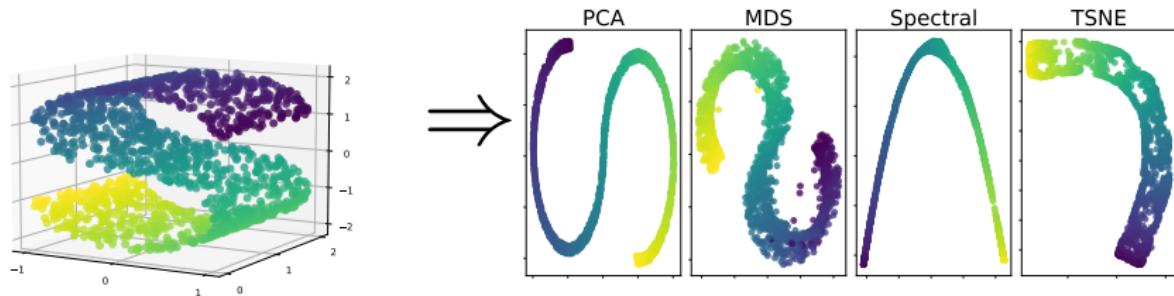
Objective

$$(\mathbf{x}_i)_{i=1}^n \quad \Rightarrow \quad (\tilde{\mathbf{x}}_i \in \mathbb{R}^k)_{i=1}^n \text{ with } k \ll d$$

- ▶ Project the data into a low dimensional space \mathbb{R}^k with $k \ll d$
- ▶ Preserve the information in the data (class, subspace, similarities)

The big picture

Original dataset



Objective

$$(\mathbf{x}_i)_{i=1}^n \Rightarrow (\tilde{\mathbf{x}}_i \in \mathbb{R}^k)_{i=1}^n \text{ with } k \ll d$$

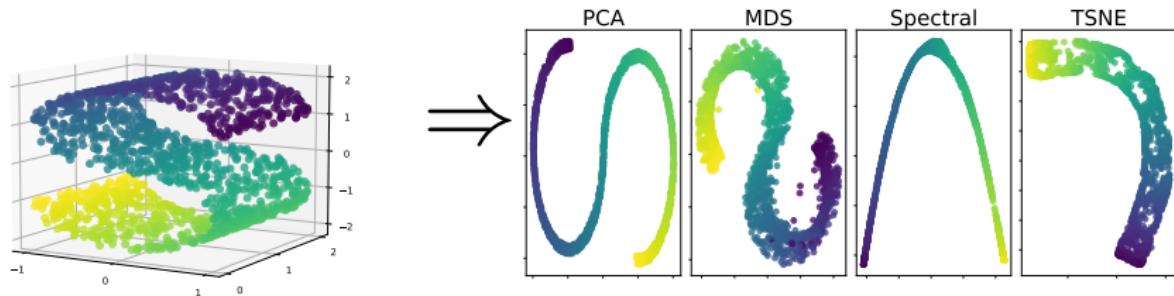
- ▶ Project the data into a low dimensional space \mathbb{R}^k with $k \ll d$
- ▶ Preserve the information in the data (class, subspace, similarities)

Hyperparameters

- ▶ Linear, non linear projection ?
- ▶ Similarity between samples.

The big picture

Original dataset



Objective

$$(\mathbf{x}_i)_{i=1}^n \Rightarrow (\tilde{\mathbf{x}}_i \in \mathbb{R}^k)_{i=1}^n \text{ with } k \ll d$$

- ▶ Project the data into a low dimensional space \mathbb{R}^k with $k \ll d$
- ▶ Preserve the information in the data (class, subspace, similarities)

Hyperparameters

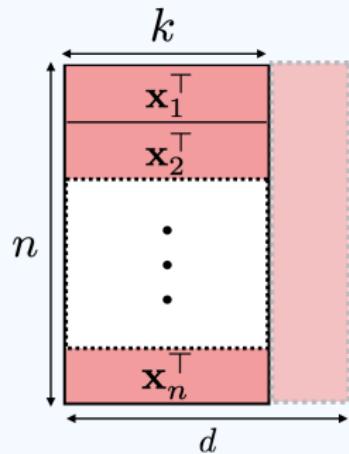
- ▶ Linear, non linear projection ?
- ▶ Similarity between samples.

Methods

- ▶ PCA, random projections.
- ▶ Non-linear methods (MDS, tSNE, Auto-Encoder)

Dimension reduction vs subsampling

Dimension reduction

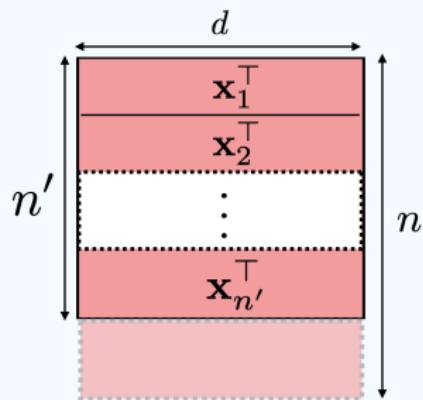


| Random projections (JL lemma)

| Feature selection

| Minimum distortion embedding, PCA

Subsampling



| Coresets

| Importance sampling

Table of contents

Why dimension reduction ?

Principal component analysis

The principle of PCA

Singular value decomposition

Minimum distortion embedding

General setting

Random projections

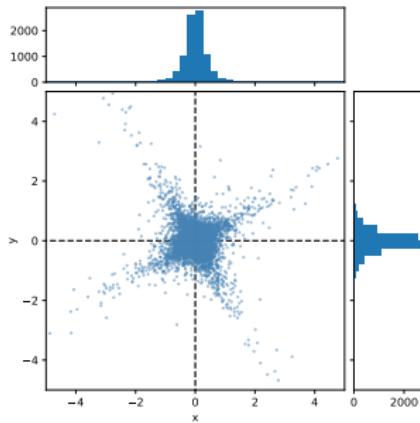
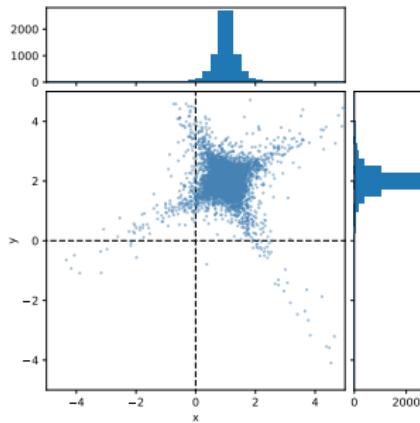
Non linear methods

Autoencoders

The principle

Setting

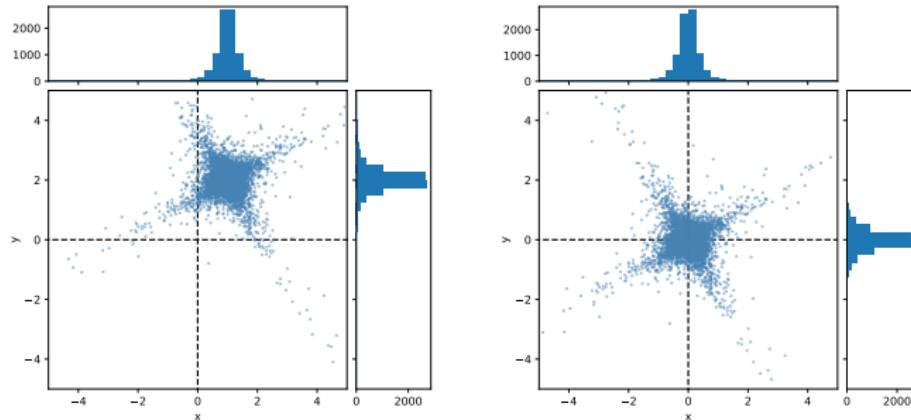
- ▶ A dataset $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \in \mathbb{R}^{n \times d}$ with d big.
- ▶ Suppose for simplicity $\sum_{i=1}^n \mathbf{x}_i = 0$ (centered data), i.e. $\mathbf{X}^\top \mathbf{1}_n = 0$.



The principle

Setting

- ▶ A dataset $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top \in \mathbb{R}^{n \times d}$ with d big.
- ▶ Suppose for simplicity $\sum_{i=1}^n \mathbf{x}_i = 0$ (centered data), i.e. $\mathbf{X}^\top \mathbf{1}_n = 0$.



Goal

- ▶ Find coordinates $\tilde{\mathbf{x}}_i = f(\mathbf{x}_i)$ in \mathbb{R}^k with $k \ll d$.
- ▶ The new data $(\tilde{\mathbf{x}}_i)_{i \in [n]}$ should “look like” \mathbf{X} (to be defined).
- ▶ First step: linear mapping f !

The principle of PCA

Linear mapping according to a reconstruction principle

- ▶ Find $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_k) \in \mathbb{R}^{d \times k}$ with $\mathbf{u}_n^\top \mathbf{u}_m = \delta_{nm}$ (orthonormal vectors)
- ▶ Dimension reduction via linear mapping: $\tilde{\mathbf{x}}_i = \mathbf{U}^\top \mathbf{x}_i \in \mathbb{R}^k$

The principle of PCA

Linear mapping according to a reconstruction principle

- ▶ Find $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_k) \in \mathbb{R}^{d \times k}$ with $\mathbf{u}_n^\top \mathbf{u}_m = \delta_{nm}$ (orthonormal vectors)
- ▶ Dimension reduction via linear mapping: $\tilde{\mathbf{x}}_i = \mathbf{U}^\top \mathbf{x}_i \in \mathbb{R}^k$
- ▶ What make a \mathbf{U} “better” than another?

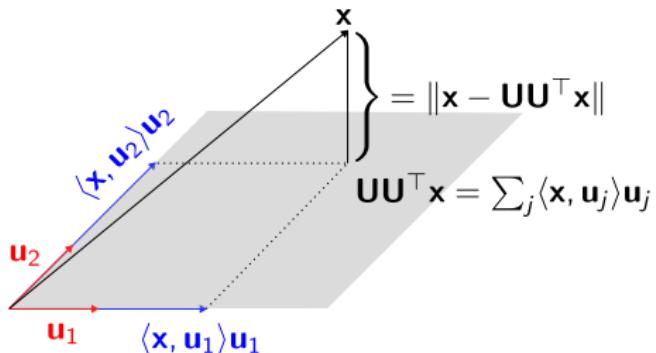
The principle of PCA

Linear mapping according to a reconstruction principle

- ▶ Find $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_k) \in \mathbb{R}^{d \times k}$ with $\mathbf{u}_n^\top \mathbf{u}_m = \delta_{nm}$ (orthonormal vectors)
- ▶ Dimension reduction via linear mapping: $\tilde{\mathbf{x}}_i = \mathbf{U}^\top \mathbf{x}_i \in \mathbb{R}^k$
- ▶ Reconstruction principle Pearson 1901:

$$\min_{\substack{\mathbf{U} \in \mathbb{R}^{d \times k} \\ \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k}} \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{U}\tilde{\mathbf{x}}_i\|_2^2 = \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{U}\mathbf{U}^\top \mathbf{x}_i\|_2^2$$

- ▶ $\mathbf{U}\mathbf{U}^\top \mathbf{x}_i$ is the orthogonal projection of \mathbf{x}_i onto $\text{span}(\mathbf{u}_1, \dots, \mathbf{u}_k)$



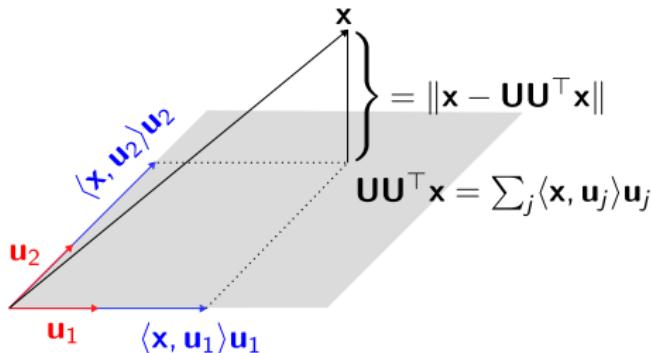
The principle of PCA

Linear mapping according to a reconstruction principle

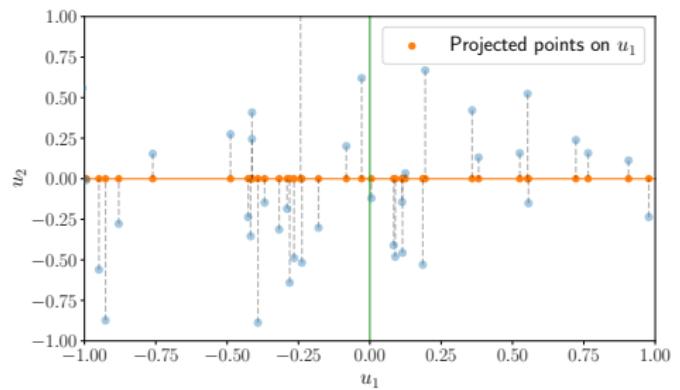
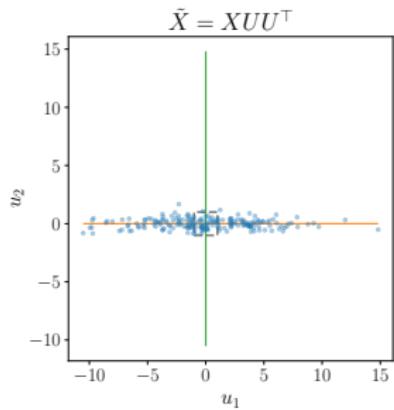
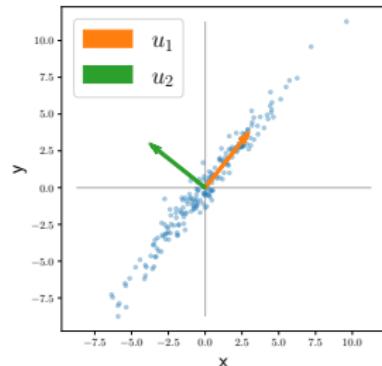
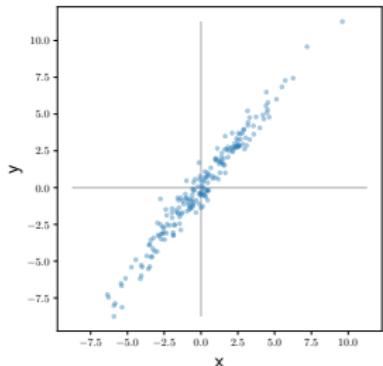
- ▶ Find $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_k) \in \mathbb{R}^{d \times k}$ with $\mathbf{u}_n^\top \mathbf{u}_m = \delta_{nm}$ (orthonormal vectors)
- ▶ Dimension reduction via linear mapping: $\tilde{\mathbf{x}}_i = \mathbf{U}^\top \mathbf{x}_i \in \mathbb{R}^k$
- ▶ Reconstruction principle Pearson 1901:

$$\min_{\substack{\mathbf{U} \in \mathbb{R}^{d \times k} \\ \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k}} \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{U}\tilde{\mathbf{x}}_i\|_2^2 = \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{U}\mathbf{U}^\top \mathbf{x}_i\|_2^2$$

- ▶ After finding a sol. \mathbf{U}^* , $\mathbf{x}_i \approx \sum_{j=1}^k \langle \mathbf{x}_i, \mathbf{u}_j^* \rangle \mathbf{u}_j^*$ (equality when $k = d$).



Illustration



Variance interpretation

Equivalent problem

- The PCA problem is equivalent to the *non-convex* quadratic problem (board):

$$\max_{\substack{\mathbf{U} \in \mathbb{R}^{d \times k} \\ \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k}} \frac{1}{n} \sum_{i=1}^n \|\mathbf{U}^\top \mathbf{x}_i\|_2^2 = \text{tr} \left(\mathbf{U}^\top \underbrace{\left(\frac{1}{n} \mathbf{X}^\top \mathbf{X} \right)}_{\hat{\Sigma}} \mathbf{U} \right)$$

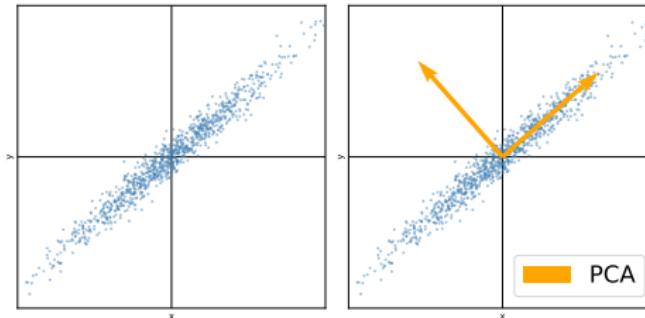
Variance interpretation

Equivalent problem

- The PCA problem is equivalent to the *non-convex* quadratic problem (board):

$$\max_{\substack{\mathbf{U} \in \mathbb{R}^{d \times k} \\ \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k}} \frac{1}{n} \sum_{i=1}^n \|\mathbf{U}^\top \mathbf{x}_i\|_2^2 = \text{tr} \left(\mathbf{U}^\top \underbrace{\left(\frac{1}{n} \mathbf{X}^\top \mathbf{X} \right)}_{\widehat{\Sigma}} \mathbf{U} \right)$$

- Equivalent to maximizing the **variance of the projected samples** $\tilde{\mathbf{x}}_i$.



- Empirical covariance matrix $\widehat{\Sigma} = \frac{1}{n} \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{d \times d}$

Recap: Two views on PCA

The PCA is the linear mapping $\mathbf{x} \mapsto \tilde{\mathbf{x}} = \mathbf{U}\mathbf{x} \in \mathbb{R}^k$ that (equivalently):

- ▶ minimizes the reconstruction error

$$\min_{\substack{\mathbf{U} \in \mathbb{R}^{d \times k} \\ \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k}} \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{U}\mathbf{U}^\top \mathbf{x}_i\|^2$$

- ▶ maximizes the variance of the projected data

$$\max_{\substack{\mathbf{U} \in \mathbb{R}^{d \times k} \\ \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k}} \frac{1}{n} \sum_{i=1}^n \|\mathbf{U}\mathbf{x}_i\|^2$$

Now how do we compute this optimal \mathbf{U} ?

Ky-Fan theorem

Fan 1949

Let $\mathbf{A} \in \mathbb{R}^{d \times d}$ **symmetric** with eigenvalues $\lambda_1 \geq \dots \geq \lambda_d$ and $k \leq d$. Then $\sum_{i=1}^k \lambda_i$ (*resp.* $\sum_{i=1}^k \lambda_{d+i-1}$) is the maximum (*resp.* minimum) of $\text{tr}(\mathbf{U}^\top \mathbf{A} \mathbf{U})$ over $\{\mathbf{U} \in \mathbb{R}^{d \times k}, \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k\}$. In particular

$$\max_{\mathbf{U} \in \mathbb{R}^{d \times k}, \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k} \text{tr}(\mathbf{U}^\top \mathbf{A} \mathbf{U}) = \sum_{i=1}^k \lambda_i. \quad (1)$$

The solution of (1) is given by $\mathbf{U}^* = (\mathbf{u}_1, \dots, \mathbf{u}_k)$ where $\mathbf{u}_1, \dots, \mathbf{u}_k$ are eigenvectors of \mathbf{A} associated to the top- k eigenvalues.

Ky-Fan theorem

Fan 1949

Let $\mathbf{A} \in \mathbb{R}^{d \times d}$ **symmetric** with eigenvalues $\lambda_1 \geq \dots \geq \lambda_d$ and $k \leq d$. Then $\sum_{i=1}^k \lambda_i$ (*resp.* $\sum_{i=1}^k \lambda_{d+i-1}$) is the maximum (*resp.* minimum) of $\text{tr}(\mathbf{U}^\top \mathbf{A} \mathbf{U})$ over $\{\mathbf{U} \in \mathbb{R}^{d \times k}, \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k\}$. In particular

$$\max_{\mathbf{U} \in \mathbb{R}^{d \times k}, \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k} \text{tr}(\mathbf{U}^\top \mathbf{A} \mathbf{U}) = \sum_{i=1}^k \lambda_i. \quad (1)$$

The solution of (1) is given by $\mathbf{U}^* = (\mathbf{u}_1, \dots, \mathbf{u}_k)$ where $\mathbf{u}_1, \dots, \mathbf{u}_k$ are eigenvectors of \mathbf{A} associated to the top- k eigenvalues.

Consequences for PCA

- ▶ Solution of PCA: find the k largest eigenvalues of $\widehat{\Sigma} = \frac{1}{n} \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{d \times d}$.
- ▶ Solution $\mathbf{U}^* = (\mathbf{u}_1^*, \dots, \mathbf{u}_k^*)$ associated to the top- k eigenvalues of $\widehat{\Sigma}$.
- ▶ $\mathbf{u}_1^* \in \mathbb{R}^d, \dots, \mathbf{u}_k^* \in \mathbb{R}^d$ are called *principal components*.

Ky-Fan theorem

Fan 1949 (proof on board)

Let $\mathbf{A} \in \mathbb{R}^{d \times d}$ **symmetric** with eigenvalues $\lambda_1 \geq \dots \geq \lambda_d$ and $k \leq d$. Then $\sum_{i=1}^k \lambda_i$ (*resp.* $\sum_{i=1}^k \lambda_{d+i-1}$) is the maximum (*resp.* minimum) of $\text{tr}(\mathbf{U}^\top \mathbf{A} \mathbf{U})$ over $\{\mathbf{U} \in \mathbb{R}^{d \times k}, \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k\}$. In particular

$$\max_{\mathbf{U} \in \mathbb{R}^{d \times k}, \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k} \text{tr}(\mathbf{U}^\top \mathbf{A} \mathbf{U}) = \sum_{i=1}^k \lambda_i. \quad (1)$$

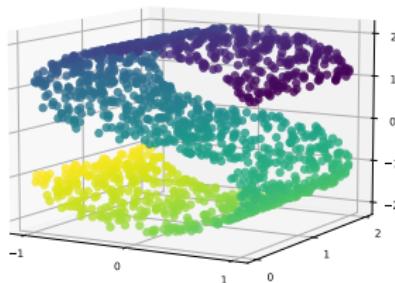
The solution of (1) is given by $\mathbf{U}^* = (\mathbf{u}_1, \dots, \mathbf{u}_k)$ where $\mathbf{u}_1, \dots, \mathbf{u}_k$ are eigenvectors of \mathbf{A} associated to the top- k eigenvalues.

Consequences for PCA

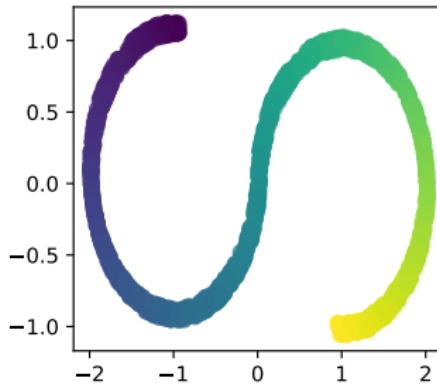
- ▶ Solution of PCA: find the k largest eigenvalues of $\widehat{\Sigma} = \frac{1}{n} \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{d \times d}$.
- ▶ Solution $\mathbf{U}^* = (\mathbf{u}_1^*, \dots, \mathbf{u}_k^*)$ associated to the top- k eigenvalues of $\widehat{\Sigma}$.
- ▶ $\mathbf{u}_1^* \in \mathbb{R}^d, \dots, \mathbf{u}_k^* \in \mathbb{R}^d$ are called *principal components*.

Example with 3D data

- ▶ Simple 3D data.

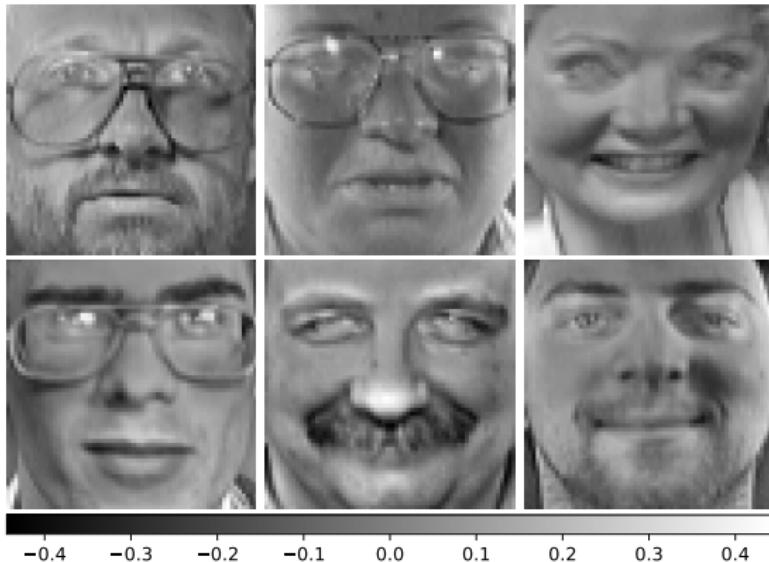


- ▶ Projection onto the two firsts principal components ($d = 3 \rightarrow k = 2$).
PCA



Example with eigenfaces

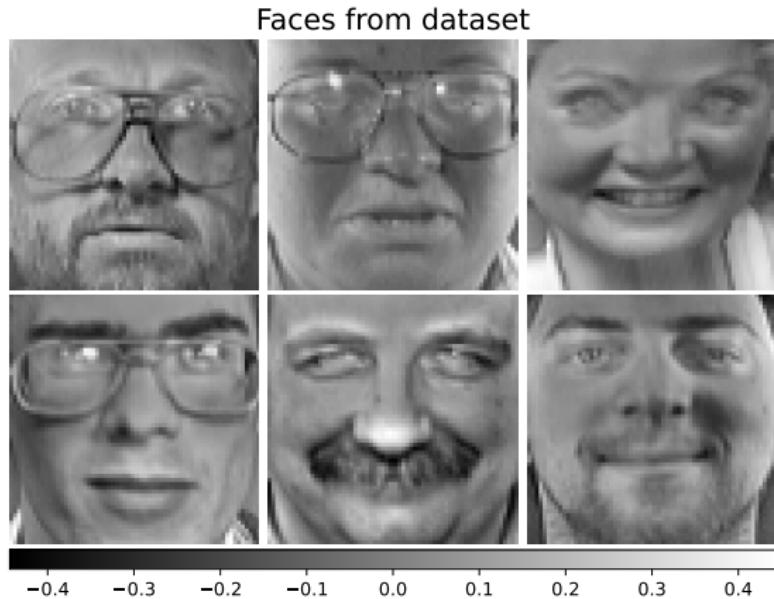
Faces from dataset



Setting

- ▶ Each image is a vector $\mathbf{x}_i \in \mathbb{R}^{4096}$ ($d = 4096$ pixels), $n = 400$ images.

Example with eigenfaces



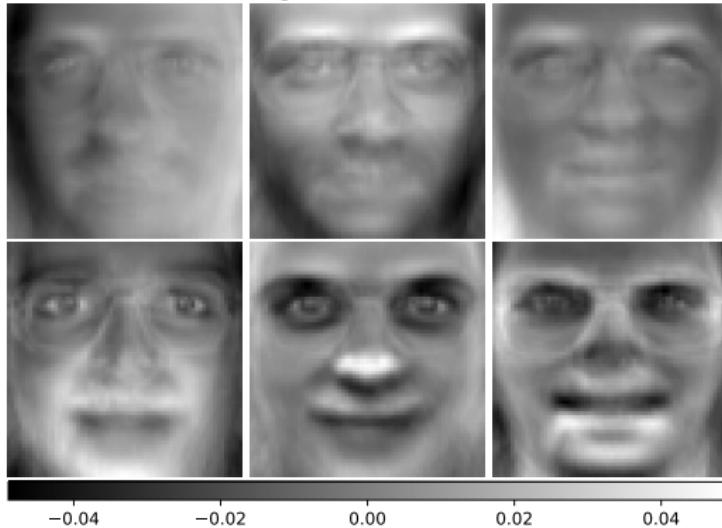
Setting

- ▶ Each image is a vector $\mathbf{x}_i \in \mathbb{R}^{4096}$ ($d = 4096$ pixels), $n = 400$ images.
- ▶ Find k “standard faces”: principal components $\mathbf{u}_1^*, \dots, \mathbf{u}_k^* \in \mathbb{R}^d$.
- ▶ Idea: explain images via $\mathbf{x}_i \approx \sum_{j=1}^k \langle \mathbf{x}_i, \mathbf{u}_j^* \rangle \mathbf{u}_j^*$, in other words
image $\approx \alpha_1 \times \text{eigenface 1} + \dots + \alpha_k \times \text{eigenface } k$

Example with eigenfaces

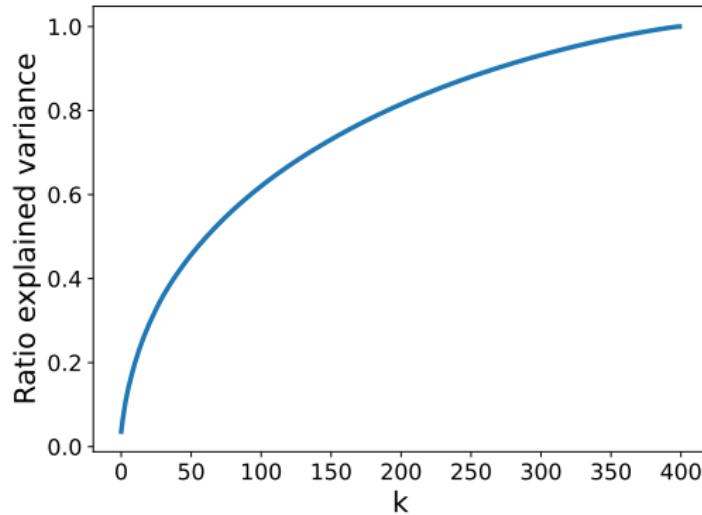
- ▶ Result with $k = 6$

Eigenfaces - SVD



Example with eigenfaces

- ▶ How to choose k ? ratio explained variance: $r = \sum_{i=1}^k \lambda_i / \sum_{i=1}^d \lambda_i$



- ▶ Explanation:

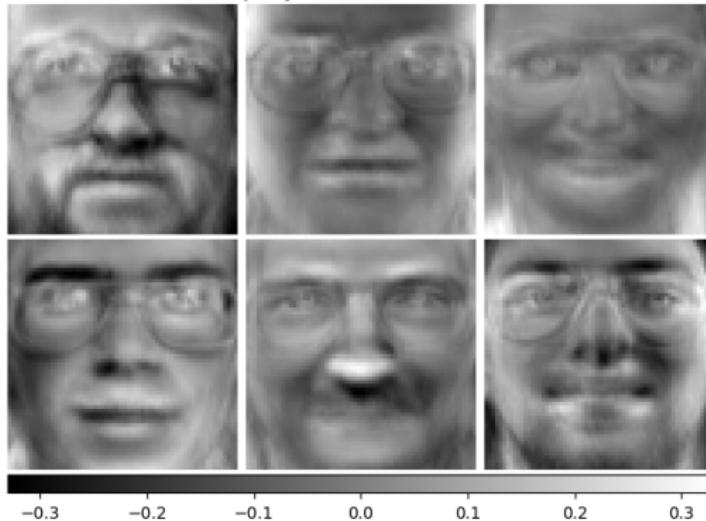
$$r = \sum_{i=1}^k \lambda_i / \sum_{i=1}^d \lambda_i = \text{tr} \left((\mathbf{U}^*)^\top \widehat{\Sigma} (\mathbf{U}^*) \right) / \text{tr}(\widehat{\Sigma})$$

$$= \frac{1}{n} \sum_{i=1}^n \|\tilde{\mathbf{x}}_i\|_2^2 / \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i\|_2^2$$

Example with eigenfaces

- ▶ Reconstruction with $k = 30$

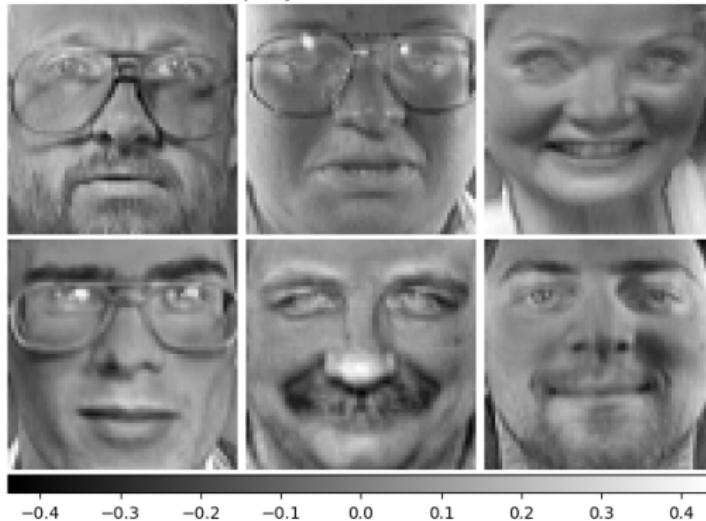
Faces proj then reconstruction



Example with eigenfaces

- ▶ Reconstruction with $k = 300$

Faces proj then reconstruction



How to compute the PCA?

The “naive” way

- ▶ Find the eigenvalue decomposition of $\widehat{\Sigma} = \frac{1}{n} \mathbf{X}^\top \mathbf{X} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top$
- ▶ Compute $\widehat{\Sigma}$: $\mathcal{O}(nd^2)$ operations.
- ▶ Eigenvalue decomposition : $\mathcal{O}(d^3)$ operations.
- ▶ Keep only the k -largest eigenvalues associated to k eigenvectors.
- ▶ Space complexity: $\mathcal{O}(d^2)$
- ▶ Time complexity: $\mathcal{O}(nd^2 + d^3)$

How to compute the PCA?

The “naive” way

- ▶ Find the eigenvalue decomposition of $\widehat{\Sigma} = \frac{1}{n} \mathbf{X}^\top \mathbf{X} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top$
- ▶ Compute $\widehat{\Sigma}$: $\mathcal{O}(nd^2)$ operations.
- ▶ Eigenvalue decomposition : $\mathcal{O}(d^3)$ operations.
- ▶ Keep only the k -largest eigenvalues associated to k eigenvectors.
- ▶ Space complexity: $\mathcal{O}(d^2)$
- ▶ Time complexity: $\mathcal{O}(nd^2 + d^3)$

The right way

- ▶ Compute the singular value decomposition (SVD) of \mathbf{X} !

Table of contents

Why dimension reduction ?

Principal component analysis

The principle of PCA

Singular value decomposition

Minimum distortion embedding

General setting

Random projections

Non linear methods

Autoencoders

One of the most useful tool in linear algebra

Singular value decomposition

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$. Then \mathbf{X} can be decomposed as

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top \tag{2}$$

where $\mathbf{U} \in \mathbb{R}^{n \times n}$, $\mathbf{V} \in \mathbb{R}^{d \times d}$ are *unitary* ($\mathbf{U}^\top \mathbf{U} = \mathbf{U}\mathbf{U}^\top = \mathbf{I}_n$, $\mathbf{V}^\top \mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}_d$) and $\Sigma \in \mathbb{R}^{n \times d}$ is a rectangular diagonal matrix with *non-negative* real numbers $(\sigma_i)_{i \in [\min\{n,d\}]}$ on the diagonal, called *singular values*.

One of the most useful tool in linear algebra

Singular value decomposition

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$. Then \mathbf{X} can be decomposed as

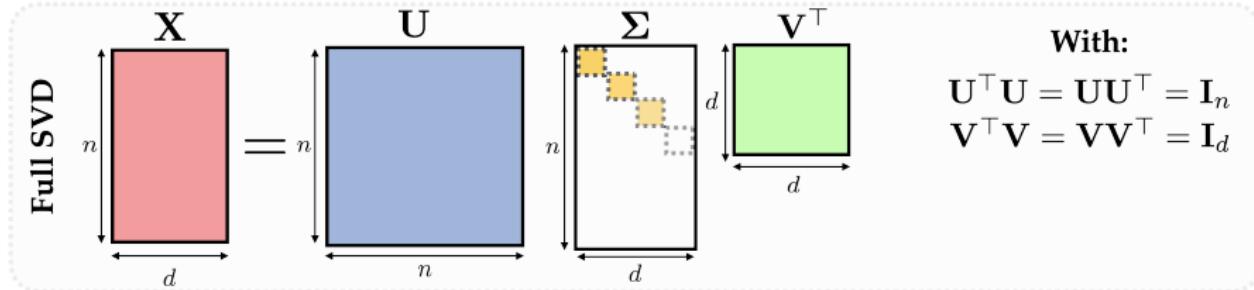
$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top \tag{2}$$

where $\mathbf{U} \in \mathbb{R}^{n \times n}$, $\mathbf{V} \in \mathbb{R}^{d \times d}$ are *unitary* ($\mathbf{U}^\top \mathbf{U} = \mathbf{U}\mathbf{U}^\top = \mathbf{I}_n$, $\mathbf{V}^\top \mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}_d$) and $\Sigma \in \mathbb{R}^{n \times d}$ is a rectangular diagonal matrix with *non-negative* real numbers $(\sigma_i)_{i \in [\min\{n,d\}]}$ on the diagonal, called *singular values*.

Properties

- ▶ $\text{rank}(\mathbf{X}) = \text{number of non-zero } \sigma'_i$ s.
- ▶ The *columns* of \mathbf{V} are eigenvectors of $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{d \times d}$.
- ▶ The *columns* of \mathbf{U} are eigenvectors of $\mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{n \times n}$.
- ▶ We have $\sigma_i = \sqrt{\text{eigenvalue}_i(\mathbf{X}^\top \mathbf{X})} = \sqrt{\text{eigenvalue}_i(\mathbf{X}\mathbf{X}^\top)}$.
- ▶ We have the relations $\mathbf{X}\mathbf{v}_i = \sigma_i\mathbf{u}_i$, $\mathbf{X}^\top \mathbf{u}_i = \sigma_i\mathbf{v}_i$.

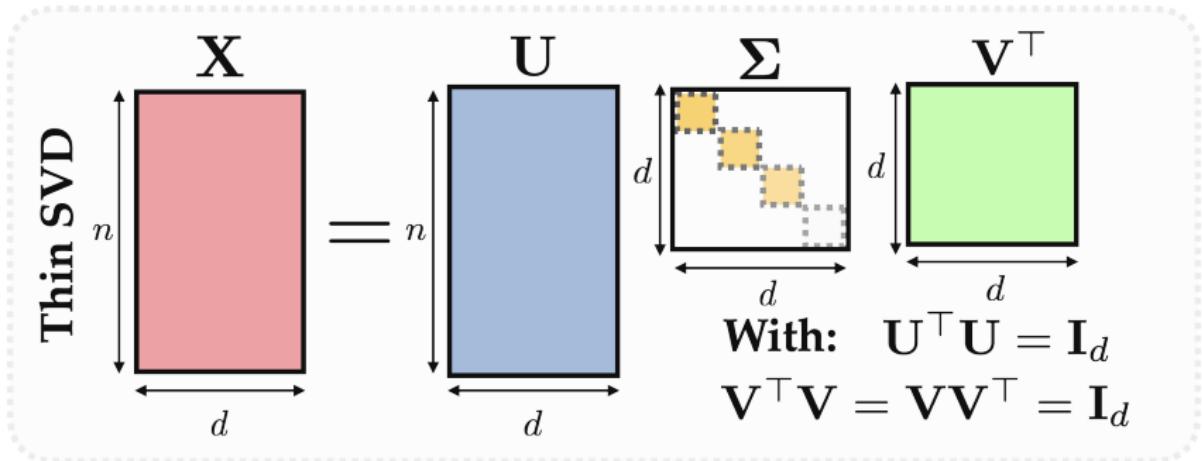
SVD: many flavors



Full SVD (image in case $n \geq d$)

- ▶ Generalizes eigenvalue decomposition for non-symmetric matrices.
- ▶ Complexity: $\mathcal{O}(nd \min\{n, d\})$ (Golub–Reinsch algorithm see [Cline and Dhillon 2006](#)).
- ▶ To find the eigenvalues of $\mathbf{X}^\top \mathbf{X}$ or $\mathbf{X} \mathbf{X}^\top$ we do not even have to compute these matrices !

SVD: many flavors

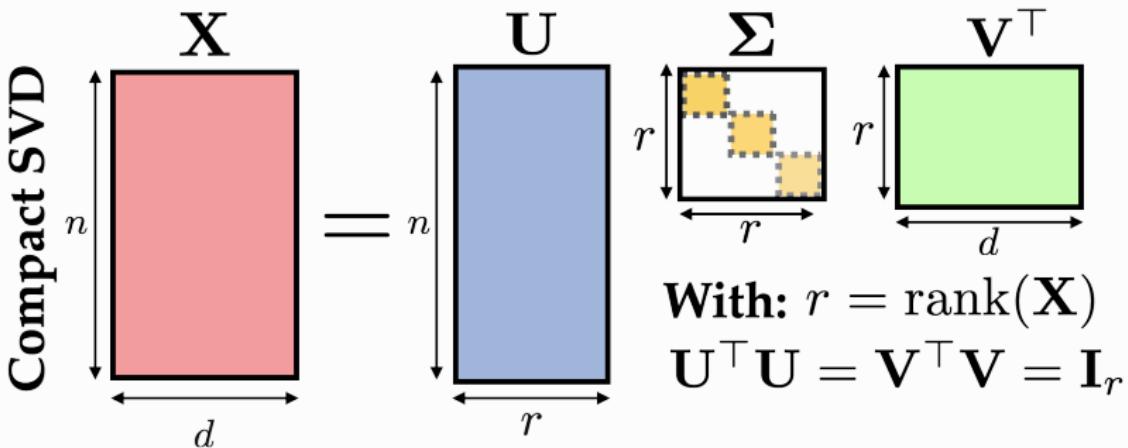


Thin SVD

- If we write $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_n)$, $\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_d)$ the full SVD, then Thin SVD gives:

$$\mathbf{X} = \sum_{i=1}^{\min\{n,d\}} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$$

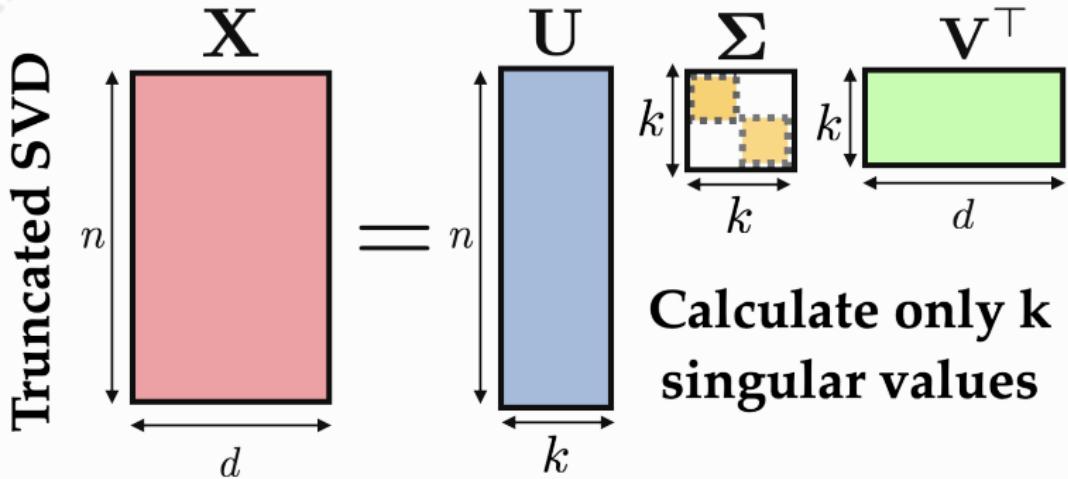
SVD: many flavors



Compact SVD

- ▶ Keep only the non-zero singular values. In particular $r = \text{rank}(\mathbf{X})$.
- ▶ The pseudo-inverse of \mathbf{X} is given by $\mathbf{X}^\dagger = \mathbf{V}\Sigma^{-1}\mathbf{U}^\top$.

SVD: many flavors



Calculate only k singular values

Truncated SVD

- ▶ Best rank- k approximation of \mathbf{X} (in the sense of $\|\cdot\|_F$, $\|\cdot\|_{2 \rightarrow 2}$).
- ▶ The solution of the PCA is given by $\mathbf{V} \in \mathbb{R}^{d \times k}$.
- ▶ The embedding $(\tilde{\mathbf{x}}_i)_{i \in [n]}$ in low dim of PCA is given by $\mathbf{U}\Sigma \in \mathbb{R}^{n \times k}$.
- ▶ Efficient algorithms $\mathcal{O}(ndk)$ Halko, Martinsson, and Tropp 2011.

PCA: a recap

- ▶ Dimension reduction $\mathbb{R}^d \rightarrow \mathbb{R}^k$ via a linear mapping.
- ▶ Defined with a matrix \mathbf{U} with orthonormal columns.
- ▶ Follows a reconstruction principle.
- ▶ Maximizes the variance of the projected samples.
- ▶ Used for compression, interpretation, robustness.
- ▶ Can be computed with SVD in $\mathcal{O}(ndk)$ time.

A word on Kernel PCA

From PCA...

- ▶ PCA solves $\max_{\substack{\mathbf{U} \in \mathbb{R}^{d \times k} \\ \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k}} \text{tr}(\mathbf{U}^\top (\mathbf{X}^\top \mathbf{X}) \mathbf{U})$.
- ▶ Eigenvalue decomposition of $\mathbf{X}^\top \mathbf{X} = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top \in \mathbb{R}^{d \times d}$.
- ▶ Same non-zero eigenvalues than $\mathbf{X} \mathbf{X}^\top = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle)_{ij} \in \mathbb{R}^{n \times n}$ (**exercise**).

A word on Kernel PCA

From PCA...

- ▶ PCA solves $\max_{\substack{\mathbf{U} \in \mathbb{R}^{d \times k} \\ \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k}} \text{tr}(\mathbf{U}^\top (\mathbf{X}^\top \mathbf{X}) \mathbf{U}).$
- ▶ Eigenvalue decomposition of $\mathbf{X}^\top \mathbf{X} = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top \in \mathbb{R}^{d \times d}$.
- ▶ Same non-zero eigenvalues than $\mathbf{X} \mathbf{X}^\top = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle)_{ij} \in \mathbb{R}^{n \times n}$ (exercise).

... to Kernel PCA Schölkopf, Smola, and Müller 2005

- ▶ PCA in a *high-dimensional non-linear* embedding $\Phi(\mathbf{x})$ of the data.
- ▶ *Kernel trick:* embedding is *implicit* we only need $\mathbf{K} = (\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle)_{ij}$.
- ▶ Kernel PCA: eigenvalue decomposition of $\mathbf{K} \in \mathbb{R}^{n \times n}$.
- ▶ More powerful but expensive for large n .

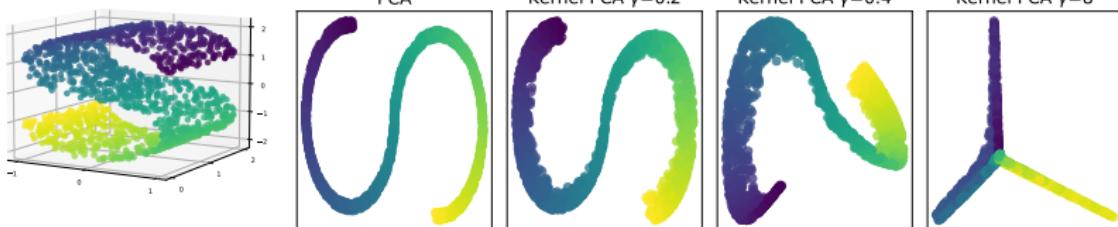
A word on Kernel PCA

From PCA...

- ▶ PCA solves $\max_{\substack{\mathbf{U} \in \mathbb{R}^{d \times k} \\ \mathbf{U}^\top \mathbf{U} = \mathbf{I}_k}} \text{tr}(\mathbf{U}^\top (\mathbf{X}^\top \mathbf{X}) \mathbf{U}).$
- ▶ Eigenvalue decomposition of $\mathbf{X}^\top \mathbf{X} = \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top \in \mathbb{R}^{d \times d}$.
- ▶ Same non-zero eigenvalues than $\mathbf{X} \mathbf{X}^\top = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle)_{ij} \in \mathbb{R}^{n \times n}$ (exercise).

... to Kernel PCA Schölkopf, Smola, and Müller 2005

- ▶ PCA in a *high-dimensional non-linear* embedding $\Phi(\mathbf{x})$ of the data.
- ▶ *Kernel trick*: embedding is *implicit* we only need $\mathbf{K} = (\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle)_{ij}$.
- ▶ Kernel PCA: eigenvalue decomposition of $\mathbf{K} \in \mathbb{R}^{n \times n}$.
- ▶ More powerful but expensive for large n .



A more general setting: dictionary learning

From PCA...

- ▶ One principle of PCA is to represent a sample as a linear combination $\mathbf{x} \approx \sum_{j=1}^k \alpha_j \mathbf{d}_j$ with $\mathbf{d}_j \in \mathbb{R}^d$.
- ▶ PCA: $\alpha_j = \langle \mathbf{x}, \mathbf{u}_j^* \rangle$, $\mathbf{d}_j = \mathbf{u}_j^*$ principal component.

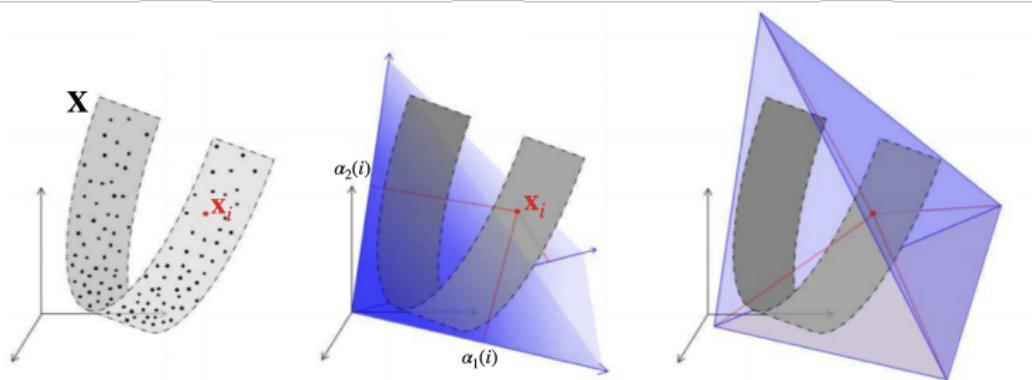
A more general setting: dictionary learning

From PCA...

- ▶ One principle of PCA is to represent a sample as a linear combination $\mathbf{x} \approx \sum_{j=1}^k \alpha_j \mathbf{d}_j$ with $\mathbf{d}_j \in \mathbb{R}^d$.
- ▶ PCA: $\alpha_j = \langle \mathbf{x}, \mathbf{u}_j^* \rangle$, $\mathbf{d}_j = \mathbf{u}_j^*$ principal component.

... to dictionary learning (DL)

- ▶ Represent \mathbf{x} in another “base”: $\mathbf{x} \approx \mathbf{D}\boldsymbol{\alpha}$ (e.g. Fourier/Wavelet basis).
- ▶ $\mathbf{D} \in \mathbb{R}^{d \times k}$ is the *dictionary*. k might be bigger than d ! (overcomplete)
- ▶ $\boldsymbol{\alpha} \in \mathbb{R}^k$ is the representation of \mathbf{x} in the dictionary \mathbf{D} .



A more general setting: dictionary learning

Find the representation

- ▶ Given a point \mathbf{x} and a dictionary \mathbf{D} :

$$\hat{\boldsymbol{\alpha}} = \arg \min_{\boldsymbol{\alpha} \in C} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \quad (3)$$

- ▶ When $C = \mathbb{R}^k$, $\mathbf{D}^\top \mathbf{D} = \mathbf{I}_k$ then $\hat{\boldsymbol{\alpha}} = \mathbf{D}^\top \mathbf{x}$.
- ▶ Can also be used with different losses than $\|\cdot\|_2^2$.

A more general setting: dictionary learning

Find the representation

- Given a point \mathbf{x} and a dictionary \mathbf{D} :

$$\hat{\boldsymbol{\alpha}} = \arg \min_{\boldsymbol{\alpha} \in C} \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \quad (3)$$

- When $C = \mathbb{R}^k$, $\mathbf{D}^\top \mathbf{D} = \mathbf{I}_k$ then $\hat{\boldsymbol{\alpha}} = \mathbf{D}^\top \mathbf{x}$.
- Can also be used with different losses than $\|\cdot\|_2^2$.

Learn the representation and the dictionary

- Given a dataset $\mathbf{x}_1, \dots, \mathbf{x}_n$, learn the dictionary and the representations

$$\hat{\mathbf{D}}, \hat{\boldsymbol{\alpha}_1}, \dots, \hat{\boldsymbol{\alpha}_n} = \arg \min_{\substack{\mathbf{D} \in \mathcal{D} \\ \boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_n \in C}} \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2. \quad (4)$$

- When $C = \mathbb{R}^k$, $\mathcal{D} = \{\mathbf{D} \in \mathbb{R}^{d \times k}, \mathbf{D}^\top \mathbf{D} = \mathbf{I}_k\}$ we retrieve the PCA.
- But various possibilities Mairal et al. 2009 !
- Scikit-learn implementation : `sklearn.decomposition.DictionaryLearning`

One example

Sparse dictionary learning

- Given a dataset $\mathbf{x}_1, \dots, \mathbf{x}_n$, learn the dictionary and the representations

$$\widehat{\mathbf{D}}, \widehat{\boldsymbol{\alpha}}_1, \dots, \widehat{\boldsymbol{\alpha}}_n = \arg \min_{\substack{\mathbf{D} \in \mathcal{D} \\ \boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_n \in C}} \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2. \quad (5)$$

- Take $\mathcal{D} = \{\mathbf{D} \in \mathbb{R}^{d \times k} : \forall i, \|\mathbf{d}_i\|_2 = 1\}$ (normalized columns).
- Take $C = \{\boldsymbol{\alpha} : \|\boldsymbol{\alpha}\|_1 \leq \lambda\}$ sparsity promoting regularization.
- Example $d = 2, k = 3$ (not dimension reduction !!)

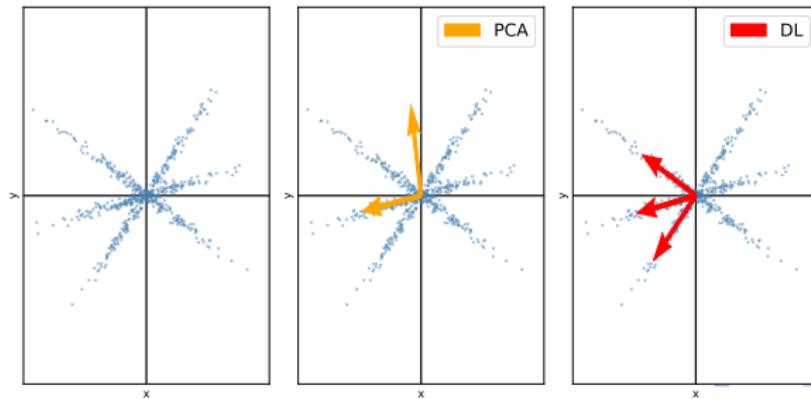


Table of contents

Why dimension reduction ?

Principal component analysis

The principle of PCA

Singular value decomposition

Minimum distortion embedding

General setting

Random projections

Non linear methods

Autoencoders

General setting

Preserve the pairwise distances Agrawal, Ali, and Boyd 2021

- ▶ A dataset $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n), \mathbf{x}_i \in \mathbb{R}^d$.
- ▶ Find a dataset $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n), \tilde{\mathbf{x}}_i \in \mathbb{R}^k, k \ll d$ such that

$$\forall (i, j), \text{similarity}_{\mathbb{R}^d}(\mathbf{x}_i, \mathbf{x}_j) \approx \text{similarity}_{\mathbb{R}^k}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) \text{ or}$$

$$\forall (i, j), \text{dissimilarity}_{\mathbb{R}^d}(\mathbf{x}_i, \mathbf{x}_j) \approx \text{dissimilarity}_{\mathbb{R}^k}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$$

- ▶ Optional: find a mapping $f : \mathbb{R}^d \rightarrow \mathbb{R}^k, \tilde{\mathbf{x}}_i = f(\mathbf{x}_i)$.

General setting

Preserve the pairwise distances Agrawal, Ali, and Boyd 2021

- ▶ A dataset $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n), \mathbf{x}_i \in \mathbb{R}^d$.
- ▶ Find a dataset $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n), \tilde{\mathbf{x}}_i \in \mathbb{R}^k, k \ll d$ such that

$$\begin{aligned}\forall (i, j), \text{ similarity}_{\mathbb{R}^d}(\mathbf{x}_i, \mathbf{x}_j) &\approx \text{similarity}_{\mathbb{R}^k}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) \text{ or} \\ \forall (i, j), \text{ dissimilarity}_{\mathbb{R}^d}(\mathbf{x}_i, \mathbf{x}_j) &\approx \text{dissimilarity}_{\mathbb{R}^k}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)\end{aligned}$$

- ▶ Optional: find a mapping $f : \mathbb{R}^d \rightarrow \mathbb{R}^k, \tilde{\mathbf{x}}_i = f(\mathbf{x}_i)$.

Example with $\|\cdot\|_2^2$

- ▶ Can we find $\delta \in [0, 1]$ and $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$ such that

$$(1 - \delta) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2^2 \leq (1 + \delta) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 ? \quad (6)$$

One necessary result

The notion of distortion Matoušek 2013

- ▶ Consider two metric spaces (X, d_X) and (Y, d_Y) . A function $f : X \rightarrow Y$ is called a μ -embedding if, $\exists r > 0$, such that

$$\forall (\mathbf{x}, \mathbf{x}') \in X, \quad r \cdot d_X(\mathbf{x}, \mathbf{x}') \leq d_Y(f(\mathbf{x}), f(\mathbf{x}')) \leq \mu \times r \cdot d_X(\mathbf{x}, \mathbf{x}'). \quad (7)$$

- ▶ The smallest $\mu > 1$ such that (7) is called the *distortion*.

One necessary result

The notion of distortion Matoušek 2013

- ▶ Consider two metric spaces (X, d_X) and (Y, d_Y) . A function $f : X \rightarrow Y$ is called a μ -embedding if, $\exists r > 0$, such that

$$\forall (\mathbf{x}, \mathbf{x}') \in X, \quad r \cdot d_X(\mathbf{x}, \mathbf{x}') \leq d_Y(f(\mathbf{x}), f(\mathbf{x}')) \leq \mu \times r \cdot d_X(\mathbf{x}, \mathbf{x}'). \quad (7)$$

- ▶ The smallest $\mu > 1$ such that (7) is called the *distortion*.
- ▶ If $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ finite metric space:

$$\forall (i, j) \in \llbracket n \rrbracket^2, \quad r \cdot d_X(\mathbf{x}_i, \mathbf{x}_j) \leq d_Y(f(\mathbf{x}_i), f(\mathbf{x}_j)) \leq \mu \times r \cdot d_X(\mathbf{x}_i, \mathbf{x}_j). \quad (8)$$

One necessary result

The notion of distortion Matoušek 2013

- ▶ Consider two metric spaces (X, d_X) and (Y, d_Y) . A function $f : X \rightarrow Y$ is called a μ -embedding if, $\exists r > 0$, such that

$$\forall (\mathbf{x}, \mathbf{x}') \in X, \quad r \cdot d_X(\mathbf{x}, \mathbf{x}') \leq d_Y(f(\mathbf{x}), f(\mathbf{x}')) \leq \mu \times r \cdot d_X(\mathbf{x}, \mathbf{x}'). \quad (7)$$

- ▶ The smallest $\mu > 1$ such that (7) is called the *distortion*.
- ▶ If $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ finite metric space:

$$\forall (i, j) \in \llbracket n \rrbracket^2, \quad r \cdot d_X(\mathbf{x}_i, \mathbf{x}_j) \leq d_Y(f(\mathbf{x}_i), f(\mathbf{x}_j)) \leq \mu \times r \cdot d_X(\mathbf{x}_i, \mathbf{x}_j). \quad (8)$$

Example

- ▶ Not always possible to have low distortion !
- ▶ Every embedding of the n -point equilateral space $X = K_n$ (every two points have distance 1) into the Euclidean plane $(\mathbb{R}^2, \|\cdot\|_2)$ has distortion at least $\Omega(\sqrt{n})$.

One necessary result

The notion of distortion Matoušek 2013

- ▶ Consider two metric spaces (X, d_X) and (Y, d_Y) . A function $f : X \rightarrow Y$ is called a μ -embedding if, $\exists r > 0$, such that

$$\forall (\mathbf{x}, \mathbf{x}') \in X, \quad r \cdot d_X(\mathbf{x}, \mathbf{x}') \leq d_Y(f(\mathbf{x}), f(\mathbf{x}')) \leq \mu \times r \cdot d_X(\mathbf{x}, \mathbf{x}'). \quad (7)$$

- ▶ The smallest $\mu > 1$ such that (7) is called the *distortion*.
- ▶ If $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ finite metric space:

$$\forall (i, j) \in \llbracket n \rrbracket^2, \quad r \cdot d_X(\mathbf{x}_i, \mathbf{x}_j) \leq d_Y(f(\mathbf{x}_i), f(\mathbf{x}_j)) \leq \mu \times r \cdot d_X(\mathbf{x}_i, \mathbf{x}_j). \quad (8)$$

Example

- ▶ Not always possible to have low distortion !
- ▶ Every embedding of the n -point equilateral space $X = K_n$ (every two points have distance 1) into the Euclidean plane $(\mathbb{R}^k, \|\cdot\|_2)$ has distortion at least $\Omega(n^{1/k})$.

Table of contents

Why dimension reduction ?

Principal component analysis

The principle of PCA

Singular value decomposition

Minimum distortion embedding

General setting

Random projections

Non linear methods

Autoencoders

Johnson-Lindenstrauss lemma

Johnson and Lindenstrauss 1984

Let $0 < \delta < 1$ and *any* dataset $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$. Providing that

$$k > 8\delta^{-2} \log(n),$$

there is a matrix $\mathbf{A} \in \mathbb{R}^{k \times d}$, such that,

$$\forall (i, j) \in \llbracket n \rrbracket^2, (1 - \delta) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \|\mathbf{A}\mathbf{x}_i - \mathbf{A}\mathbf{x}_j\|_2^2 \leq (1 + \delta) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2.$$

Johnson-Lindenstrauss lemma

Johnson and Lindenstrauss 1984

Let $0 < \delta < 1$ and *any* dataset $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$. Providing that

$$k > 8\delta^{-2} \log(n),$$

there is a matrix $\mathbf{A} \in \mathbb{R}^{k \times d}$, such that,

$$\forall (i, j) \in \llbracket n \rrbracket^2, (1 - \delta) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \|\mathbf{Ax}_i - \mathbf{Ax}_j\|_2^2 \leq (1 + \delta) \|\mathbf{x}_i - \mathbf{x}_j\|_2^2.$$

Important comments

- ▶ The mapping is linear + works for any dataset !
- ▶ k does not depend on the dimension d !
- ▶ Magical: \mathbf{A} can be drawn randomly: $A_{ij} \sim \mathcal{N}(0, \frac{1}{k})$.
- ▶ This is tight in some sense [Larsen and Nelson 2017](#).
- ▶ Caveat: $n = 300$ samples, $\delta = 10\%$ already requires $k > 4560$.

Johnson-Lindenstrauss in practice

- ▶ Real dataset in $\mathbb{R}^{38 \times 7129}$, $\delta = 0.15$.
- ▶ For various choices of k , draw random Gaussian $A \sim \mathcal{N}(0, \frac{1}{k}) \in \mathbb{R}^{k \times d}$.
- ▶ Compare distances between \mathbf{Ax}_i s to distances between \mathbf{x}_i s.

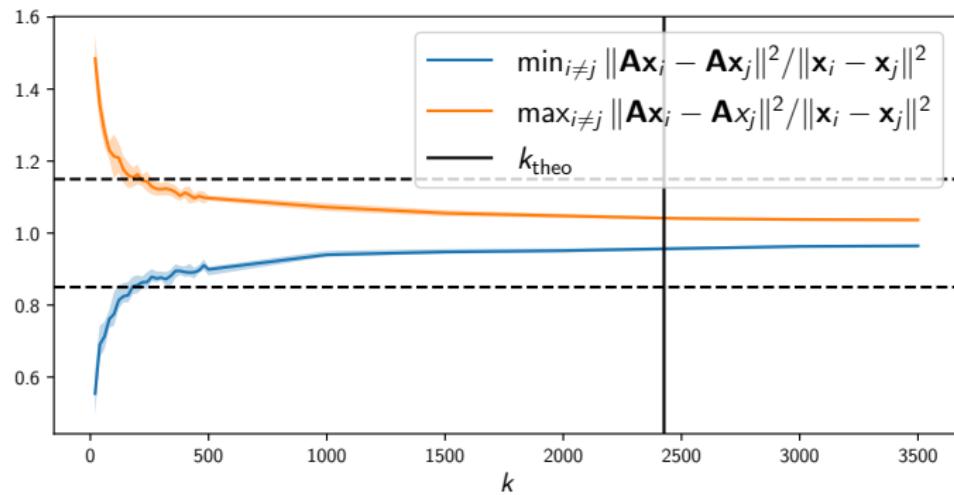


Table of contents

Why dimension reduction ?

Principal component analysis

The principle of PCA

Singular value decomposition

Minimum distortion embedding

General setting

Random projections

Non linear methods

Autoencoders

Multidimensional scaling (MDS)

Learn from pairwise distances

- ▶ Distances in the big space: $D_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$ for some “metric” D .
- ▶ Find $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n \in \mathbb{R}^k$ that minimizes:

$$\text{stress}_D(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n) = \sum_{i \neq j} (D_{ij} - \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2)^2 \quad (9)$$

- ▶ Eq. (9) usually called *stress minimization*: $\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2 \approx D_{ij}$.
- ▶ Can also be used for embedding nodes \mathbf{x}_i of a graph (not only Euclidean).
- ▶ Can be solved with eigenvalue decomposition.
- ▶ No map from the high dim space to the lower dim space.

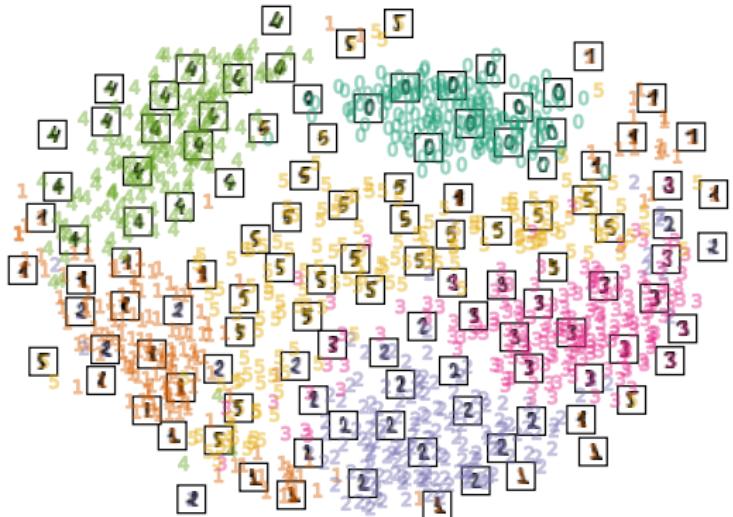
Multidimensional scaling (MDS)

- With digits dataset:

A selection from the 64-dimensional digits dataset

0	1	2	3	4	5	0	1	3	
4	5	0	1	2	3	4	5	0	5
5	5	0	4	1	3	5	1	0	0
2	2	2	0	1	2	3	3	3	3
4	4	1	5	0	5	2	4	0	0
1	3	2	1	4	3	4	3	1	4
3	4	4	6	0	5	7	4	5	4
2	2	2	5	5	4	4	0	0	1
2	3	4	5	0	1	2	3	4	5
0	1	2	3	4	5	0	5	5	5

MDS embedding (time 2.959s)



Stochastic neighbor embedding (SNE)

Learn from pairwise similarities

- ▶ One of the most used algorithm G. E. Hinton and Roweis 2002.
- ▶ Similarities in the high-dim space:

$$P_{ij} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|_2^2/2\sigma_i^2)}, P_{ii} = 0.$$

- ▶ Similarities in the low-dim space:

$$Q_{ij} = \frac{\exp(-\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2^2)}{\sum_{k \neq i} \exp(-\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_k\|_2^2)}, Q_{ii} = 0.$$

- ▶ SNE: find $(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n)$ that minimizes $\text{KL}(\mathbf{P}|\mathbf{Q}) = \sum_{ij} P_{ij} \log(\frac{P_{ij}}{Q_{ij}})$.
- ▶ σ_i local scaling, tuned with *entropic affinities* with fixed *perplexity* Vladymyrov and Carreira-Perpinan 2013.
- ▶ t-SNE variant for the kernel \mathbf{Q} (t-Student) Van der Maaten and G. Hinton 2008.

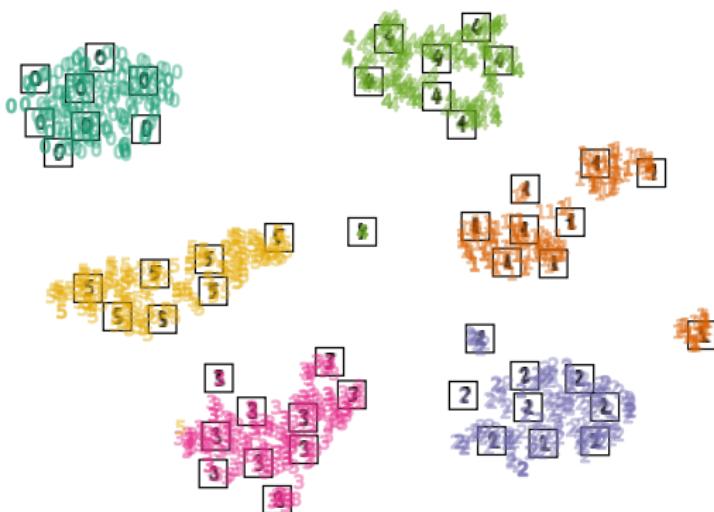
Stochastic neighbor embedding (SNE)

- With digits dataset:

A selection from the 64-dimensional digits dataset

0	1	2	3	4	5	0	1	1	3
4	5	0	1	2	3	4	5	0	5
5	5	0	4	1	3	5	1	0	0
2	2	2	0	1	2	3	3	3	3
4	4	1	5	0	5	2	4	0	0
1	3	2	1	4	3	1	3	1	4
3	4	4	6	0	5	7	4	5	4
2	2	2	2	5	5	4	4	0	0
2	3	4	5	0	1	2	3	4	5
0	1	2	3	4	5	0	5	5	5

t-SNE embedding (time 1.324s)



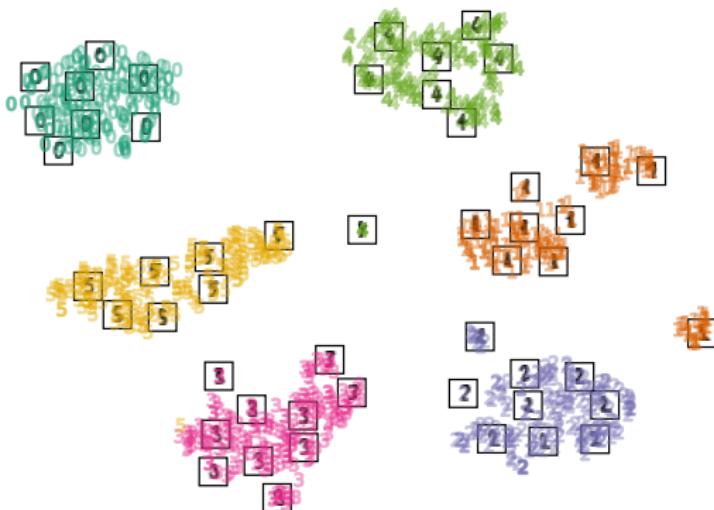
Stochastic neighbor embedding (SNE)

- With digits dataset:

A selection from the 64-dimensional digits dataset

0	1	2	3	4	5	0	1	1	3
4	5	0	1	2	3	4	5	0	5
5	5	0	4	1	3	5	1	0	0
2	2	2	0	1	2	3	3	3	3
4	4	1	5	0	5	2	4	0	0
1	3	2	1	4	3	1	3	1	4
3	4	4	6	0	5	7	1	5	4
2	2	2	2	5	5	4	4	0	0
2	3	4	5	0	1	2	3	4	5
0	1	2	3	4	5	0	5	5	5

t-SNE embedding (time 1.324s)



Stochastic neighbor embedding (SNE)

On the perplexity parameter

- ▶ $(\sigma_i)_i$ local scalings are found so that:

$$\forall i \in \llbracket n \rrbracket, H(P_{i,:}) = - \sum_j P_{ij} \log(P_{ij}) = \log(\text{perp})$$

Stochastic neighbor embedding (SNE)

On the perplexity parameter

- ▶ $(\sigma_i)_i$ local scalings are found so that:

$$\forall i \in \llbracket n \rrbracket, H(P_{i,:}) = - \sum_j P_{ij} \log(P_{ij}) = \log(\text{perp})$$

Be aware of ...

- ▶ (T)SNE has tendency to show non-existent clusters for small perplexity
- ▶ (T)SNE struggles in high-dim! In practice: PCA first.
- ▶ No geometrical relations between clusters.
- ▶ Difficult to interpret, sensitive to perplexity.
- ▶ No mapping from the high dim space to the lower dim space.

t-SNE perp=3



t-SNE perp=50



t-SNE perp=200



t-SNE perp=500



Table of contents

Why dimension reduction ?

Principal component analysis

The principle of PCA

Singular value decomposition

Minimum distortion embedding

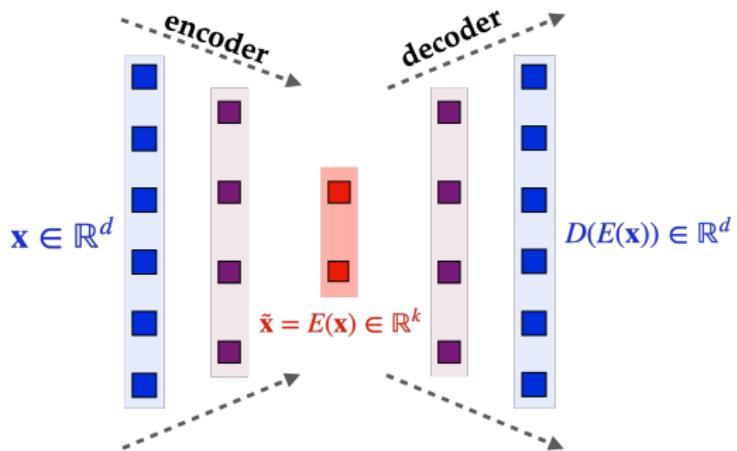
General setting

Random projections

Non linear methods

Autoencoders

Autoencoders



Principle

- ▶ Send the point x from \mathbb{R}^d to \mathbb{R}^k with an *encoder* E .
- ▶ Map back the *code/latent variable* $E(x)$ to the original space with a *decoder* D .
- ▶ Decoded code should be close to the original point.
- ▶ Code dimension $k \ll$ original dimension d .
- ▶ Autoencoder: E and D are neural networks!

Learning autoencoders

- ▶ Architecture of E and D is fixed (number of layers, non-linearity, type of layers)
- ▶ Typical fully-connected neural networks are a combination of matrix multiplication + bias with pointwise non-linearity:

$$g_K \circ \dots \circ g_1 \text{ where } g_k(\mathbf{x}) = \sigma(\mathbf{W}_k \mathbf{x} + \mathbf{b}_k)$$

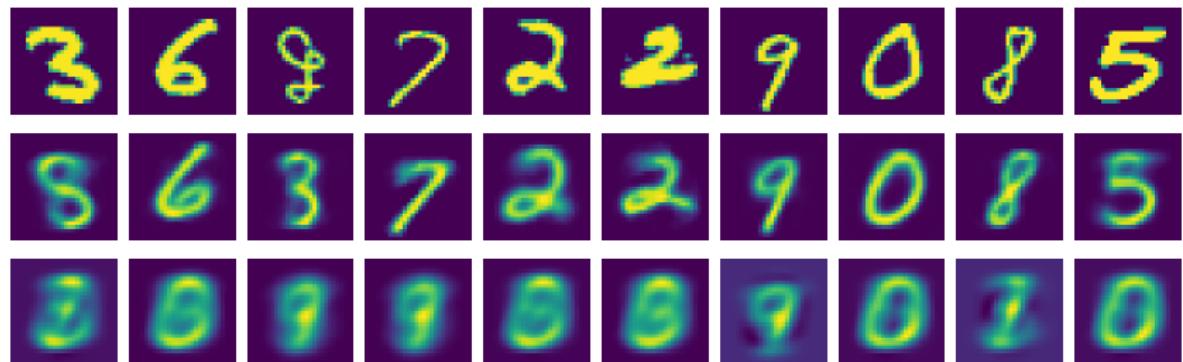
- ▶ Weights are learned by solving:

$$\min_{D,E} \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - D(E(\mathbf{x}_i))\|_2^2$$

- ▶ Optimisation is performed with first-order methods (SGD, Adam).

Autoencoder application to MNIST

- ▶ MNIST data: 28×28 images (784 pixels)
- ▶ Compressed into \mathbb{R}^2 with Autoencoder or PCA



Top: original, middle: autoencoder, bottom: PCA.

References I

-  Agrawal, Akshay, Alnur Ali, and Stephen Boyd (2021). "Minimum-Distortion Embedding". In: *Foundations and Trends® in Machine Learning* 14.
-  Cline, Alan Kaylor and Inderjit S. Dhillon (2006). *Computation of the Singular Value Decomposition*. CRC Press.
-  Fan, Ky (1949). "On a Theorem of Weyl Concerning Eigenvalues of Linear Transformations I*". In: *Proceedings of the National Academy of Sciences* 35.11.
-  Halko, N., P. G. Martinsson, and J. A. Tropp (2011). "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions". In: *SIAM Review* 53.2, pp. 217–288.
-  Hinton, Geoffrey E and Sam Roweis (2002). "Stochastic neighbor embedding". In: *Advances in neural information processing systems* 15.
-  Johnson, William and Joram Lindenstrauss (1984). "Extensions of Lipschitz maps into a Hilbert space". In: *Contemporary Mathematics* 26.
-  Larsen, Kasper Green and Jelani Nelson (2017). "Optimality of the Johnson-Lindenstrauss Lemma". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 633–638.

References II

-  Mairal, Julien et al. (2009). "Online Dictionary Learning for Sparse Coding". In: *International Conference on Machine Learning*.
-  Matoušek, Jiri (2013). *Lecture notes on metric embeddings*. Tech. rep. Technical report, ETH Zürich.
-  Pearson, Karl (1901). "On lines and planes of closest fit to systems of points in space". In: *Philosophical Magazine Series 1 2*, pp. 559–572.
-  Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller (2005). "Kernel principal component analysis". In: *Artificial Neural Networks—ICANN'97: 7th International Conference Lausanne, Switzerland, October 8–10, 1997 Proceedings*. Springer, pp. 583–588.
-  Van der Maaten, Laurens and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.11.
-  Vladymyrov, Max and Miguel Carreira-Perpinan (2013). "Entropic affinities: Properties and efficient numerical computation". In: *International conference on machine learning*. PMLR, pp. 477–485.