Introduction to Data Science
Unsupervised Learning

Benoit Gaüzère - Romain Hérault

INSA Rouen Normandie - Laboratoire LITIS

March 4, 2024

What is Machine Learning ? I

Artificial Intelligence

A program or a device that displays or mimics cognitive behaviors attributed to humans.
Examples : **deducing** or **inferring**.

This definition is highly imprecise and human-centric.

Symbolic AI

- **Deducing** new decision rules from known **rules**,

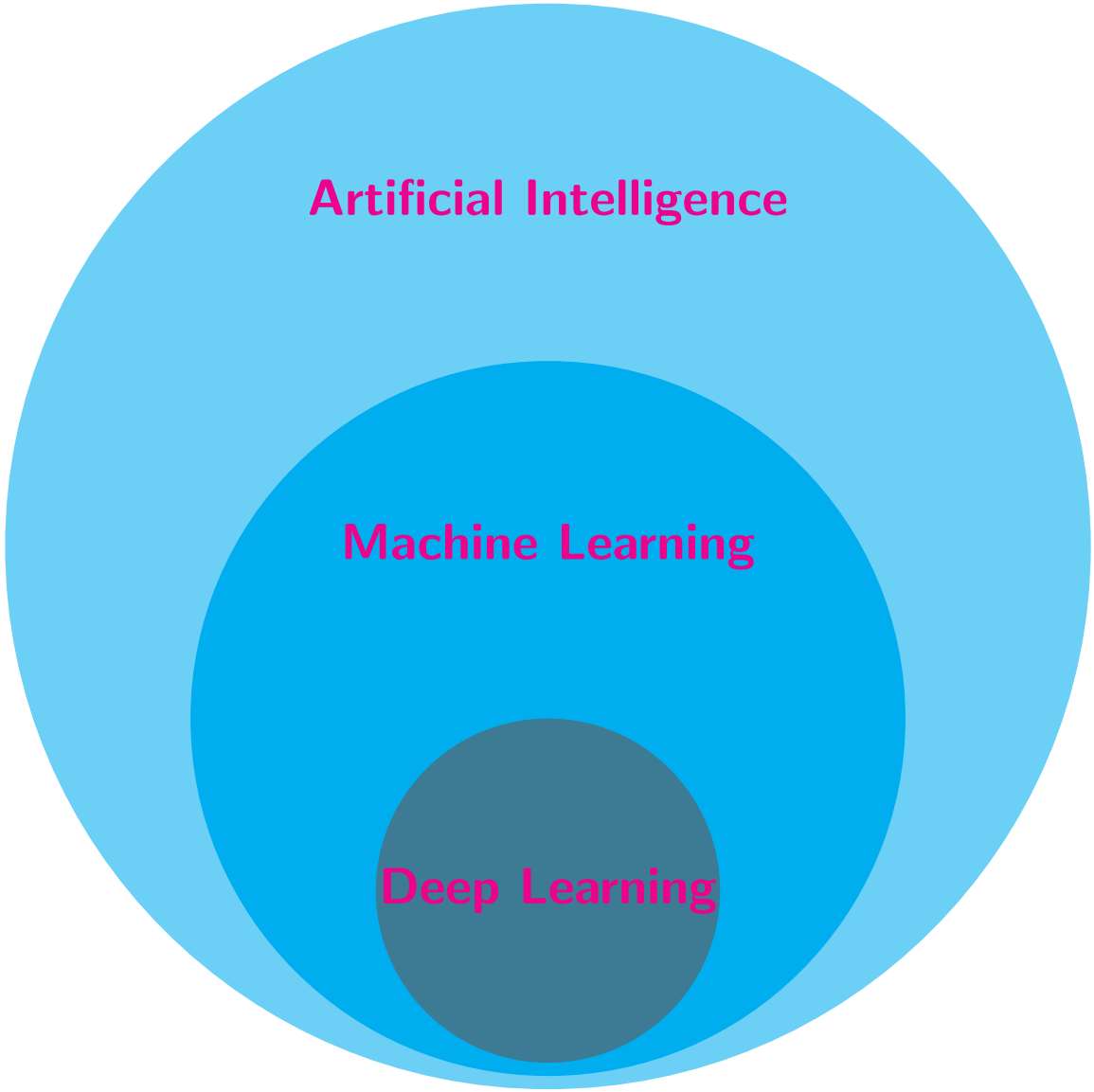
$\Rightarrow$ Discrete / finite domain optimization.

Machine Learning

- **Inferring** new decision rules from **observations**,

$\Rightarrow$ Continuous / mixed optimization.

What is Machine Learning ? II



What is Machine Learning ? III

Different tasks

Supervised Learning predicts the property of an observation considering pairs of (observations, property),

Unsupervised Learning detect patterns or data structure only from observations

Semi-Supervised Learning predicts data property only considering a small set of (observations, property)

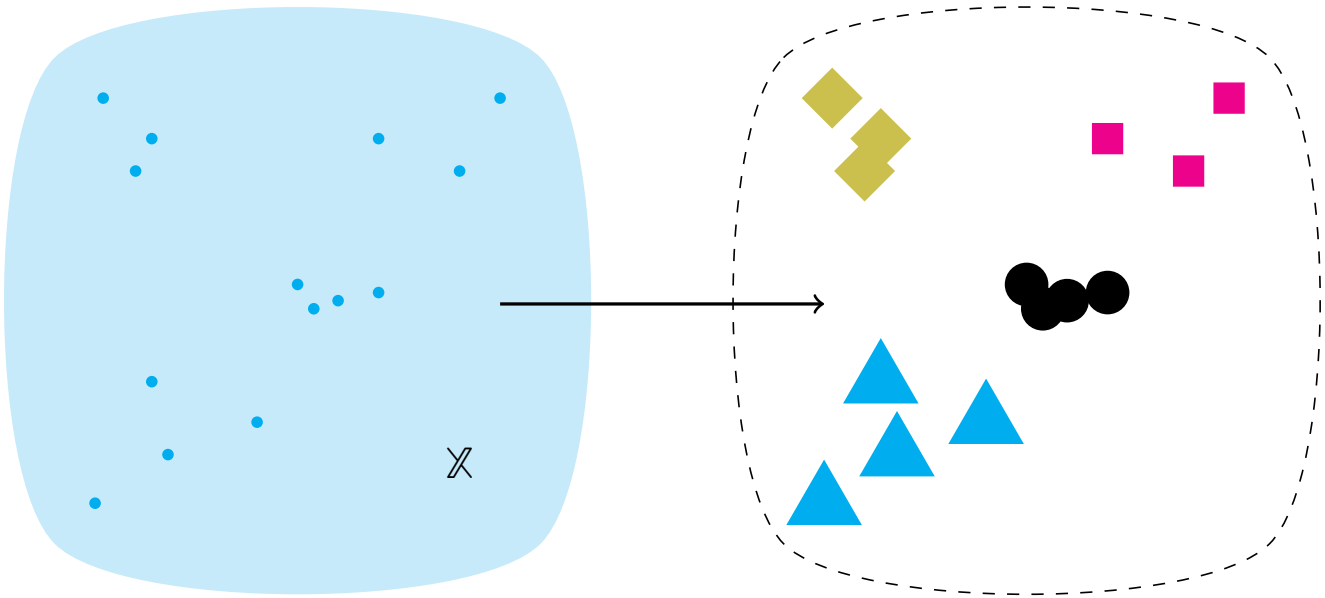
and more : Reinforcement Learning, Active Learning, Self supervised learning, ...

Unsupervised learning I

Only use observations!

Clustering

Detect groups of homogen data

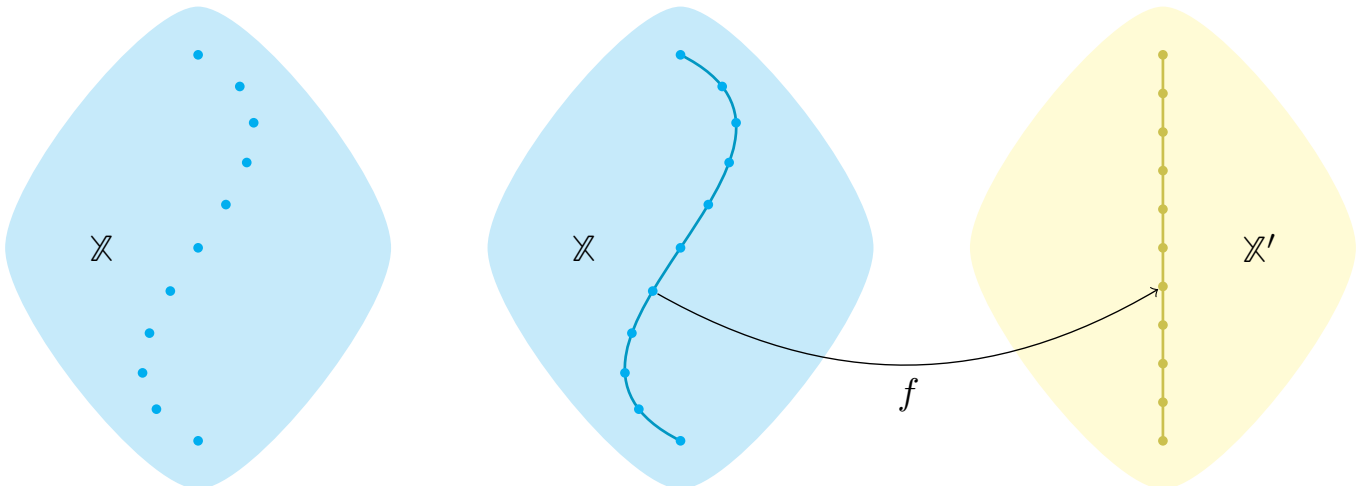


- Methods: Kmeans, Hierarchical clustering, DBSCAN, ...
- Used to segment client profiles, areas of images, ...

Unsupervised learning II

Dimensionality Reduction

Discover low-dimensional representation that captures most of the information in the data.



- Methods: PCA, t-SNE, UMAP
- Data compression, Noise removing, Visualization, Latent space

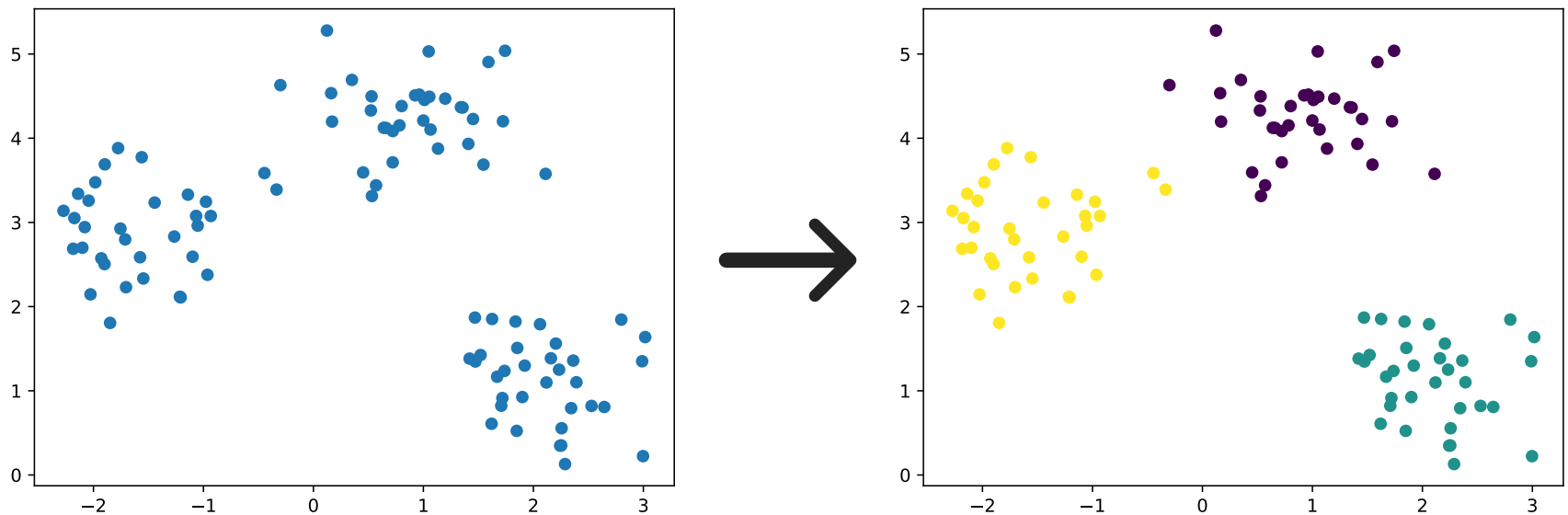
and others : novelty detection, data generation, ...

Unsupervised Learning
Clustering

Clustering

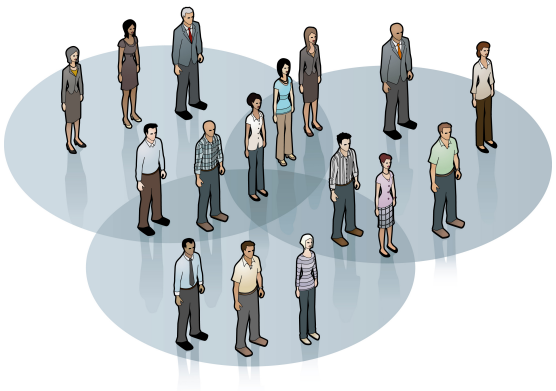
Clustering in few lines

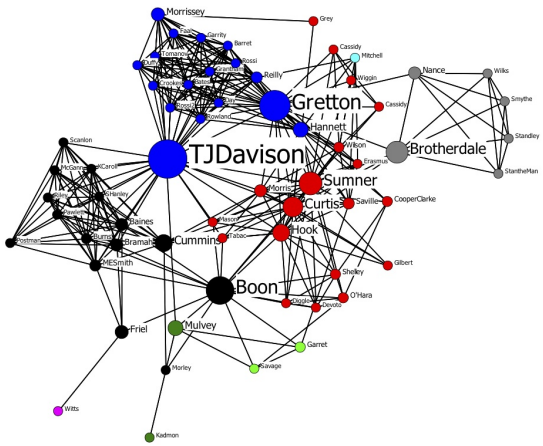
- Dataset : n observations, each observation is encoded by a vector $\in \mathbb{R}^d$.
- Purpose: Organize the data into homogen groups



Application Domains

- Natural Language Processing: Find set of texts
- Documents: Automatic classification (Driver License, ID, Passport)
- Marketing: Client profiles
- ...





Questions raised I

- What is a cluster ?
- What is a good cluster(ing) ?
- What is similar data ?
- How many clusters ?
- How to evaluate clusters ?
- Which method to compute clusters ?

Questions raised II

What is a cluster ?

A set of observations grouped together.

- Formally, we assign a value $c \in \mathbb{N}$ to each observation
- The clustering encoded by a vector $\mathbf{c} \in \mathbb{N}^n$.

Questions raised III

What is a good cluster(ing)

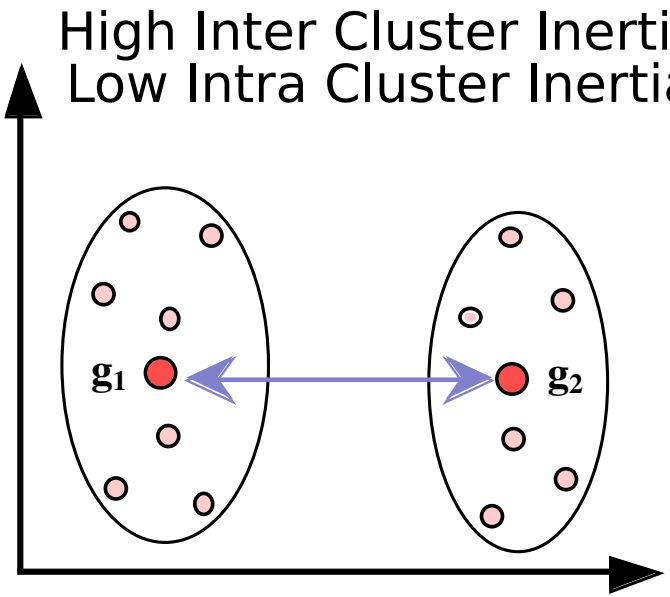
A good cluster:

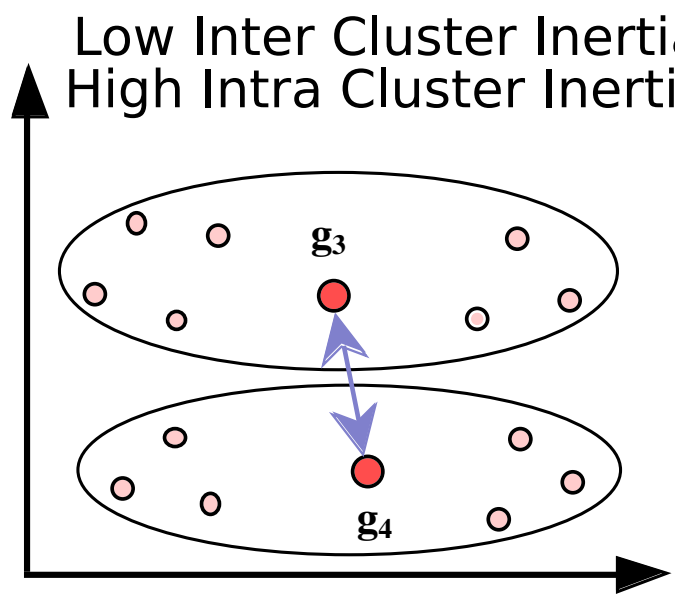
- A cluster is homogen if all data are similar
  - $\Rightarrow$ Need to define the (dis)similarity between data
  - Euclidean distance.
- Intra cluster variance : measure of dispersion of the data inside a cluster. Lower the better.

A good clustering:

- A good clustering is when clusters are very different
  - Need to define a similarity between clusters
  - Based on similarity between data
  - Use the closest, farthest pairs, distance of barycenters, sum of all distances between pairs, etc
- Inter cluster variance : measure of dispersion of centers of clusters. Higher the better.

Questions raised IV





Questions raised V

How many clusters ?

- Depends on the application
- May be known a priori ...
- If not, need to be optimized
- How to evaluate a good clustering ?
  - Hard because we don't have ground truth values
  - $\Rightarrow$ Minimize Intra cluster variance, maximize inter cluster variance.
  - Silhouette score, and more

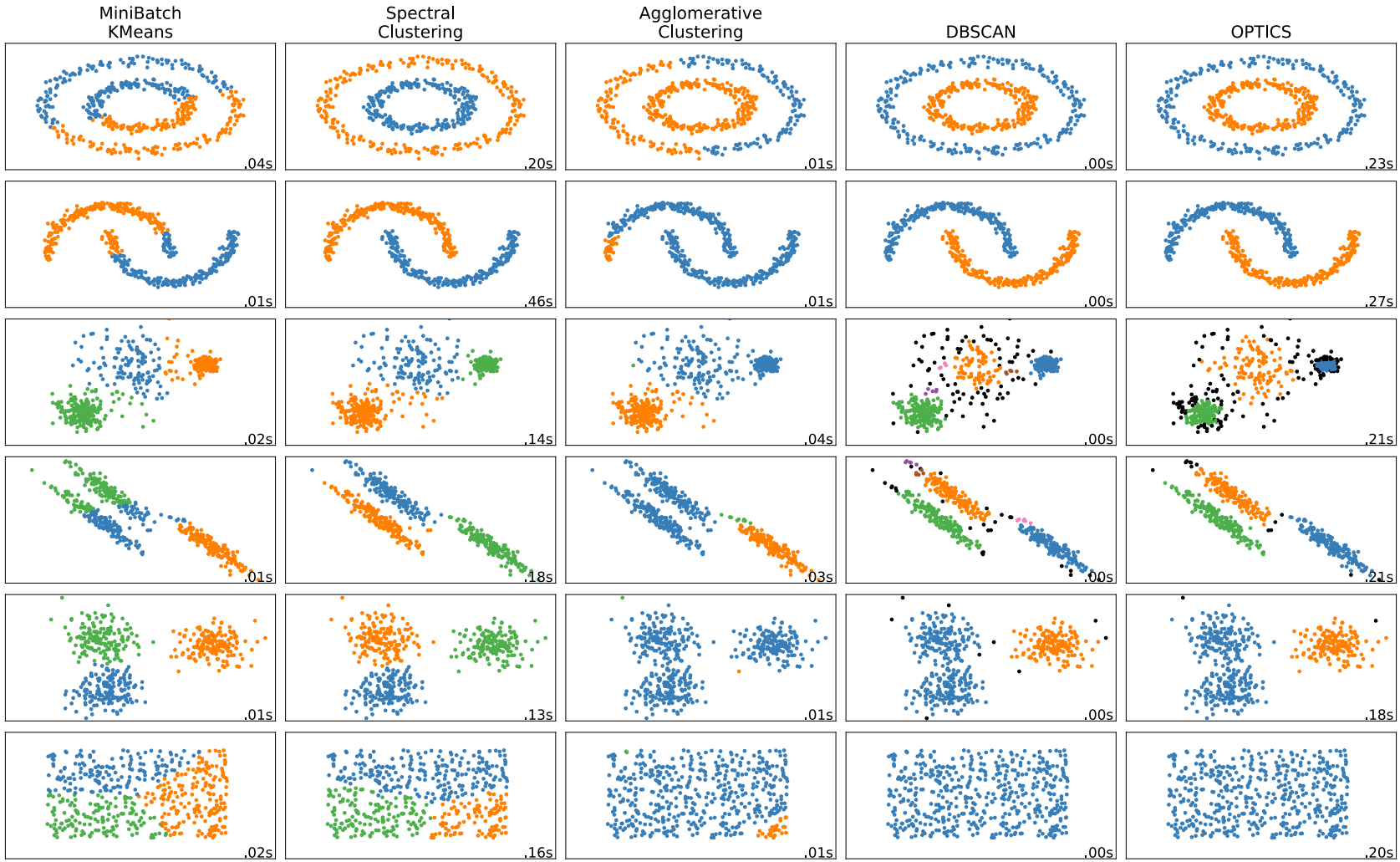
Questions raised VI

What method to use ?

Many methods exists (https://scikit-learn.org/stable/modules/clustering.html)

- **Kmeans** : iterative optimization of centers
- **Hierarchical Clustering** : iterative merge of clusters
- **DBSCAN** : Identify and expand areas of high density

Questions raised VII



K-means

K-means for Clustering

Purpose

- $\mathcal{D} = \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1,\cdots,N}$
- Clustering in $K < N$ clusters $\mathcal{C}_k$

Brute Force

1. Build all possible partitions
2. Evaluate each clustering et keep the best clustering

Problem

Number of possible clusterings increases exponentially

$$\#\text{Clusters} = \frac{1}{K!}\sum_{k=1}^{K}(-1)^{K-k}C_k^K k^N$$

For $N = 10$ and $K = 4$, we have 34105 possible clusterings ! For $N = 100$ and $K = 4$, we have $\simeq 6,69 * 10^{58}$ possible clusterings !

K-means for Clustering

A better solution

- Minimizing intra-class inertia, w.r.t. $\boldsymbol{\mu}_k, k = 1, \ldots, K$

$$J_w = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} {d_m}^2(\boldsymbol{x}_i, \boldsymbol{\mu}_k)$$

- Use of an heuristic: we will have a good clustering but not necessarily the best one according to J_w

K-means for clustering

A famous algorithm: K-means

1. Consider we have gravity centers $\boldsymbol{\mu}_k, k = 1, \cdots, K$
2. we affect each $\boldsymbol{x}_i$ to the closest cluster $\mathcal{C}_\ell$:

$$\ell = \arg\min_k d_m(x_i, \boldsymbol{\mu}_k)$$

3. We recompute $\boldsymbol{\mu}_k$ for each $\mathcal{C}_k, k = 1, \cdots, K$
4. We continue until we reach convergence

K-means algorithm

- Initialise gravity centers $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K$
- Repeat
  - Affect each point to its closest cluster

$$\mathcal{C}_\ell \leftarrow \boldsymbol{x}_i \quad \text{such as} \quad \ell = \arg \min_k d_m(\boldsymbol{x}_i, \boldsymbol{\mu}_k)$$

  - Compute $J_w = \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} {d_m}^2(\boldsymbol{x}_i, \boldsymbol{\mu}_k)$
  - Compute $\boldsymbol{\mu}_k$ for each cluster

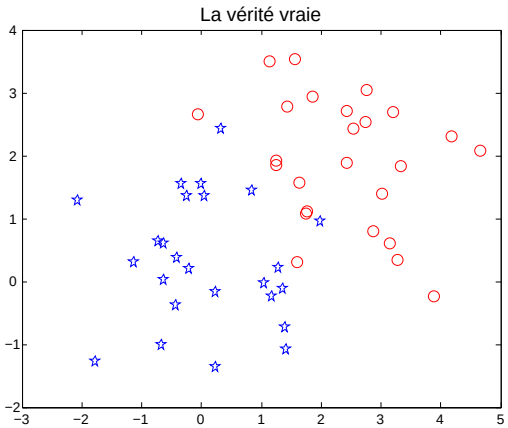
$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i \in \mathcal{C}_k} \boldsymbol{x}_i \quad \text{with} \quad N_k = \text{card}(\mathcal{C}_k)$$

- Until $\| \Delta\boldsymbol{\mu} \| > \epsilon$ ou $\|J_w\| > \epsilon_2$

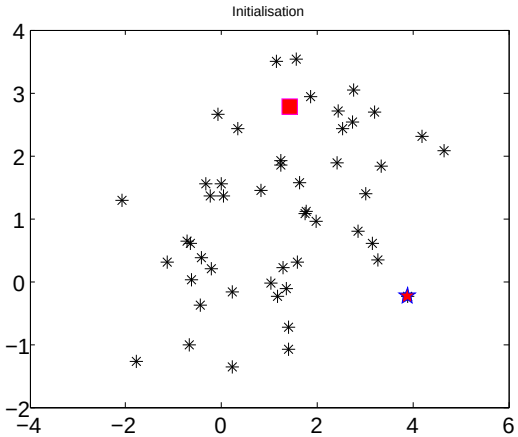
K-Means : illustration

Clustering in $K = 2$ clusters

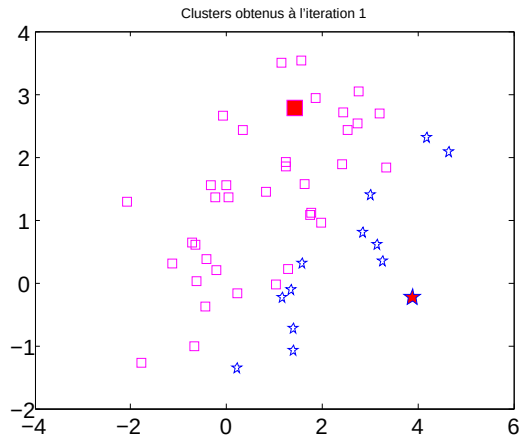
Data



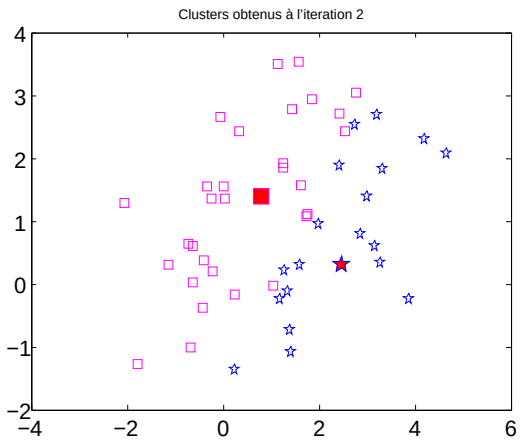
Initialisation



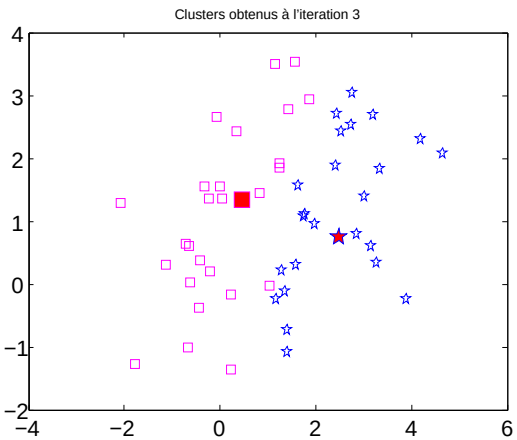
Iteration 1



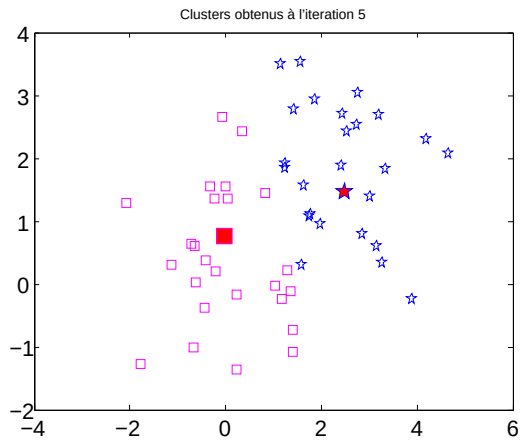
Iteration 2



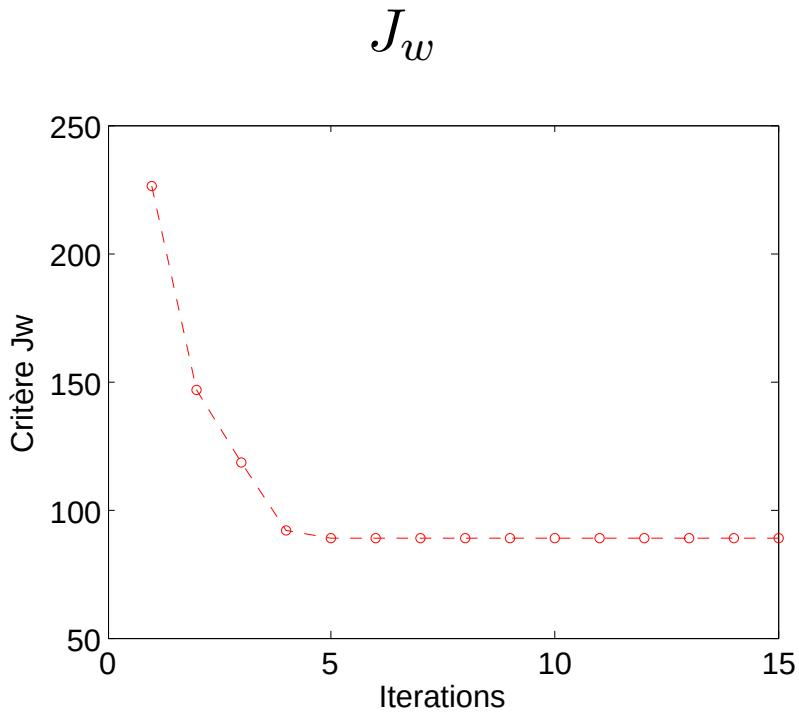
Iteration 3



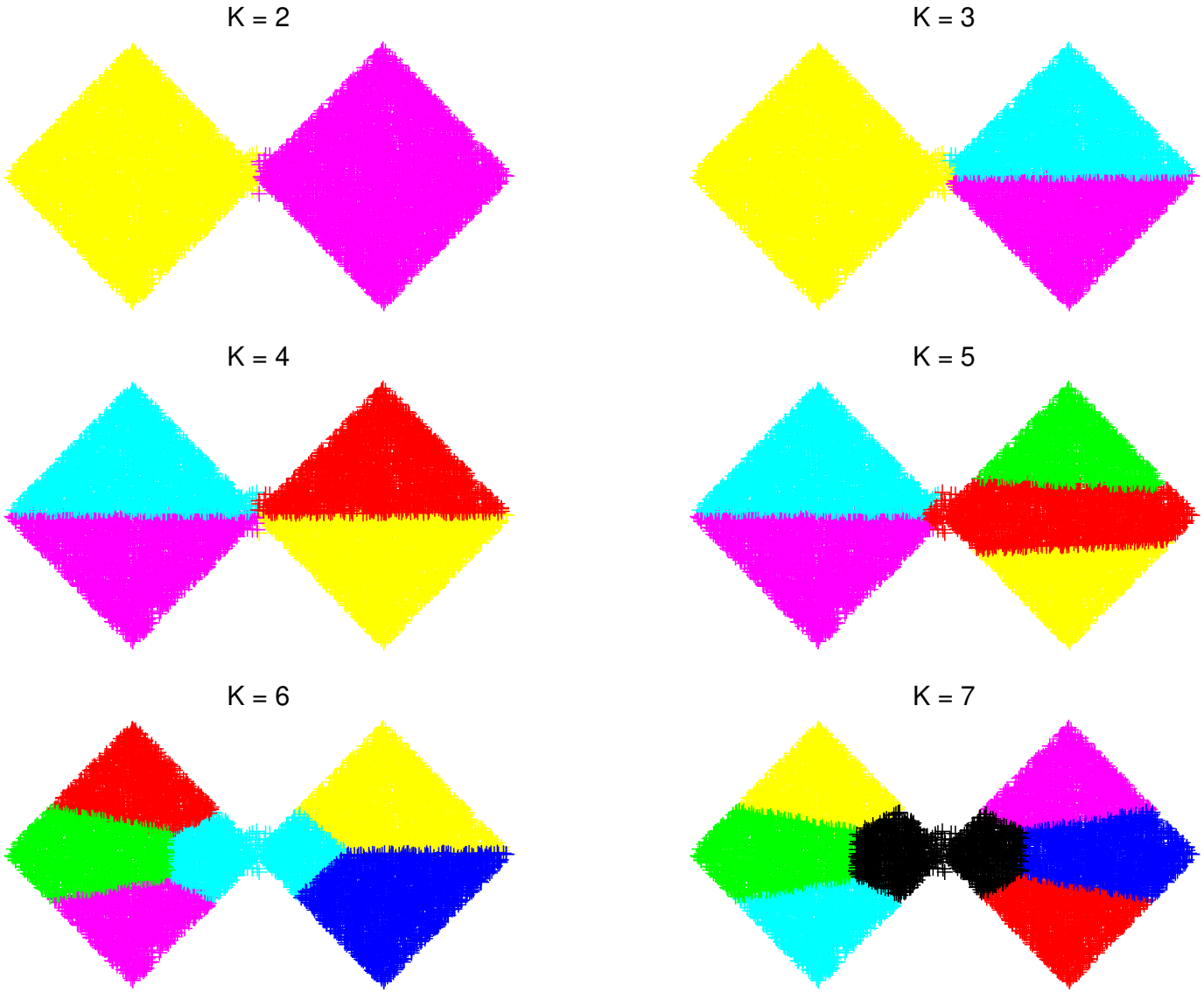
Iteration 5



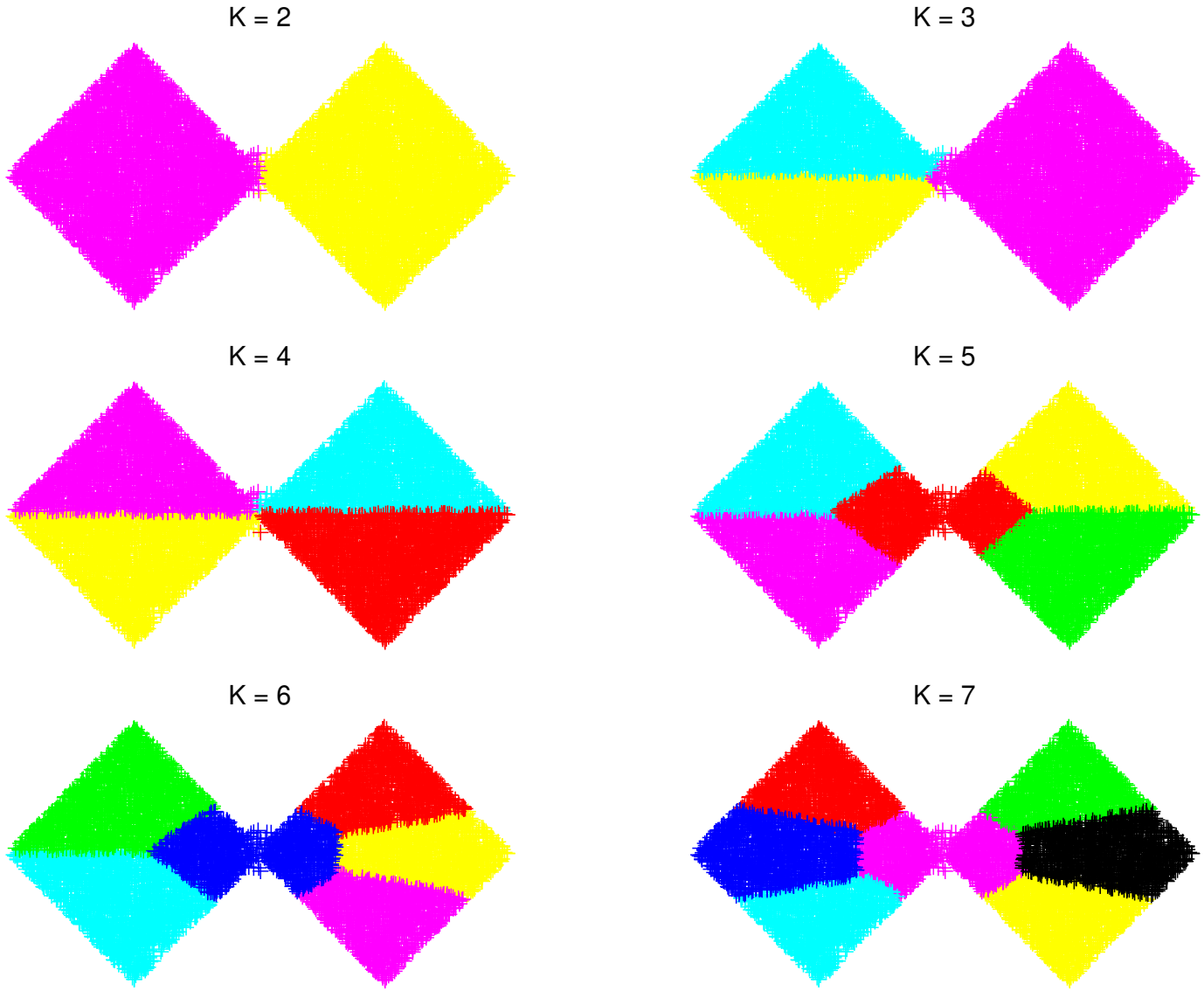
K-Means : J_w



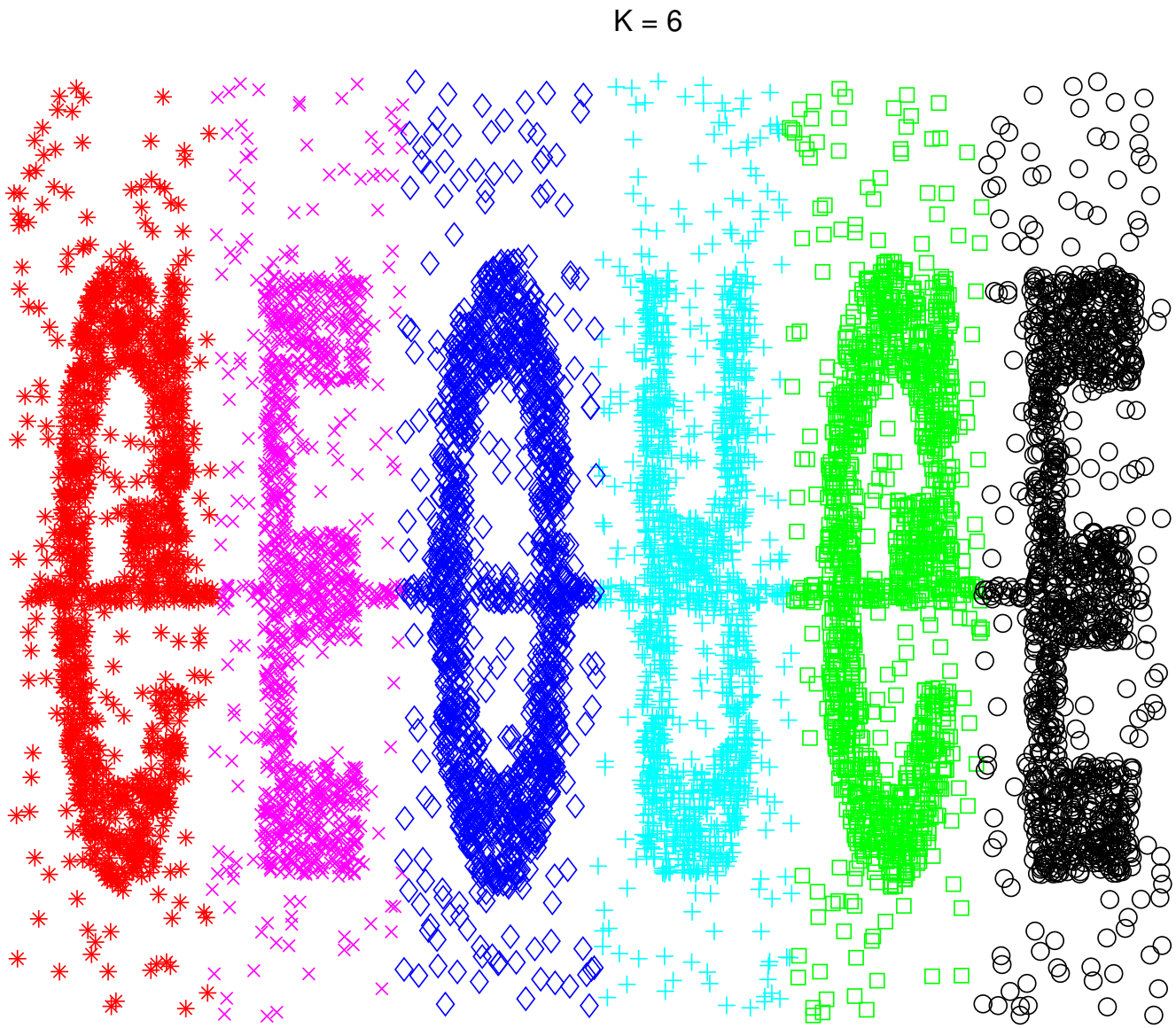
K-Means : example



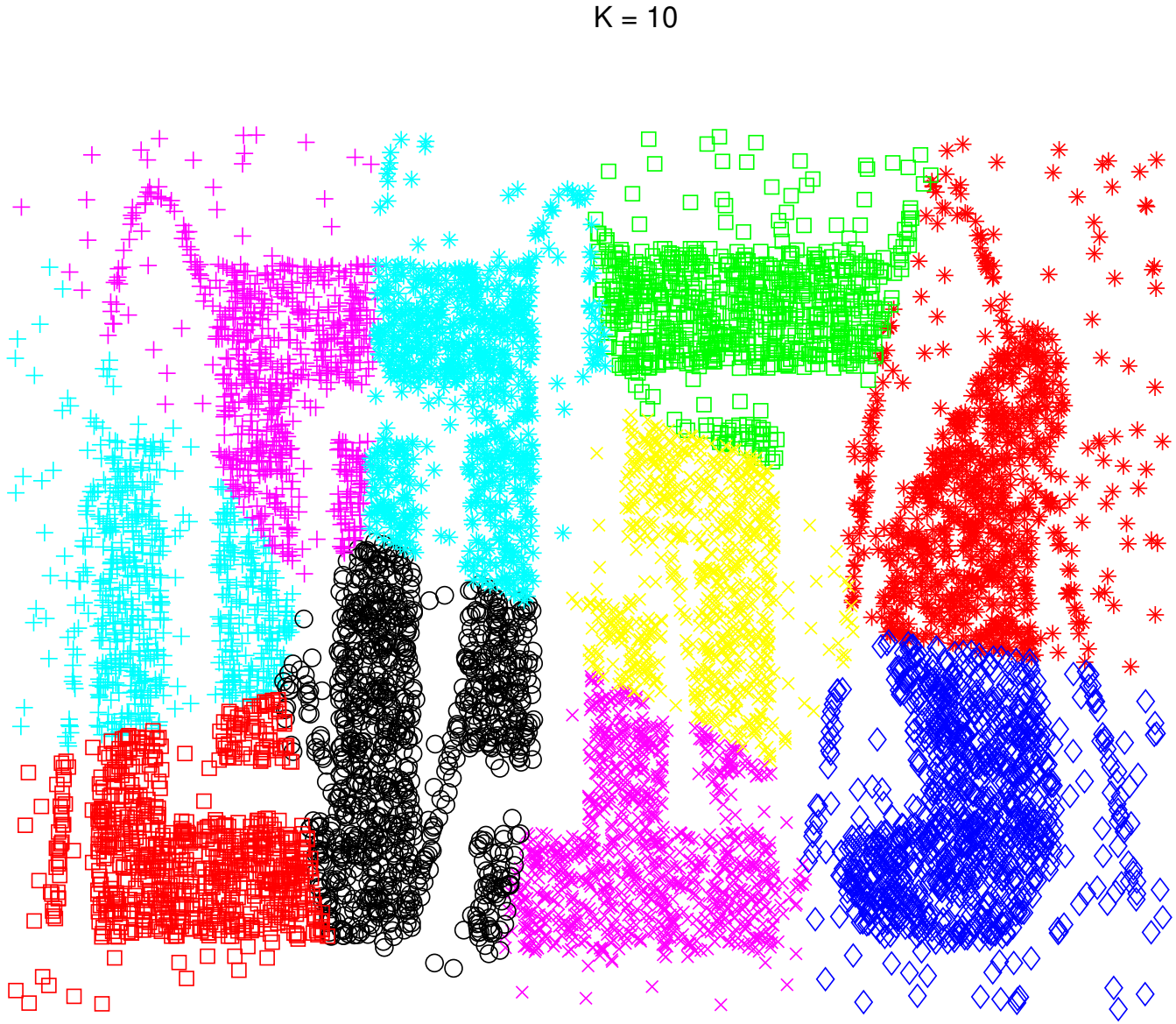
K-Means : example



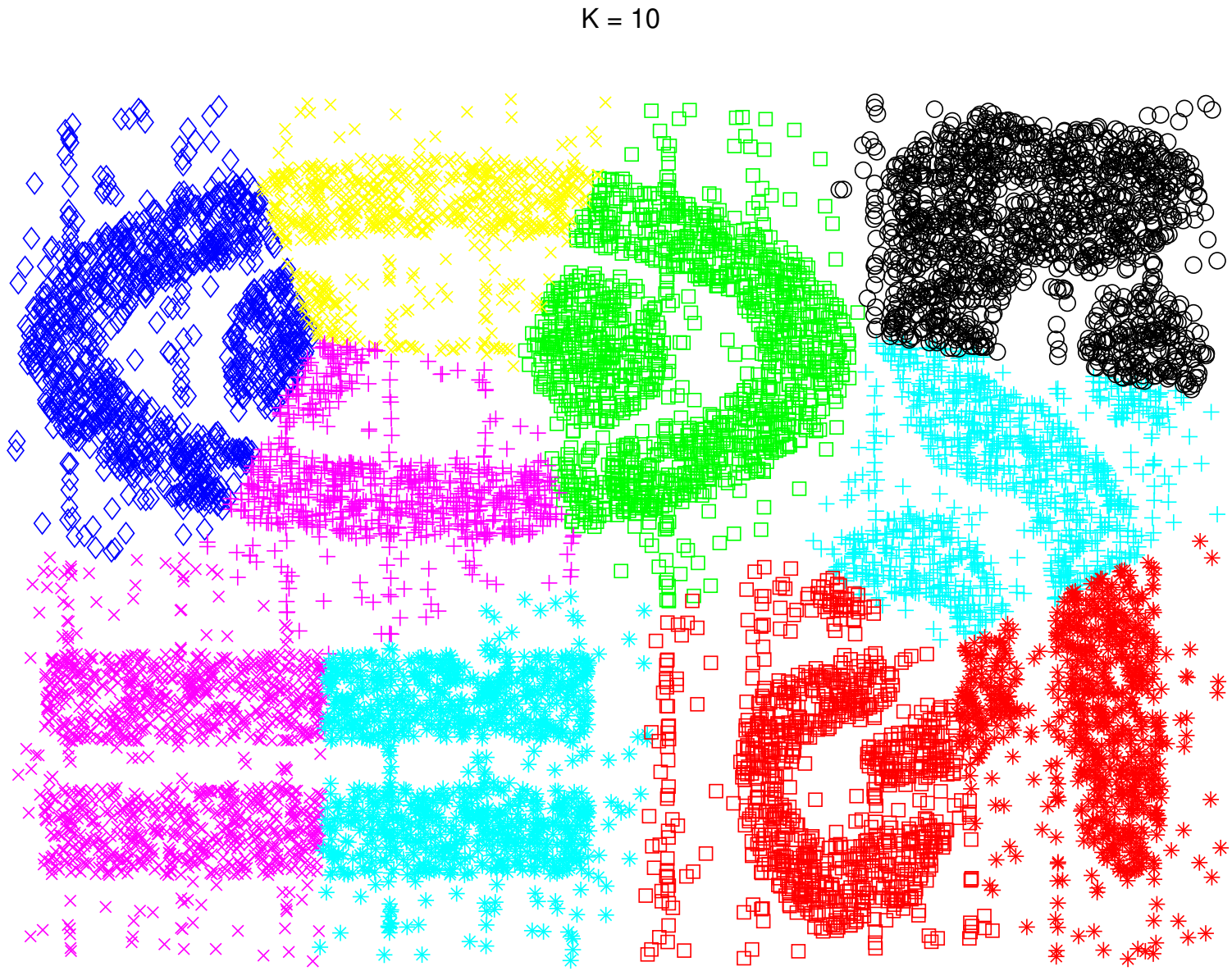
K-Means : example



K-Means : example



K-Means : example



K-Means : Color Quantification

Passage de 89944 à 32 couleurs.

K-Means : Discussion

Observations on K-Means

- J_w decreases at each iteration
- It converges towards a local minimum of J_w
- Initialisation of $\boldsymbol{\mu}_k$:
  - Randomly within $\boldsymbol{x}_i$ domain
  - Randomly K among $\boldsymbol{X}$
- Different initializations may lead to different clustering
- scikit-learn : test of different initalisations and choose the lowest J_w

Kmeans Implementation

```
1  from sklearn.datasets import make_blobs
2  import matplotlib.pyplot as plt
3  from sklearn.cluster import KMeans
4  X, y = make_blobs(n_samples=100, centers=3,
5                                          n_features=2, cluster_std=0.5)
6  # K = 3 clusters. By default, Kmeans uses multiple initializations
7  clustering = KMeans(n_clusters=3)
8  clustering.fit(X) # computes the centroids
9  clusters = clustering.predict(X) # assigns each point to a cluster
10 # plot the points with the cluster color
11 plt.scatter(X[:, 0], X[:, 1],c=clusters)
```

- sklearn documentation
- How to use Kmeans

K-Means conclusion

Pros

- Intuitive and easy to understand algorithm
- Very effective in practice
- Improved computational time : Mini Batch KMeans
- Solutions for the problem of initialization

Cons

- Need to define K
- Only convex clusters

Other methods for Clustering

Hierarchical Clustering

Iteratively merge clusters

- Need to define a cluster's distance
- Number of cluster can be choosen a posteriori
- sklearn

DBScan

Identify high density areas and expand them

- Number of clusters is deduced from the data
- Some points may be outside clusters
- sklearn

and others.

Unsupervised Learning
Dimensionnality Reduction

Principal component analysis (PCA) I

Principle

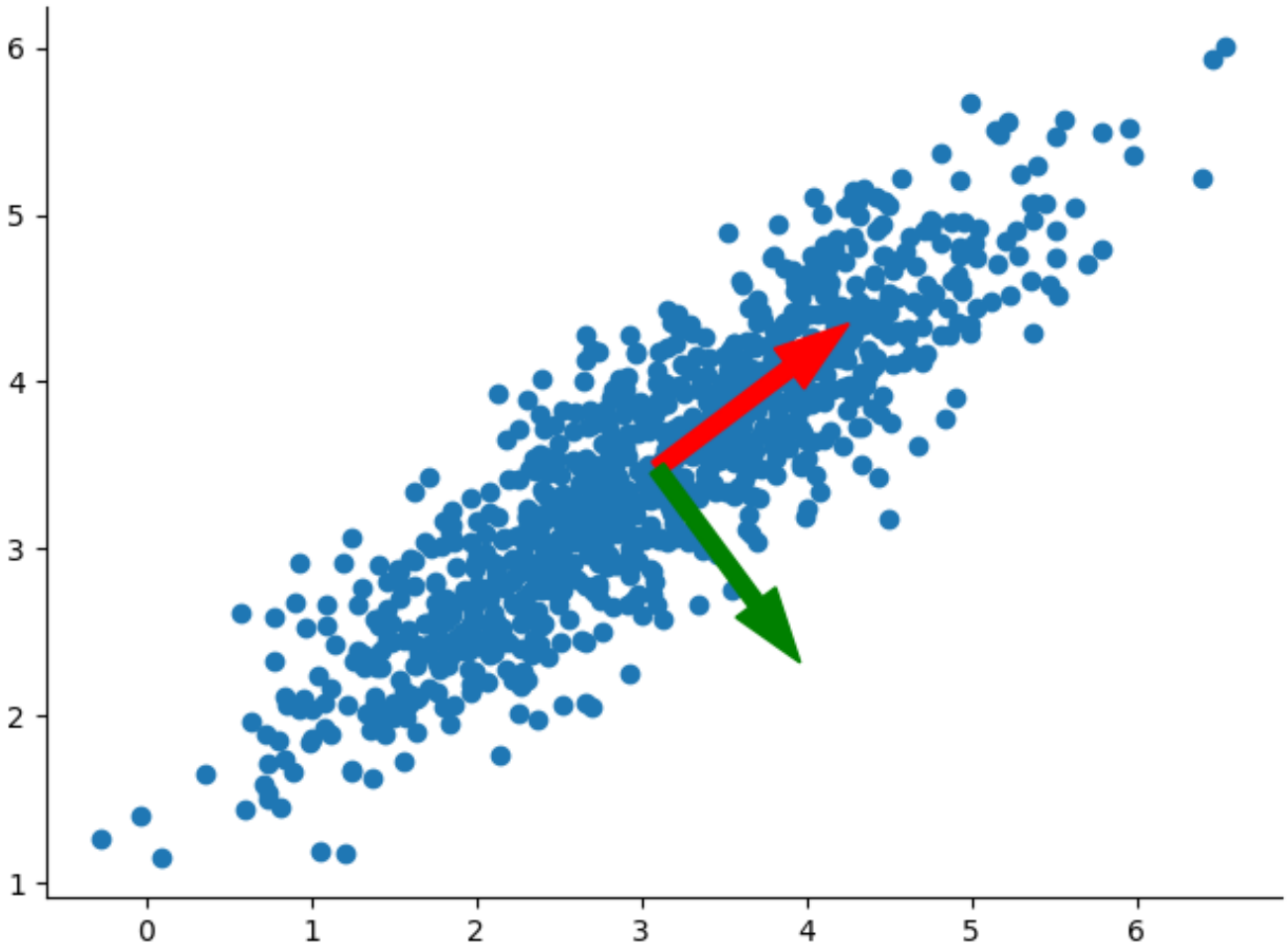
Find a new base where first dimensions encodes most of the information.

- Information $\Rightarrow$ Variance, Dispersion of the data.
- First components will contain information
- Last components will contain noise
- Need to tune the number of components to keep :
  - 2 or 3 for visualization
  - for data compression : as many as required to keep a decent quality,
  - for data transmission : as few as possible to limit the amount of data,
  - for data analysis : enough to only keep information and discards noise.

Principal component analysis (PCA) II

PCA on point cloud

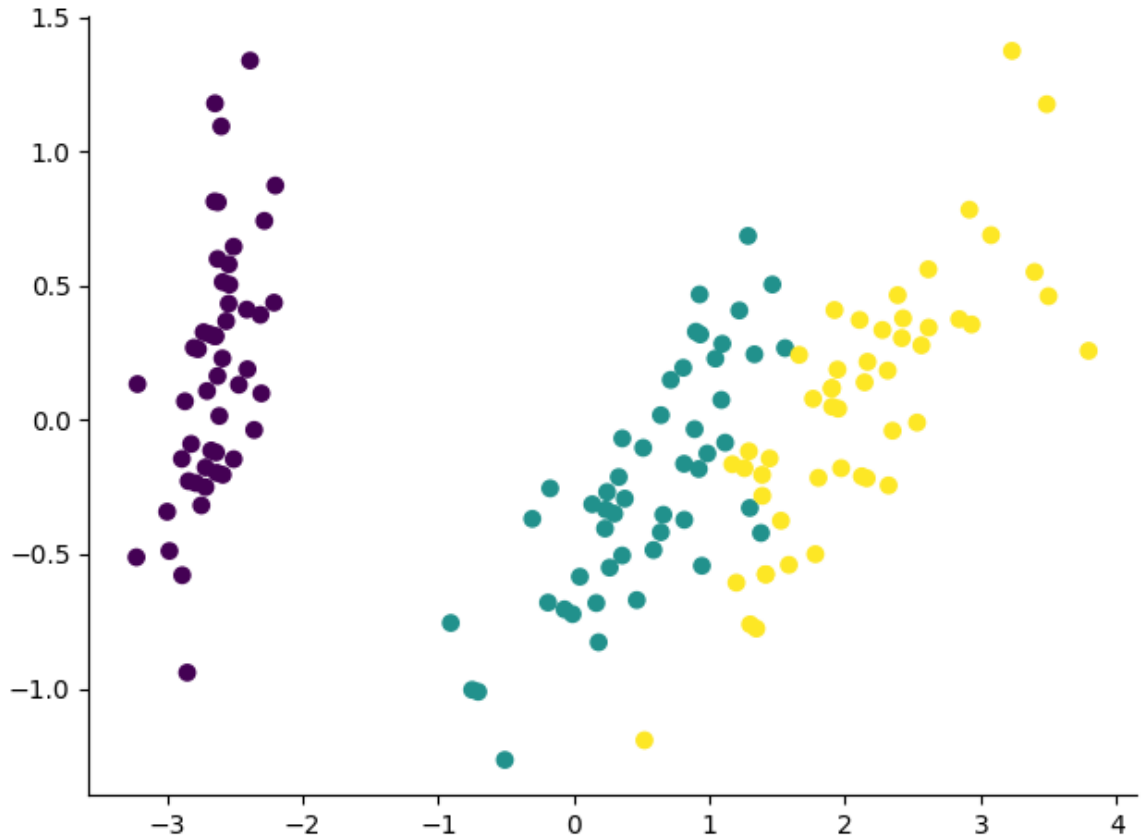
Identify the axes where the information is spread



Principal component analysis (PCA) III

PCA on Iris Dataset

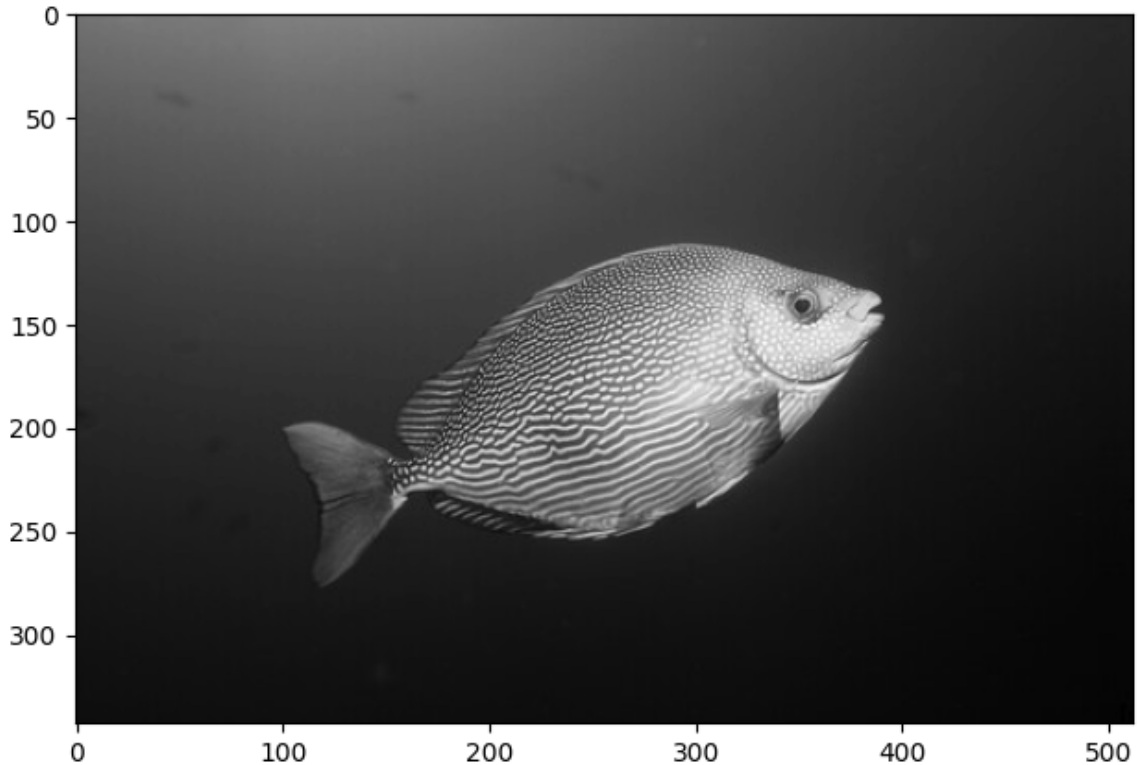
Iris dataset is a dataset with 150 flowers encoded by 4 values.
Impossible to visualize, we need to reduce its dimension.



Principal component analysis (PCA) IV

PCA for images

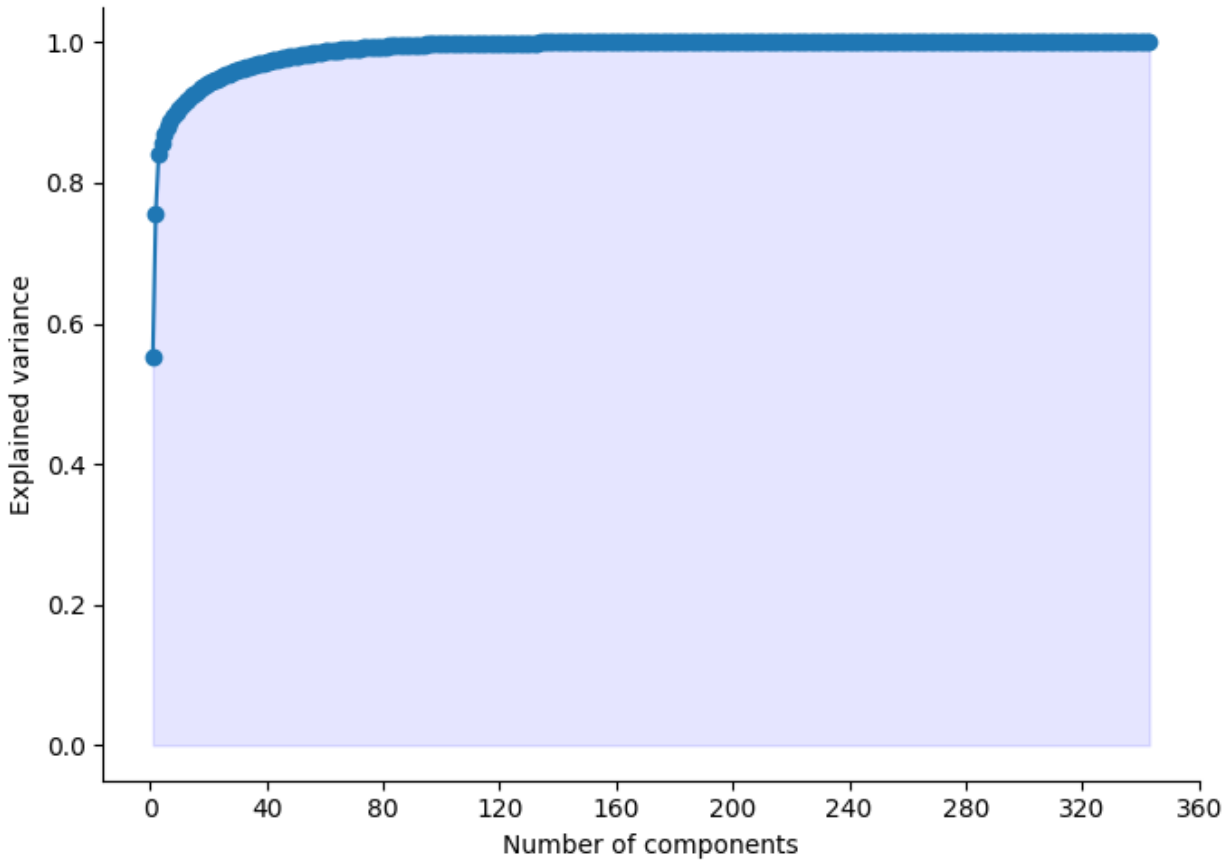
Original image is 343×512, i.e. 175616 pixels.



Principal component analysis (PCA) V

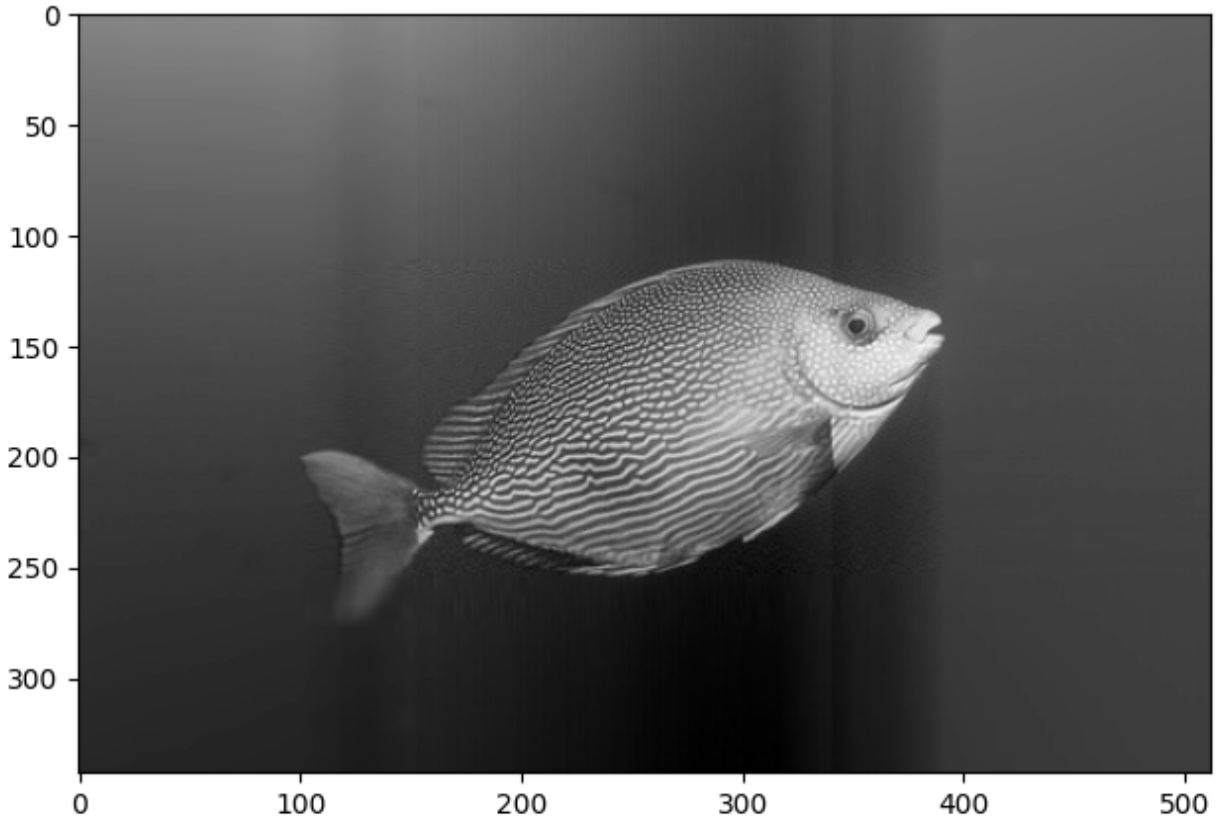
Amount of encoded info

Using 50 components, most of information is kept .



Principal component analysis (PCA) VI

Reconstruction of the image using twice less data.



PCA Implementation

```
1  from sklearn.decomposition import PCA
2  from sklearn.datasets import load_iris
3  X,y = load_iris(return_X_y=True) # load the data
4  pca = PCA(n_components=2) # define the PCA with 2 components to keep
5  pca.fit(X) # learn the new base for representation
6  # compute the representation of the data in this new base
7  X_pca = pca.transform(X)
8  # display the data in the new base (here 2 dimensions)
9  plt.plot(X_pca[:,0], X_pca[:,1],'o')
10 # reconstruct the compressed data in the original base
11 X_reconstruct = pca.inverse_transform(X_pca)
```

- full documentation
- sklearn example

Extensions to non linear reprensentation

Non linear representations

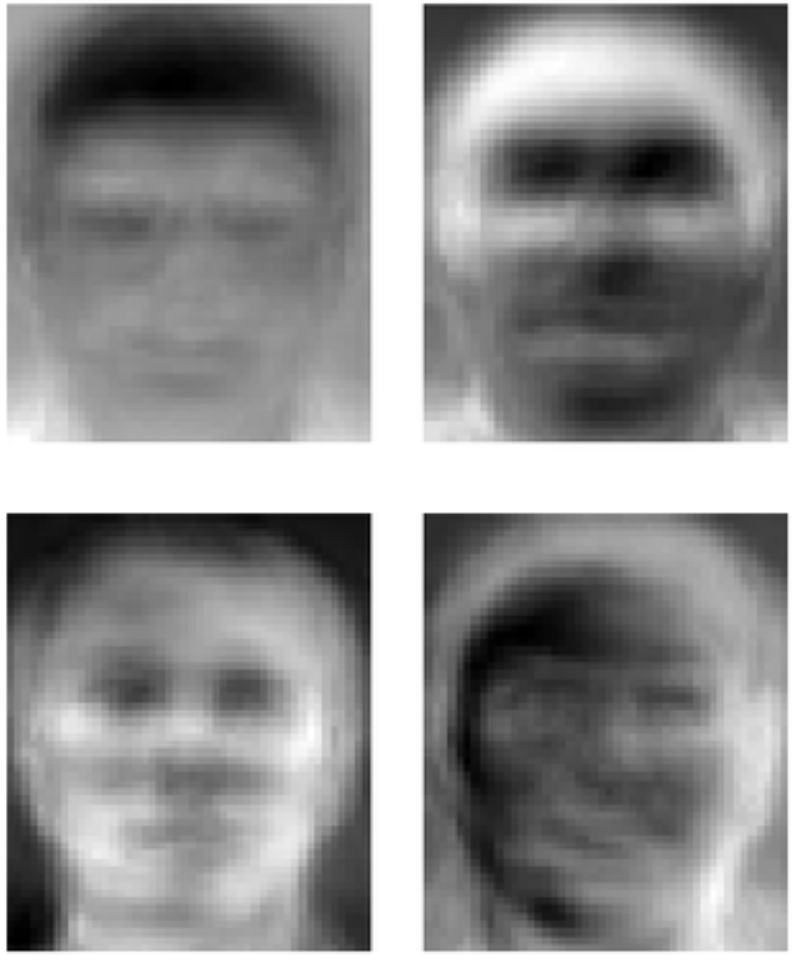
PCA is linear by design : limited to simple transformations.
This principle has been extented to non linear transformations :

- **KernelPCA** : Abstraction of the dot product to non linear functions,
- **t-SNE** : Compute a space where local relationships are kept,
- **UMAP** : More complex . . . details here.

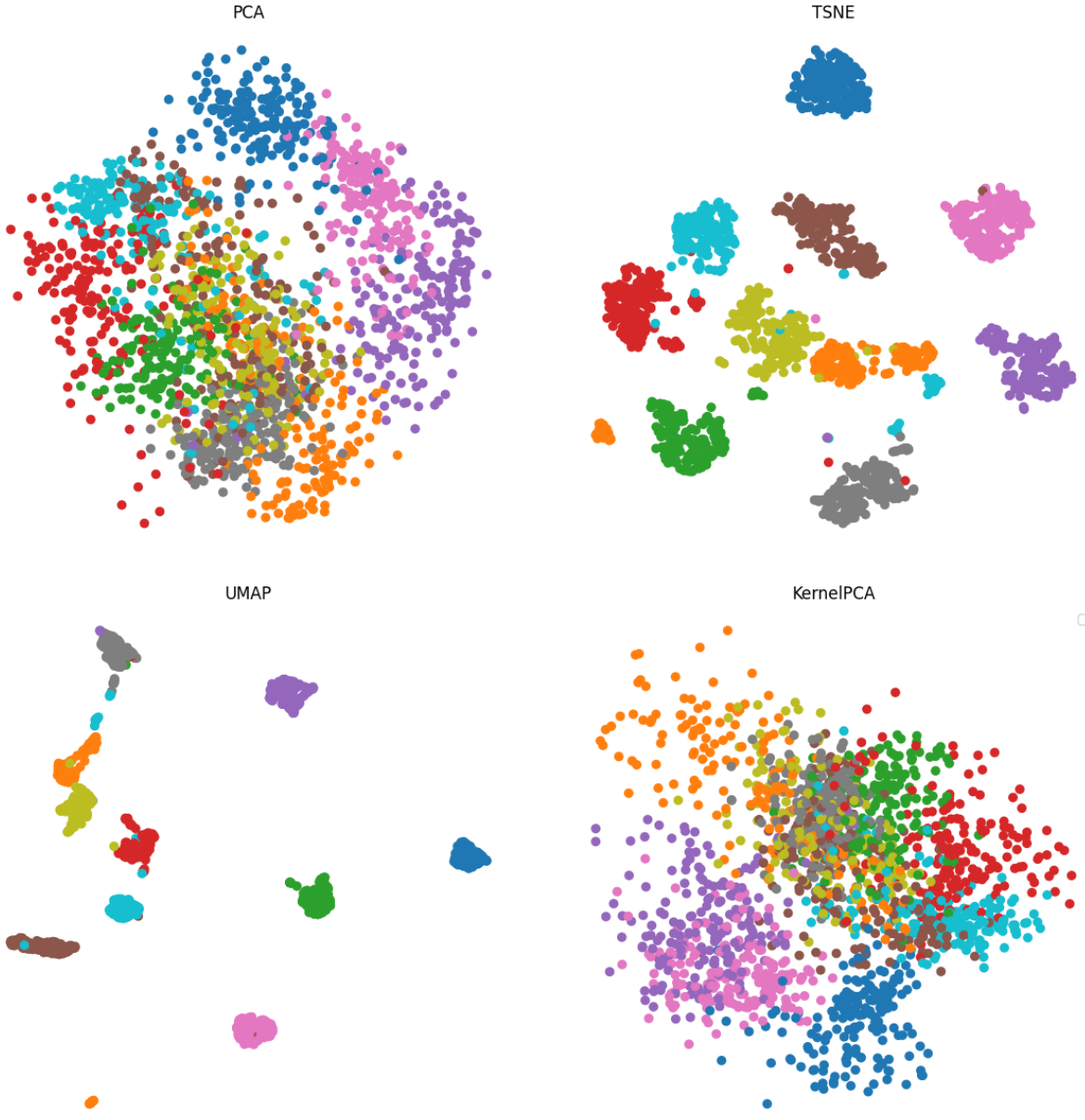
Illustrative comparison: `https://projector.tensorflow.org/`

Examples of non linear dimension reductions I

Facial recognition : EigenFaces



Examples of non linear dimension reductions II



Dimensionality Reduction - Conclusion

Pros

- Used to visualize the data
- Data compression:
  - → Data denoising
  - → Accelerate computation
  - → Latent space

,

Cons

- Need to handcraft how data is compared
- PCA is limited to linear transformations
- May be long to be computed

Unsupervised learning - Conclusion

We're limited and blind with unsupervised Learning 😔

But:

- We can understand how the data is structured
- In the real world, most of the data is unlabeled
- unsupervised learning can serve as a preprocessing step for more complex tasks.

Next step: use the data's properties !