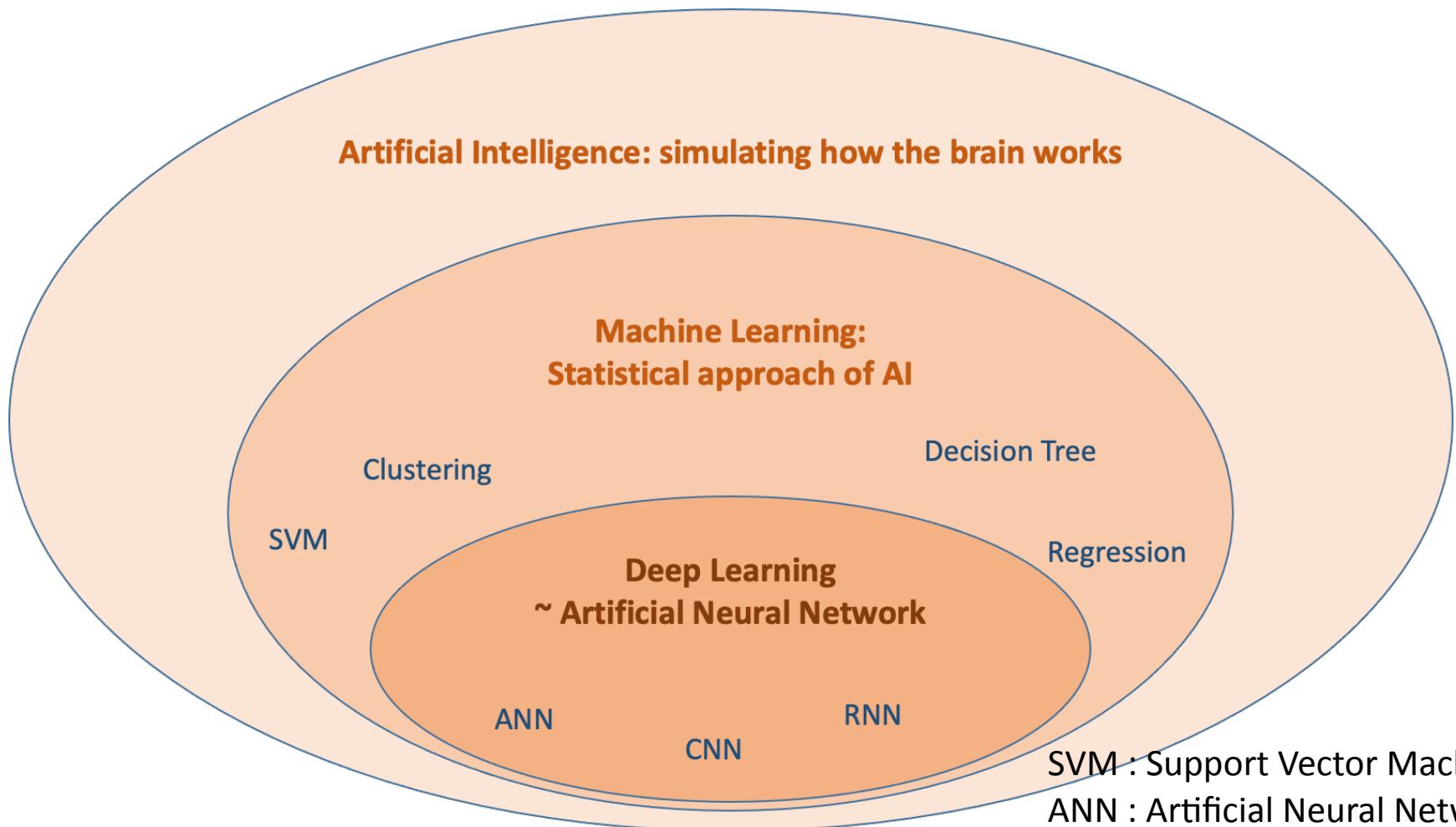

Machine Learning non-supervisé

Entrainement non supervisé



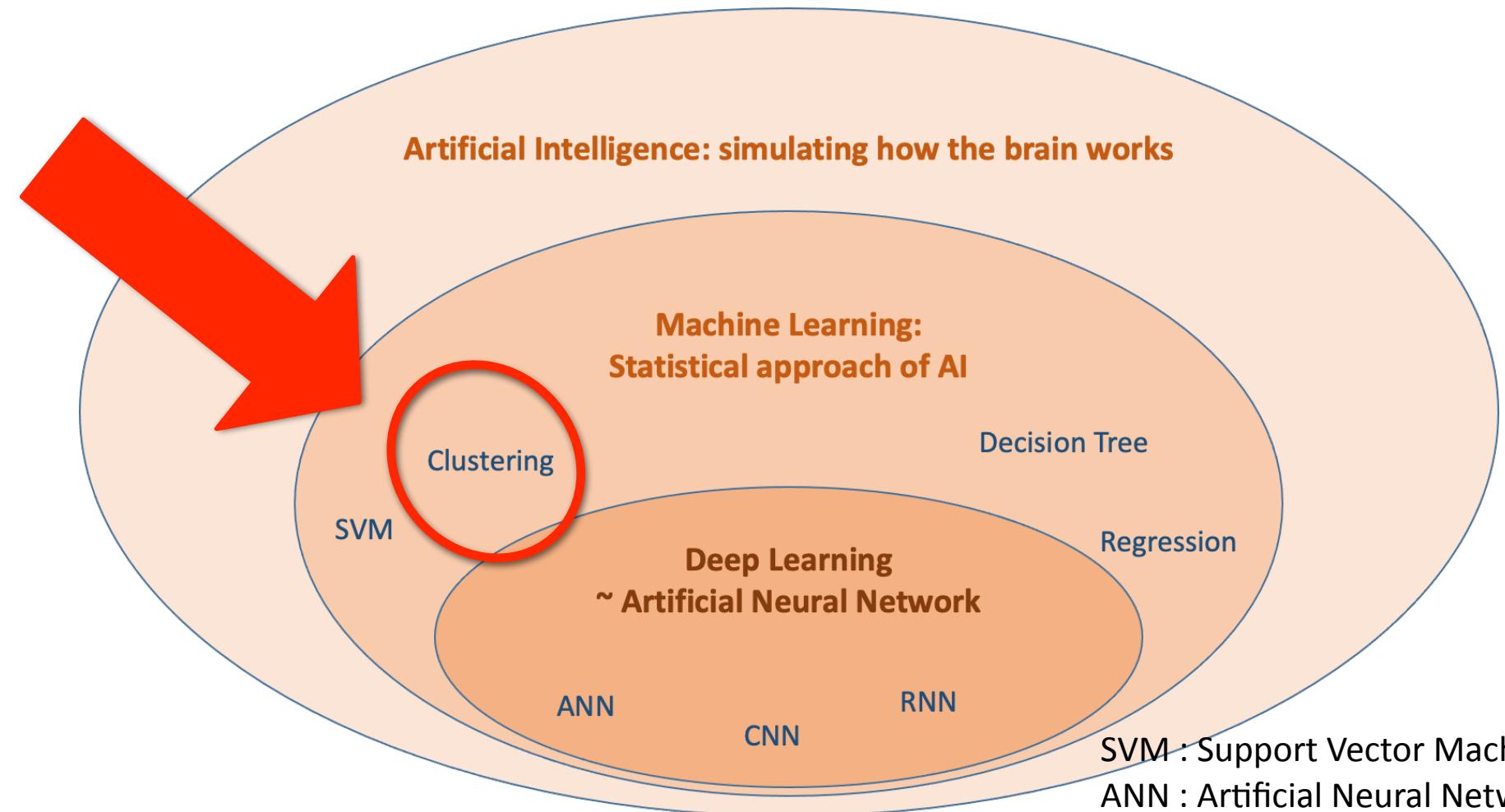
SVM : Support Vector Machine

ANN : Artificial Neural Network

CNN : Convolution NN

RNN : Recurrent NN

Entrainement non supervisé



SVM : Support Vector Machine
ANN : Artificial Neural Network
CNN : Convolution NN
RNN : Recurrent NN

Entrainement non supervisé

Non Supervisé : Aucun label / information sur les données d'entrés



$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad (m) \text{ Vecteurs de Dimension (n)}$$

$y^{(i)}$ peut être de n'importe quel type (nombre, vecteur, label...)

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

Un mot sur scikit-learn



<https://scikit-learn.org/>

- Bibliothèque python pour le Machine learning
- Inclue :
 - Processing ➔ cf demain
 - Entrainement supervisé ➔ vu ce matin
 - Entrainement non supervisé ➔ maintenant
 - Visualisation de donnée
- Principalement maintenue par l'Inria (sur le plateau)

Un mot sur scikit-learn

```
import sklearn  
  
# ou de manière générale  
  
from sklearn.submodule1 import AlgoA  
from sklearn.submodule2 import AlgoB
```

```
mon_algo = AlgorithmeA(param1, param2)
```

Création de l'algorithme

```
mon_algo.fit(X_train)
```

Entrainement sur les
donnée d'entraînement

```
X_trans = mon_algo.transform(X_test)
```

Application sur les
donnée de test

En non supervisé $X_{train} = X_{test}$

Deux Familles d'algorithme non supervisé :

Réduction de dimension :

Partant de donnée de dimension N , obtient des données en dimension M sans perdre (trop) d'information (avec $M < N$)

Pourquoi :

- Compresse l'information
- Élimine les paramètres inutiles
- Facilite la visualisation
- Préparation des données pour le ML et le DL

Instinctivement :

- Passage en coord sphérique/cylindrique
- Sélection des colonnes pertinentes d'une base de donnée

Clustering (Partitionnement) :

Partant de donnée de dimension N , étiquette ensemble les éléments proche

Pourquoi :

- Gagne de l'information sur les données d'entrée
- Classification / segmentation
- Préparation des données pour le ML et le DL

Instinctivement :

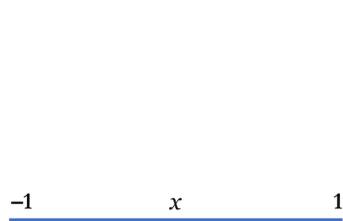
- Groupement d'individus par caractéristiques (plantes, individus...)

Réduction de dimension

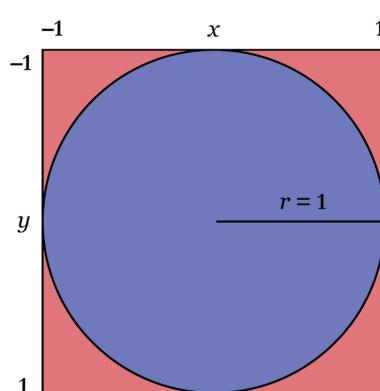
La « Malédiction de la dimension »

Chaque dimension que l'on ajoute à un espace de paramètre fait grandir le volume de manière exponentielle.

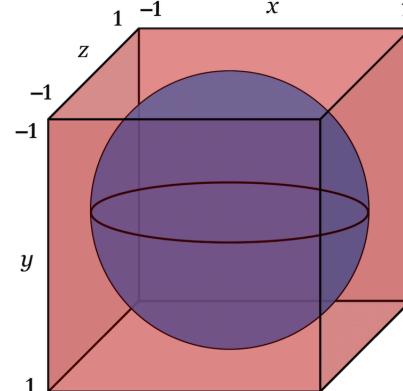
Étudier de manière homogène l'ensemble du volume ainsi créé devient rapidement impossible quand la dimension augmente.



1 dimension ($D = 1$)



2 dimensions ($D = 2$)



3 dimensions ($D = 3$)

Points uniformément distribués $x^n \in [0,1]$

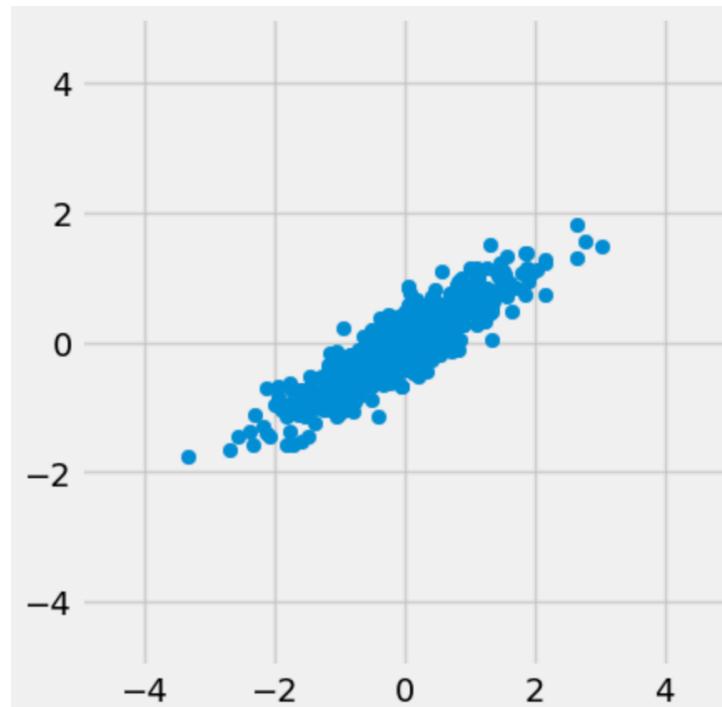
- 1D : 100 points
- 3D : 1000000 points
- 10D : $100^{10} = 10^{20}$ points

Analyse en composantes principales

En anglais : principal component analysis (PCA)

Objectif : détermination des axes favorables qui maximise la variance de nos données

➔ Trouve le premier axe avec la variance la plus large

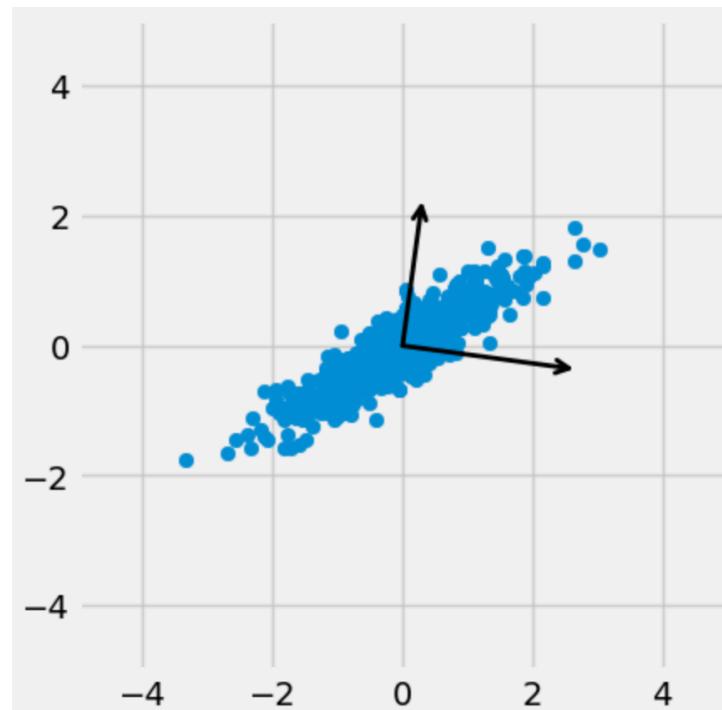


Analyse en composantes principales

En anglais : principal component analysis (PCA)

Objectif : détermination des axes favorables qui maximise la variance de nos données

- ➔ Trouve le premier axe avec la variance la plus large
- ➔ Regarde les axes non corrélés (perpendiculaire) et trouve celui avec la variance la plus large

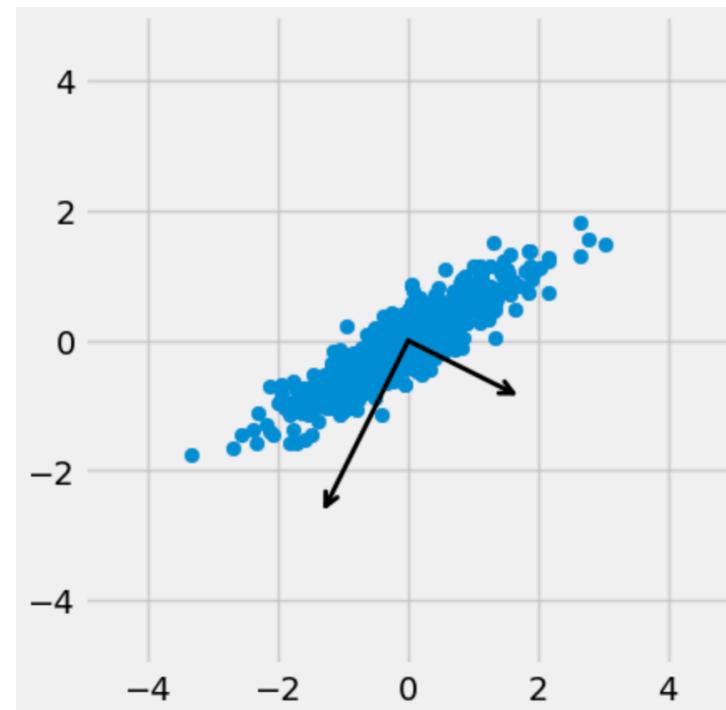


Analyse en composantes principales

En anglais : principal component analysis (PCA)

Objectif : détermination des axes favorables qui maximise la variance de nos données

- ➔ Trouve le premier axe avec la variance la plus large
- ➔ Regarde les axes non corrélés (perpendiculaire) et trouve celui avec la variance la plus large
- ➔ Continue jusqu'à avoir fait tous les axes

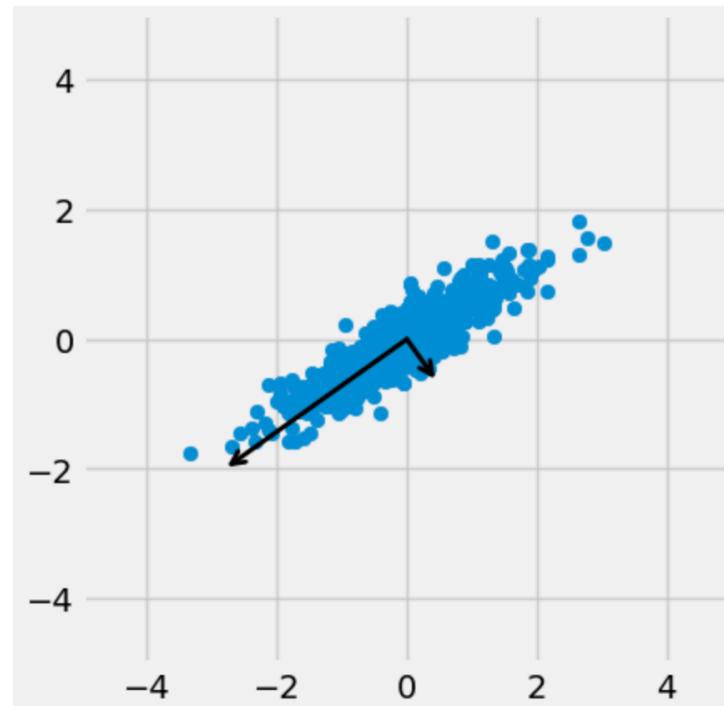


Analyse en composantes principales

En anglais : principal component analysis (PCA)

Objectif : détermination des axes favorables qui maximise la variance de nos données

- ➔ Trouve le premier axe avec la variance la plus large
- ➔ Regarde les axes non corrélés (perpendiculaire) et trouve celui avec la variance la plus large
- ➔ Continue jusqu'à avoir fait tous les axes

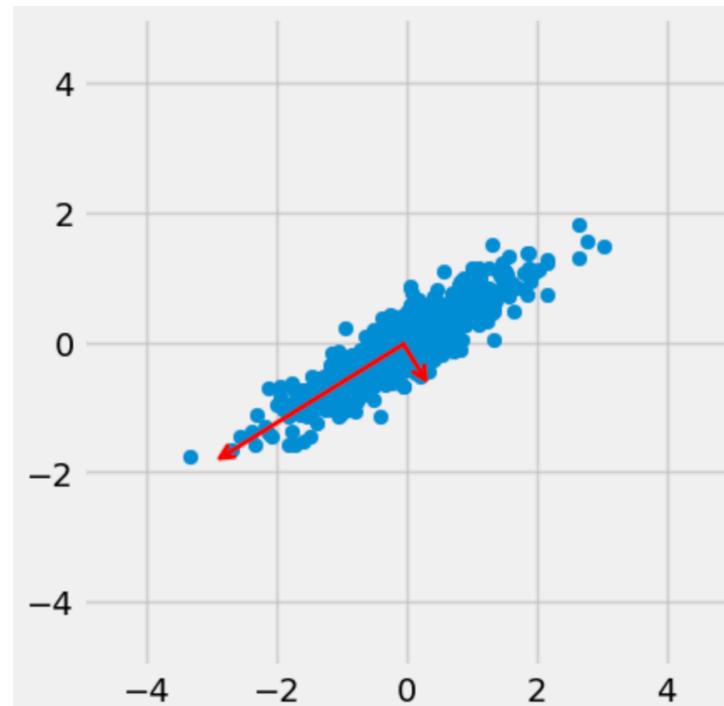


Analyse en composantes principales

En anglais : principal component analysis (PCA)

Objectif : détermination des axes favorables qui maximise la variance de nos données

- ➔ Trouve le premier axe avec la variance la plus large
- ➔ Regarde les axes non corrélés (perpendiculaire) et trouve celui avec la variance la plus large
- ➔ Continue jusqu'à avoir fait tous les axes



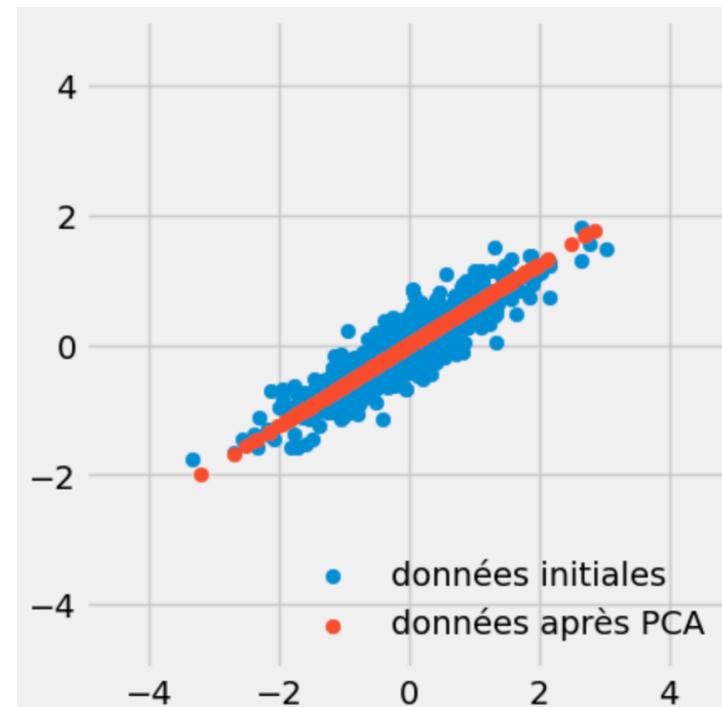
Analyse en composantes principales

En anglais : principal component analysis (PCA)

Objectif : détermination des axes favorables qui maximise la variance de nos données

- Trouve le premier axe avec la variance la plus large
- Regarde les axes non corrélés (perpendiculaire) et trouve celui avec la variance la plus large
- Continue jusqu'à avoir fait tous les axes

On peut ensuite supprimer les axes de faible variance en projetant nos données



Analyse en composantes principales

Mathématiquement :

Calcule la matrice de covariance des données initiale :

$$\begin{pmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_p) \\ \text{Cov}(X_2, X_1) & \ddots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_p, X_1) & \cdots & \cdots & \text{Var}(X_p) \end{pmatrix}$$



Les données d'entrée doivent être normalisées

À partir de celle-ci, on trouve les vecteurs et valeurs propres ➔
Correspondes aux directions principales et la variance associées

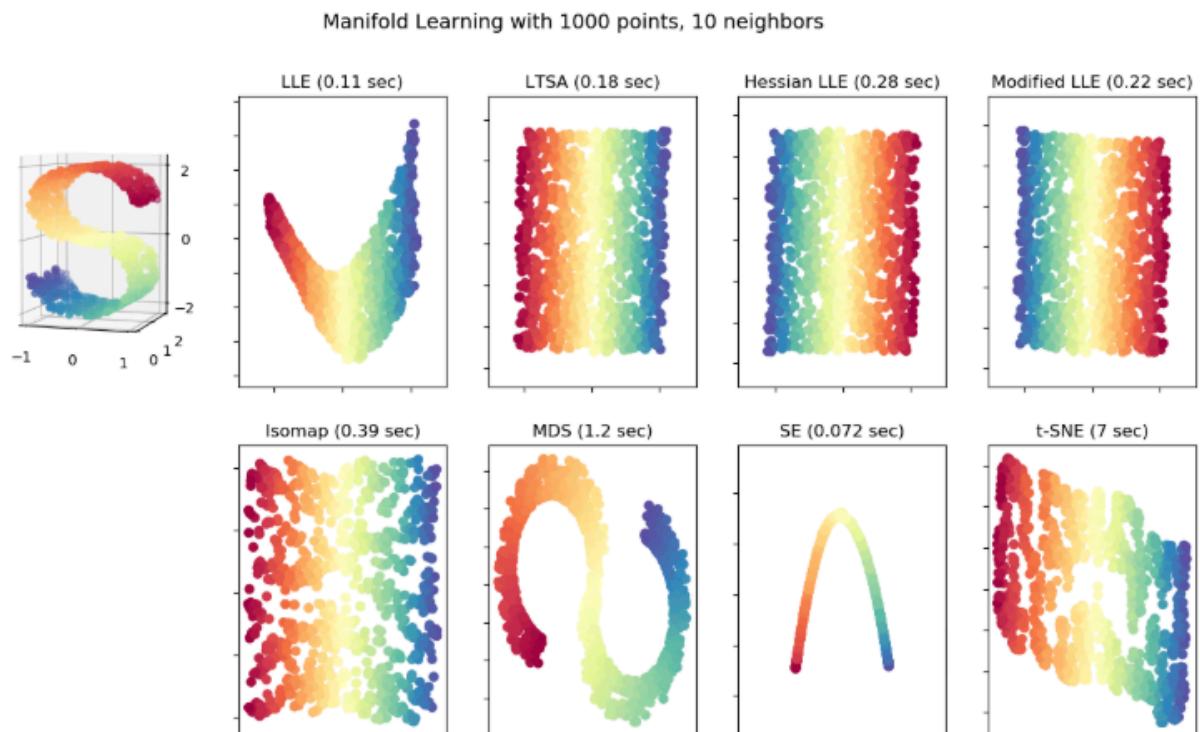
$$\text{Var}(X) = \text{Cov}(X, X) = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

$$\text{cov}(X, Y) = \frac{1}{n} \sum_{i=1}^n (x_i - E(X))(y_i - E(Y))$$

Manifold learning

La PCM est très rapide et efficace, mais elle ne fonctionne que les données sont séparables linéairement

Manifold learning : trouve une représentation 2D de donnée de haute dimension



Méthode très utile pour visualiser simplement des données de haute dimension

Algorithme t-SNE

t-SNE (t-distributed stochastic neighbour embedding)



Extension des algorithmes de SNE :

- Dans P : calcule la probabilité que i et j soit choisis comme voisin (choix de voisin basé sur une distribution gaussienne)
-

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/2\sigma_i^2)},$$

Algorithme t-SNE

t-SNE (t-distributed stochastic neighbor embedding)

Espace P : ND



t-SNE



Espace Q : 2D

Extension des algorithmes de SNE :

- Dans P : calcule la probabilité que i et j soit choisis comme voisin (choix de voisin basé sur une distribution gaussienne)
- Même chose dans Q

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}$$

Algorithme t-SNE

t-SNE (t-distributed stochastic neighbor embedding)



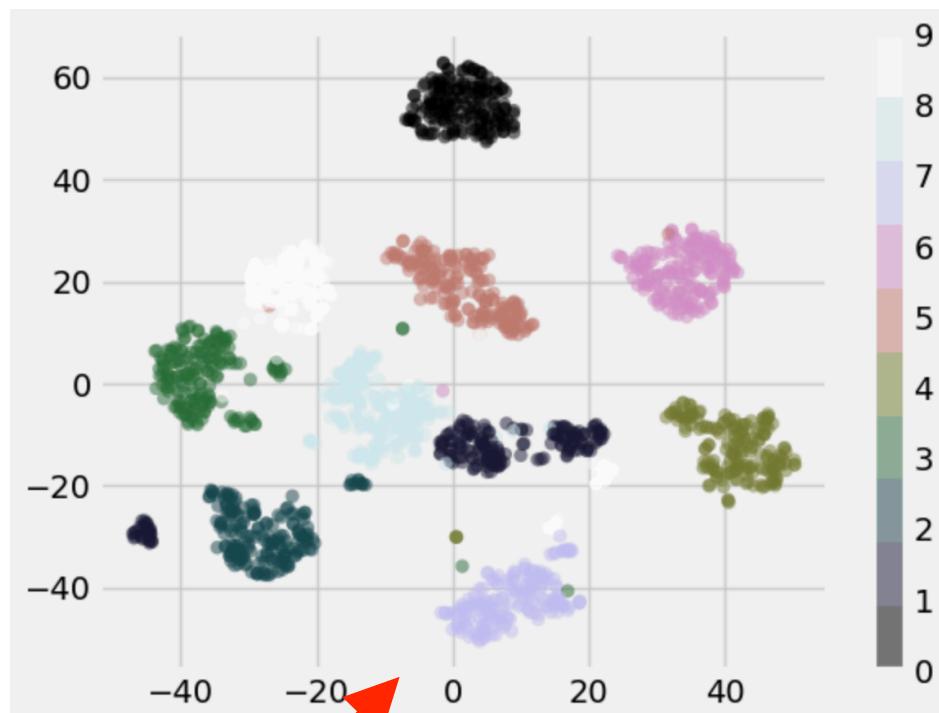
Extension des algorithmes de SNE :

- Dans P : calcule la probabilité que i et j soit choisis comme voisin (choix de voisin basé sur une distribution gaussienne)
- Même chose dans Q
- Si Q est une bonne représentation de P, alors $\frac{p_{j|i}}{q_{j|i}}$ tend vers 1
- On trouve les positions dans Q par descente de gradient

$$C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

Algorithme t-SNE

Exemple avec des chiffres



Le clustering

Objectif :

Regrouper ensemble les éléments similaires d'un jeu de donnée :

- Identifications de patterns
- Préparation des données

Algorithmes :

Énormément d'algorithmes différents (12 dans scikit-learn) :

- K-Means
- Gaussian Mixture
- DBSCAN

K means

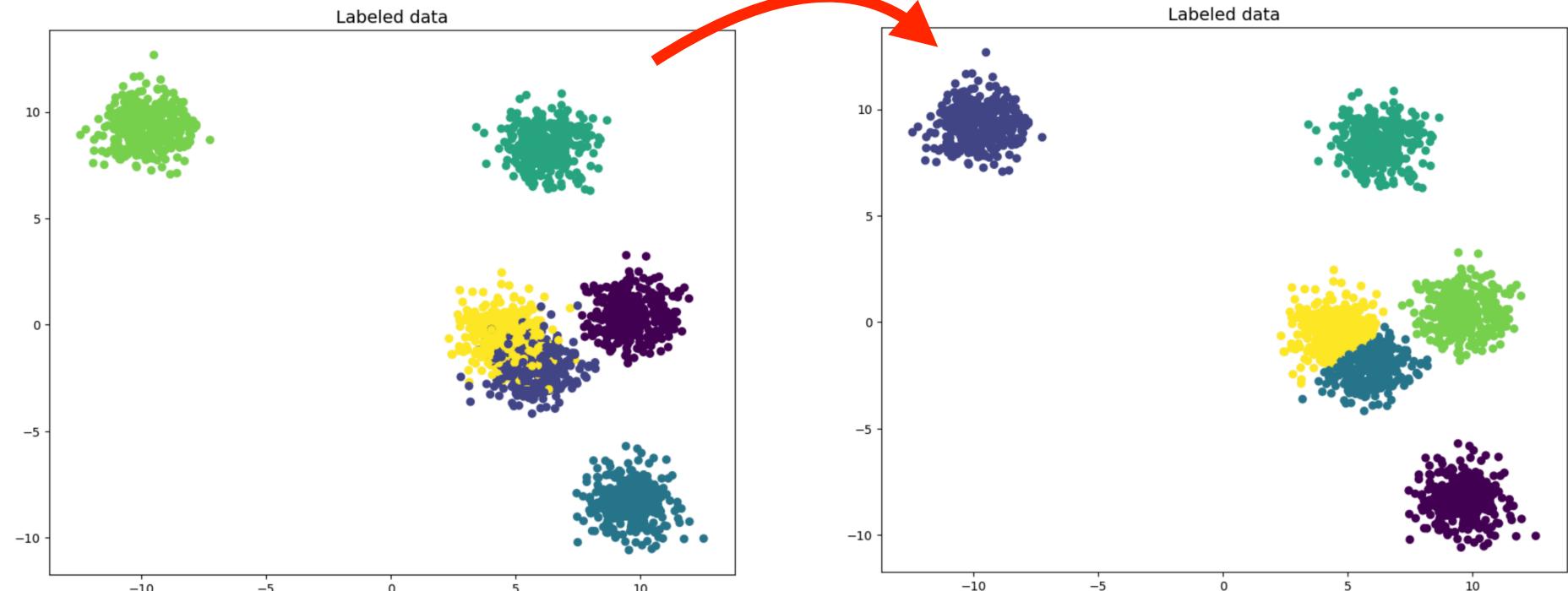
Divise les N données en **K clusters**

Extrêmement rapide et efficace

Nombreuse application

Spécifié initialement par l'utilisateur !

K = 6



K means

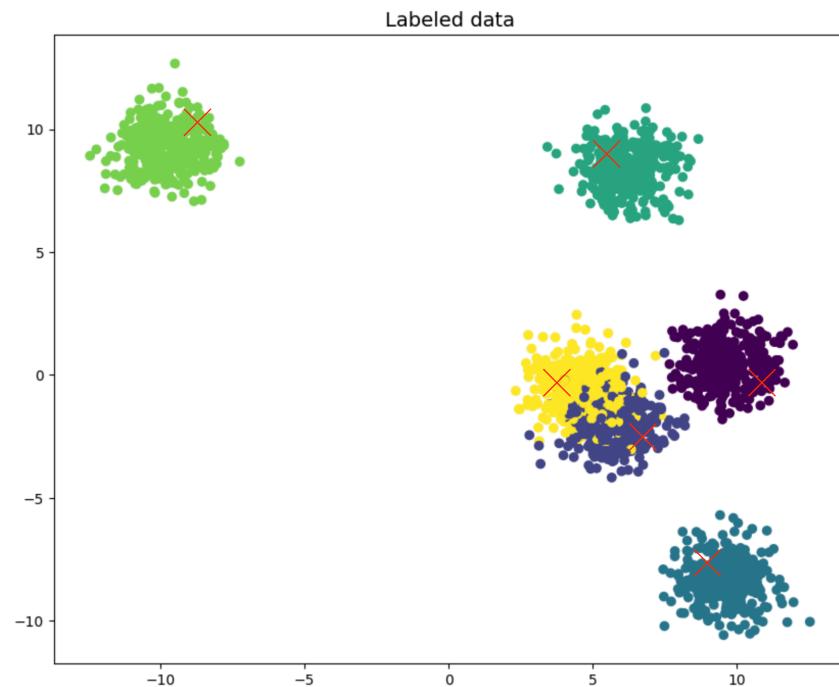
Comment ça marche ?

L'algorithme recherche K centroids minimisant l'inertie du système

$$\sum_{i=0}^n \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

Trois étapes :

- Sélection de la position initiale des centroids (K points aléatoire)



K means

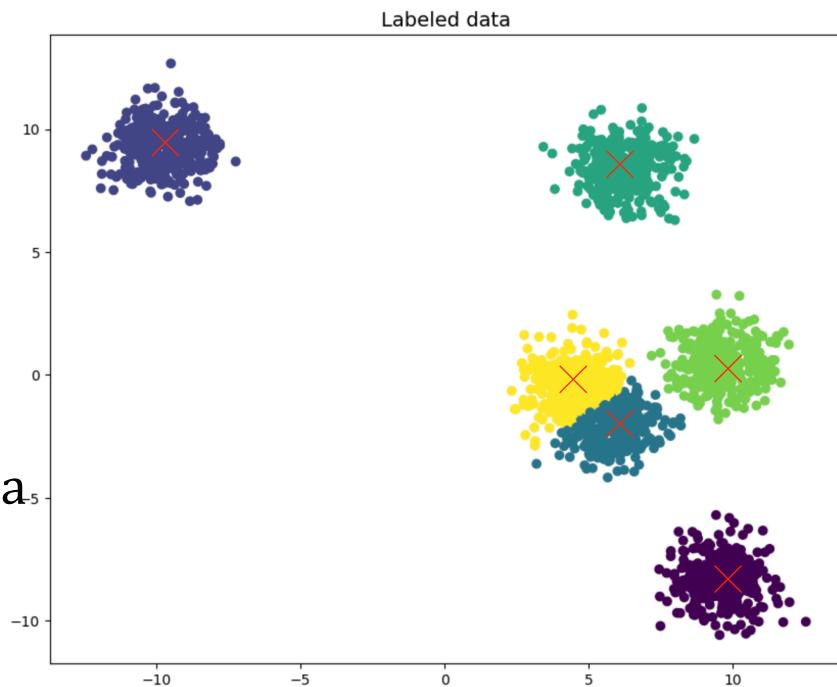
Comment ça marche ?

L'algorithme recherche K centroids minimisant l'inertie du système

$$\sum_{i=0}^n \min_{\mu_j \in C} (\|x_i - \mu_j\|^2)$$

Trois étapes :

- Sélection de la position initiale des centroids (K points aléatoire)
- Associe les points au centroid le plus proche
- Crée de nouveau centroid en prenant la valeur moyenne de chaque cluster



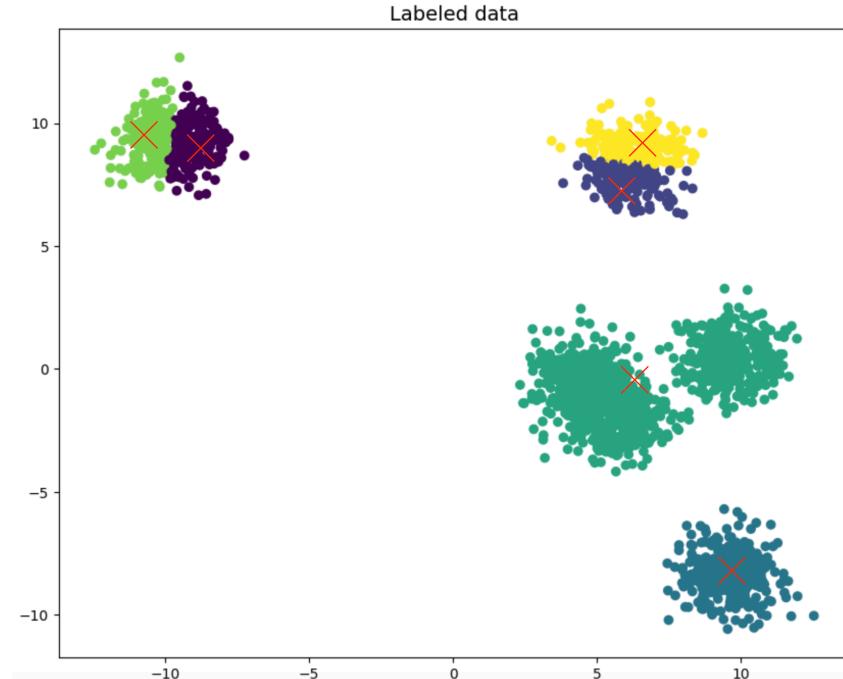
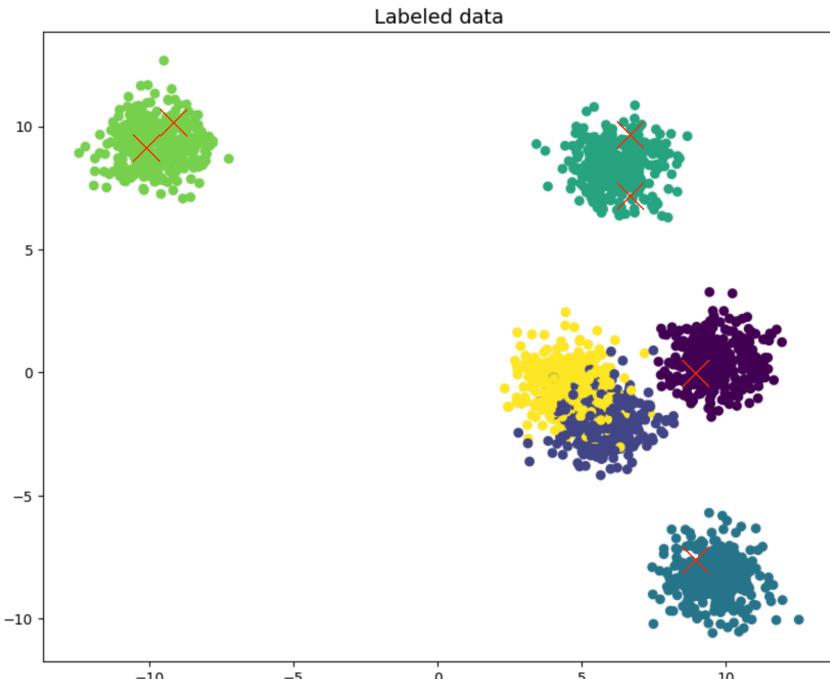
K means

Convergence ?

Le résultat dépend grandement des points choisis pour l'initialisation et la structure des données

L'algorithme étant rapide il est répété n fois avec différente initialisation

Des techniques d'initialisation différentes existent aussi (k-means++)

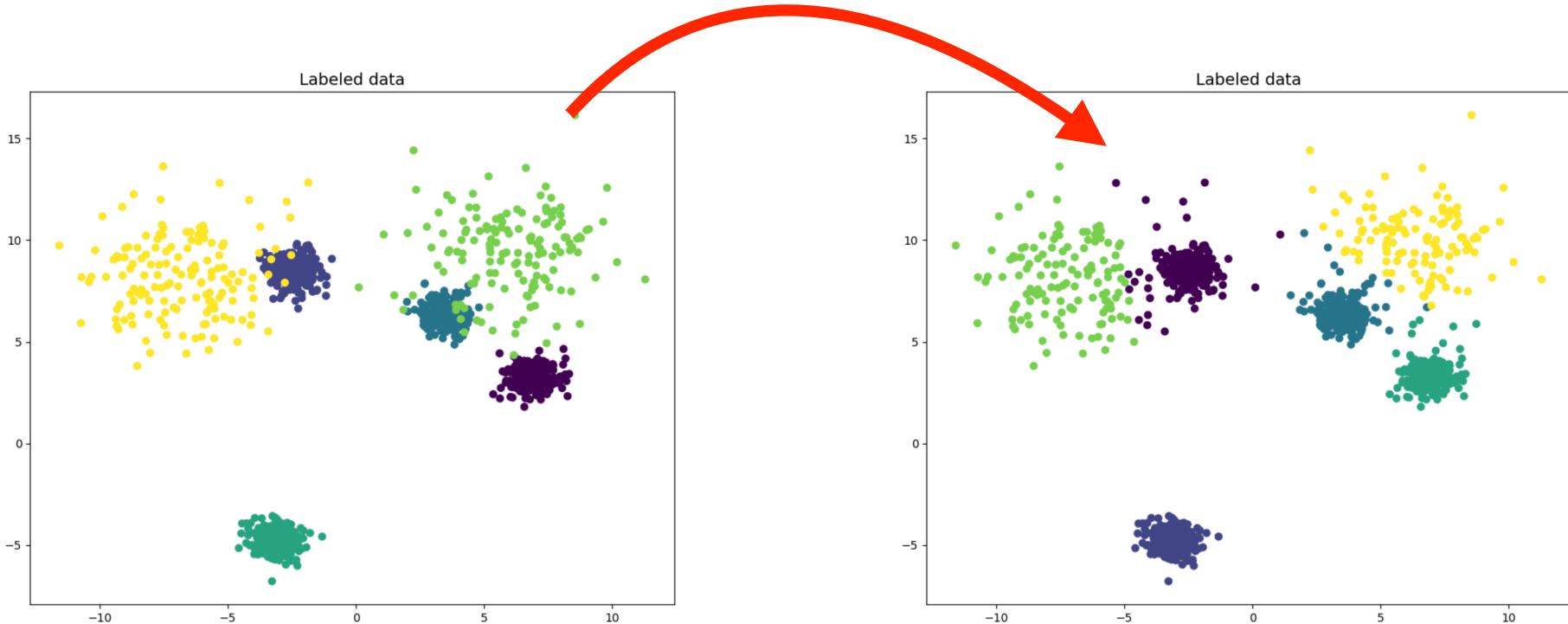


GaussianMixture

Inefficacité du k-mean pour les données in-homogène

Données générées selon deux types de distribution différentes

Problème à l'interface des deux distributions

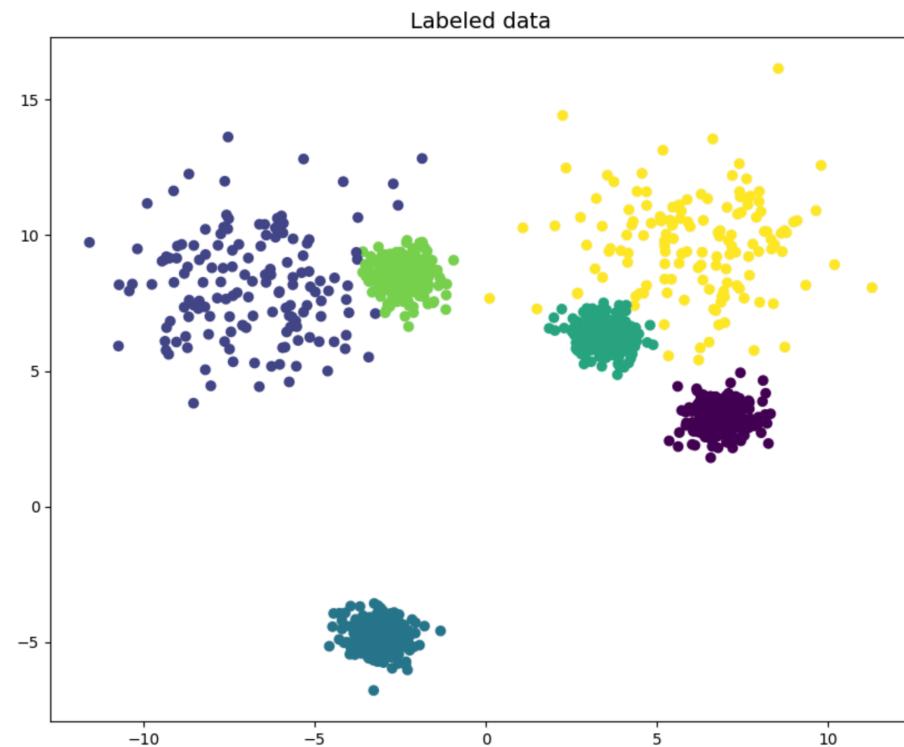


GaussianMixture

Gaussian mixture model

Assume que les données proviennent d'une combinaison de distributions gaussiennes

- Implémentation proche du k-mean : part de K distribution gaussienne
- Calcule la probabilité de chaque point d'avoir été générés par chaque distribution
- Mise à jour des poids des distributions pour maximiser la vraisemblance des données



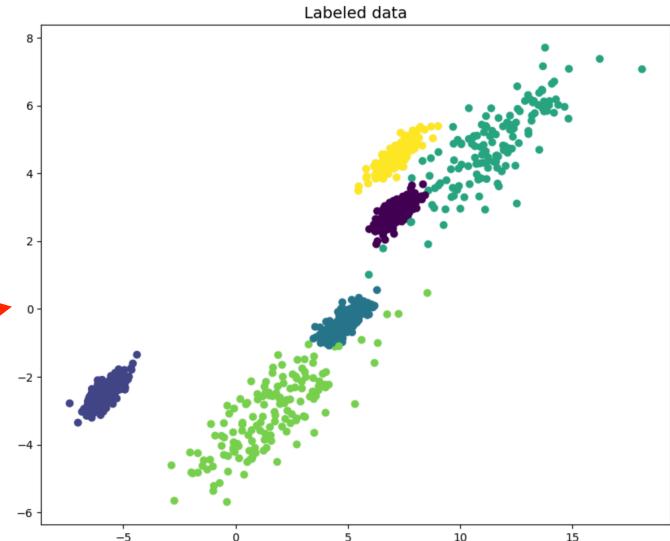
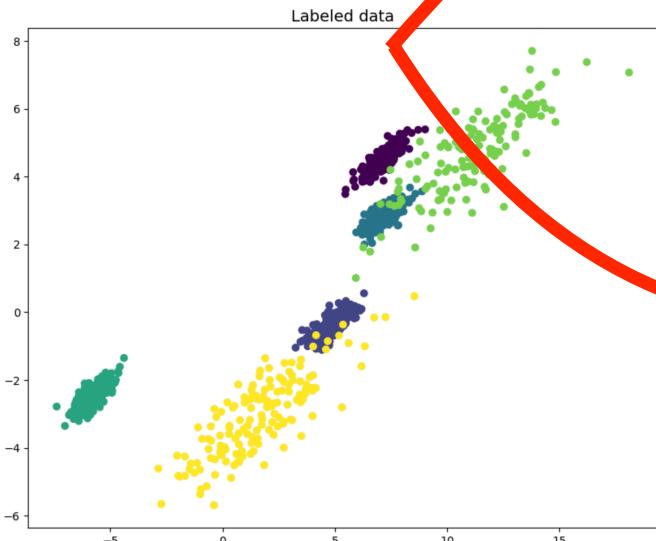
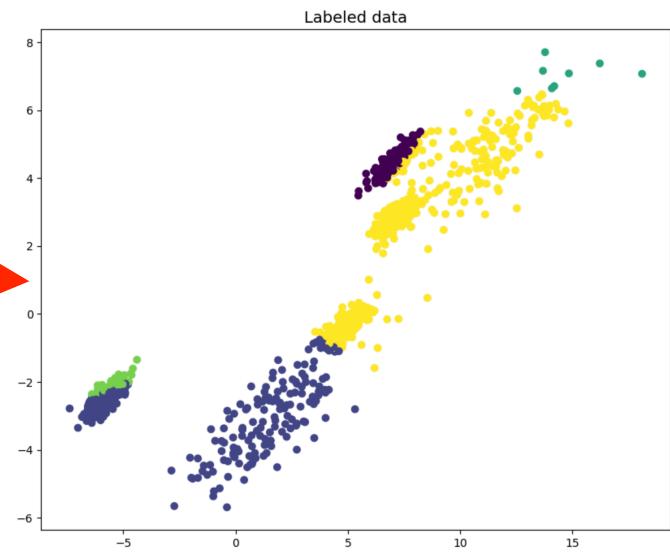
- Chaque point : probabilité d'appartenance au différent cluster

GaussianMixture

Données corrélées

GMM bien meilleur pour les données corrélées (non circulaire)

kMean

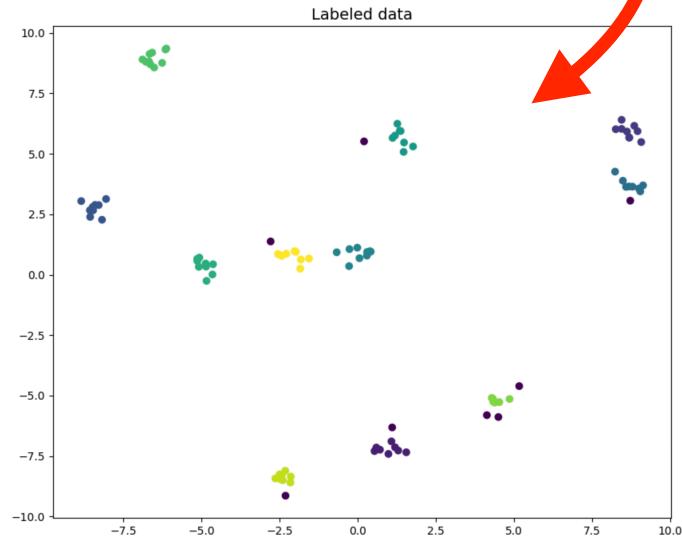
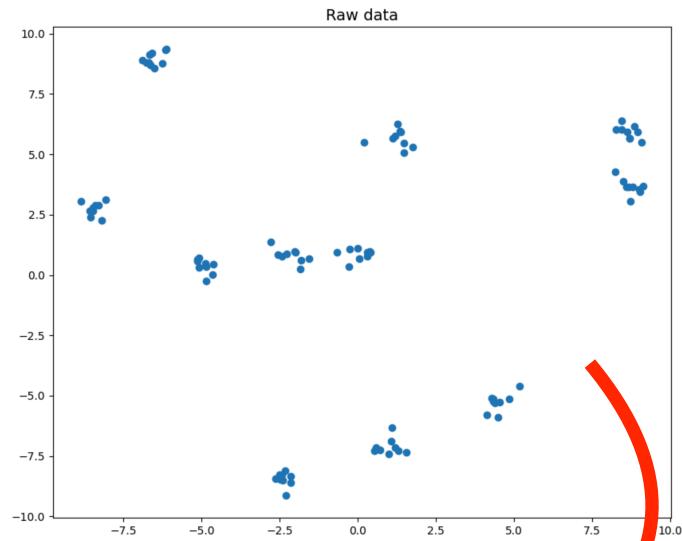


DBSCAN

Nombre de cluster libre

kMean, GMM ➔ On connaît le nombre de clusters et on veut leurs propriétés

DBSCAN ➔ On ignore le nombre de clusters, mais on connaît / peut approximer leurs propriétés



DBSCAN

Implementation :

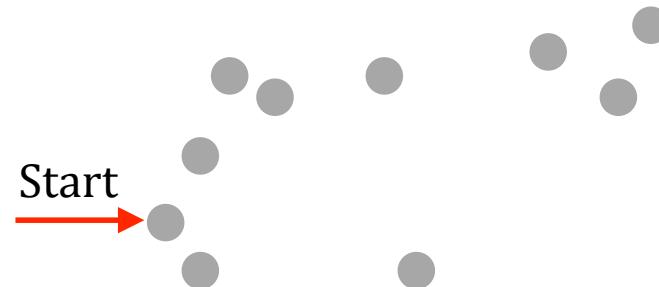
Basé sur la densité de donnée (ne marche pas pour des clusters superposés)

Deux paramètres d'entrée :

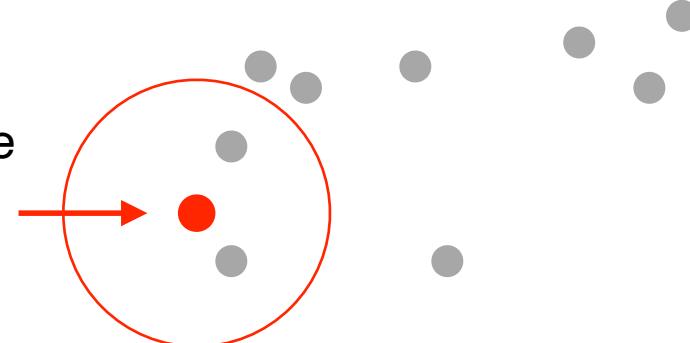
- ε : distance max entre 2 voisins
- $\text{Min}_{\text{sample}}$: nombre min de voisins

→ Sélectionne un point pour commencer le clustering

→ S'il a plus de $\text{Min}_{\text{sample}}$ voisin à une distance ε : crée un cluster



$\text{Min}_{\text{sample}}=3$



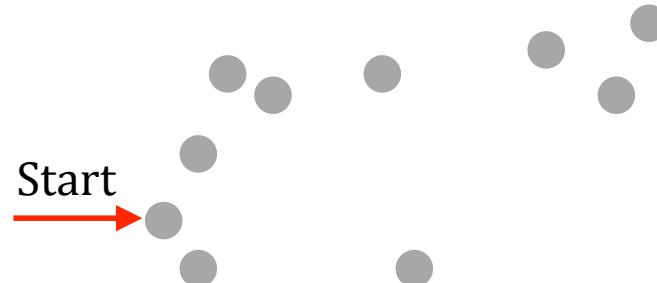
DBSCAN

Implementation :

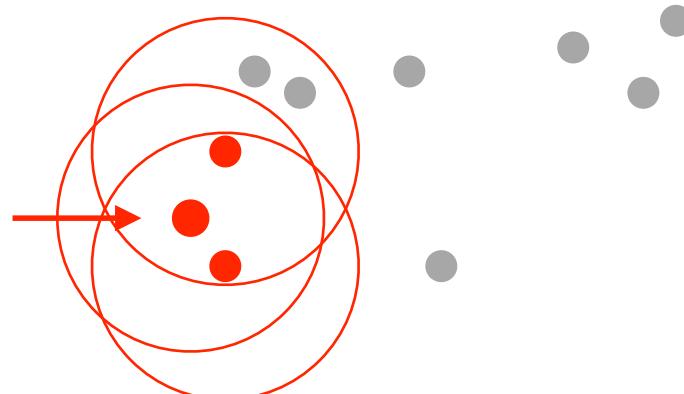
Basé sur la densité de donnée (ne marche pas bien pour des clusters superposés)

Deux paramètres d'entrée :

- ε : distance max entre 2 voisins
 - $\text{Min}_{\text{sample}}$: nombre min de voisins
- Sélectionne un point pour commencer le clustering
- Cluster : point plus voisin ($d < \varepsilon$)
- Répète le processus pour les voisins



$\text{Min}_{\text{sample}}=3$



DBSCAN

Implementation :

Basé sur la densité de donnée (ne marche pas bien pour des clusters superposés)

Deux paramètres d'entrée :

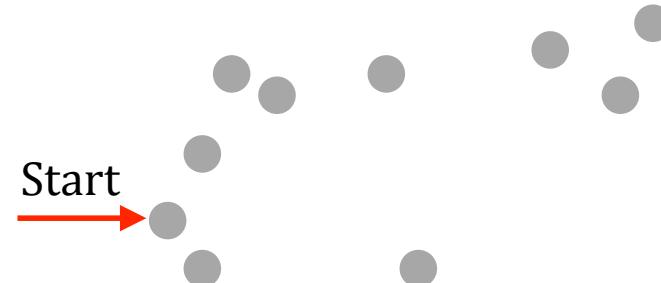
- ε : distance max entre 2 voisins
- $\text{Min}_{\text{sample}}$: nombre min de voisins

→ Sélectionne un point pour commencer le clustering

→ Cluster : point plus voisin ($d < \varepsilon$)

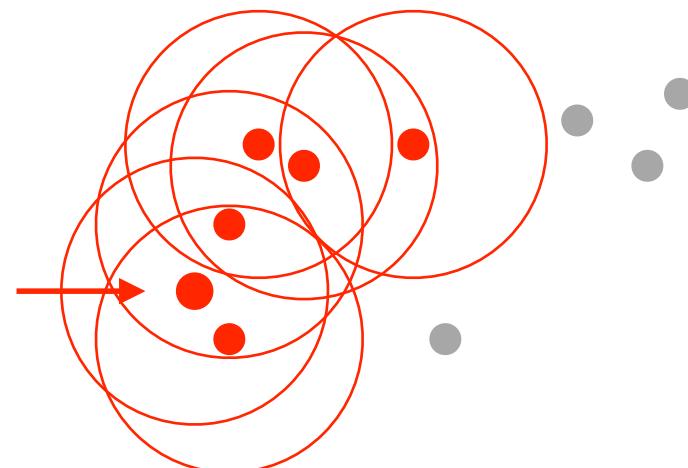
→ Répète le processus pour les voisins

→ Une fois le cluster fini continu avec un autre point



Min_{sample}=3

Cluster A



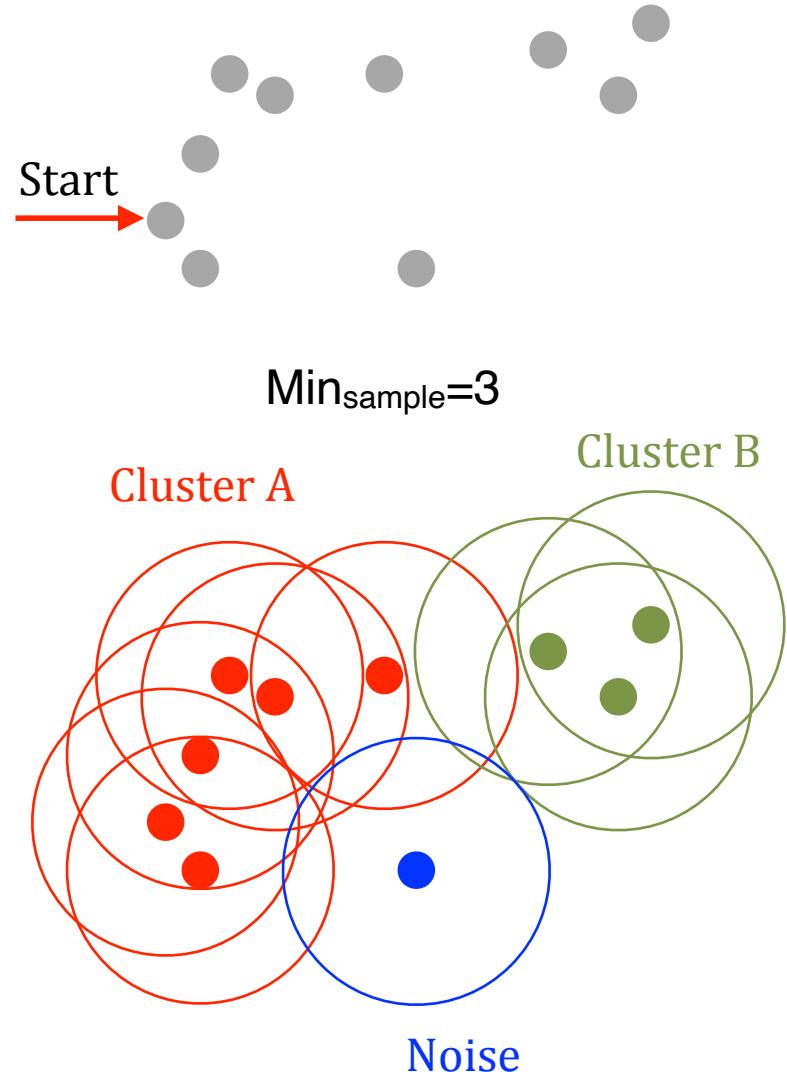
DBSCAN

Implementation :

Basé sur la densité de donnée (ne marche pas bien pour des clusters superposés)

Deux paramètres d'entrée :

- ε : distance max entre 2 voisins
 - $\text{Min}_{\text{sample}}$: nombre min de voisins
- Sélectionne un point pour commencer le clustering
- Cluster : point plus voisin ($d < \varepsilon$)
- Répète le processus pour les voisins
- Une fois le cluster fini continu avec un autre point

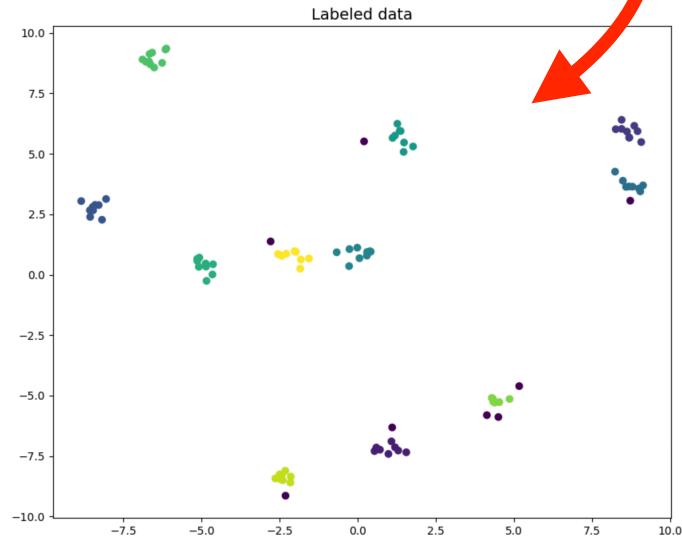
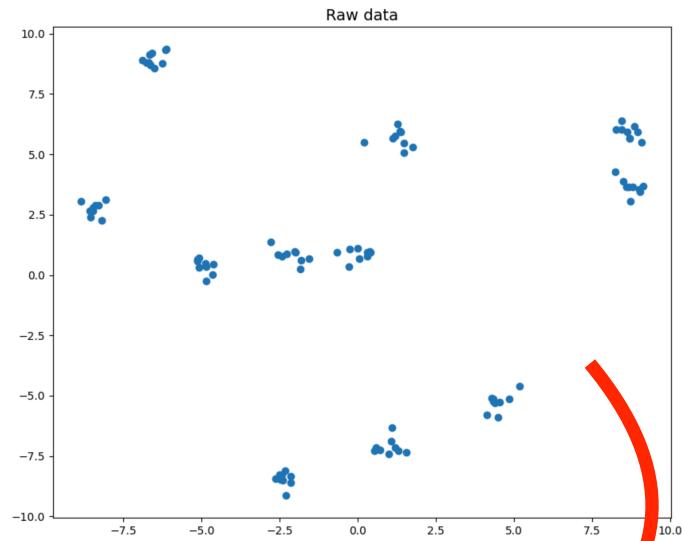


DBSCAN

Nombre de cluster libre

kMean, GMM ➔ On connaît le nombre de clusters et on veut leurs propriétés

DBSCAN ➔ On ignore le nombre de clusters, mais on connaît / peut approximer leurs propriétés



BACKUP