cnrs formation entreprises

Fundamentals of Machine Learning

Direction de l'innovation et des relations avec les entreprises

June 22nd-29th 2023

Aurélien Garivier, Rémi Gribonval, Mathurin Massias, Titouan Vayer

Session 2

Unsupervised learning & dimension reduction

Session program

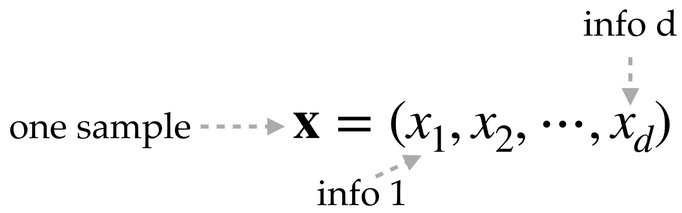
- **Dimension reduction**
  - with focus on Principal Component Analysis
- **Clustering**
  - with focus on K-means

Previous session

- What is Machine Learning ?
- What are the two main types of problem ?
- What's the difference between classification and regression ?
- Examples of ML problems ?
- What types of data can be encountered in ML ?
- What is overfitting ?
- What standard methodology can be used to limit it ?

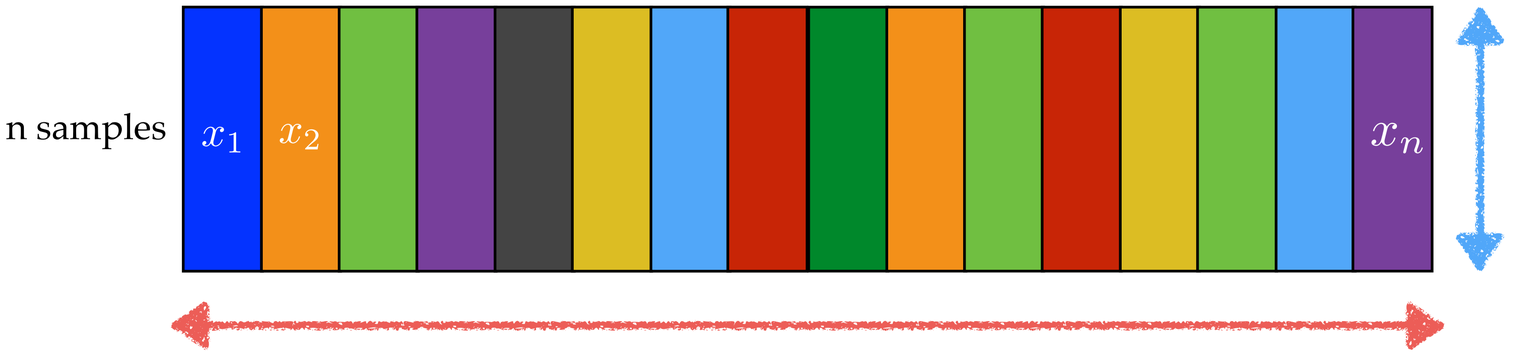
Part 1 : Dimension reduction

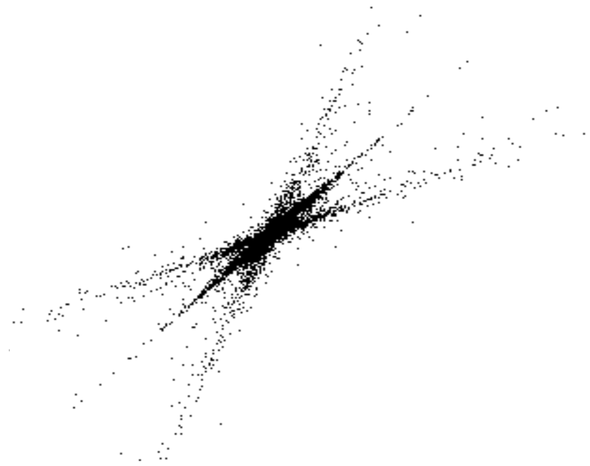
Motivation



Motivation

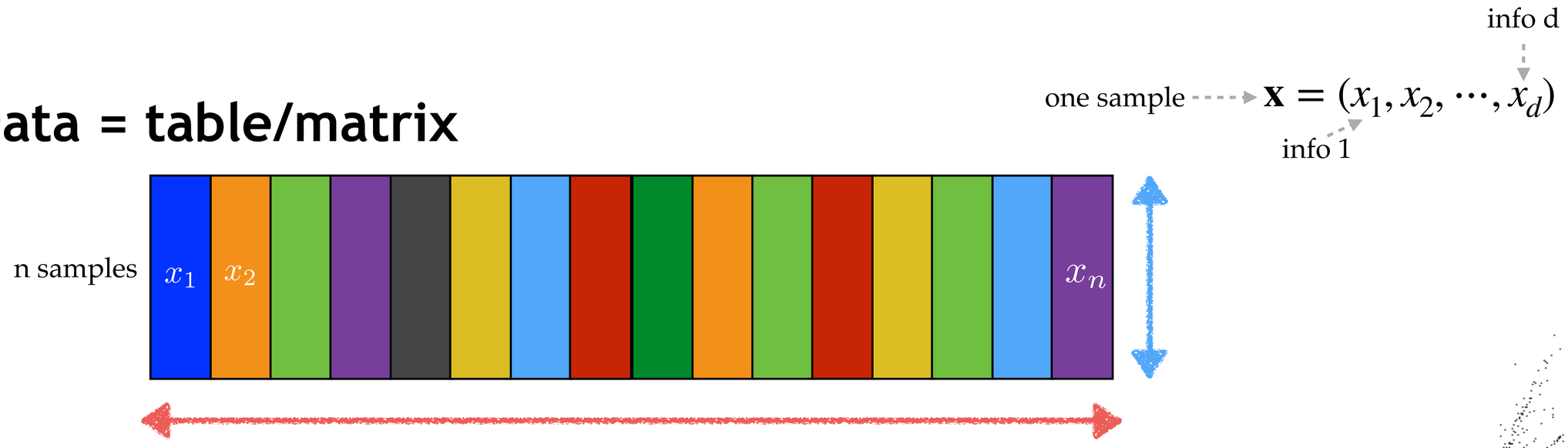
- **Data = table/matrix**



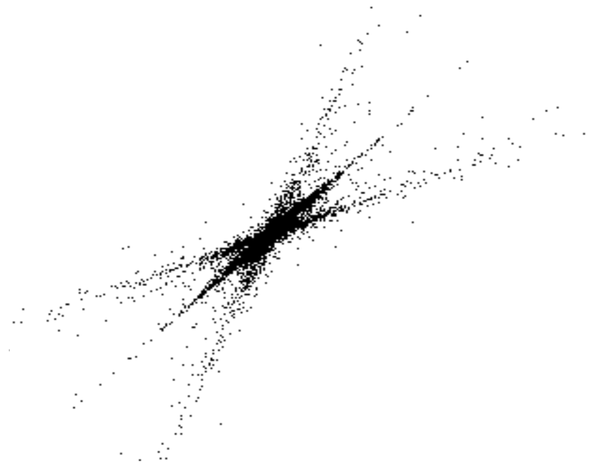


Motivation

- **Data = table/matrix**

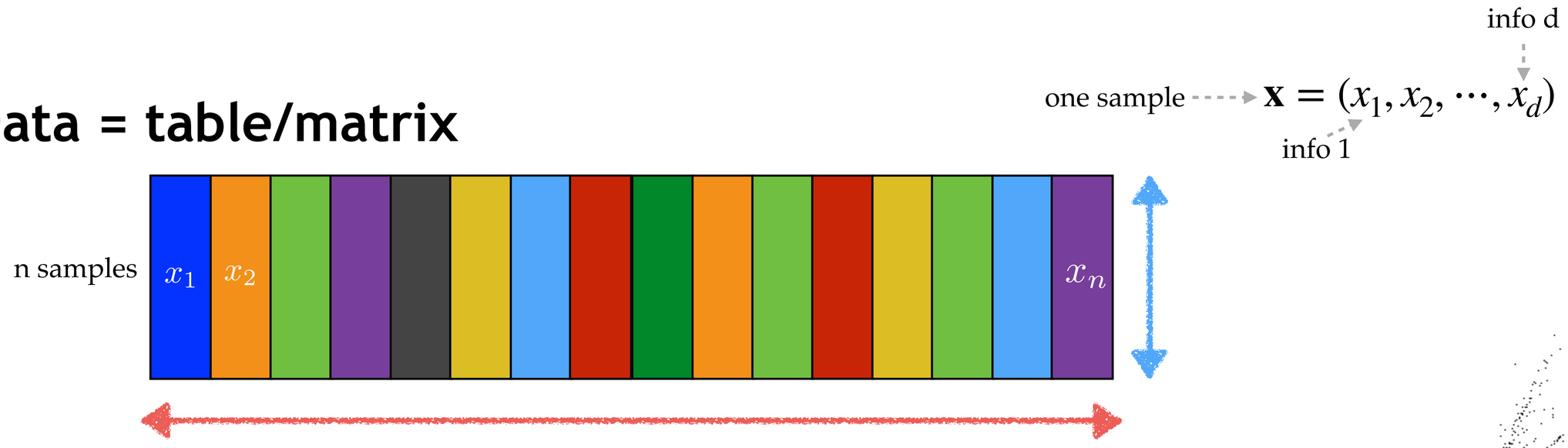


- High-dimensional features *d*
- Large sample size *n* = *"volume"*

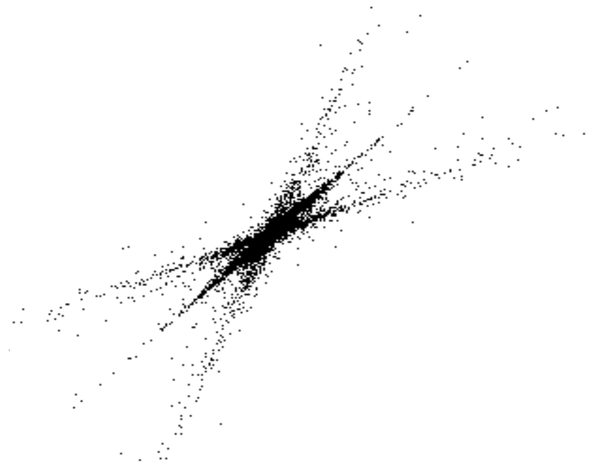


Motivation

- **Data = table/matrix**



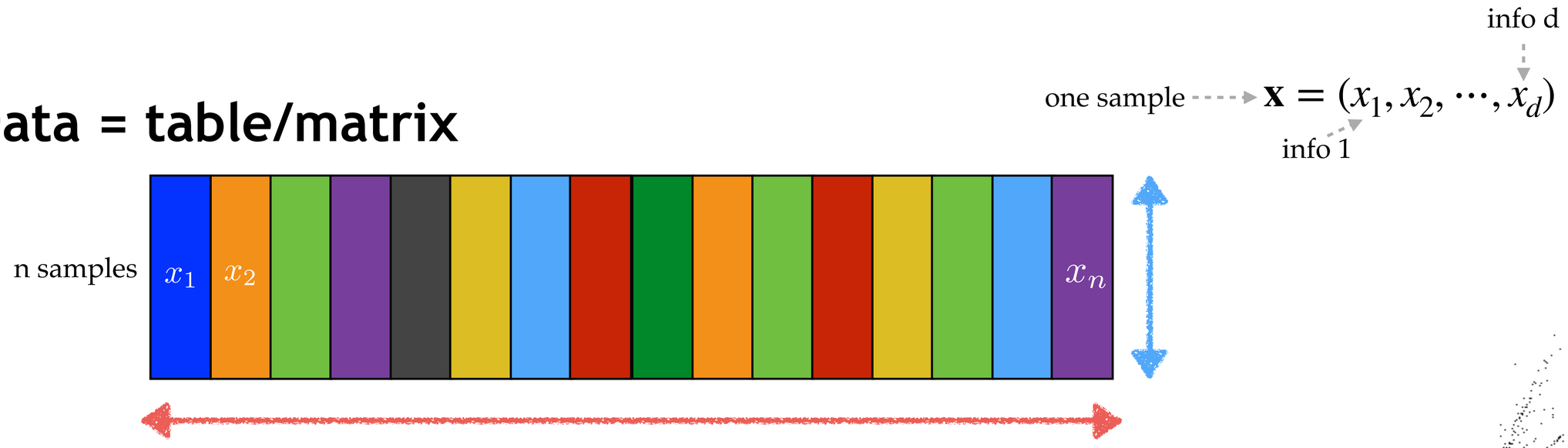
  - High-dimensional features *d*
  - Large sample size *n* = *"volume"*



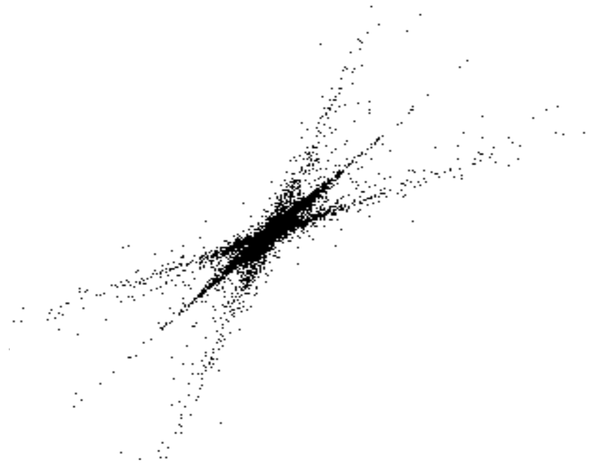
- How to **visualize/understand** the data ?
  - Dimension $d > 2$!

Motivation

- **Data = table/matrix**



  - High-dimensional features *d*
  - Large sample size *n* = *"volume"*



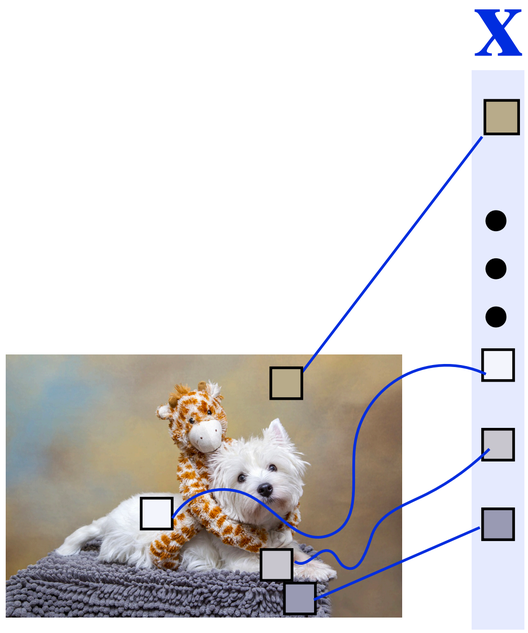
- How to **visualize/understand** the data ?
  - Dimension $d > 2$!
- How to **compress** (storage and transfer) ?

Motivation

- Images:
  - ImageNet

$d = 469 \times 387$

$n = 14 \times 10^6$

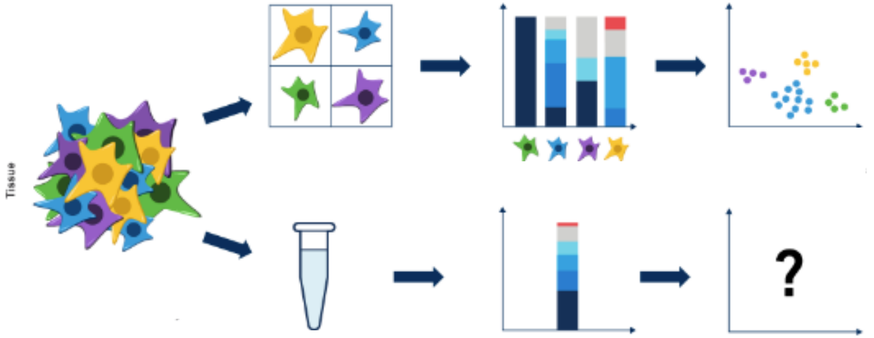


$d = 648 \times 449 = 290952$

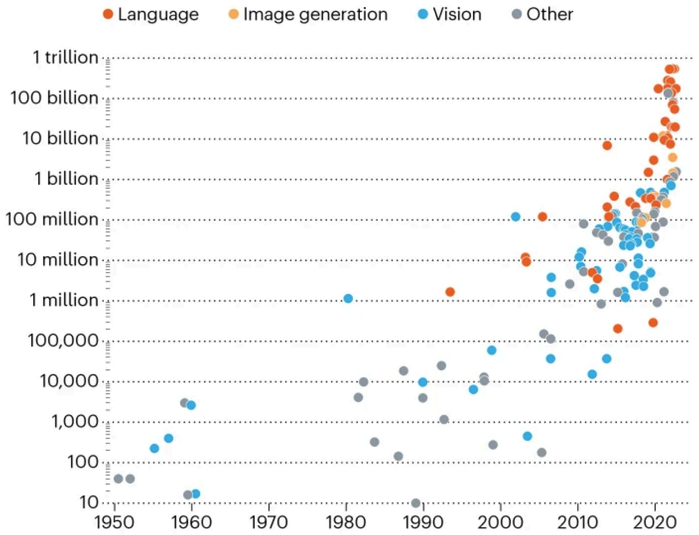


- Biology:
  - Single Cell

$d \approx 10^4 \quad n > 10^6$



- Text: $d \approx 10^4$ (number of possible words) $n =$ bilions (Wikipedia)



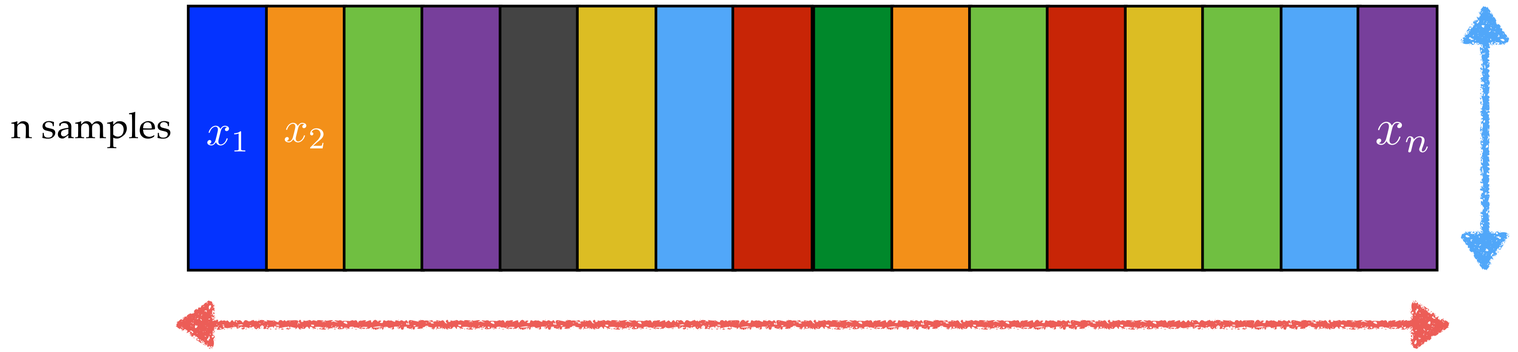
Motivation

info d

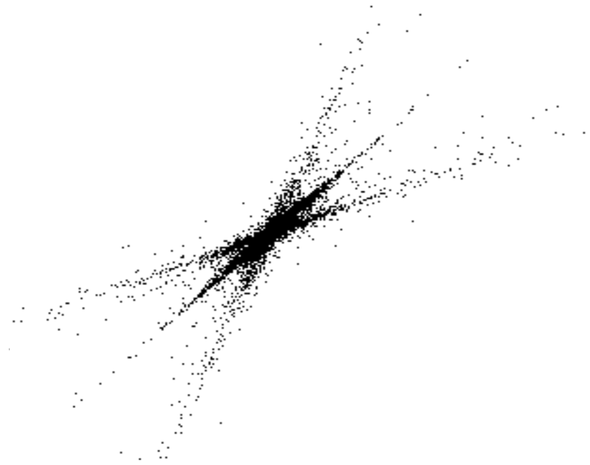
one sample $\mathbf{x} = (x_1, x_2, \cdots, x_d)$

info 1

- **Data = table/matrix**



  - High-dimensional features *d*
  - Large sample size *n* = *"volume"*



- How to **visualize/understand** the data ?
  - Dimension $d > 2$!
- How to **compress** (storage and transfer) ?

➡Projection into a lower-dimensional space = ***dimension reduction***

Which projection to choose ?



Which projection to choose ?





Which projection to choose ?

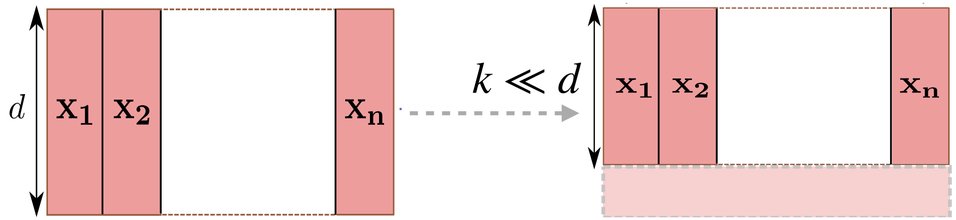


Dimension reduction

- **Training data**
  - $x_i, 1 \leq i \leq n$

Dimension reduction

- **Training data**
  - $x_i, 1 \leq i \leq n$
- **Task** : find a dimension reduction function
  - **Formally:** build $f : x \in \mathbb{R}^d \mapsto \hat{y} = f(x) \in \mathbb{R}^k$
  - **With** $k \ll d$
  - **Unsupervised** : no output y_i available



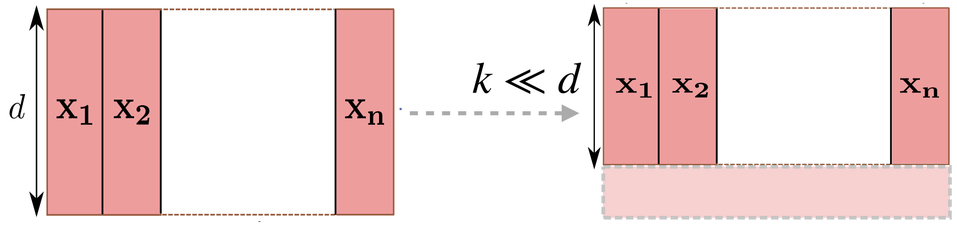
Dimension reduction

Training data

- $x_i, 1 \leq i \leq n$

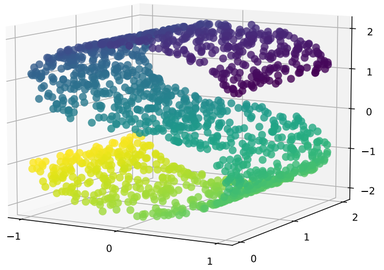
Task : find a dimension reduction function

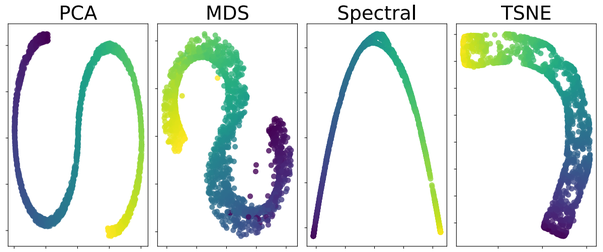
- **Formally:** build $f : x \in \mathbb{R}^d \mapsto \hat{y} = f(x) \in \mathbb{R}^k$
- **With** $k \ll d$
- **Unsupervised** : no output y_i available



For what ?

- ***Visualization***
- ***Compress*** before further use
- But also « ***reveal*** the data structure »
  - ex: spectral approaches (end of the presentation)





First contact - IRIS dataset

➡Notebook

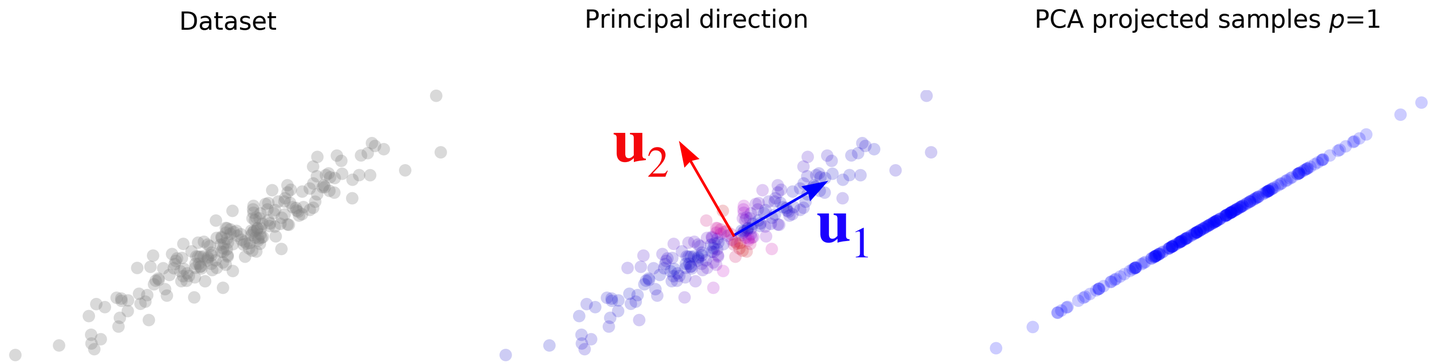
Answers

- **Question 1**
  - `X.shape`
- **Question 2**
  - `df_iris.head()`
  - `df_iris.hist(figsize=(20, 10))`
- **Question 3**
  - `target_column = 'target';`
    `df_iris[target_column].value_counts()`
- **Question 4**
  - `plot2d(X[:,2:4],y,labels[2:4])`
  - ou other coordinates
- **Question 5**
  - `plot2d(X_reduced[:,:2],y); plot2d(X[:,:2],y,labels)`

Part 2 : Principal Component Analysis (PCA)

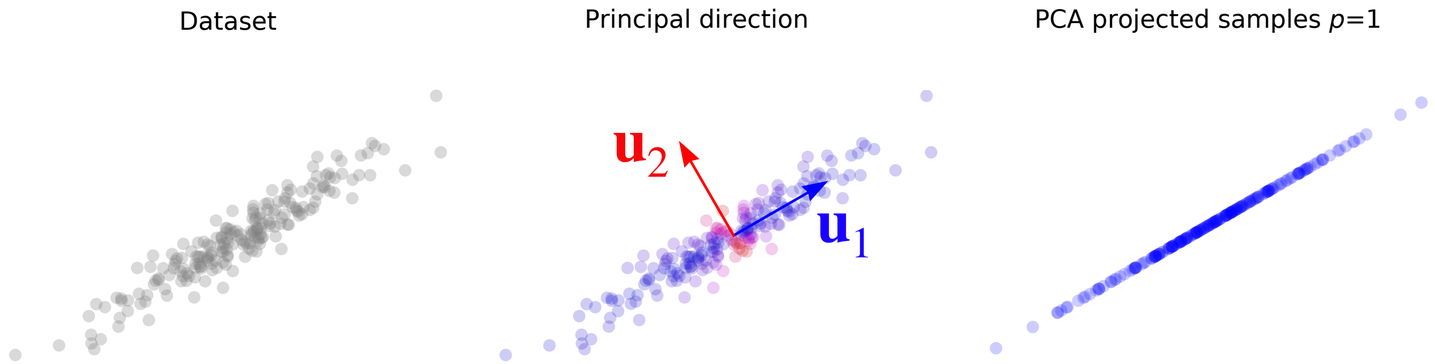
Principal Component Analysis

Principle seen on a point cloud



Principal Component Analysis

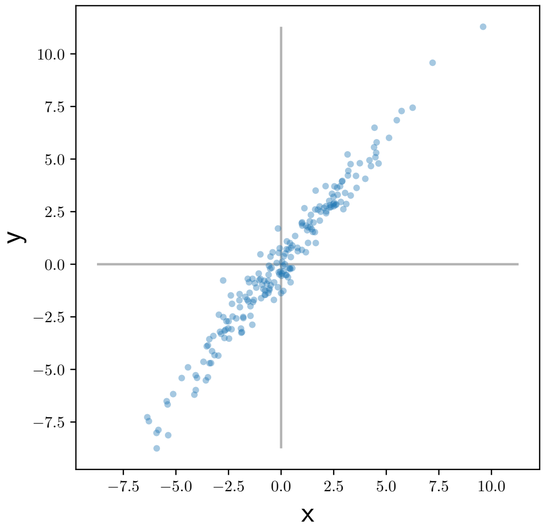
Principle seen on a point cloud

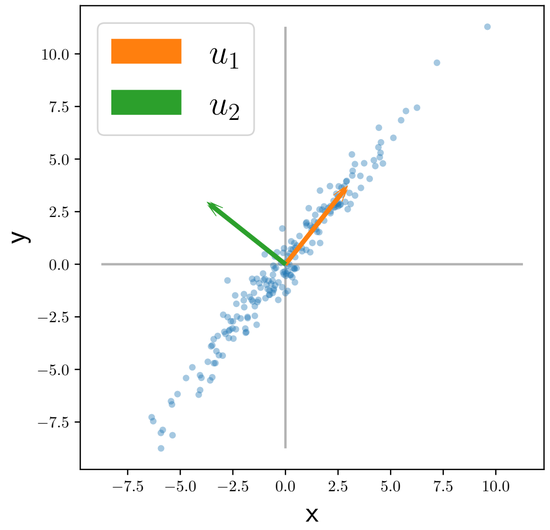


First definition (after centering the data)

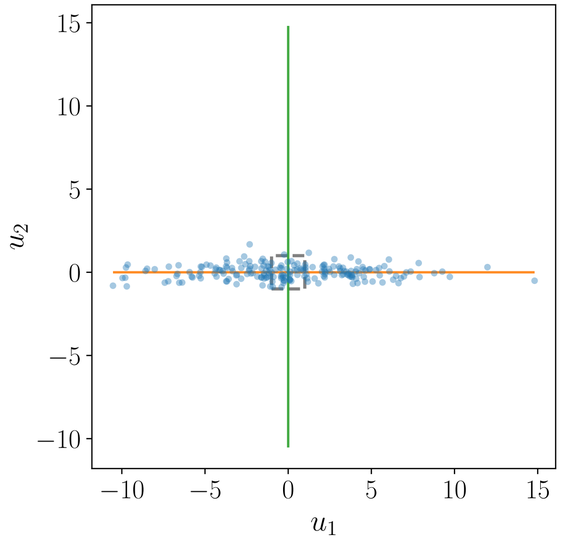
- 1st principal component = vector $\mathbf{u}_1$
  - direction in which the projected data has the greatest variance
- 2nd principal component = vector $\mathbf{u}_2$
  - largest residual variance, after « removing » the contribution of the 1st component
- and so on …

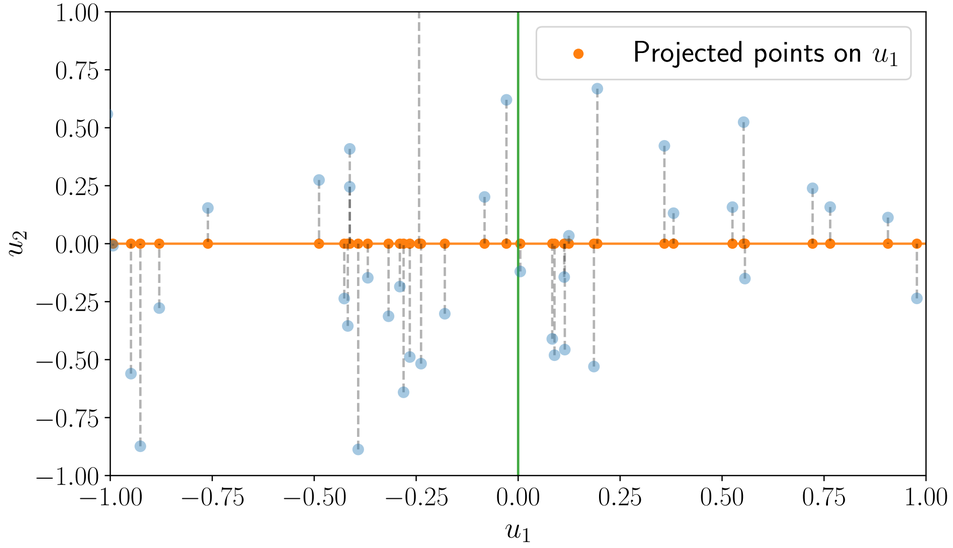
Principal Component Analysis



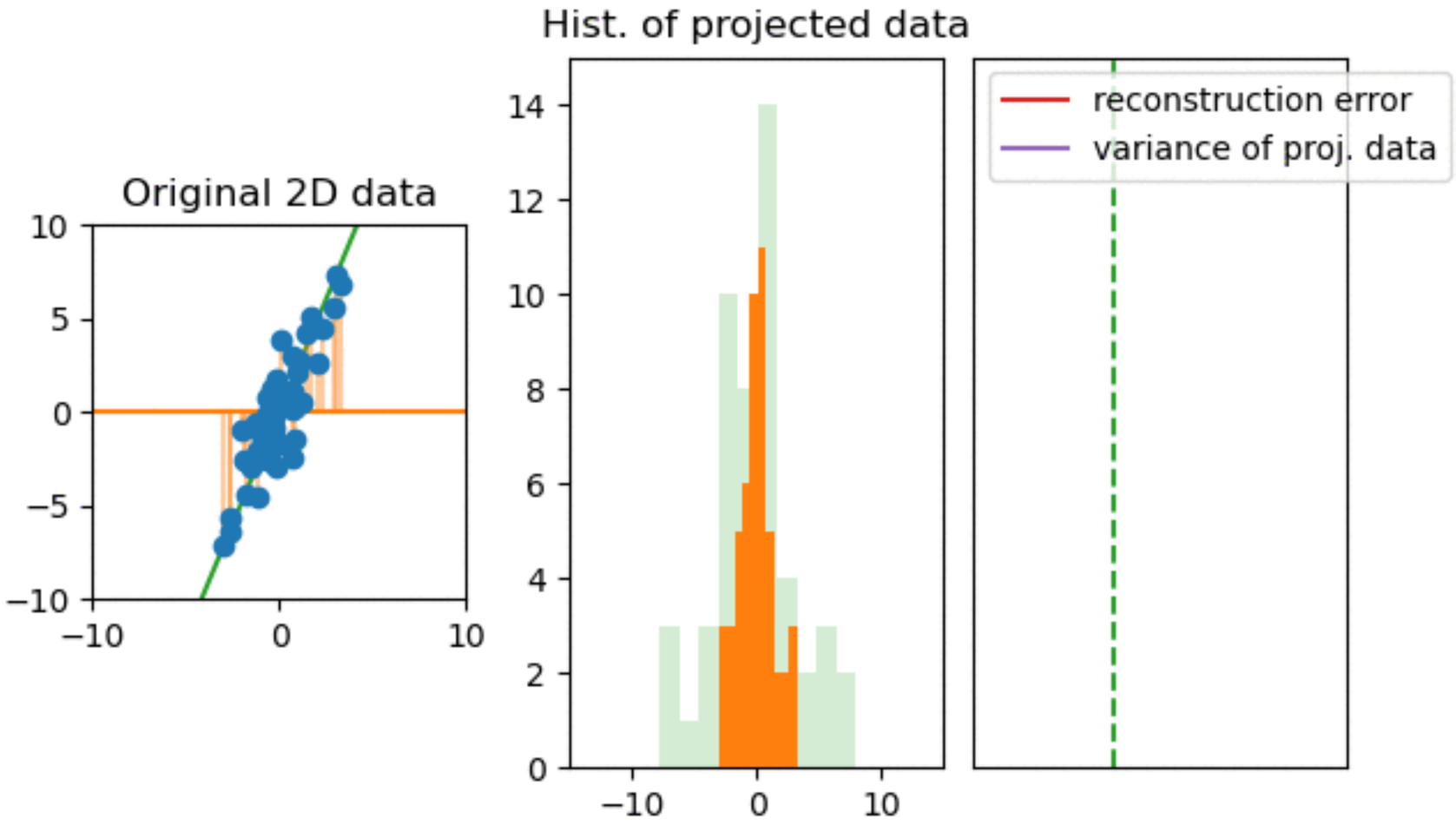


$\tilde{X} = XUU^\top$



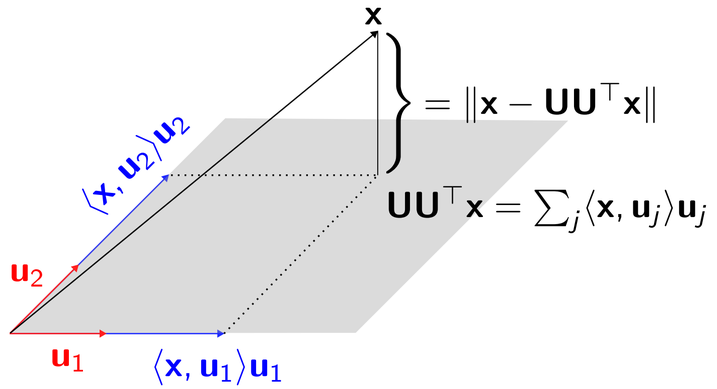


Which projection to choose ?



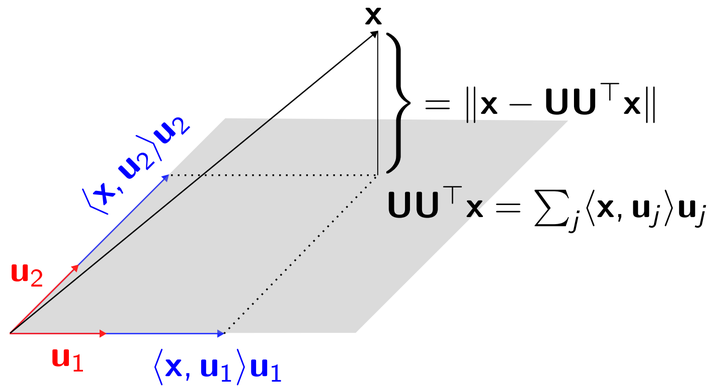
Principal Component Analysis

- **k first principal components $\mathbf{u}_1, \ldots, \mathbf{u}_k$**
  - define a subspace of dim K = image of the matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$
  - The matrix $\mathbf{P}_k = \mathbf{U}\mathbf{U}^\top$ = orthogonal projection onto this subspace
  - dimension reduction via $\mathbf{U}^\top \mathbf{x} \in \mathbb{R}^k$



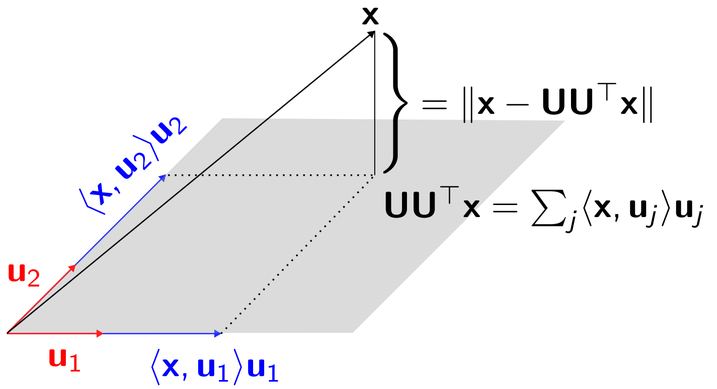
Principal Component Analysis

- **k first principal components $\mathbf{u}_1, \ldots, \mathbf{u}_k$**
  - define a subspace of dim K = image of the matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$
  - The matrix $\mathbf{P}_k = \mathbf{U}\mathbf{U}^\top$ = orthogonal projection onto this subspace
  - dimension reduction via $\mathbf{U}^\top \mathbf{x} \in \mathbb{R}^k$
- **Loss function: reconstruction error**
  - Squared error between a sample and its projection $\|\mathbf{x} - \mathbf{U}\mathbf{U}^\top \mathbf{x}\|_2^2$
  - mean squared error on all samples = risk $R(\mathbf{U}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{U}\mathbf{U}^\top \mathbf{x}_i\|_2^2$



Principal Component Analysis

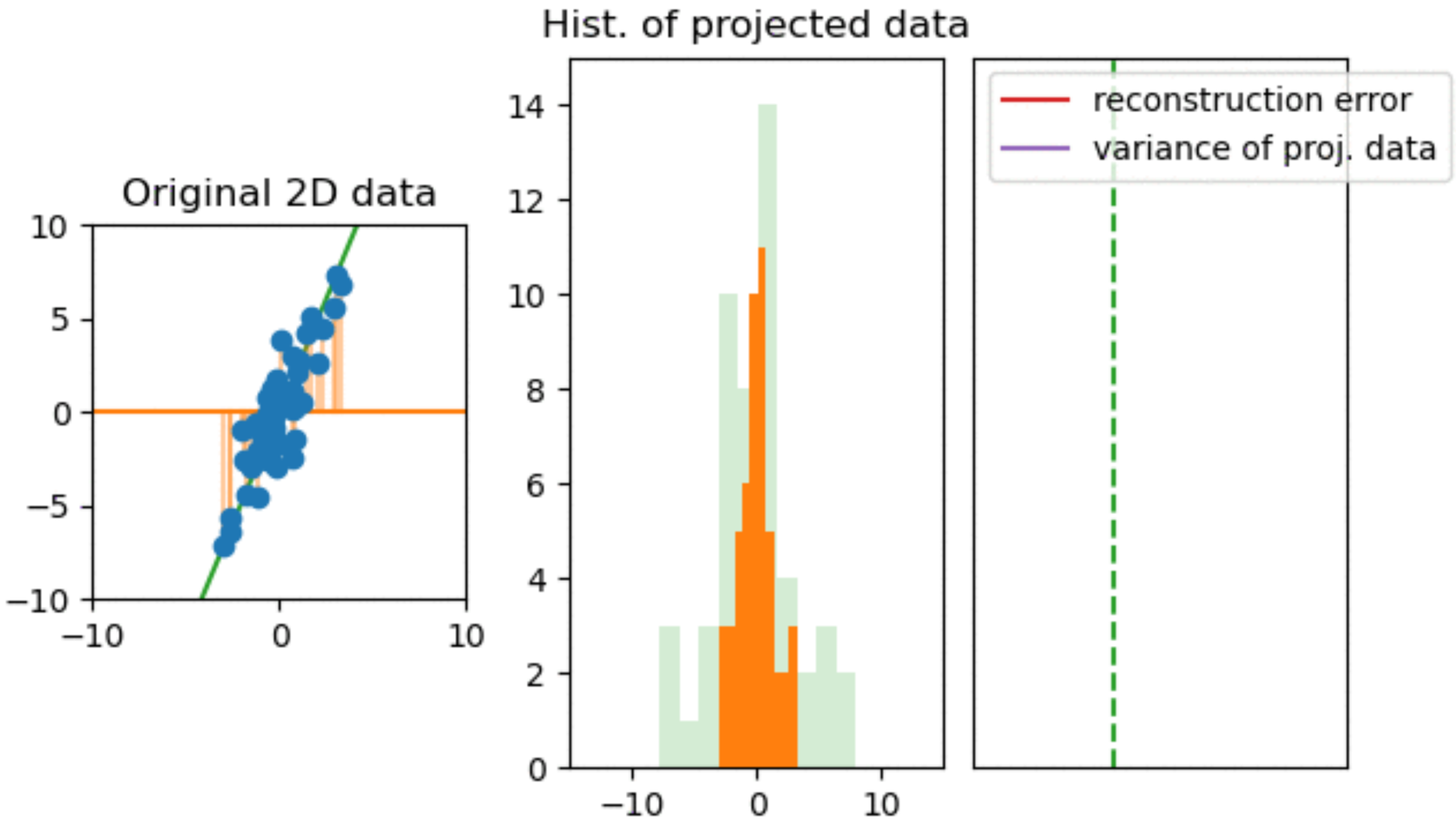
- **k first principal components $\mathbf{u}_1, \ldots, \mathbf{u}_k$**
  - define a subspace of dim K = image of the matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$
  - The matrix $\mathbf{P}_k = \mathbf{U}\mathbf{U}^\top$ = orthogonal projection onto this subspace
  - dimension reduction via $\mathbf{U}^\top \mathbf{x} \in \mathbb{R}^k$



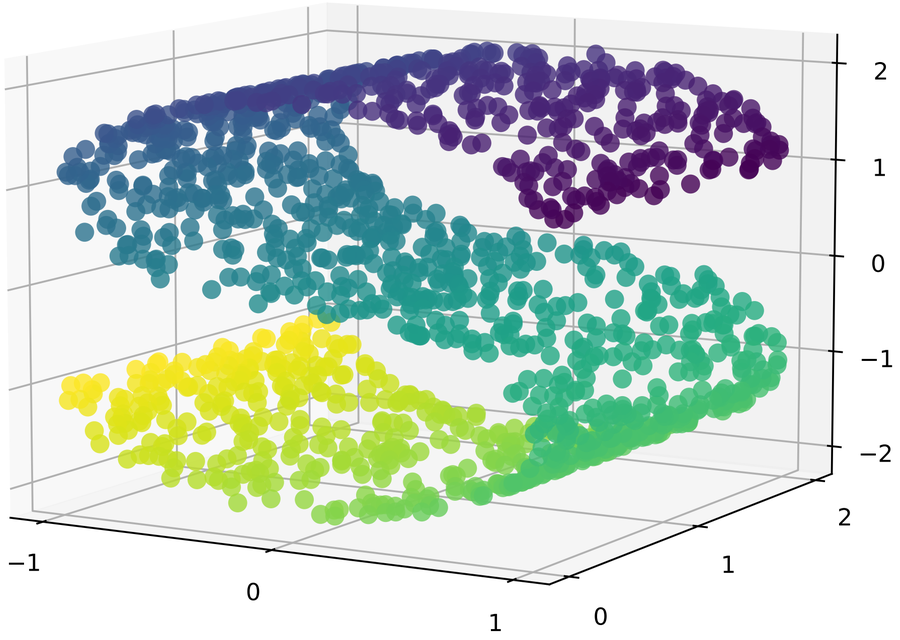
- **Loss function: reconstruction error**
  - Squared error between a sample and its projection $\|\mathbf{x} - \mathbf{U}\mathbf{U}^\top \mathbf{x}\|_2^2$
  - mean squared error on all samples = risk $R(\mathbf{U}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{U}\mathbf{U}^\top \mathbf{x}_i\|_2^2$

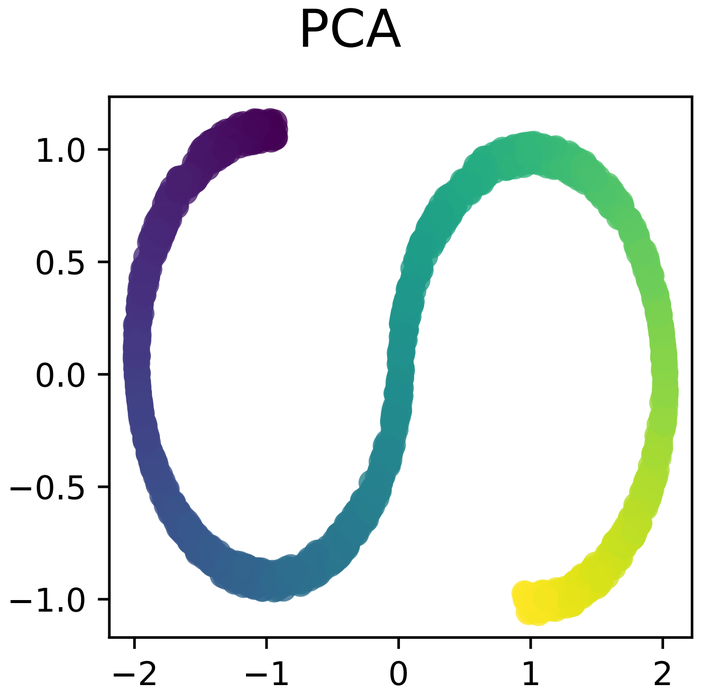
- **How to compute this ?**
  - `sklearn.decomposition.PCA`
  - Data centering, covariance computation, eigenvector decomposition
  - « Without optimisation » but complexity of $(n + k) \times d^2$, beware of big dimension !

Principal Component Analysis



Examples

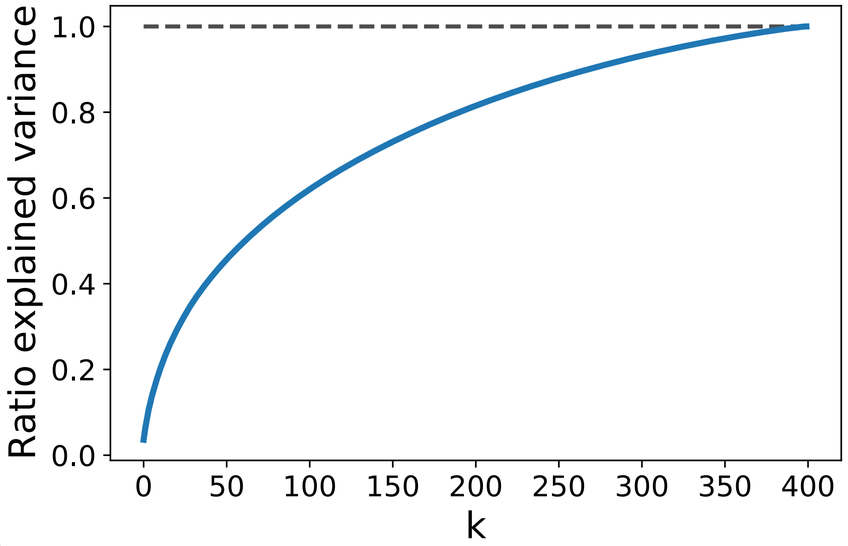




Choice of K

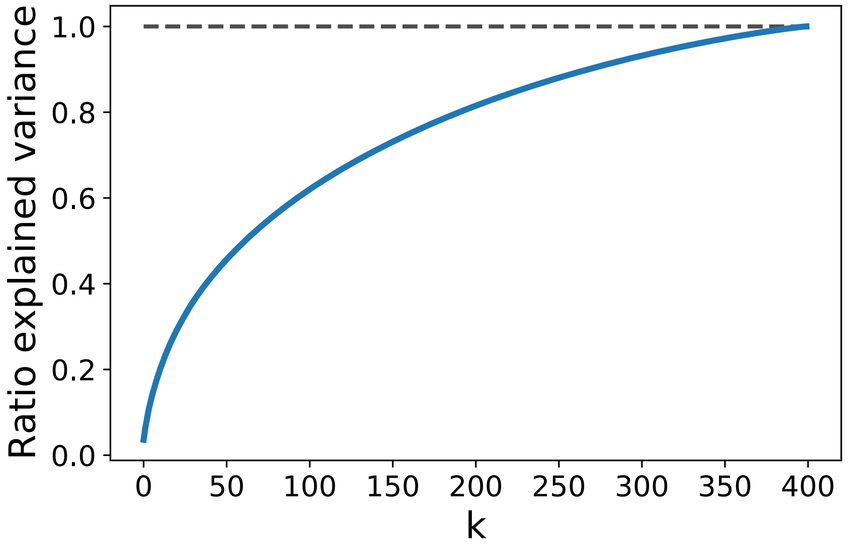
- k=2 or 3 for visualization

Choice of K



- k=2 or 3 for visualization
- Given k,
  - measure % de explained variance $\frac{\sum_{i=1}^{n} \|\mathbf{U}^\top \mathbf{x}_i\|_2^2}{\sum_{i=1}^{n} \|\mathbf{x}_i\|_2^2}$
  - choose k to get a %

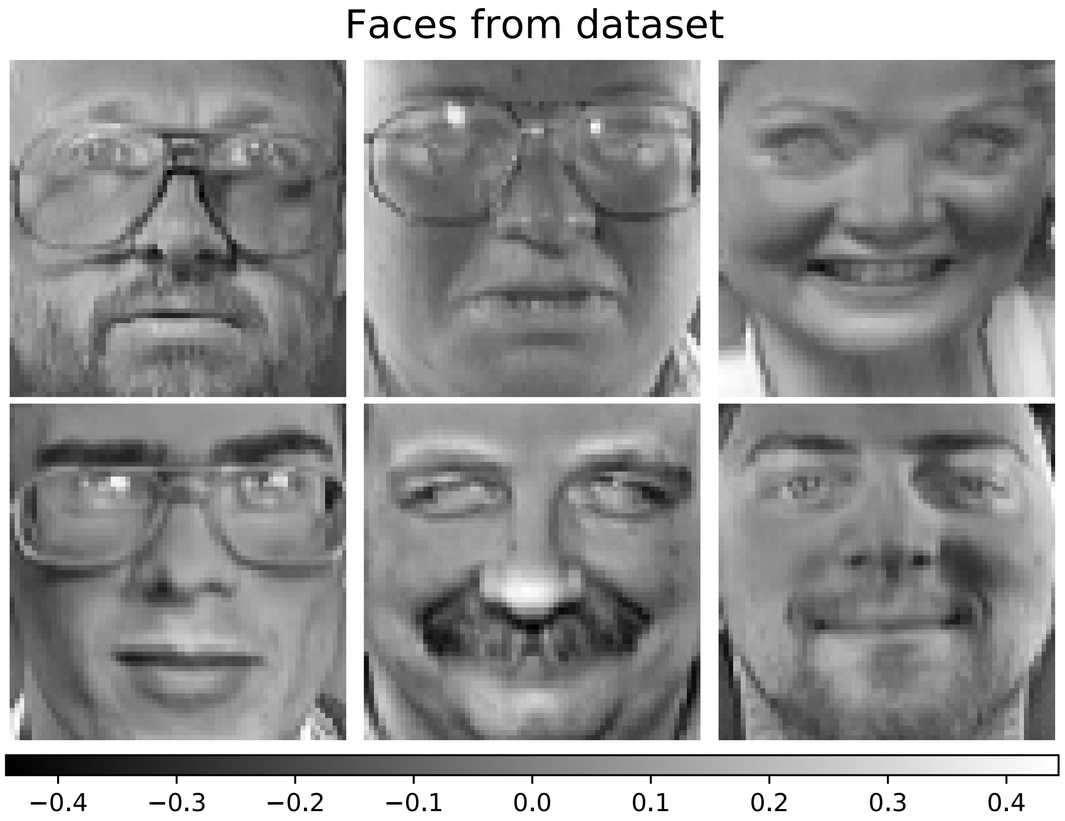
Choice of K



- k=2 or 3 for visualization
- Given k,
  - measure % de explained variance $\frac{\sum_{i=1}^{n} \|\mathbf{x}_i\|_2^2}{\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{U}\mathbf{U}^\top \mathbf{x}_i\|_2^2}$
  - choose k to get a %
- Insert in a complete pipeline and select hyper-parameter K

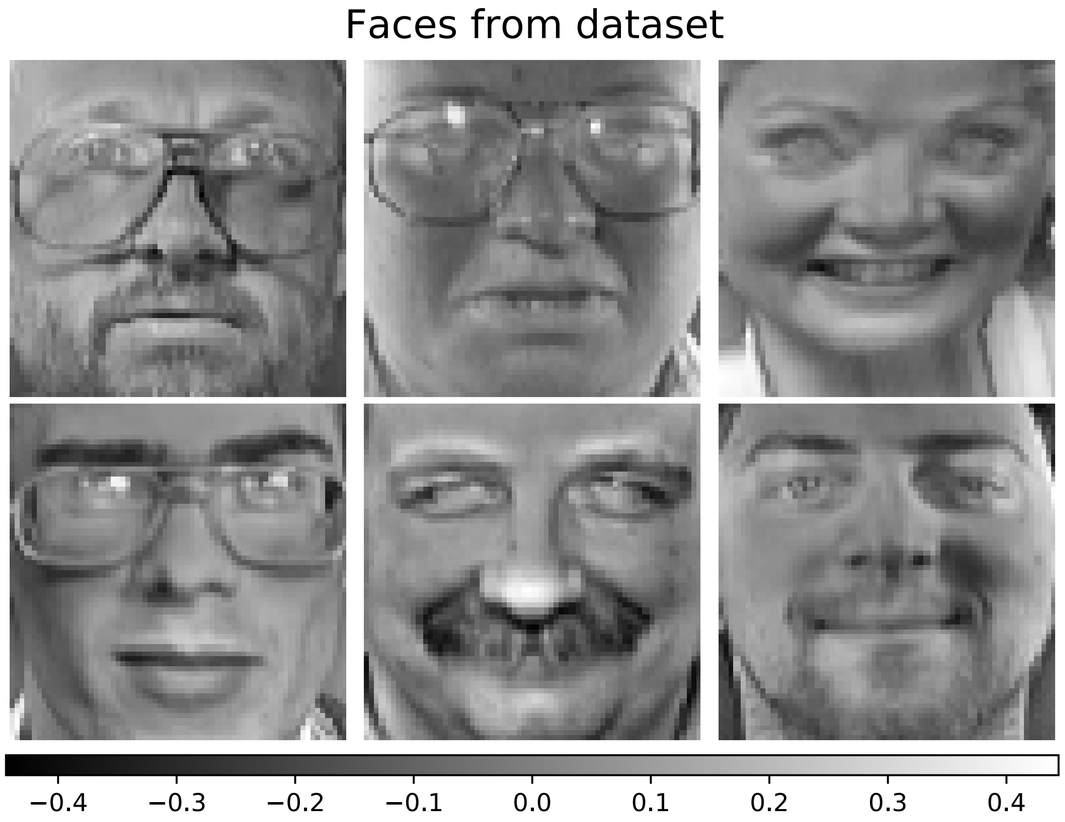
Examples

- Faces dataset
- Each image is $\mathbf{x}_i \in \mathbb{R}^{4096}$, $n = 400$ images

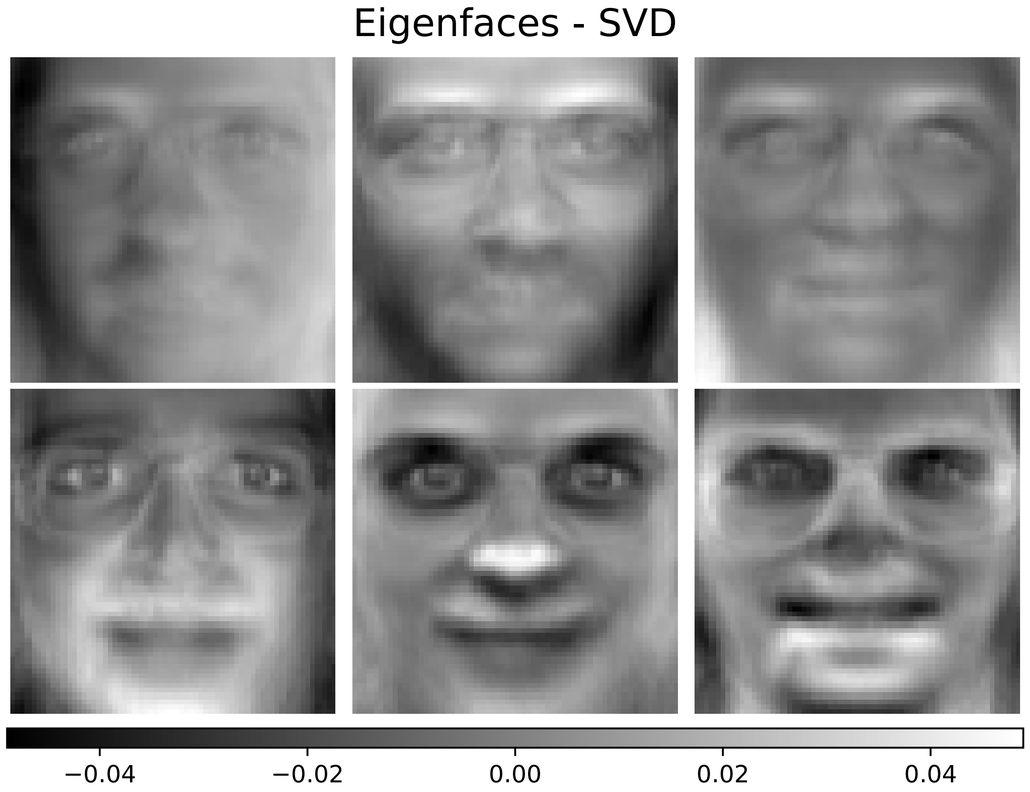


Examples

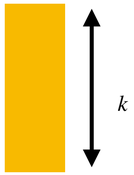
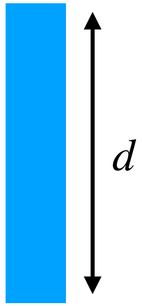
- Faces dataset
- Each image is $\mathbf{x}_i \in \mathbb{R}^{4096}$, $n = 400$ images



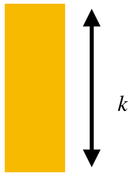
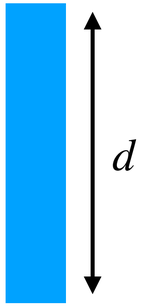
- $k = 6$ principal components



Dimension reduction - main approaches

- **Linear:** $\hat{y} = f(\mathbf{x}) = \mathbf{A}\mathbf{x} \in \mathbb{R}^k$  **from** $\mathbf{x} \in \mathbb{R}^d$ 

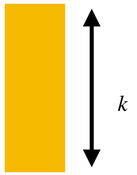
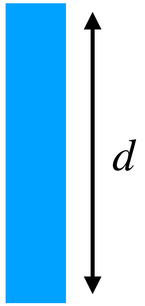
Dimension reduction - main approaches

- **Linear:** $\hat{y} = f(\mathbf{x}) = \mathbf{A}\mathbf{x} \in \mathbb{R}^k$  **from** $\mathbf{x} \in \mathbb{R}^d$ 

- ***Features selection***
  - handcrafted after visualization
  - `sklearn.feature_selection`
  - ***see also : LASSO, random forests***



Dimension reduction - main approaches

- **Linear:** $\hat{y} = f(\mathbf{x}) = \mathbf{A}\mathbf{x} \in \mathbb{R}^k$  **from** $\mathbf{x} \in \mathbb{R}^d$ 

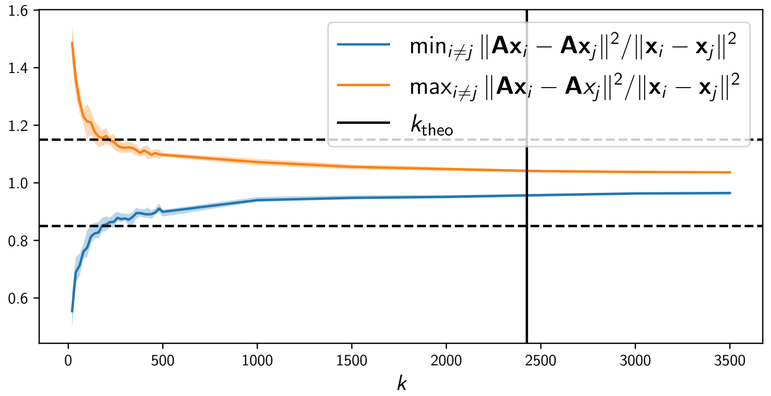
- ***Features selection***
  - handcrafted after visualization
  - `sklearn.feature_selection`
  - *see also : LASSO, random forests*



- ***Principal component analysis (ACP)***
  - `sklearn.decomposition.PCA`



- ***Random projections: random matrix*** $\mathbf{A}$
  - distances are approx. preserved if $k \propto \log n$
  - beware of the constant (can be huge)
  - typically useful for $d \gtrsim 50000$

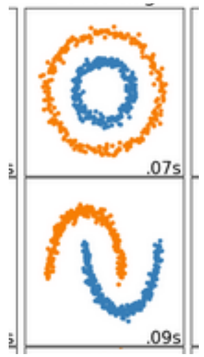


Dimension reduction - main approaches

- **But also:**
  - sparse PCA : sparsity *on* $\mathbf{D}$ *(few non-zero coordinates)*
  - sparse dictionary learning : sparsity on the projected samples y_i
  - independent component analysis (ICA) : statistical independence between the coordinates y_i
  - non-negative matrix factorization : positivité de $\mathbf{D}$ et des y_i (aide interprétabilité)

Dimension reduction - main approaches

- **But also:**
  - sparse PCA : sparsity *on* $\mathbf{D}$ *(few non-zero coordinates)*
  - sparse dictionary learning : sparsity on the projected samples y_i
  - independent component analysis (ICA) : statistical independence between the coordinates y_i
  - non-negative matrix factorization : positivité de $\mathbf{D}$ et des y_i (aide interprétabilité)
- **and non linear approaches:**
  - ***Principle: taking into account the « geometry of the data » by replacing the Euclidean distance.*** $\|x - x'\|_2$
  - **Kernel** methods and neighborhood graphs
    - Kernel PCA
    - Multidimensional scaling MDS
    - t-SNE



- `sklearn.decomposition`
- `sklearn.manifold`

t-Stochastic Neighbor Embedding

Heuristic principle of SNE

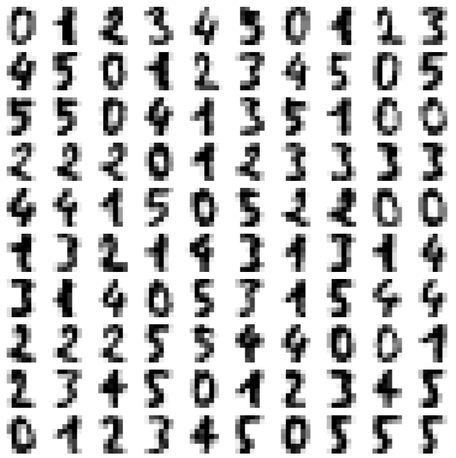
- construct conditional probabilities from $x_i \in \mathbb{R}^d$
  - $P_{j|i} = \frac{e^{-\|x_i - x_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|_2^2 / 2\sigma_i^2}}; P_{i|i} = 0$
- look for $y_i \in \mathbb{R}^k$ such that $Q_{j|i}$ are « close » to $P_{j|i}$
  - $Q_{j|i} = \frac{e^{-\|y_i - y_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|y_i - y_k\|_2^2 / 2\sigma_i^2}}; Q_{i|i} = 0$
- **t-SNE = different variant for $Q_{j|i}$**

t-Stochastic Neighbor Embedding

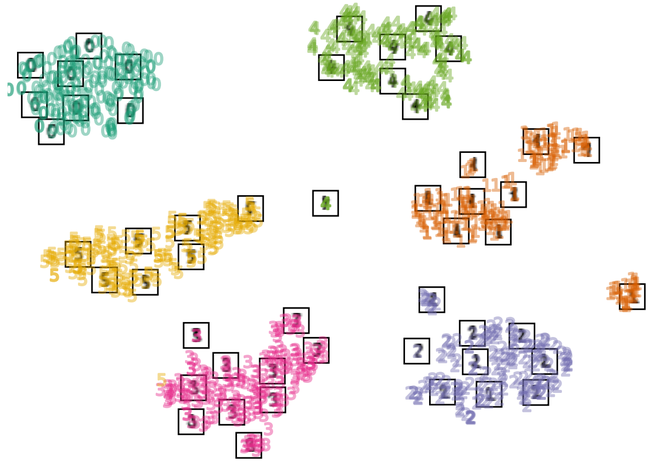
Heuristic principle of SNE

- construct conditional probabilities from $x_i \in \mathbb{R}^d$
  - $$P_{j|i} = \frac{e^{-\|x_i - x_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|_2^2 / 2\sigma_i^2}}; P_{i|i} = 0$$
- look for $y_i \in \mathbb{R}^k$ such that $Q_{j|i}$ are « close » to $P_{j|i}$
  - $$Q_{j|i} = \frac{e^{-\|y_i - y_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|y_i - y_k\|_2^2 / 2\sigma_i^2}}; Q_{i|i} = 0$$
- **t-SNE = different variant for $Q_{j|i}$**

A selection from the 64-dimensional digits dataset



t-SNE embeedding (time 1.324s)



t-Stochastic Neighbor Embedding

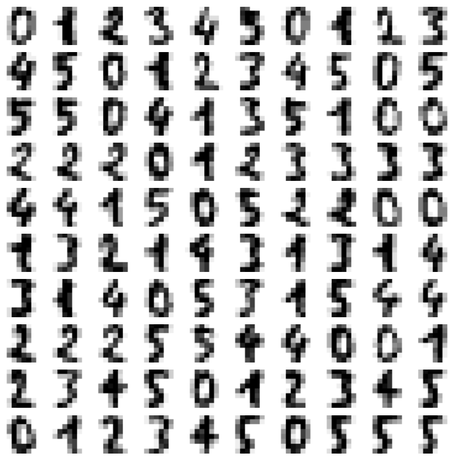
Heuristic principle of SNE

- construct conditional probabilities from $x_i \in \mathbb{R}^d$
  - $$P_{j|i} = \frac{e^{-\|x_i - x_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|_2^2 / 2\sigma_i^2}}; P_{i|i} = 0$$
- look for $y_i \in \mathbb{R}^k$ such that $Q_{j|i}$ are « close » to $P_{j|i}$
  - $$Q_{j|i} = \frac{e^{-\|y_i - y_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|y_i - y_k\|_2^2 / 2\sigma_i^2}}; Q_{i|i} = 0$$
- **t-SNE = different variant for $Q_{j|i}$**

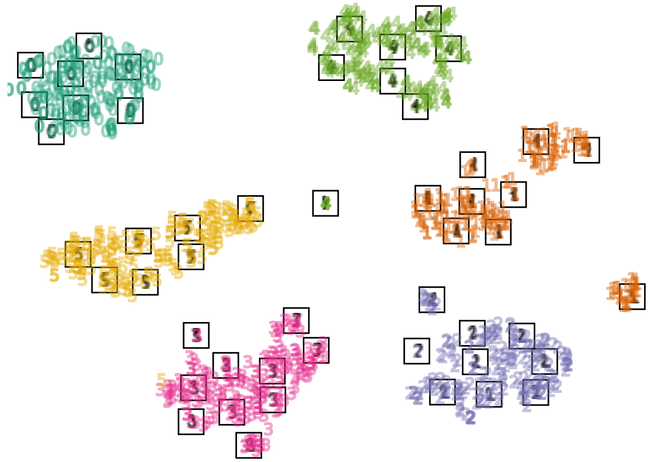
In practice

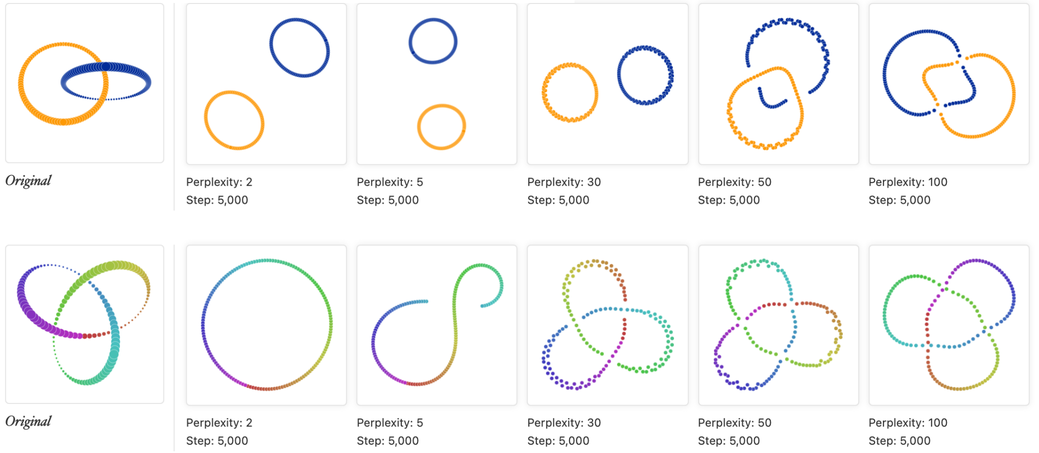
- very powerful method
- can be slow / unstable
  - depending of the parameters
- few examples :

A selection from the 64-dimensional digits dataset



t-SNE embeedding (time 1.324s)





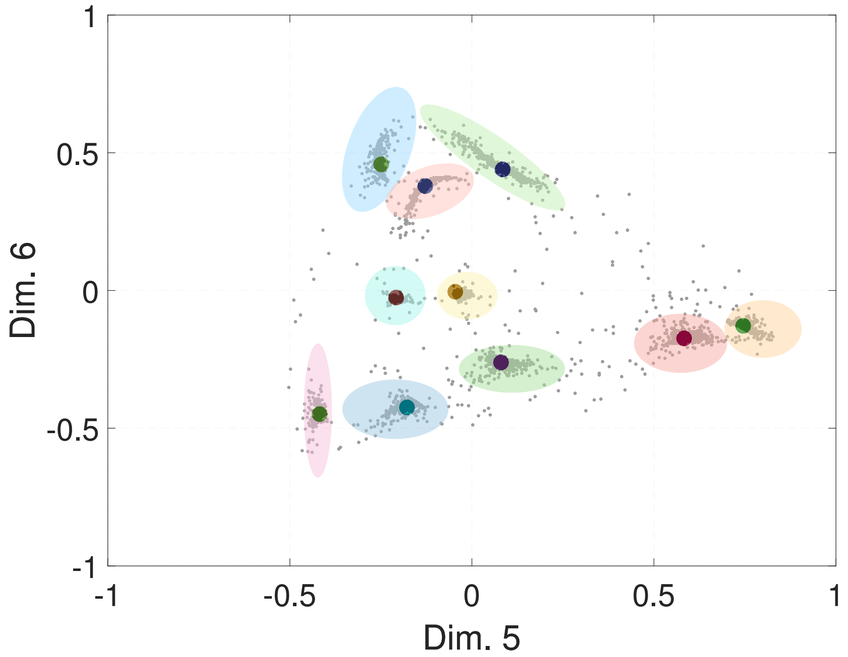
Representation depending on t-SNE parameters

Part 3 : Clustering

Clustering

Example

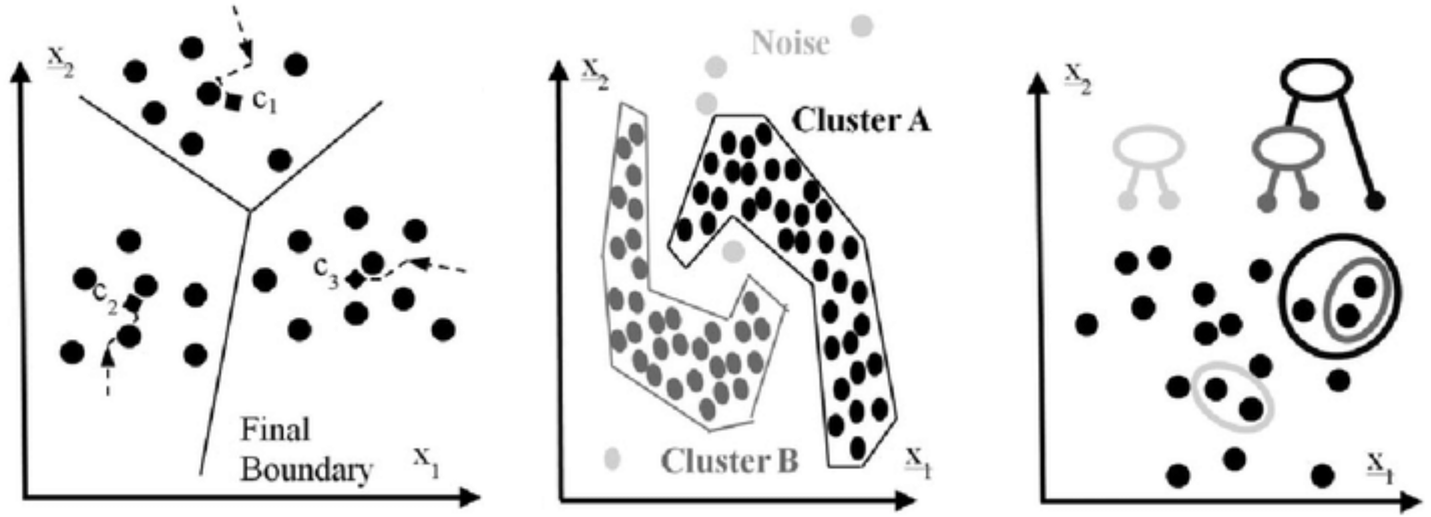
- Données $x_i, 1 \leq i \leq n$
- We « see » groups = clusters
- ***But*** = how to find them
  - formally: construct $f : x \mapsto \hat{y} = f(x) \in \{1, \ldots, K\}$



For what ?

- Interpreting groups as classes
- Subsequent use
  - ex = labels for « auto-supervised » learning
  - ex = compression : replace x by a group « representative »
    - reduce the storage cost

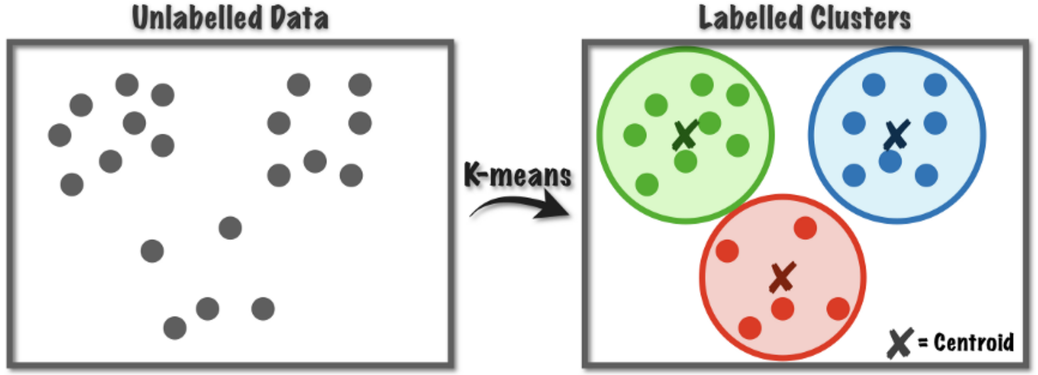
What is a cluster ?



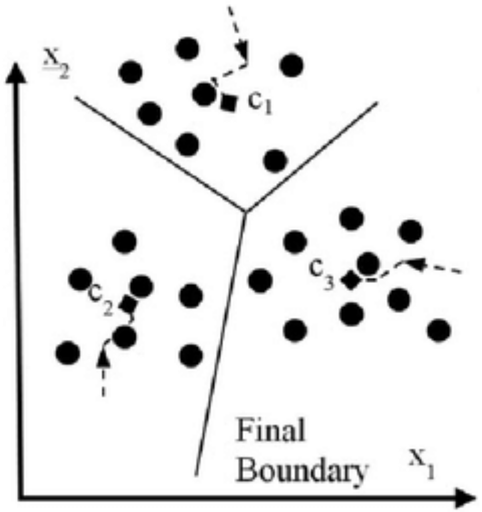
- **As a starting point …**
  - **No unique definition**
  - Need a notion of **similarity / difference**
- **Three principal approaches**
  - Cluster = collection of examples resembling a **reference prototype**
  - Cluster = set of examples that can be **connected through continuity**
  - Cluster = obtained by **merging smaller groups together**

Prototype-based approach

- **Cluster = collection of examples resembling a reference prototype**

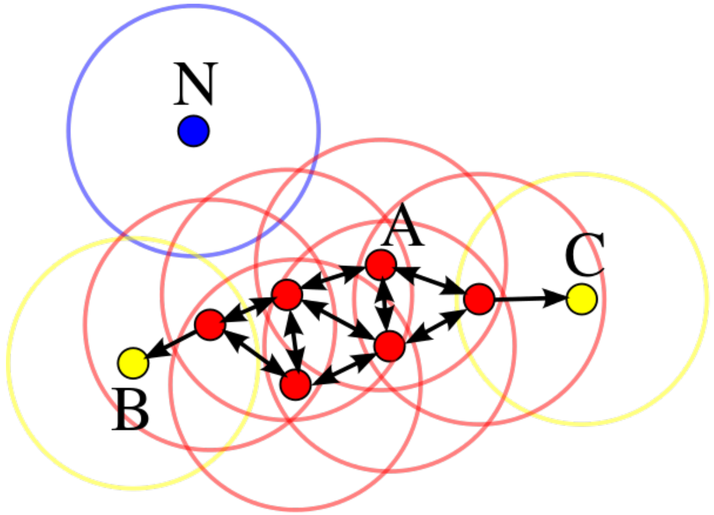


- **Well known approach = *K-means***
  - Alternating principle
    - Selection of prototypes for the current clusters
    - Updating the groups based on these prototypes.
  - The number of clusters K ***must be known in advance***



Continuity-based approach

- **Cluster = set of examples that can be connected through continuity**

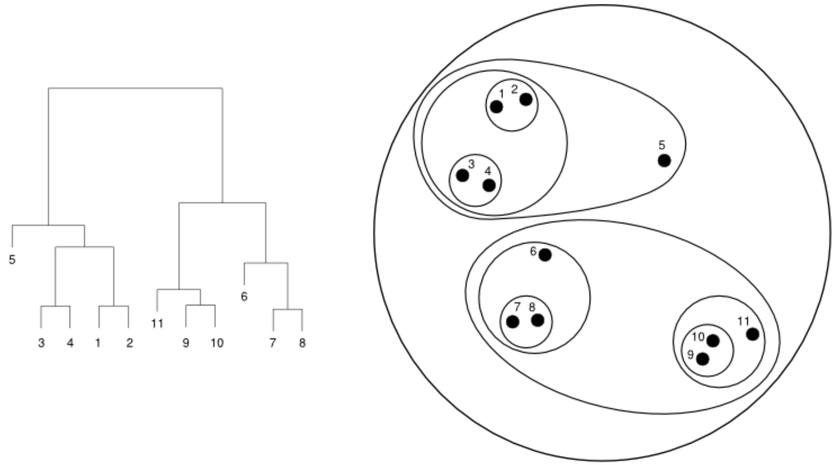


`sklearn.cluster.DBSCAN(eps=0.5,min_samples=5)` .

- **The most classical method: DBSCAN**
  - Principle :
    - Grouping examples by continuity, if possible (based on proximity and density)
    - Some samples can be isolated

Agglomerative approach

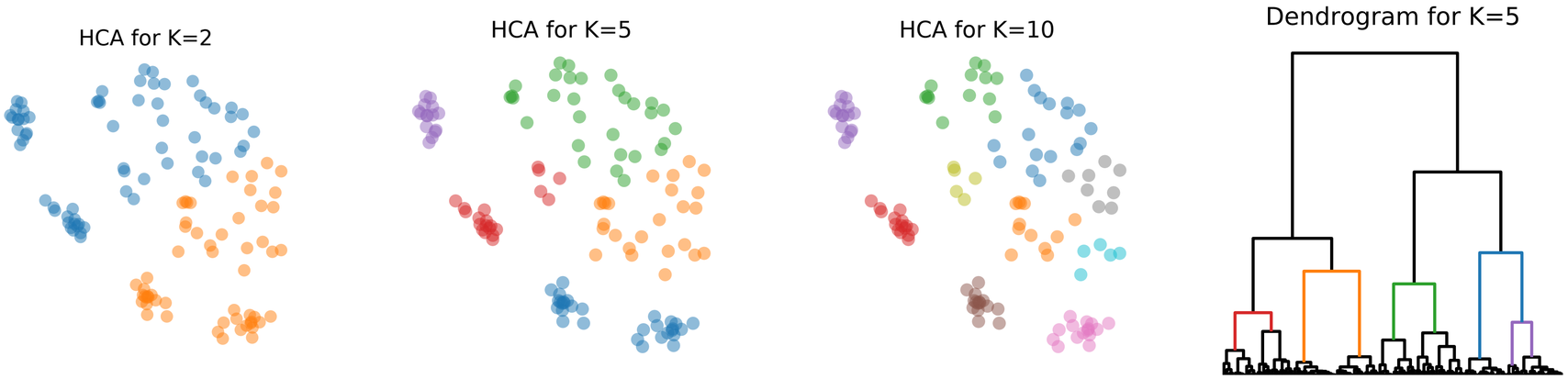
- **Cluster = obtained by merging smaller groups together**



- **Classical method : *HCA* (Hierarchical Clustering Analysis)**
  - **Principle** : hierarchical clustering
    - Utilizing pairwise 'distances' between examples/groups to merge groups and create a tree
    - many variant for the fusion criteria
  - The number of clusters K is determined a posteriori by cutting the obtained tree

Agglomerative approach

Cluster = obtained by merging smaller groups together



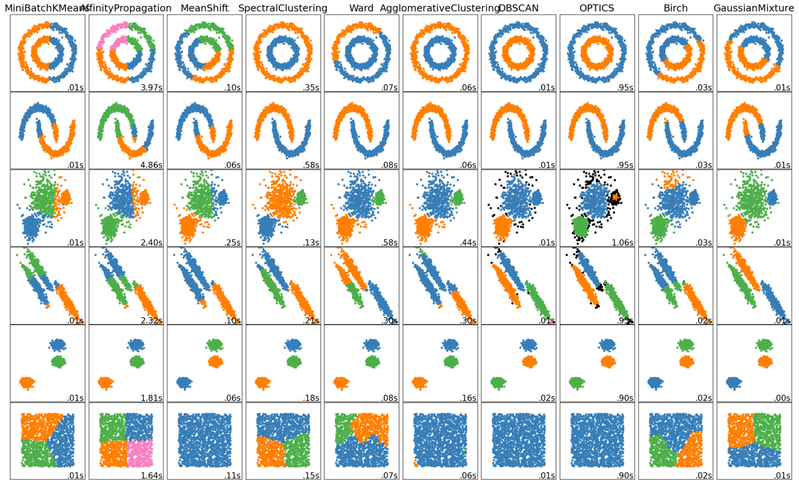
Classical method : *HCA* (Hierarchical Clustering Analysis)

- **Principle** : hierarchical clustering
  - Utilizing pairwise 'distances' between examples/groups to merge groups and create a tree
  - many variant for the fusion criteria

- The number of clusters K is determined a posteriori by cutting the obtained tree

Choice of the method and number of clusters

https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html



To remember

- No method is systematically better than another
- No explicit criteria sometimes

Choice of the number of clusters K ?

- a priori
- heuristics (BIC criteria…)
- based on the intended future use …

if labels/classes:

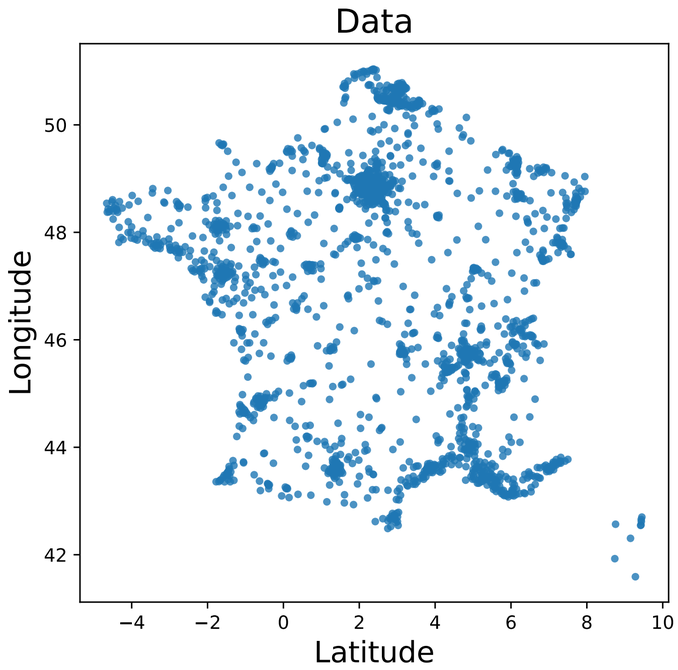
Rand Index (RI) = # pairwise correct pred. / # possible pairs

if not:

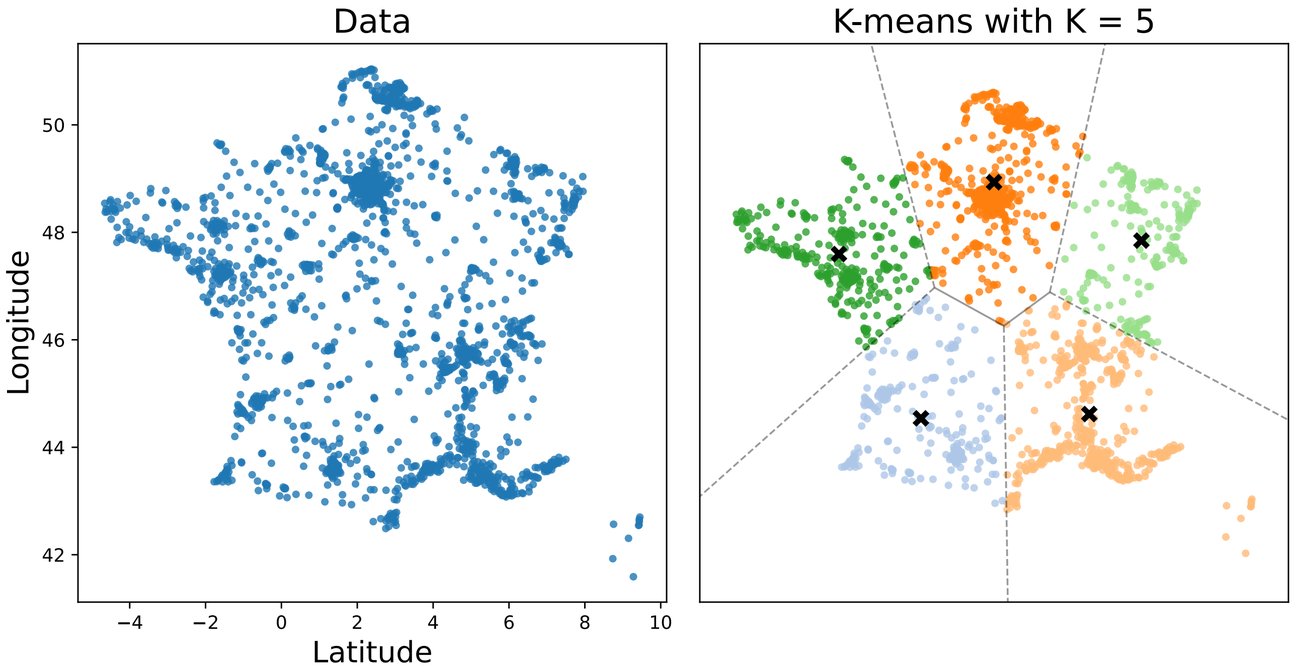
Silhouette score = average dist. to points within the same group
- average dist. to points in neighboring groups

Part 4 : K-means

K-means

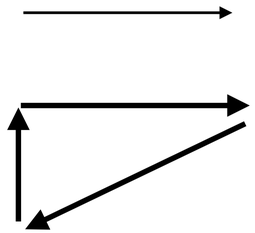


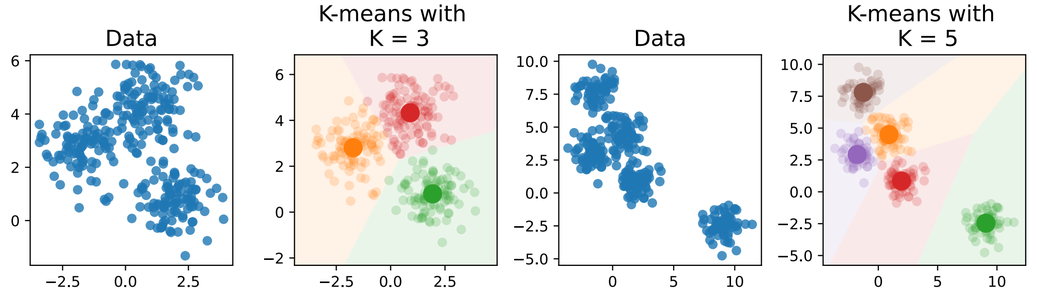
K-means


Data
K-means with K = 5
Longitude
Latitude
50
48
46
44
42
−4
−2
0
2
4
6
8
10

K-means

- **Principle :** clustering into K group
  - 1 cluster → 1 mean = *centroid* c_k
  - K clusters → K centroids

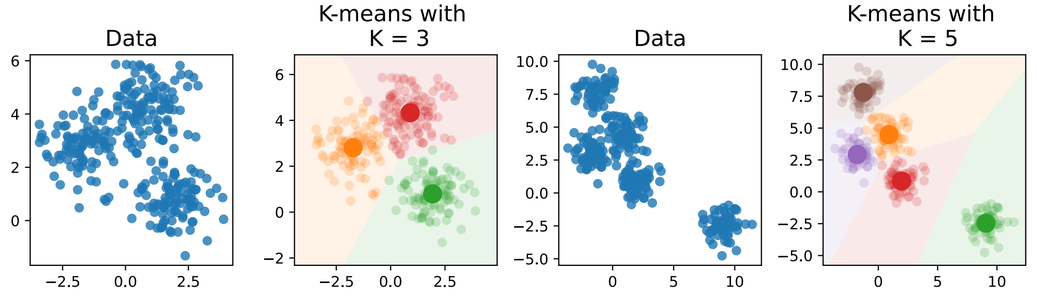




K-means

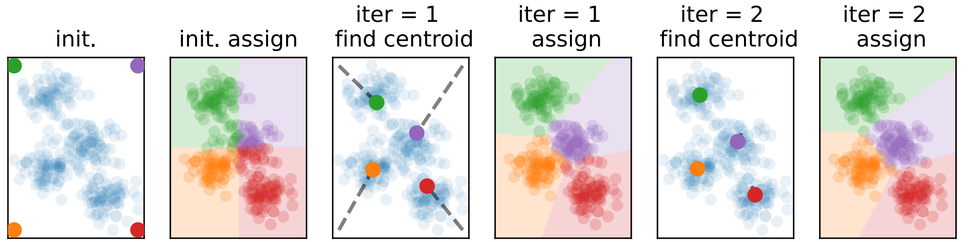
Principle : clustering into K group

- 1 cluster → 1 mean = *centroid* c_k
- K clusters → K centroids



Formulation as an optimization problem

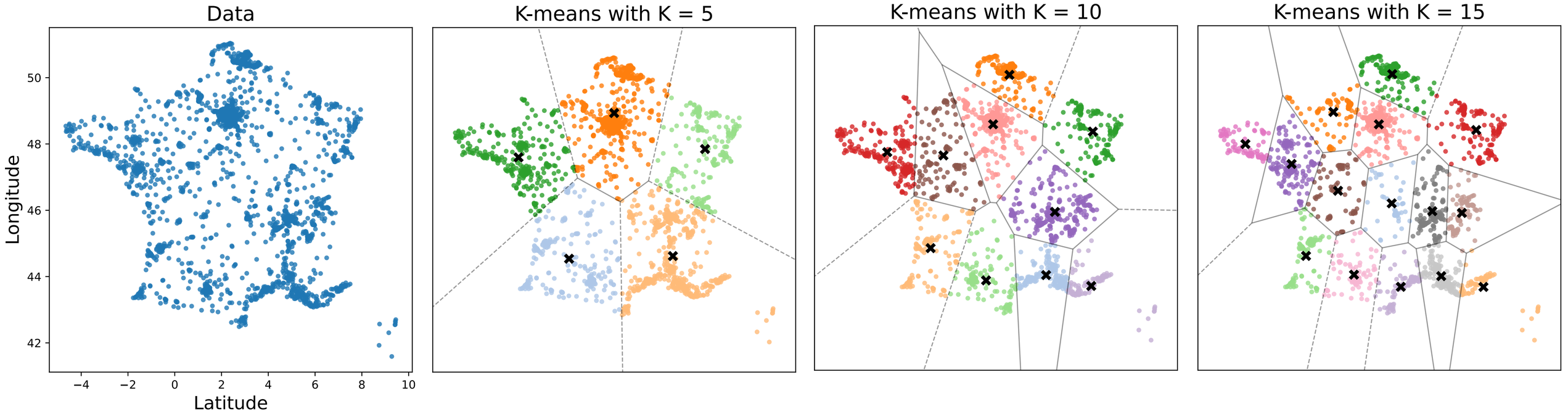
- **Parametrisation** = position of the K centroids, $\theta = (c_1, \ldots, c_K)$
- **Loss function** for one sample $f(x, \theta) := \min_{1 \leq k \leq K} \|x - c_k\|_2^2$
- **Risk to minimize** $R(\theta) := \frac{1}{n} \sum_{i=1}^{n} f(x_i, \theta)$
- On average the dist. between the samples and its closest centroid is minimum.



Lloyd iterative algorithm (from [Steinhaus 1956])

K-means

On average the dist. between the samples and its closest centroid is minimum.



Example on France map

- Notebook

Python code

```python
from sklearn.cluster import KMeans

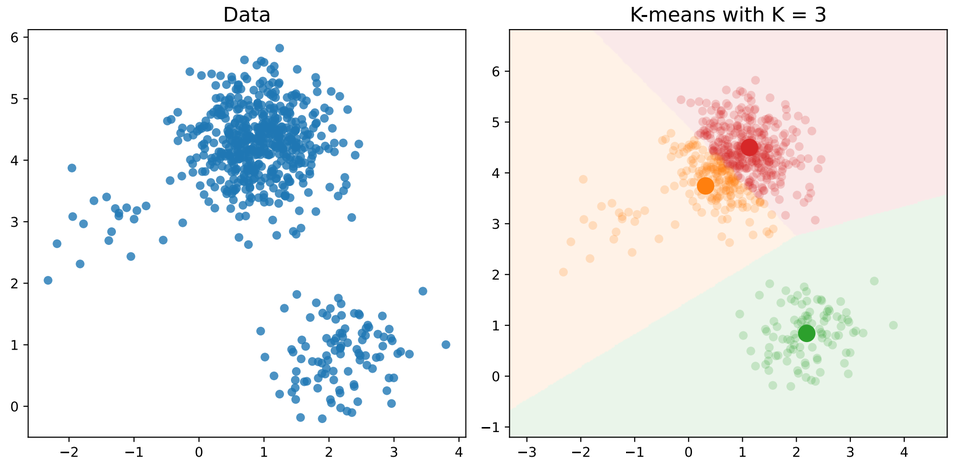
# K-means with K=2
clf = KMeans(2)

# fit the model et predict classes
y = clf.fit_predict(X)

# distance from samples to clusters
dist = clf.transform(X)

# get the centroids
C = clf.cluster_centers_
```

K-means - best practices



Good to know …

- Risk minimization = non convex problem
- ***No guarantee*** *to find the best (global) optimum*
  - the result depend on the initialization of the algorithm

A « good » initialization is crucial !

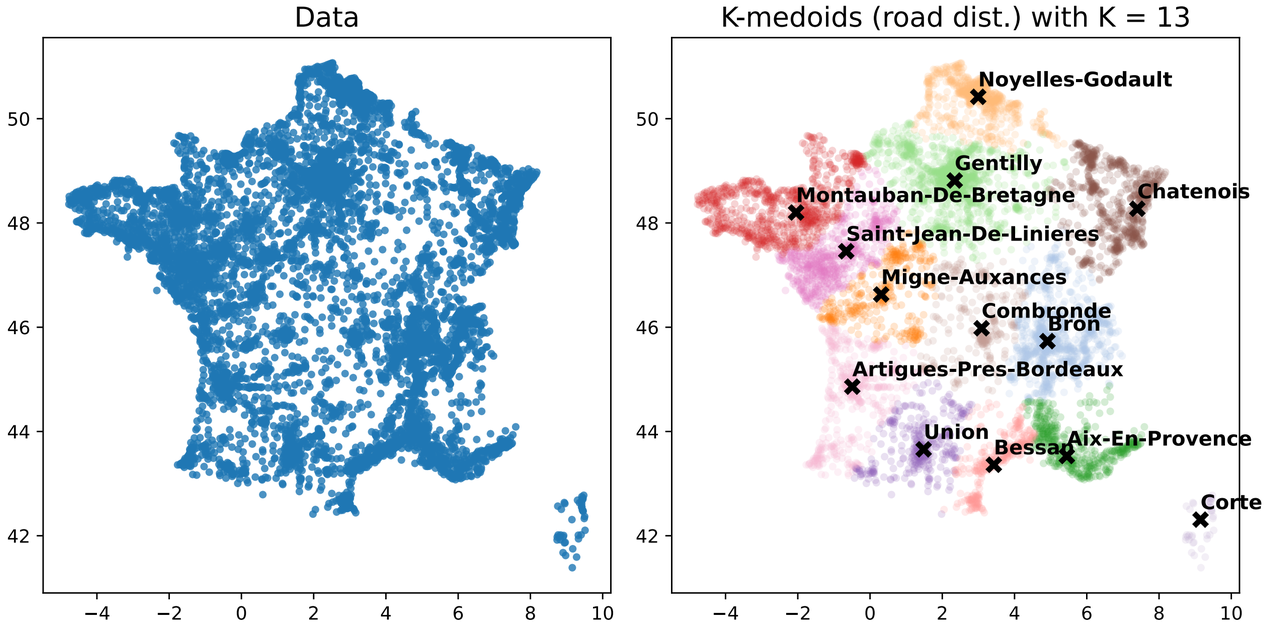
- Good practice
  - Initialize with samples from the dataset x_i
  - K**-means++** (random, spreads out the K initial cluster centers) = *default in scikit-learn*
- But also :
  - ***Repeat multiple draws and keep the result that minimizes the risk!***

Complexity : of the order of $n \times d \times K \times T$

- with T number of step
- proportional to d because of the distance calculus $\|x - c\|_2^2 = \sum_{j=1}^{d} (x(j) - c(j))^2$

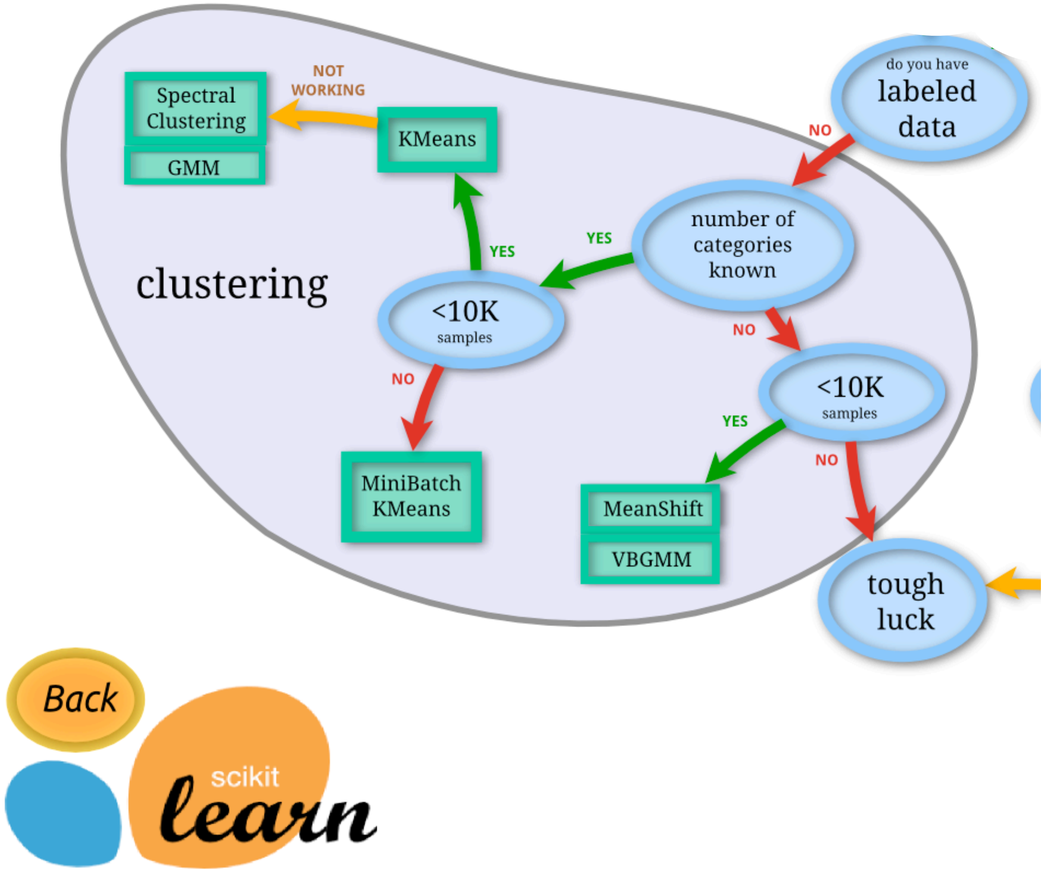
K-means : variants

- **If lot of samples n**
  - Minibatch $K-means$
- **If outliers**
  - K-median : « centroids » are …
    - Less influenced by outliers than the K-means ones.
    - … but more challenging to calculate for each group
- **But also**
  - K-medoids
    - centroids are ***amoung the data samples***
      - *more costly*



The guide - scikit-learn

https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html



Clustering - in practice

- First step « know your data »
  - Domain expertise
  - vizualisation
- Standard approaches :
  - K-means si the number of clusters is known, DBScan otherwise
  - K-means works well on data that are
    - « well clusterable », and isotropic
  - DBScan or spectral clustering
    - if the groups are nonlinearly separated or there are outliers

Recap

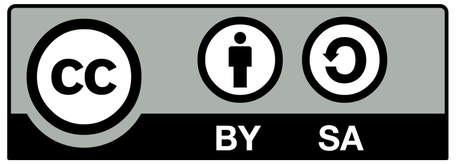
Unsupervised learning

- **Why ?**
  - **visualize** the data, understand the structure
  - **compress**, reduce storage
  - **prepare data** for subsequent use
- **How ?**
  - **dimension reduction** (feature selection, PCA, or non-linear methods)
  - **clustering** (K-means, DBscan, spectral clustering)

Crédits

- Scikit-learn
- **Slides:** inspired from
  - Rémi Flamary
    - https://remi.flamary.com/cours/intro_ml.html
  - E. Le Pennec and collegues

Licence and Contributors



Creative Commons Attribution-ShareAlike (CC BY-SA 4.0)

- You are free to:
  - **Share:** copy and redistribute the material in any medium or format
  - **Adapt:** remix, transform, and build upon the material for any purpose, even commercially.
- Under the following terms:
  - **Attribution:** You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **ShareAlike:** If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions:** You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Contributors

- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.

118

Voronoi cells

