Introduction à l'Apprentissage automatique
Apprentissage non supervisé

Benoit Gaüzère - Romain Hérault

INSA Rouen Normandie - Laboratoire LITIS

March 21, 2025

Qu'est-ce que le Machine Learning ? I

Intelligence Artificielle

Un programme ou un dispositif qui présente ou imite des comportements cognitifs attribués aux humains.
Exemples : **déduire** ou **inférer**.

Cette définition est très imprécise et centrée sur l'humain.

IA Symbolique

- **Déduire** de nouvelles règles de décision à partir de **règles** existantes,

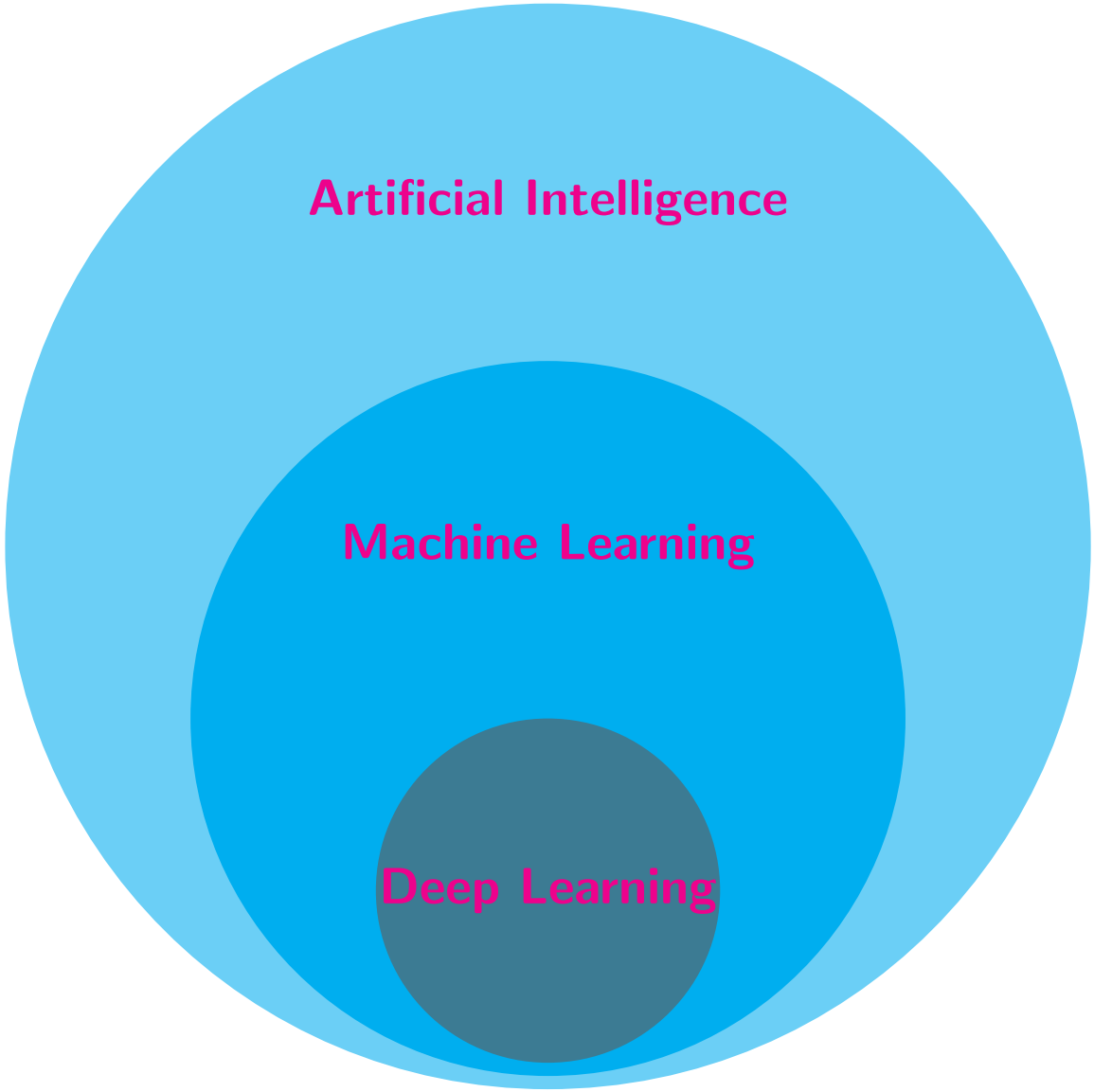
$\Rightarrow$ Optimisation sur un domaine discret/limité.

Apprentissage Automatique

- **Inférer** de nouvelles règles de décision à partir d'**observations**,

$\Rightarrow$ Optimisation continue ou mixte.

Qu'est-ce que le Machine Learning ? II



Qu'est-ce que le Machine Learning ? III

Tâches différentes

Apprentissage supervisé prédit la propriété d'une observation en s'appuyant sur des paires (observation, propriété),

Apprentissage non supervisé détecte des motifs ou la structure des données uniquement à partir des observations,

Apprentissage semi-supervisé prédit la propriété de données en ne considérant qu'un petit ensemble de paires (observation, propriété)

et bien d'autres : Apprentissage par renforcement, Apprentissage actif, Apprentissage auto-supervisé, ...

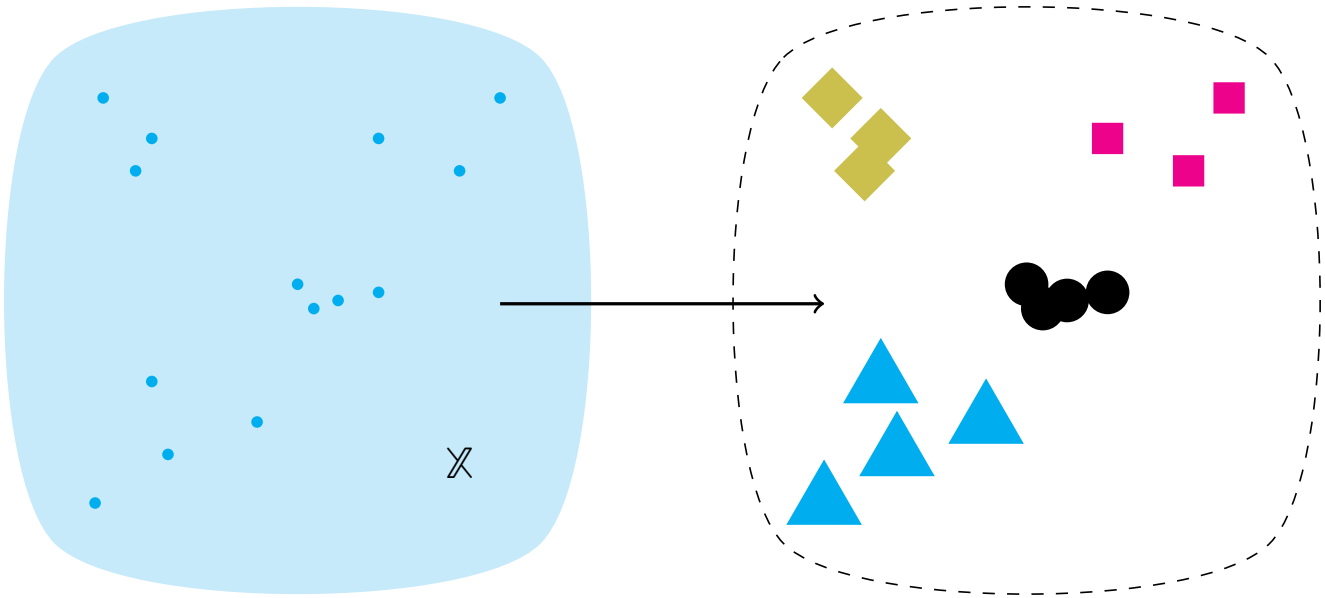
Apprentissage non supervisé I

Utilise uniquement les observations !

Clustering (Regroupement)

Apprentissage non supervisé II

Détecter des groupes de données homogènes

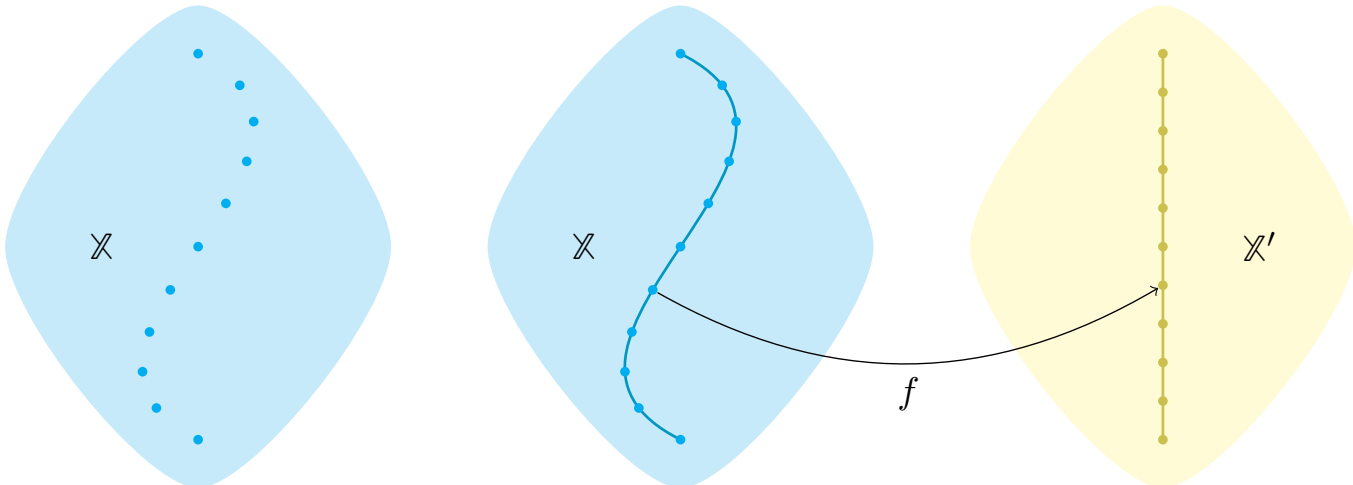


- Méthodes : Kmeans, Clustering hiérarchique, DBSCAN, ...
- Utilisé pour segmenter des profils clients, des zones d'images, ...

Apprentissage non supervisé III

Réduction de dimension

Découvrir une représentation de faible dimension qui capture l'essentiel de l'information contenue dans les données.



- Méthodes : ACP (PCA), t-SNE, UMAP
- Compression des données, élimination du bruit, visualisation, espace latent

Apprentissage non supervisé IV

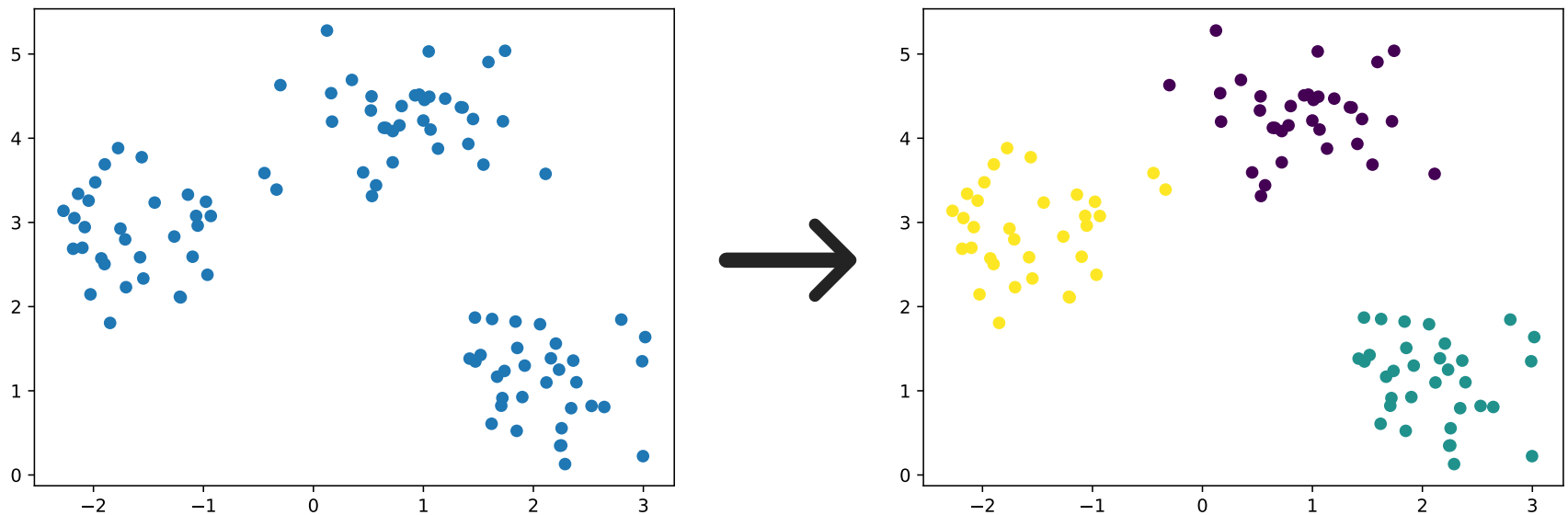
et d'autres encore : détection de nouveauté, génération de données, ...

Apprentissage non supervisé
Clustering

Clustering

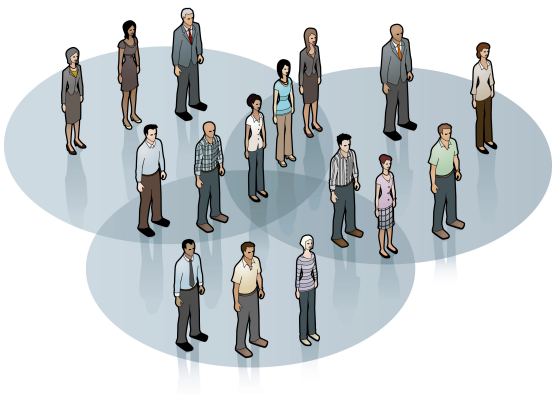
Clustering en quelques mots

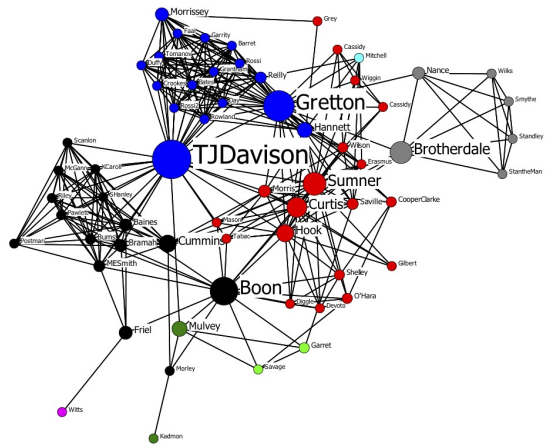
- Jeu de données : n observations, chaque observation est encodée par un vecteur $\in \mathbb{R}^d$.
- Objectif : Organiser les données en groupes homogènes



Domaines d'application

- Traitement automatique du langage naturel : regrouper des ensembles de textes
- Documents : classification automatique (Permis de conduire, Carte d'identité, Passeport)
- Marketing : profils clients
- ...





Questions soulevées I

- Qu'est-ce qu'un cluster ?
- Qu'est-ce qu'un bon cluster (ou un bon clustering) ?
- Que signifie ≪ données similaires ≫ ?
- Combien de clusters ?
- Comment évaluer les clusters ?
- Quelle méthode pour calculer les clusters ?

Questions soulevées II

Qu'est-ce qu'un cluster ?

Un ensemble d'observations regroupées ensemble.

- Formellement, on assigne une valeur $c \in \mathbb{N}$ à chaque observation.
- Le clustering est encodé par un vecteur $\mathbf{c} \in \mathbb{N}^n$.

Questions soulevées III

La notion de distance en apprentissage automatique

- Une **distance** mesure la dissimilarité entre deux objets.
- Elle doit satisfaire 3 propriétés fondamentales :
  - **Symétrie** : $d(A, B) = d(B, A)$
  - **Positivité** : $d(A, B) \geq 0$, et $d(A, A) = 0$
  - **Inégalité triangulaire** : $d(A, C) \leq d(A, B) + d(B, C)$

Exemples de distances usuelles :

- **Distance euclidienne** : $d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$
- **Distance de Manhattan** : $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$

Applications :

- Clustering : pour regrouper des données similaires
- K plus proches voisins : pour prédire à partir des voisins les plus proches

Questions soulevées IV

Capsule de rappel : Variance et Inertie

La **variance** mesure la dispersion des données :
plus elle est élevée, plus il y a de différences entre les observations.

Variance faible

- Variations fines entre individus d'un même groupe
- Peu d'information discriminante
- Souvent assimilée à du **bruit**

Variance élevée

- Différences majeures entre groupes
- Portée informative plus forte
- Peut refléter une structure des données

À retenir

Variance élevée $\Rightarrow$ *information discriminante*
Variance faible $\Rightarrow$ *variabilité inter-individus (bruit)*

Questions soulevées V

Qu'est-ce qu'un bon cluster (ou un bon clustering) ?

Questions soulevées VI

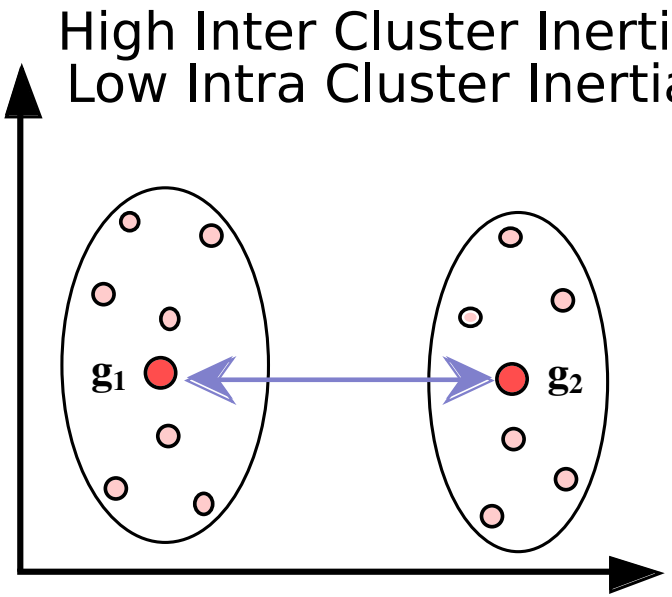
Un bon cluster :

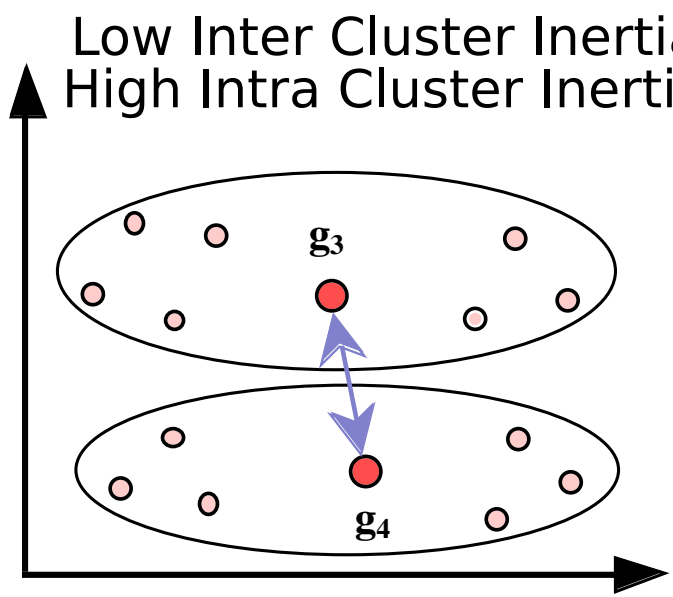
- Un cluster est homogène si toutes les données sont similaires
  - $\Rightarrow$ Il faut définir la (dis)similarité entre les données
  - Par exemple : distance euclidienne
- Variance intra-cluster : mesure la dispersion des données dans un cluster. Plus elle est faible, mieux c'est.

Un bon clustering :

- Un bon clustering se caractérise par des clusters très différents les uns des autres
  - Il faut définir une similarité entre clusters
  - Basée sur la similarité des données
  - Utiliser la distance entre les plus proches/éloignés, distance entre barycentres, somme des distances, etc.
- Variance inter-cluster : mesure la dispersion des centres des clusters. Plus elle est élevée, mieux c'est.

Questions soulevées VII





Questions soulevées VIII

Combien de clusters ?

- Cela dépend de l'application
- Peut être connu a priori ...
- Sinon, il faut l'optimiser
- Comment évaluer un bon clustering ?
  - Difficile en l'absence de vérité terrain (ground truth)
  - $\Rightarrow$ Minimiser la variance intra-cluster, maximiser la variance inter-cluster
  - Score de silhouette, et d'autres encore

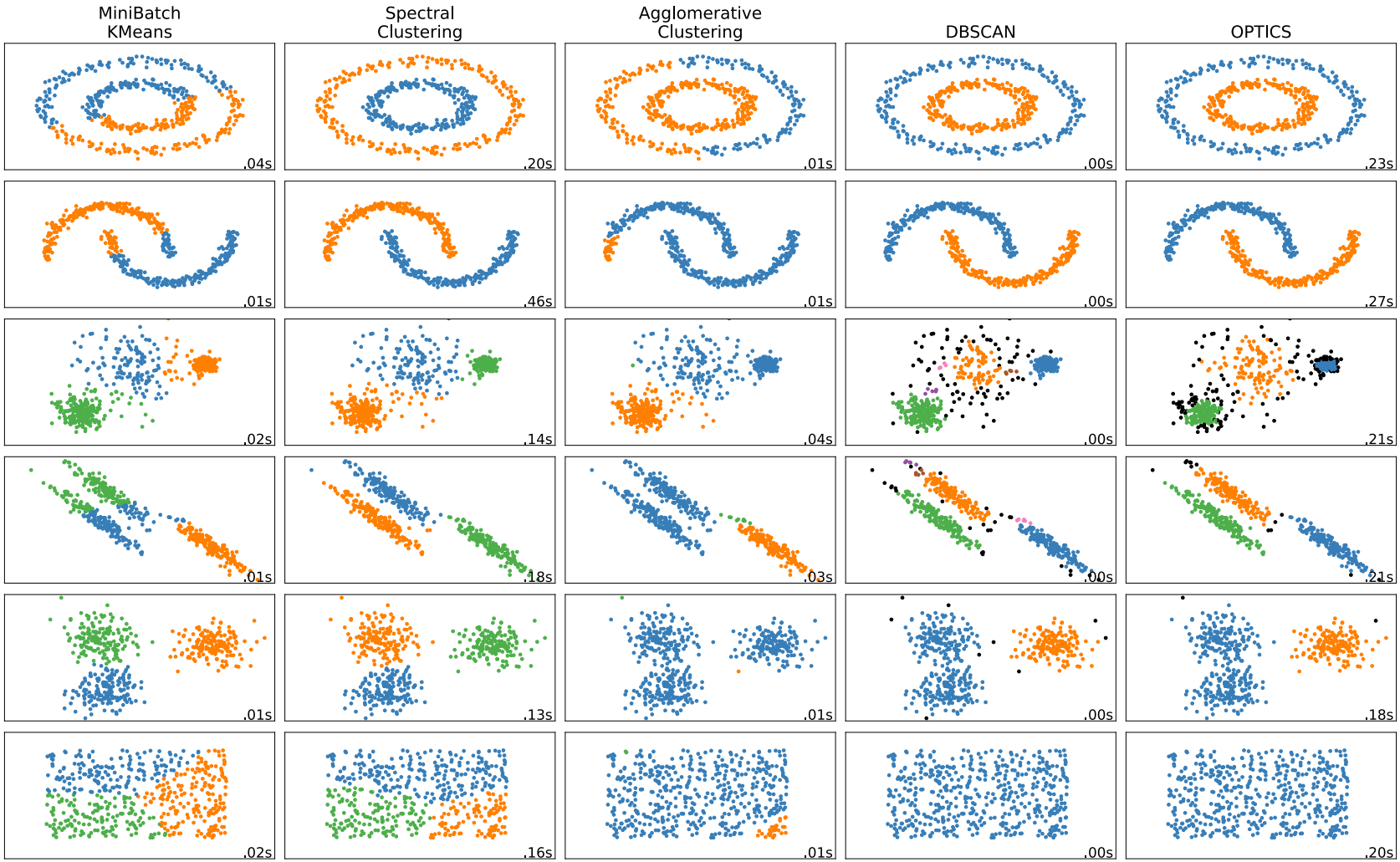
Questions soulevées IX

Quelle méthode utiliser ?

De nombreuses méthodes existent (https://scikit-learn.org/stable/modules/clustering.html)

- **Kmeans** : optimisation itérative des centres
- **Clustering hiérarchique** : fusion itérative des clusters
- **DBSCAN** : identification et expansion des zones à forte densité

Questions soulevées X



K-means

K-means pour le Clustering

Objectif

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

Approche « Brute Force »

1. Construire toutes les partitions possibles
2. Évaluer chaque clustering et conserver le meilleur

Problème

Le nombre de clusterings possibles augmente de manière exponentielle :

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

Pour $N = 10$ et $K = 4$, on obtient 34105 clusterings possibles !
Pour $N = 100$ et $K = 4$, on a environ $\simeq 6{,}69 \cdot 10^{58}$ clusterings

K-means pour le Clustering

Une meilleure solution

- Minimiser l'inertie intra-classe, par rapport aux centres de gravité $\boldsymbol{\mu}_k, k = 1, \ldots, K$:

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

- Utilisation d'une heuristique : on obtient un bon clustering mais pas forcément le meilleur selon J_w

K-means pour le Clustering

Un algorithme célèbre : K-means

1. On considère que l'on dispose de centres de gravité $\boldsymbol{\mu}_k, k = 1, \cdots, K$
2. On affecte chaque $\boldsymbol{x}_i$ au cluster le plus proche $\mathcal{C}_\ell$:

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

3. On recalcule $\boldsymbol{\mu}_k$ pour chaque $\mathcal{C}_k, k = 1, \cdots, K$
4. On répète jusqu'à atteindre la convergence

Algorithme K-means

- Initialiser les centres de gravité $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K$
- Répéter :
  - Affecter chaque point au cluster le plus proche :

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

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

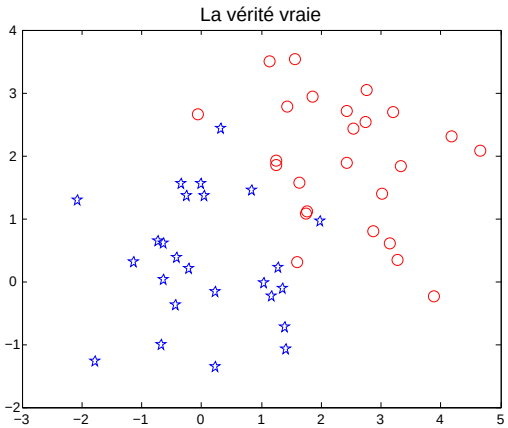
$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i \in \mathcal{C}_k} \boldsymbol{x}_i \quad \text{où} \quad N_k = \text{card}(\mathcal{C}_k)$$

- Jusqu'à ce que $\| \Delta\boldsymbol{\mu} \| > \epsilon$ ou $\|J_w\| > \epsilon_2$

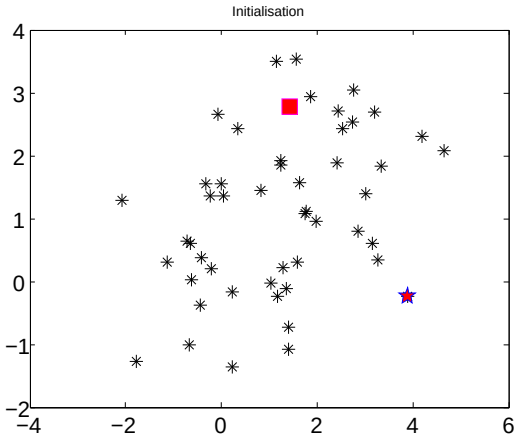
K-Means : illustration

Clustering avec $K = 2$ clusters

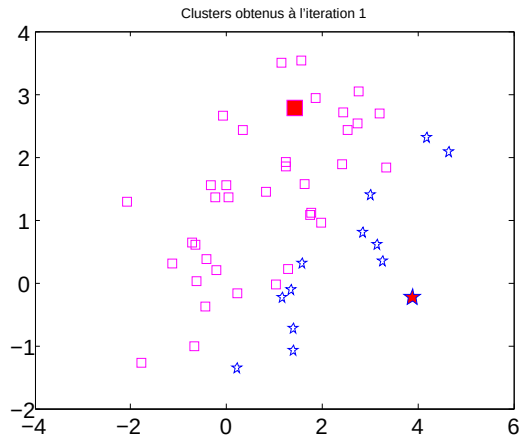
Données



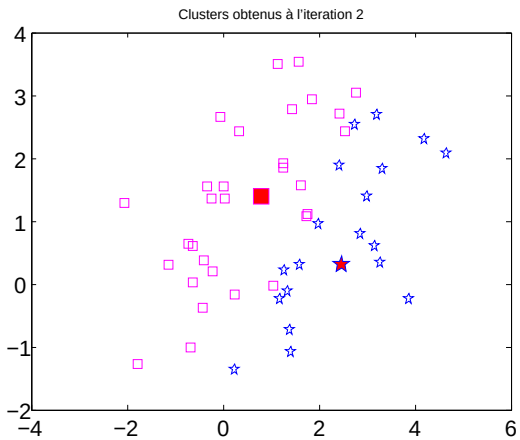
Initialisation



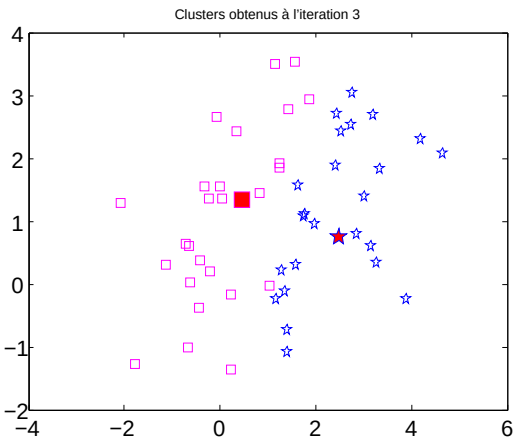
Itération 1



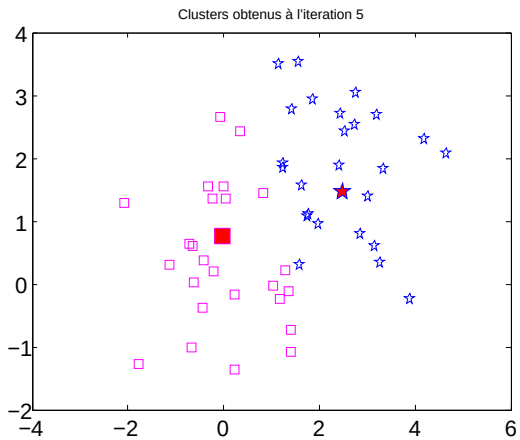
Itération 2



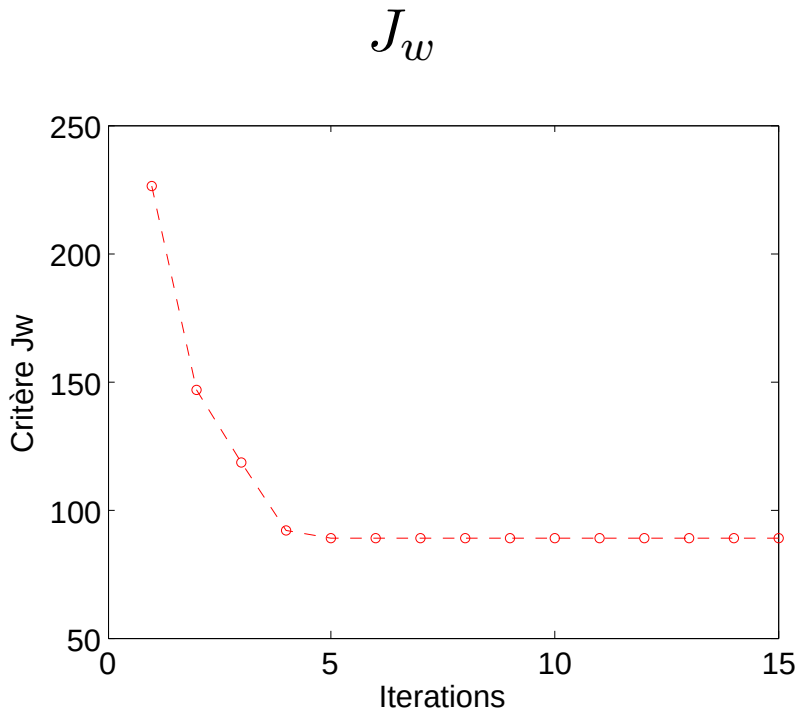
Itération 3



Itération 5

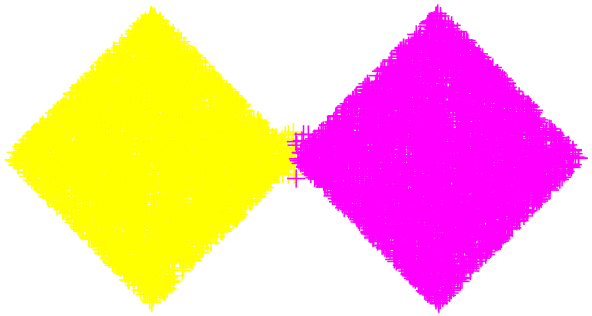


K-Means : J_w

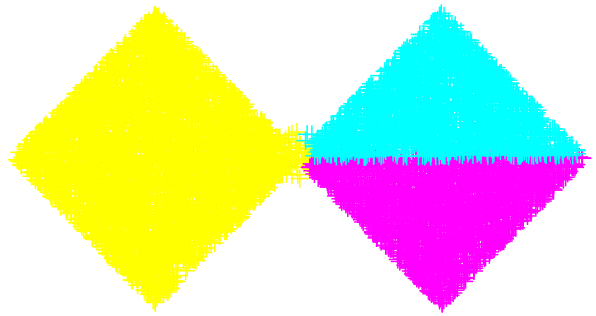


K-Means : exemple

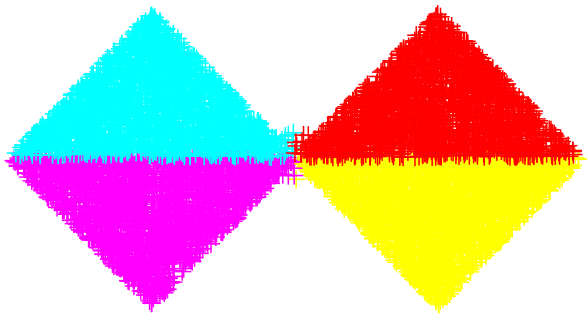
K = 2



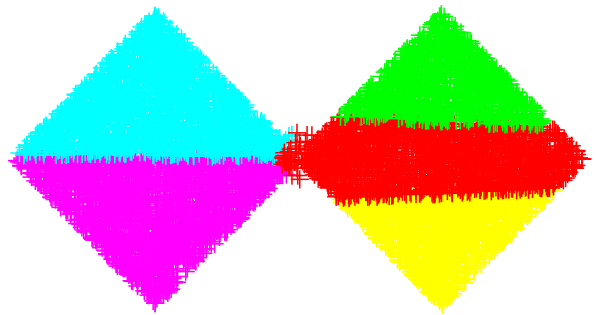
K = 3



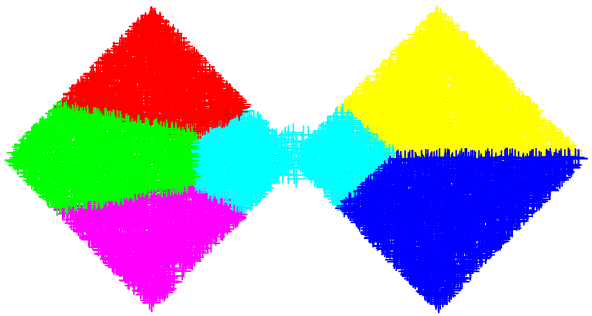
K = 4



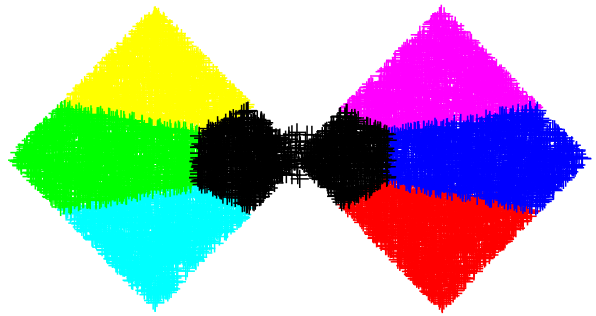
K = 5



K = 6

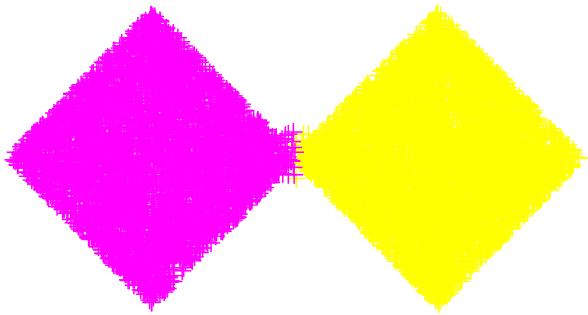


K = 7

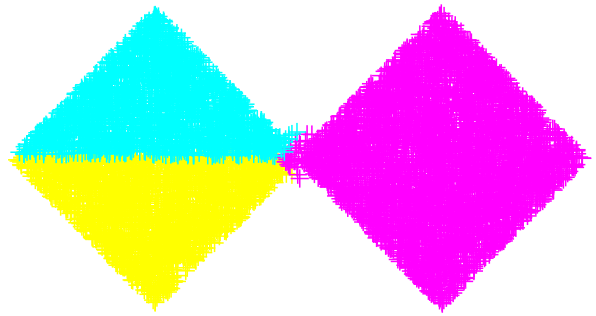


K-Means : exemple

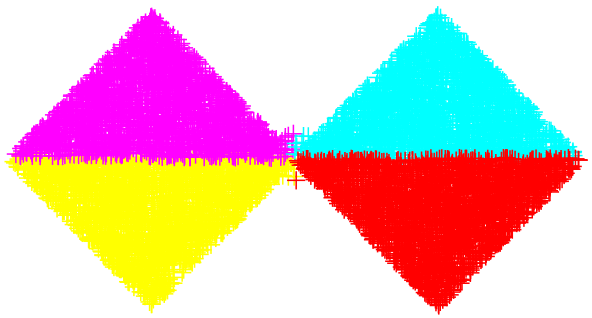
K = 2



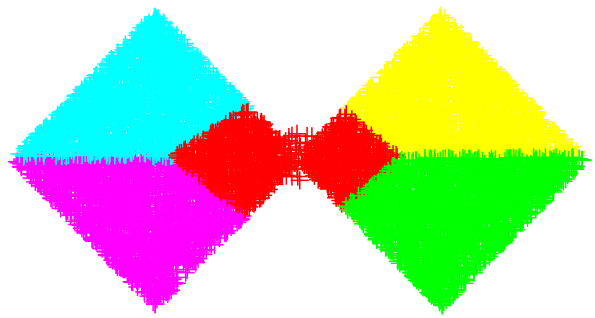
K = 3



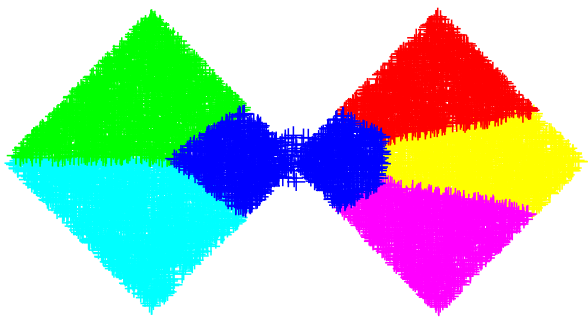
K = 4



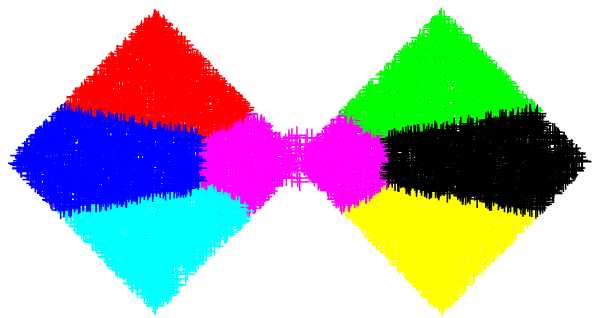
K = 5



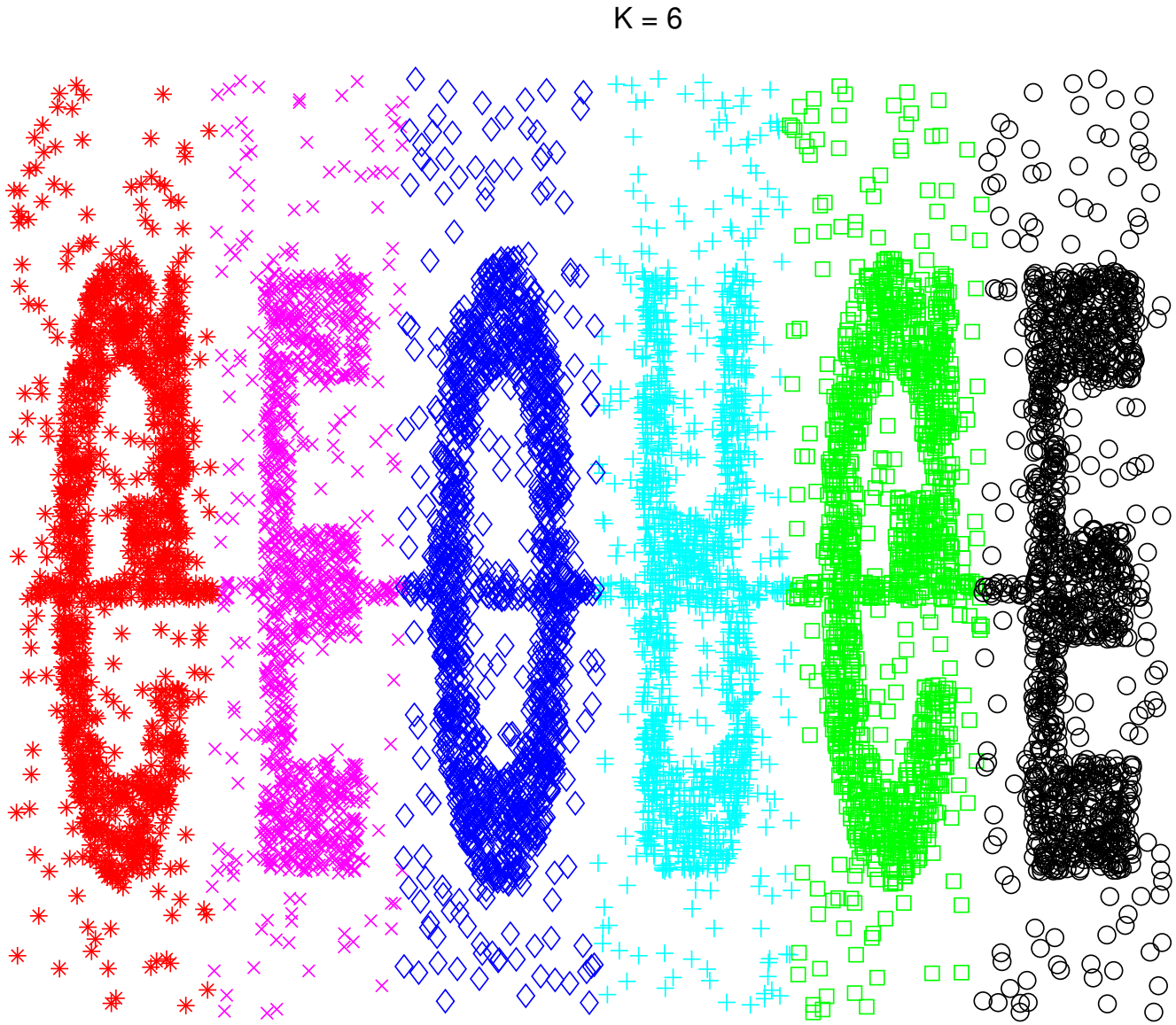
K = 6



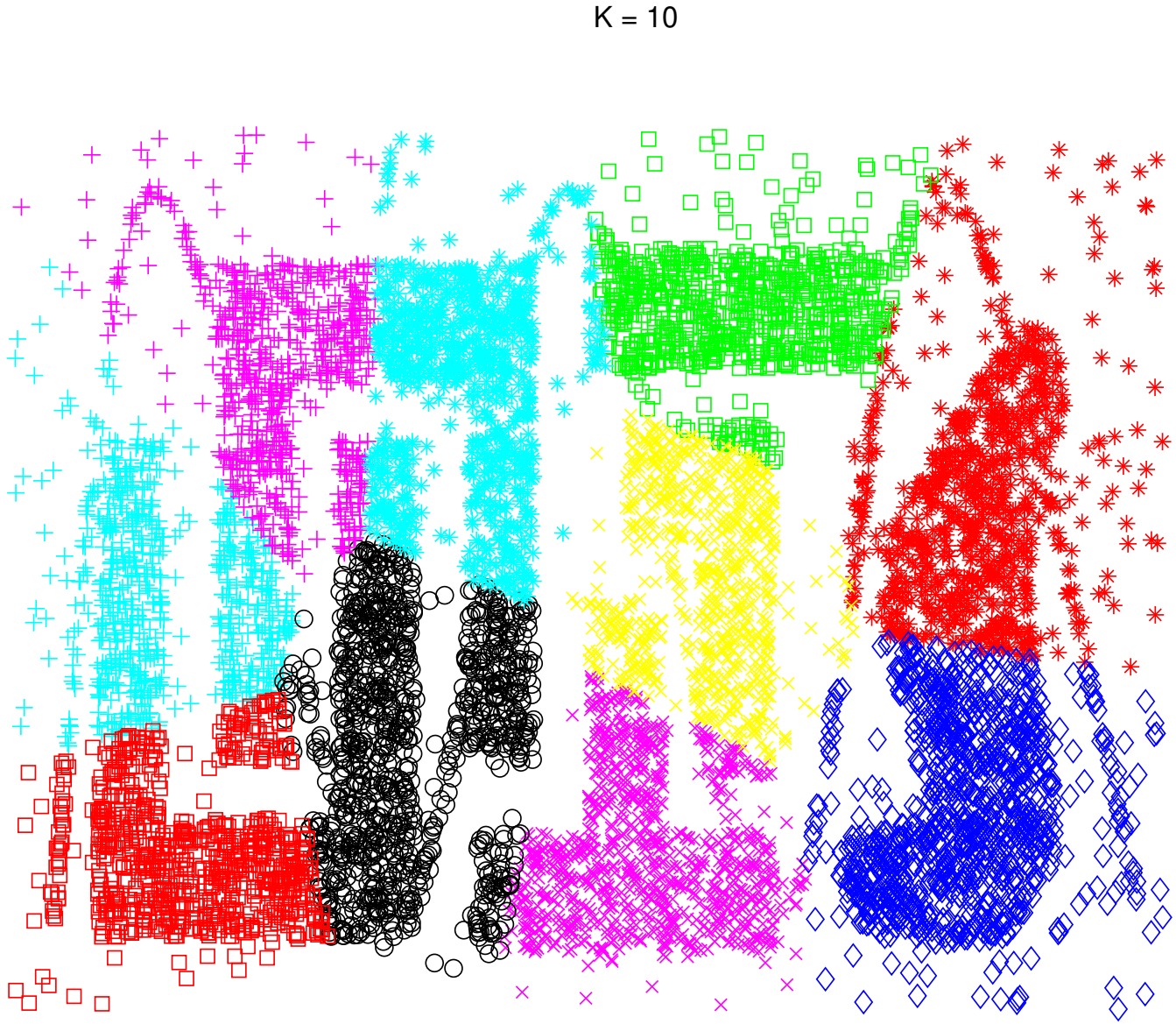
K = 7



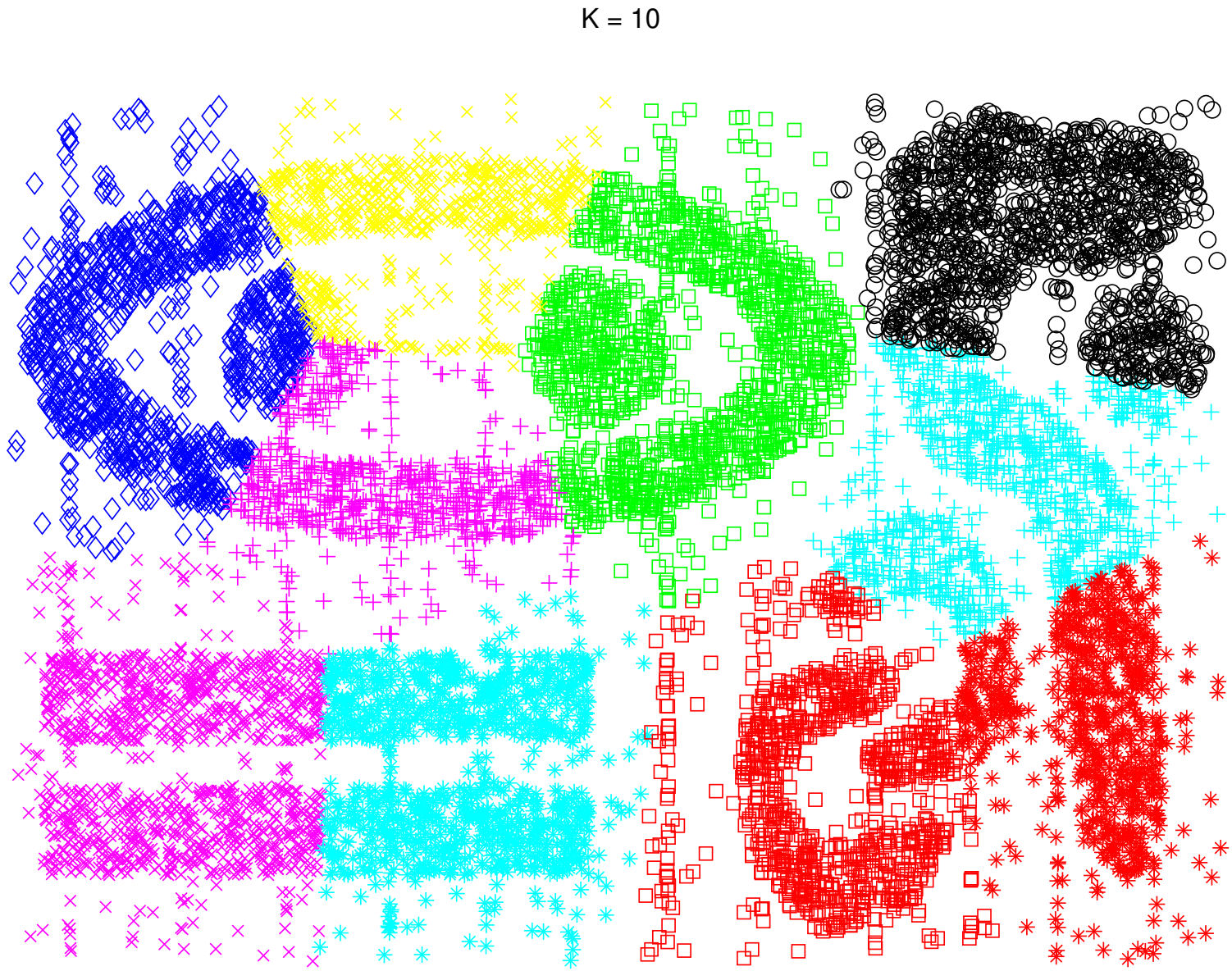
K-Means : exemple



K-Means : exemple



K-Means : exemple



K-Means : Quantification de couleurs

Passage de 89944 à 32 couleurs.

K-Means : Discussion

Observations sur K-Means

- J_w diminue à chaque itération
- Il converge vers un minimum local de J_w
- Initialisation des $\boldsymbol{\mu}_k$:
  - Aléatoirement dans le domaine des $\boldsymbol{x}_i$
  - Sélection aléatoire de K points parmi $\boldsymbol{X}$
- Différentes initialisations peuvent conduire à des clusterings différents
- scikit-learn : teste plusieurs initialisations et sélectionne celle avec le plus petit J_w

Implémentation de K-means

```
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. Par défaut, KMeans utilise plusieurs initialisations
7  clustering = KMeans(n_clusters=3)
8  clustering.fit(X) # calcule les centroïdes
9  clusters = clustering.predict(X) # assigne chaque point à un cluster
10 # affichage des points avec la couleur du cluster
11 plt.scatter(X[:, 0], X[:, 1],c=clusters)
```

- Documentation sklearn
- Utiliser KMeans

K-Means : conclusion

Avantages

- Algorithme intuitif et facile à comprendre
- Très efficace en pratique
- Temps de calcul réduit : Mini Batch KMeans
- Des solutions existent pour le problème d'initialisation

Inconvénients

- Il faut définir le nombre de clusters K
- Ne permet que des clusters convexes

Autres méthodes de Clustering

Clustering hiérarchique

Fusion itérative de clusters

- Nécessite de définir une distance entre clusters
- Le nombre de clusters peut être choisi a posteriori
- sklearn

DBSCAN

Identification et expansion des zones de forte densité

- Le nombre de clusters est déduit des données
- Certains points peuvent ne faire partie d'aucun cluster
- sklearn

et d'autres encore.

Unsupervised Learning
Dimensionnality Reduction

Analyse en Composantes Principales (ACP) I

Principe

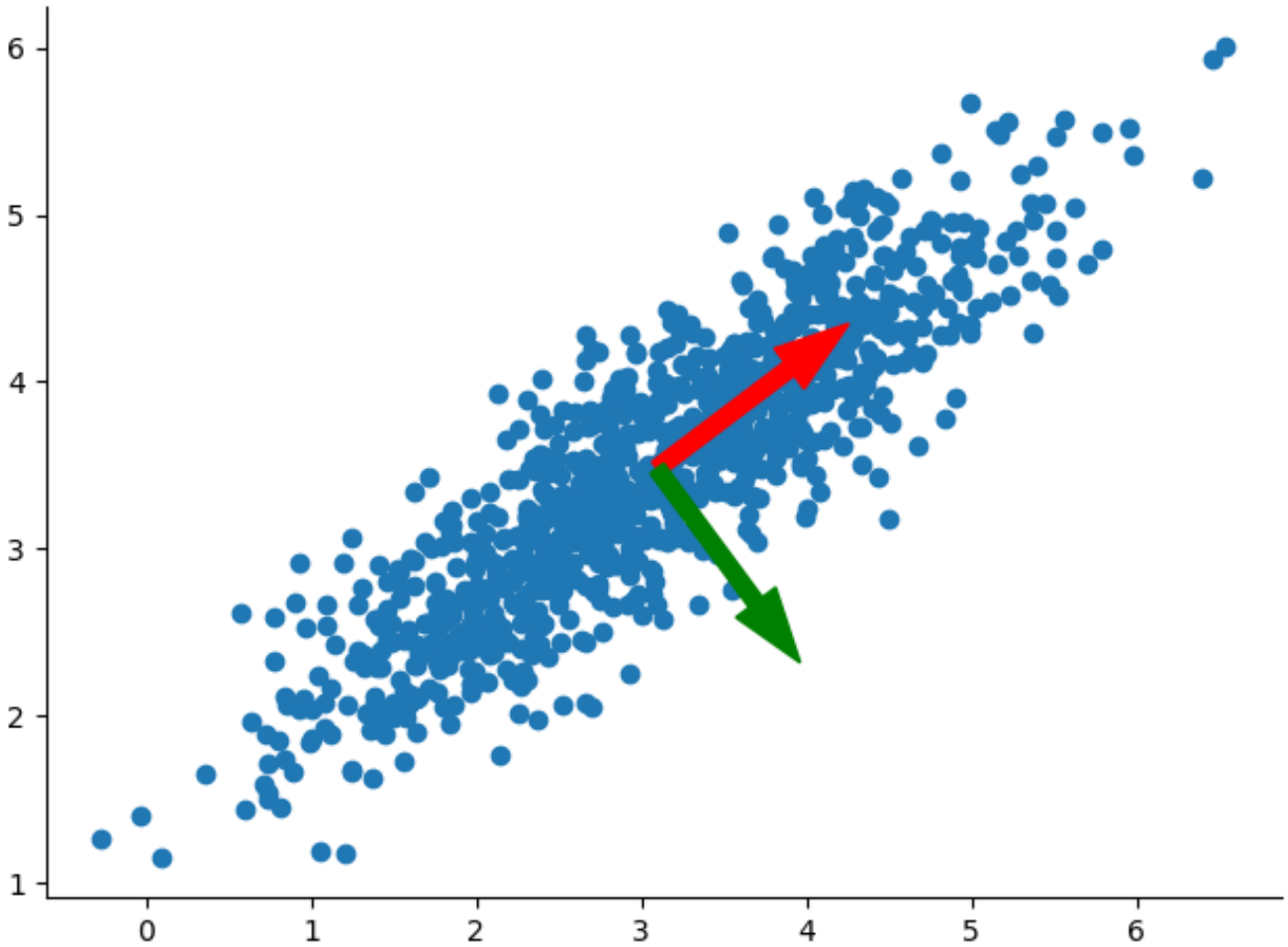
Trouver une nouvelle base dans laquelle les premières dimensions capturent l'essentiel de l'information.

- Information $\Rightarrow$ Variance, dispersion des données.
- Les premières composantes contiendront l'information.
- Les dernières composantes contiendront le bruit.
- Il faut choisir le nombre de composantes à conserver :
  - 2 ou 3 pour la visualisation,
  - pour la compression de données : autant que nécessaire pour garder une qualité acceptable,
  - pour la transmission : aussi peu que possible pour limiter le volume de données,
  - pour l'analyse : assez pour garder l'information et éliminer le bruit.

Analyse en Composantes Principales (ACP) II

ACP sur un nuage de points

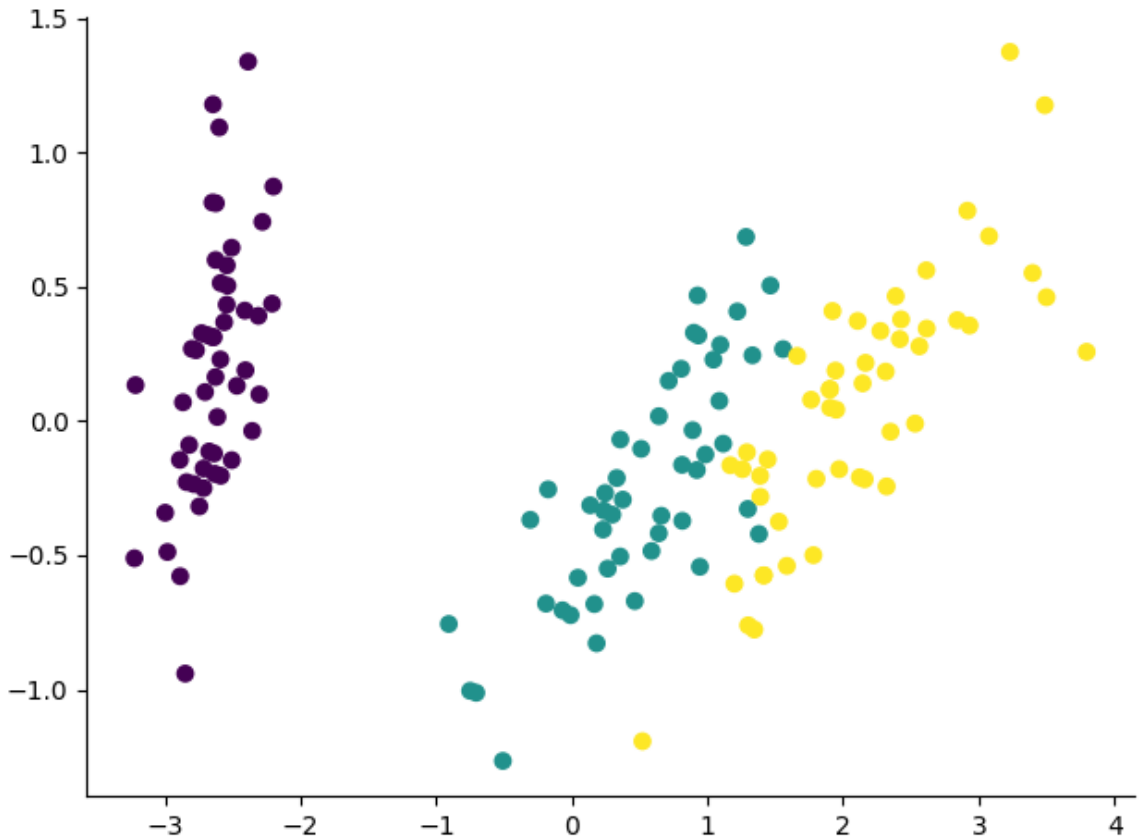
Identifier les axes sur lesquels l'information est répartie.



Analyse en Composantes Principales (ACP) III

ACP sur le jeu de données Iris

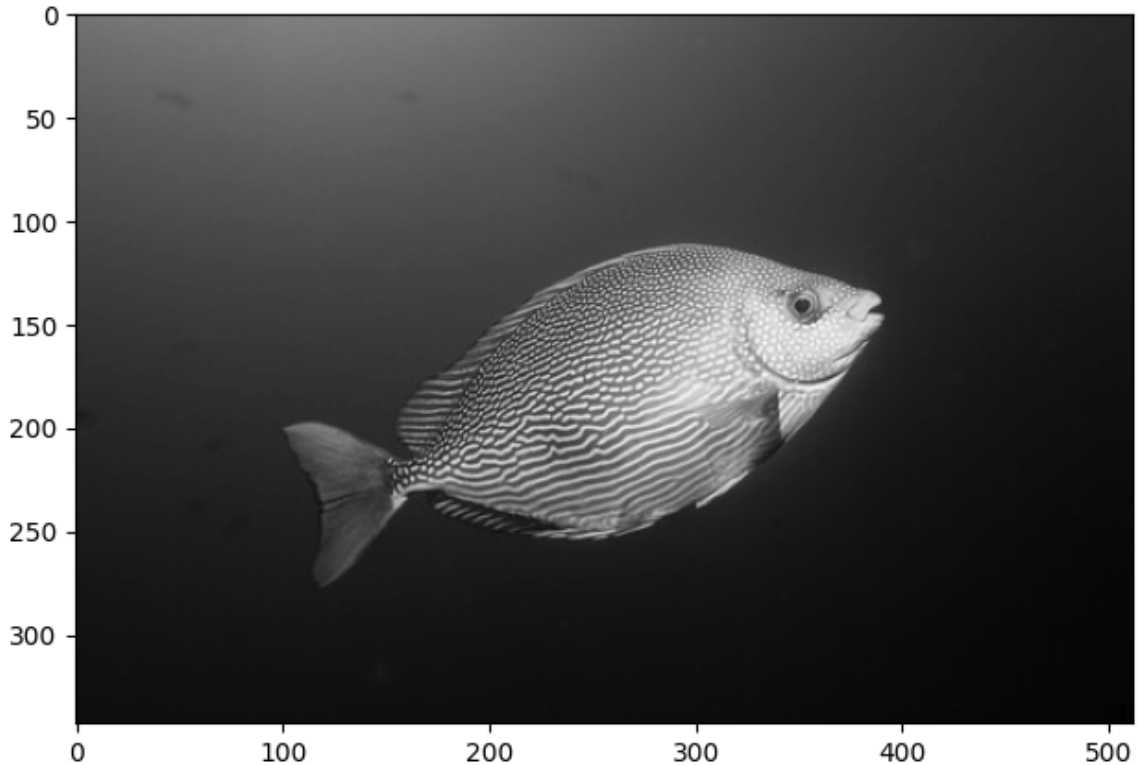
Le jeu de données Iris contient 150 fleurs décrites par 4 variables.
Il est difficile de le visualiser sans réduction de dimension.



Analyse en Composantes Principales (ACP) IV

ACP pour les images

L'image originale fait 343×512, soit 175616 pixels.

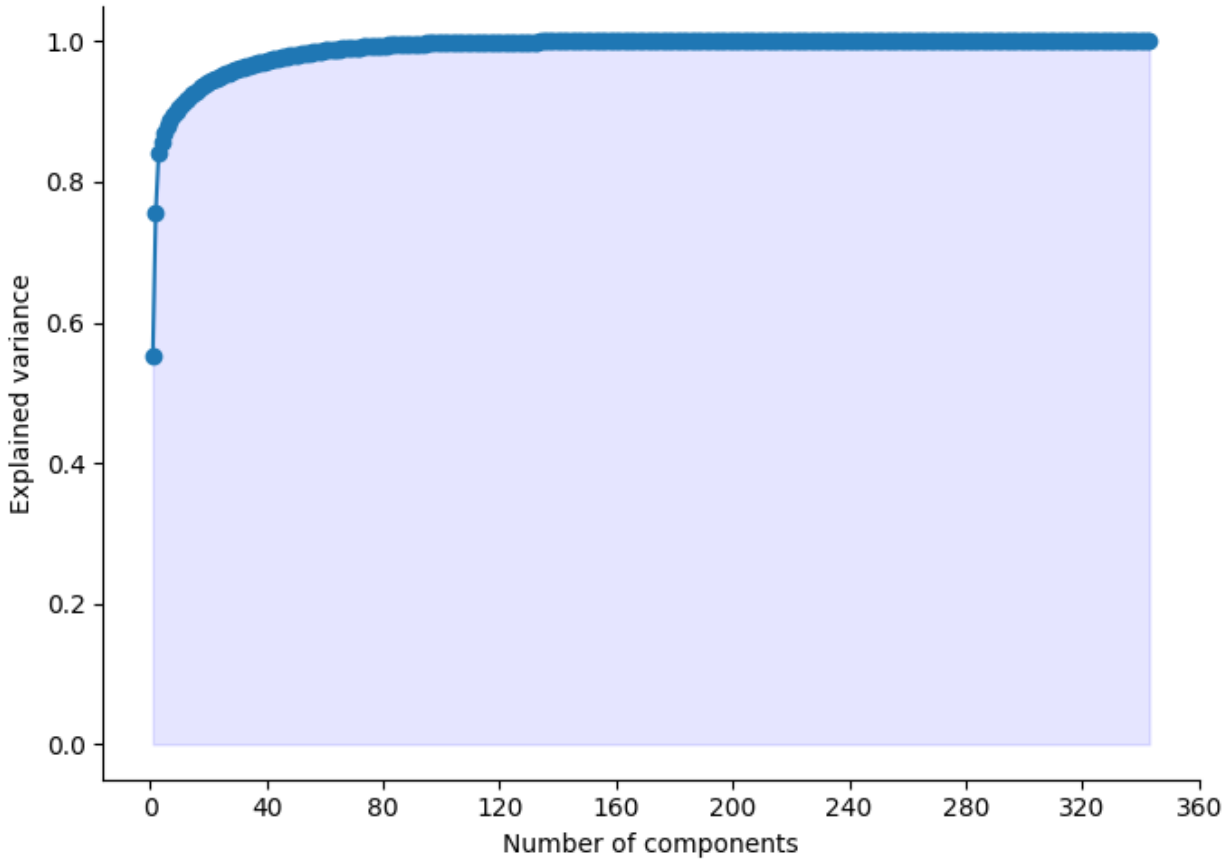


Analyse en Composantes Principales (ACP) V

Quantité d'information encodée

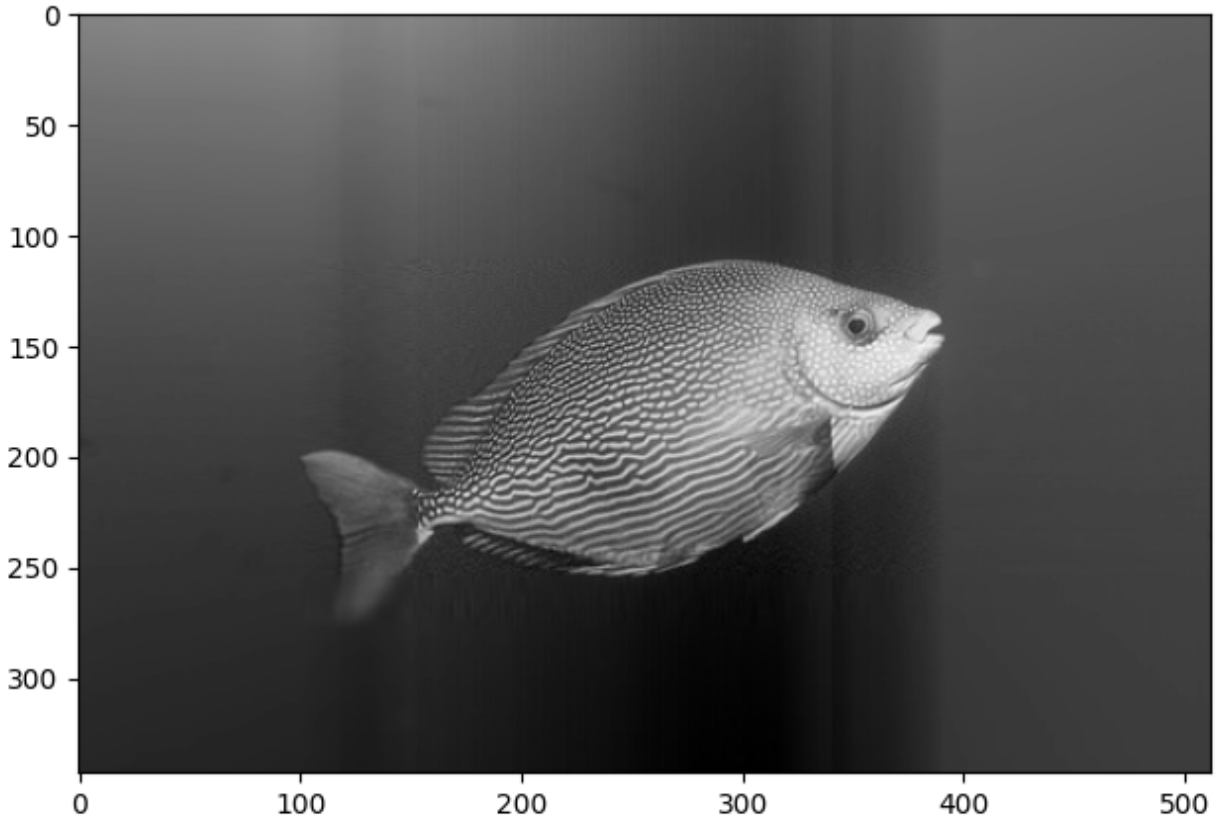
Analyse en Composantes Principales (ACP) VI

En utilisant 50 composantes, on conserve la majeure partie de l'information.



Analyse en Composantes Principales (ACP) VII

Reconstruction de l'image en utilisant deux fois moins de données :



Implémentation de l'ACP

```python
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
X,y = load_iris(return_X_y=True) # charger les données
pca = PCA(n_components=2) # définir l'ACP avec 2 composantes à conserve
pca.fit(X) # apprentissage de la nouvelle base de représentation
# calcul de la représentation des données dans cette base
X_pca = pca.transform(X)
# affichage des données dans la nouvelle base (2 dimensions ici)
plt.plot(X_pca[:,0], X_pca[:,1],'o')
# reconstruction des données compressées dans la base originale
X_reconstruct = pca.inverse_transform(X_pca)
```

- Documentation complète
- Exemple sklearn

Extensions aux représentations non linéaires

Représentations non linéaires

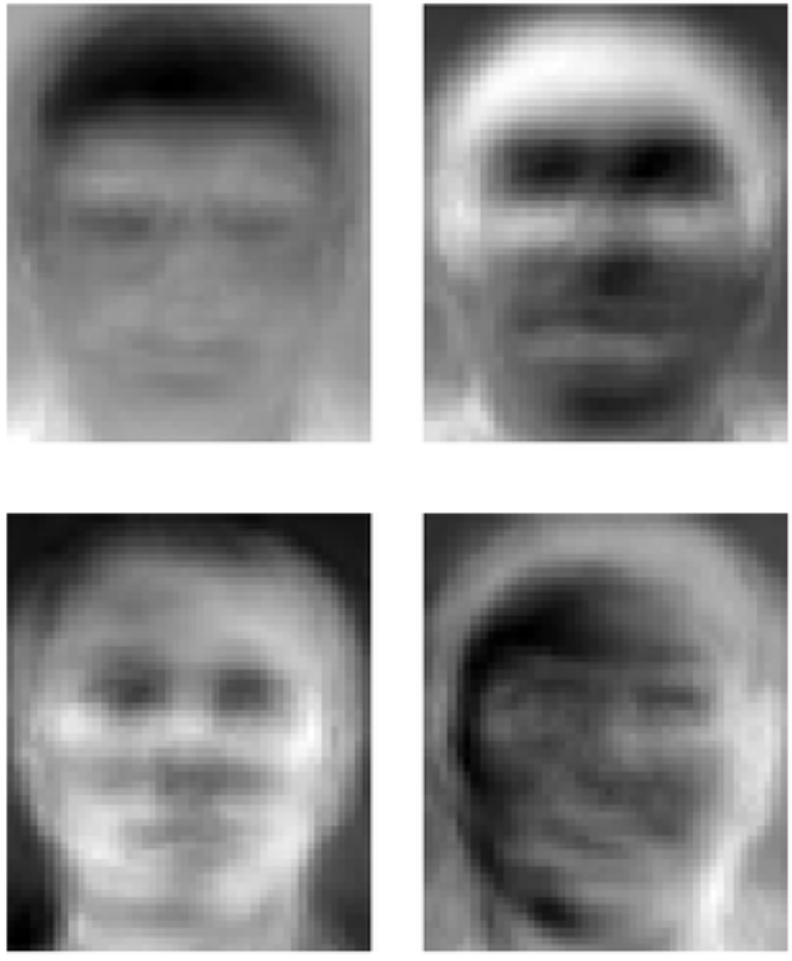
L'ACP est linéaire par nature : elle est donc limitée à des transformations simples.
Ce principe a été étendu à des transformations non linéaires :

- **KernelPCA** : abstraction du produit scalaire vers des fonctions non linéaires,
- **t-SNE** : crée un espace où les relations locales sont conservées,
- **UMAP** : plus complexe . . . voir ici.

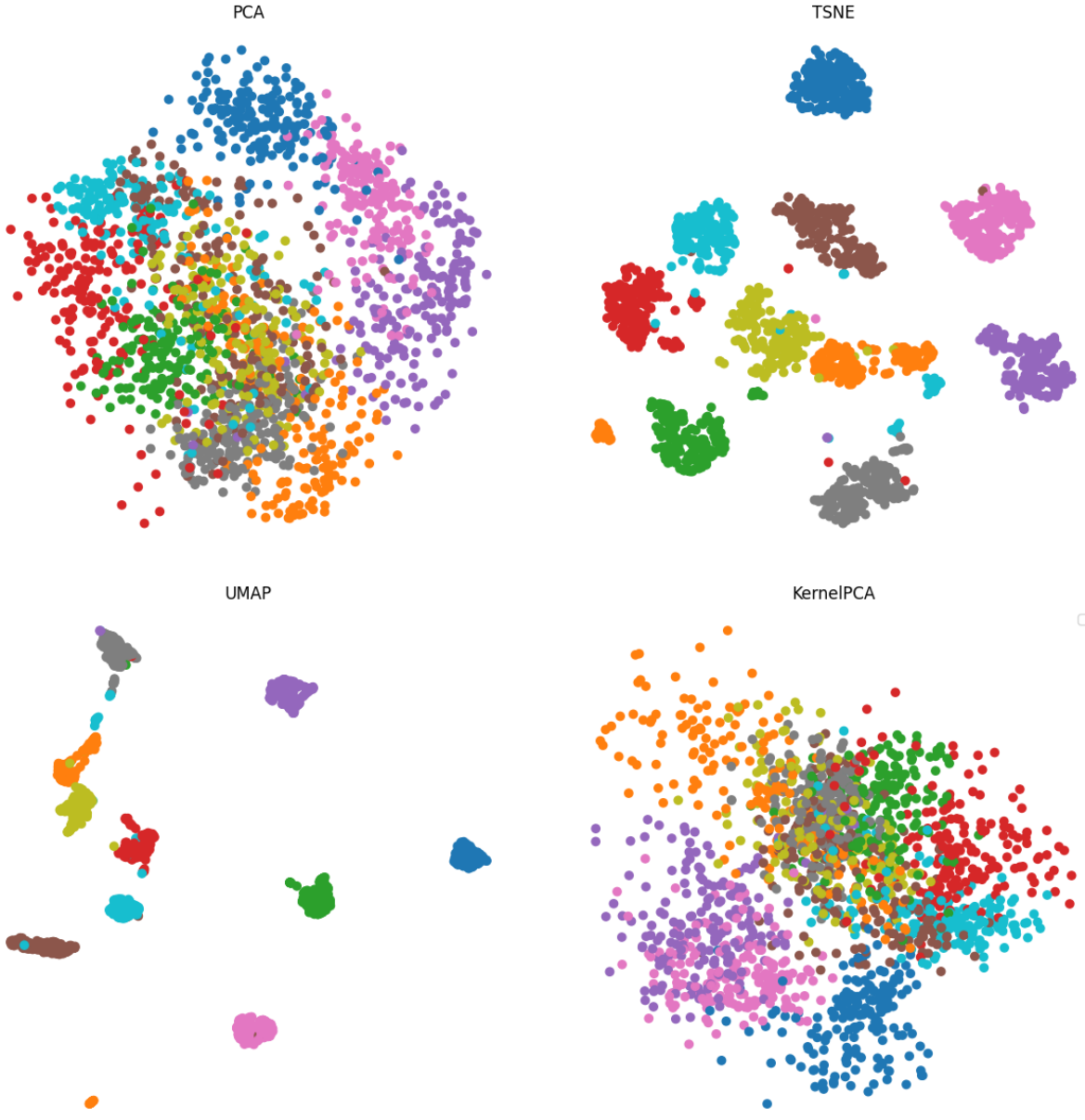
Comparaison illustrative :
https://projector.tensorflow.org/

Exemples de réductions non linéaires de dimension I

Reconnaissance faciale : EigenFaces



Exemples de réductions non linéaires de dimension II



Réduction de dimension — Conclusion

Avantages

- Utilisée pour visualiser les données
- Compression des données :
  - → Réduction du bruit
  - → Accélération des calculs
  - → Création d'un espace latent

Inconvénients

- Nécessité de définir manuellement comment comparer les données
- L'ACP est limitée aux transformations linéaires
- Peut être coûteuse en calcul

Apprentissage non supervisé — Conclusion

Nous sommes limités et "aveugles" avec l'apprentissage non supervisé 😔

Mais :

- Cela permet de comprendre la structure des données
- Dans le monde réel, la majorité des données ne sont pas annotées
- L'apprentissage non supervisé peut servir d'étape de prétraitement pour des tâches plus complexes

Prochaine étape : exploiter les propriétés des données !