Introduction à la Science des données
Classification non supervisée (partie 2)

V. Monbet, A. Widmer - fortement inspiré de ...

[1] Université de Rennes/UFR Mathématiques

Outline

Introduction

K-means

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Rappels

On dispose d'un ensemble de n individus $\mathbf{X} = \{x_1, ..., x_n\}$ un caractérisés par p variables, avec $x_i \in \mathbb{R}^p$.

- La **classification non supervisée** : regrouper entre eux les individus qui se ressemblent.
  On obtient **une partition de l'ensemble $\mathbf{X}$** telle que deux individus d'une même partie se ressemblent plus que deux individus de parties différentes.

- La **classification hiérarchique ascendante** est basée sur un algorithme glouton [1].
  Au départ chaque individu forme une classe à lui seul. A chaque étape, les deux classes les plus proches sont agglomérées (critère local), pour finir avec une classe unique qui regroupe tous les individus.
  L'arbre obtenu est coupé a posteriori de façon à obtenir un bon **compromis entre le nombre de classes et la variance intra-classe** (critère global).

1. a chaque itération, un critère local est optimisé

Similarité entre individus

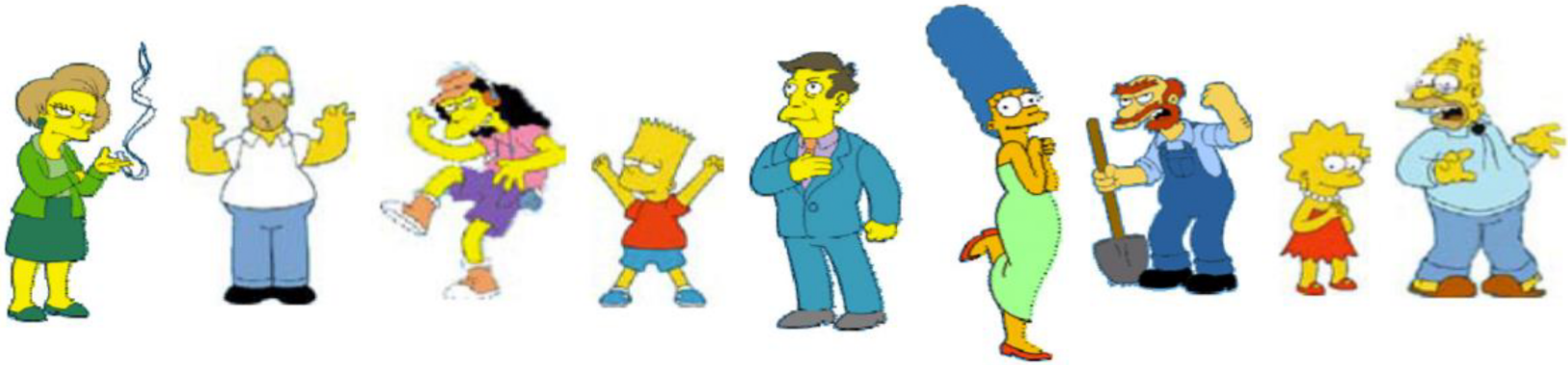
Pour classer les individus, on a besoin d'une **mesure de similarité** entre individus. Il y a différentes façons de définir la similarité.

Are they similar or not?

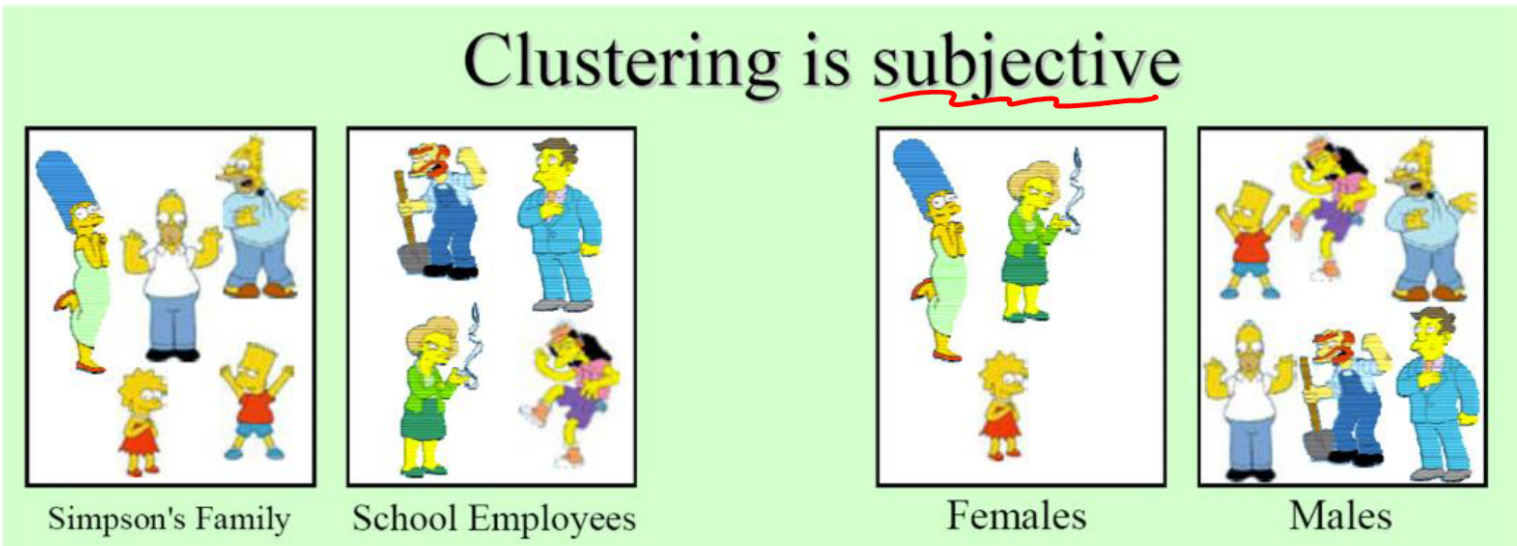


Similarité entre individus

Le choix de la mesure impacte la classification.



What is consider similar/dissimilar?



Outline

Introduction

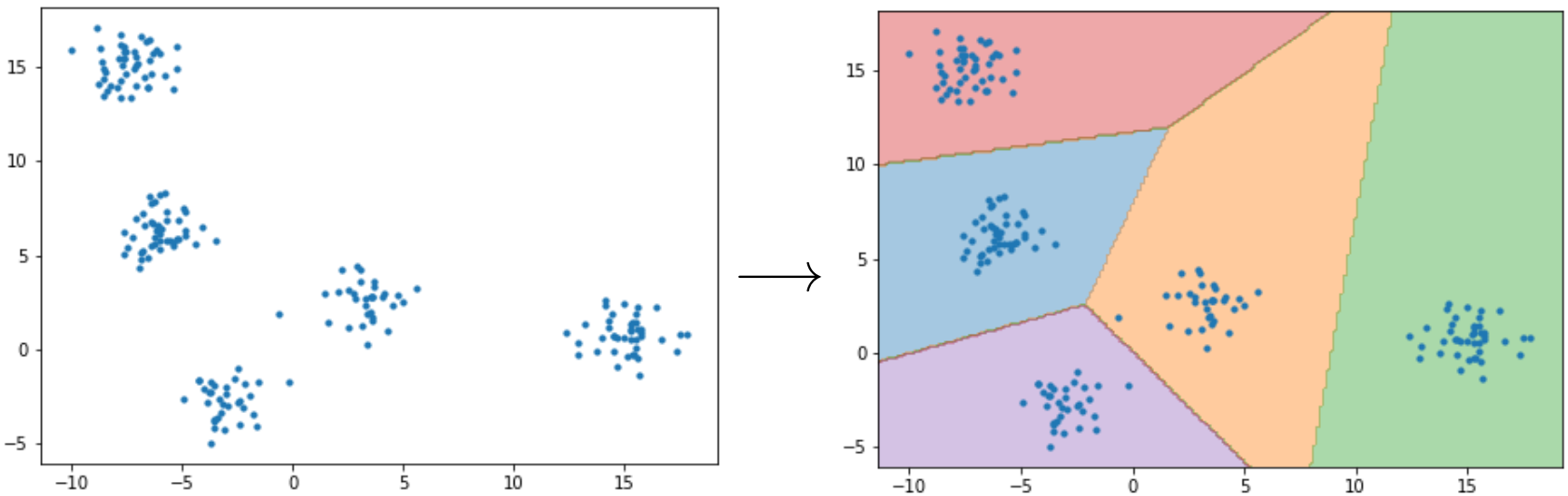
K-means
- Algorithme des kmeans
- Algorithme des kmeans++
- Exemple sur données simulées
- Choix du nombre de classes
- Un exemple de données réelles

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Objectifs

Regrouper des individus (points de $\mathbb{R}^p$), en groupes (classes) d'individus proches en un certain sens.

Par exemple :



Outline

K-means
- Algorithme des kmeans
- Algorithme des kmeans++
- Exemple sur données simulées
- Choix du nombre de classes
- Un exemple de données réelles

Principe

- On donne : les points et un nombre n_c de groupes attendus
- L'algorithme doit rendre : n_c centres

A chaque point on associe le centre le plus proche, ce qui détermine les groupes, et des régions.

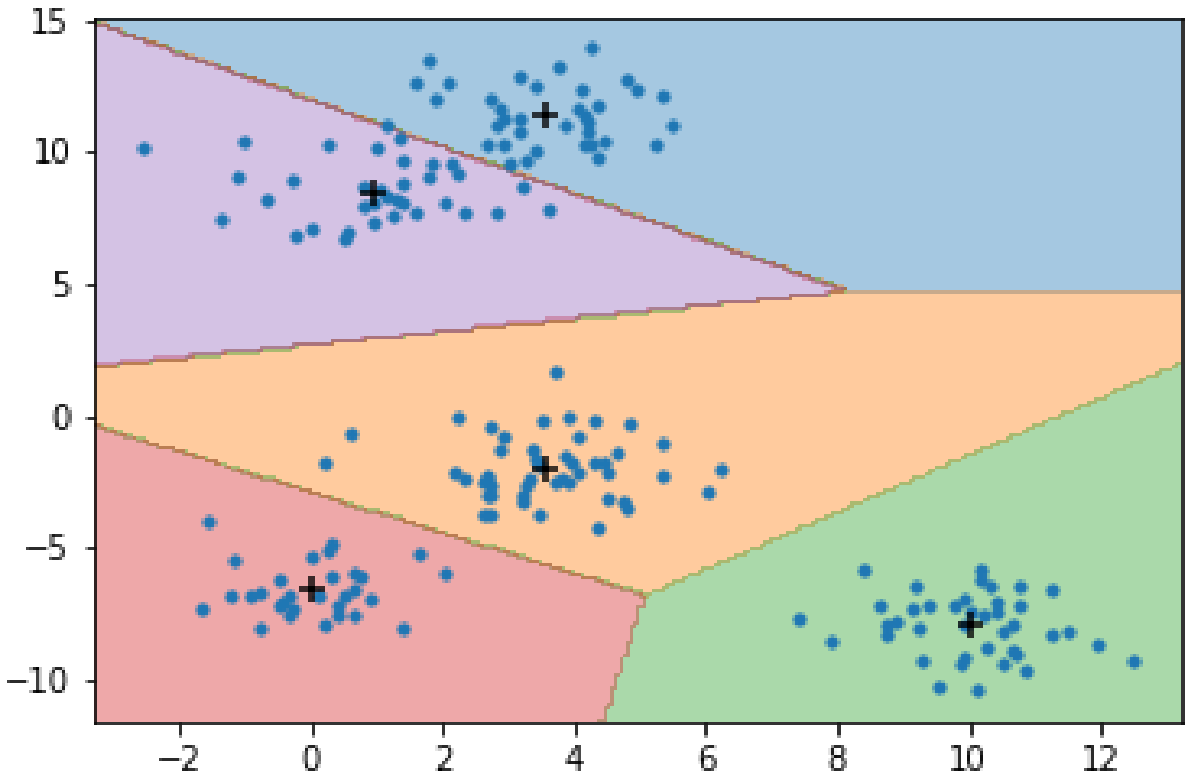
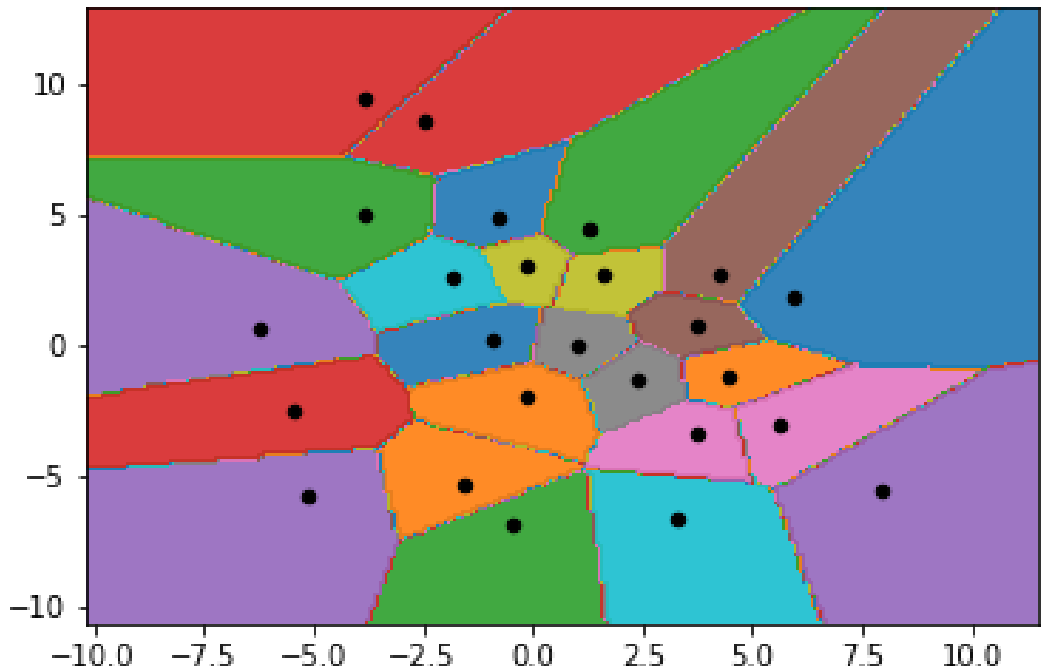


Diagramme de Voronoï associé à une famille de centres



Chaque région est l'ensemble des points pour lesquels le centre (centroïde) est le plus proche.
On remarque que les frontières sont linéaires par morceaux.

Objectif à optimiser

On cherche les centres $c_1, \ldots c_{n_c}$ qui minimisent la somme des distances au centre le plus proche

$$\sum_x \min_{j=1,\cdots,n_c} \|x - c_j\|^2 = \sum_x \|x - \widehat{x}\|^2$$

On peut voir le centre le plus proche $\widehat{x}$ comme un codage, une approximation, un arrondi, de x.

Algorithme des k-means

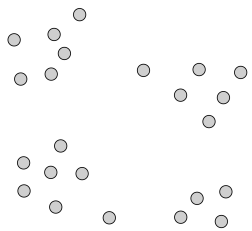
(1) Partir de n_c centres tirés aléatoirement
(2) Associer à chaque donnée x_{ij} à son centre le plus proche
(3) Recalculer chaque centre comme moyenne des données qui lui ont été associées
(4) Revenir à (2), jusqu'à convergence

La limite peut dépendre de l'initialisation.

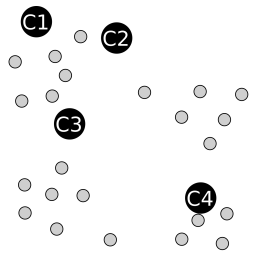
Il y a des variantes.

Algorithme des k-means

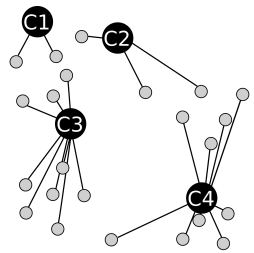
Exemple [2]



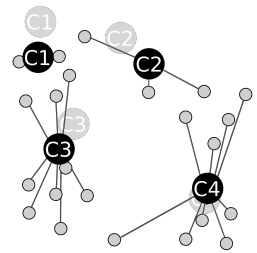
0a. Données d'entrée



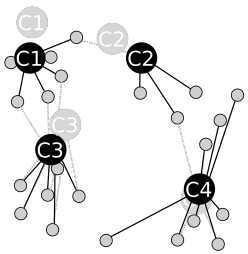
0b. intialisation



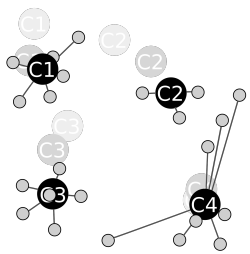
1a. assignation



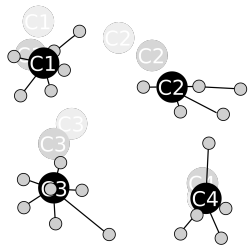
1b. calcul des points moyens



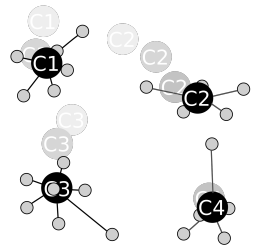
2a. assignation



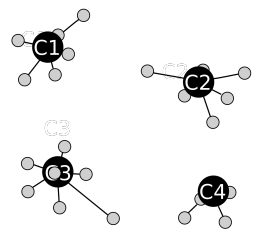
2b. calcul des points moyens



3a. assignation



3b. calcul des points moyens



4a. assignation

clusters stables (fin)

2. Par Mquantin — CC BY-SA 4.0, https ://commons.wikimedia.org/w/index.php ?curid=61321400

Outline

K-means
- Algorithme des kmeans
- Algorithme des kmeans++
- Exemple sur données simulées
- Choix du nombre de classes
- Un exemple de données réelles

Une initialisation plus élaborée de l'algorithme des k-means

L'algorithme kmeans++ d'Arthur et Vassilvitskii (2006) :

- Choisir le premier centre c_1 aléatoirement dans les données.
- Pour $j = 1, ...N - 1$:
  - Calculer pour chaque point x_i la distance au centre le plus proche

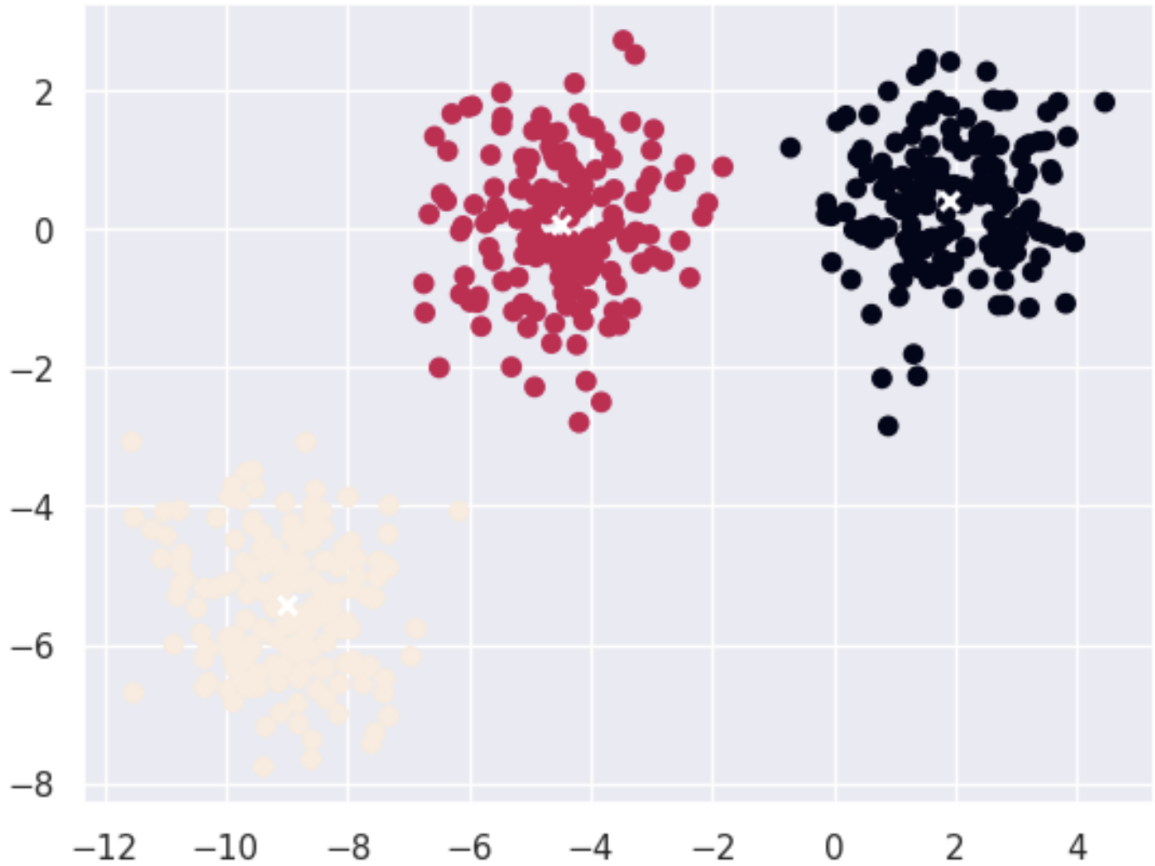
$$d_i^2 = \min_{k \leqslant j} \|x_i - c_k\|^2$$

  - Tirer aléatoirement c_{j+1} parmi les x_i selon les probabilités $d_i^2 / \sum_k d_k^2$.

Outline

K-means
- Algorithme des kmeans
- Algorithme des kmeans++
- Exemple sur données simulées
- Choix du nombre de classes
- Un exemple de données réelles

Exemple avec données simulées



Le premier exemple était facile : les classes sont bien séparées. Augmentons la variance de chacun des classes pour corser un peu l'affaire ! On a la répartition de la figure ci dessous.

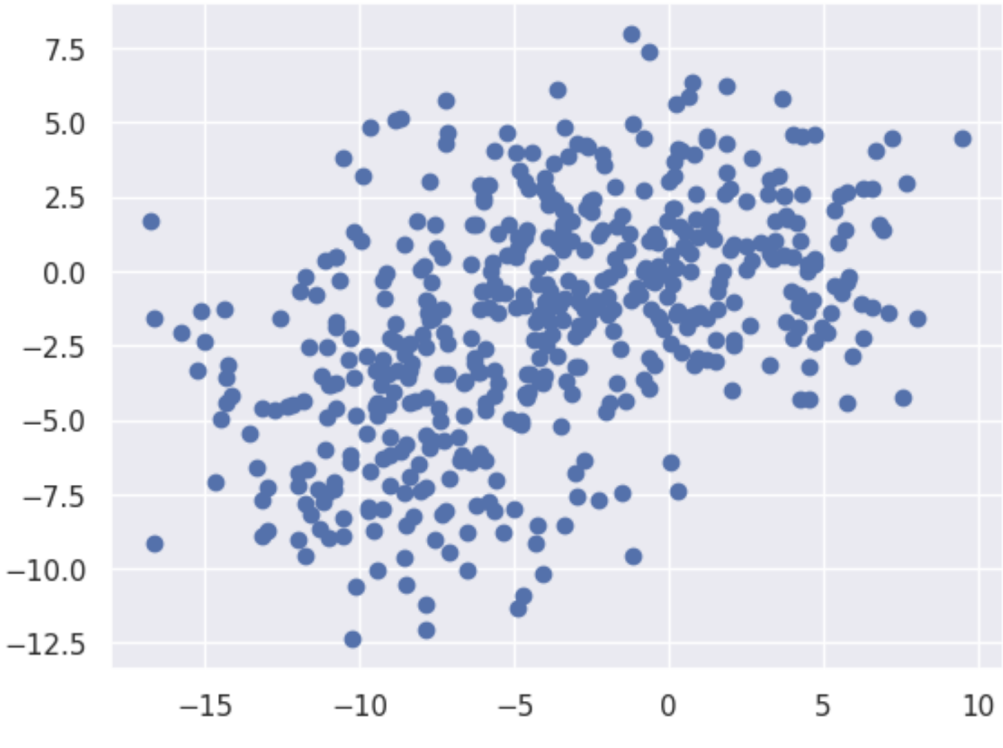


Quelle solution peut-on attendre ?

CODE

Si on oublie qu'on connait les classes, on voit ceci.

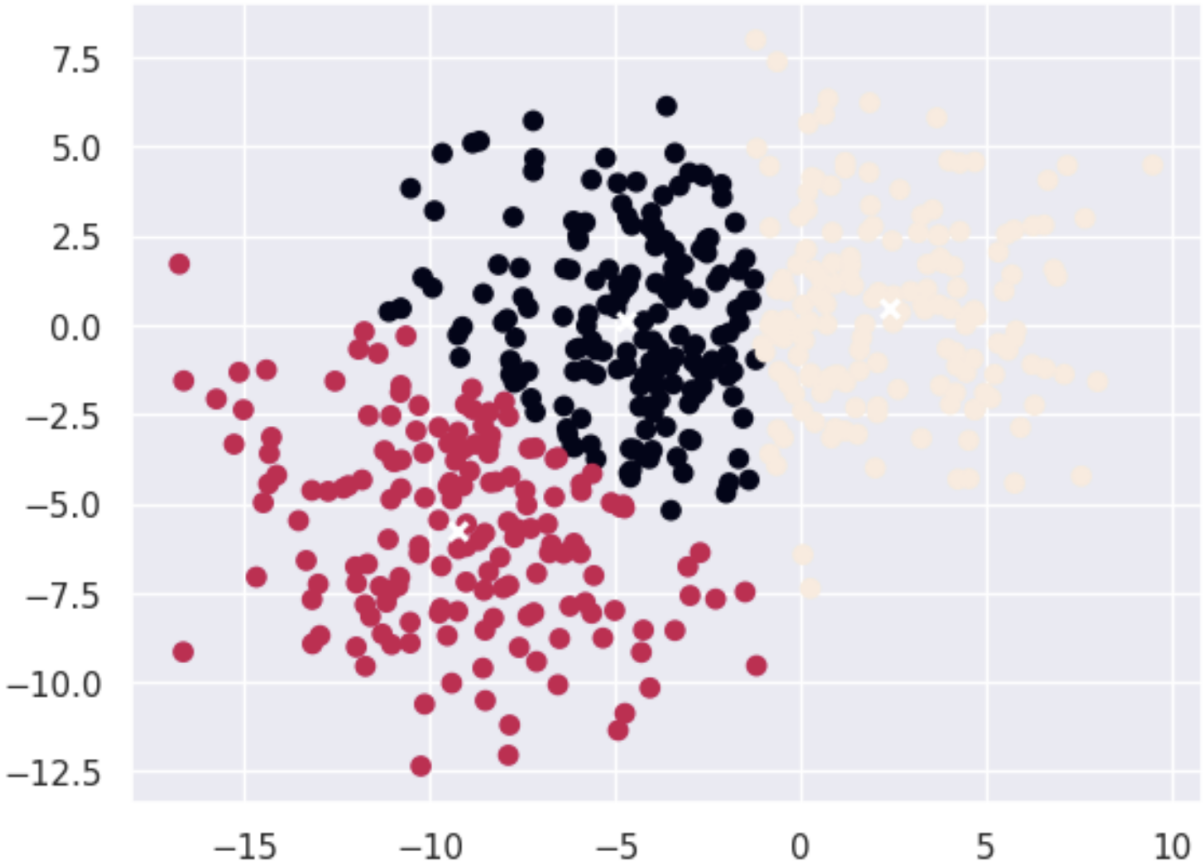




Solution du kmeans (3 classes)

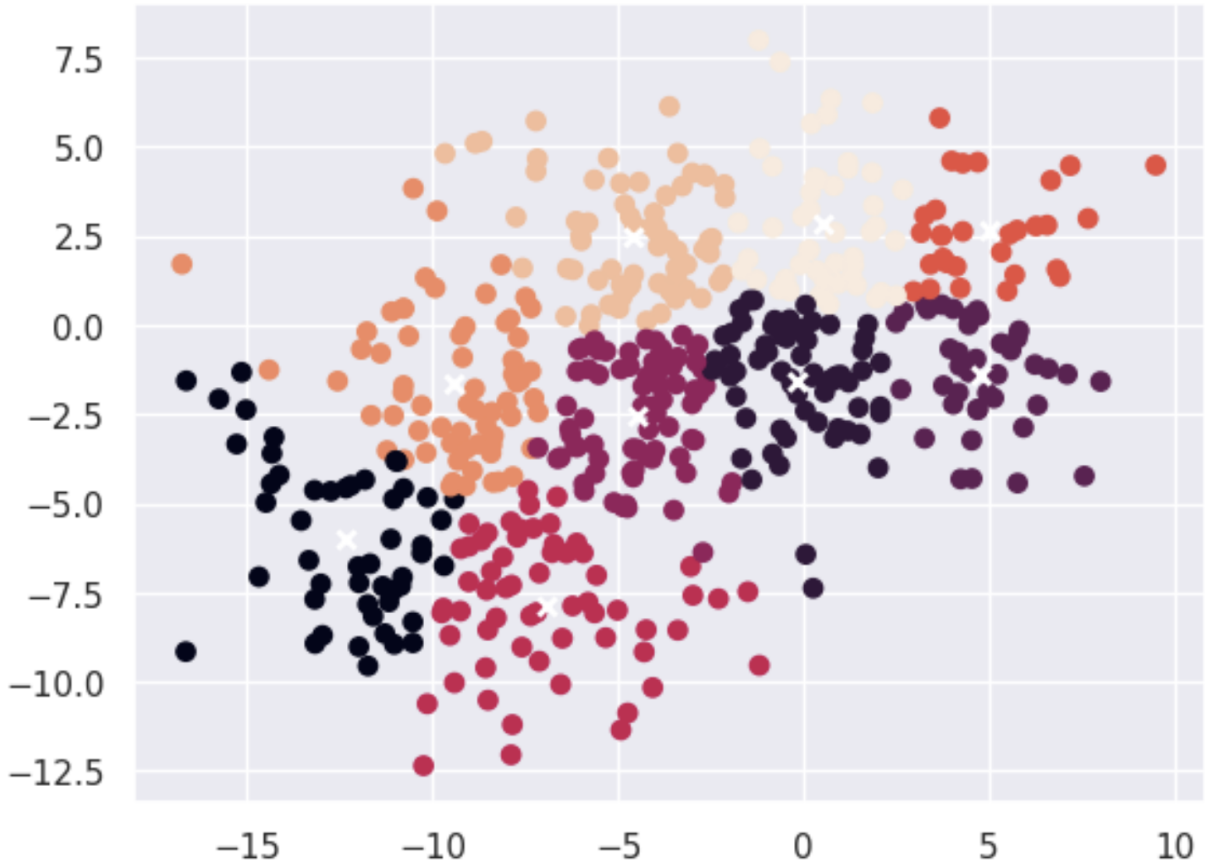
✗**les frontières sont linéaires et les classes de taille similaire.**

✗**Les classes sont données à une permutation près des labels.**



CODE

La solution dépend bien sûr du nombre de classes demandé. Par exemple, si on choisit 9 classes (après tout, pourquoi pas ?).



CODE

Outline

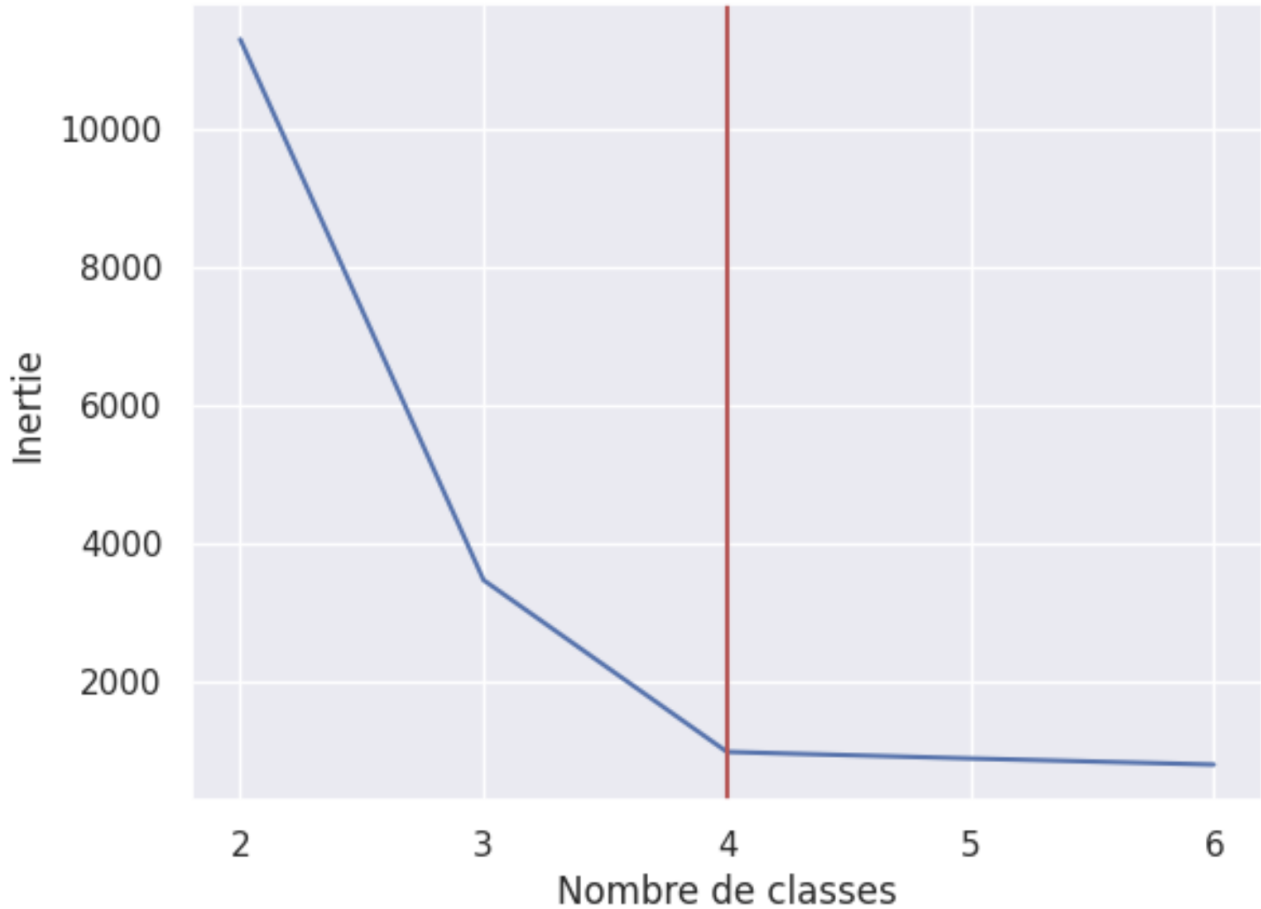
K-means
- Algorithme des kmeans
- Algorithme des kmeans++
- Exemple sur données simulées
- Choix du nombre de classes
- Un exemple de données réelles

Comment choisir le nombre de classes ?



Variance totale et règle du coude

On simule 500 points en dimension 2, regroupés en 4 classes (voir CODE).
On fait le kmeans pour différents nombres de classes et on choisit le meilleur compromis entre la variance intra-classe et le nombre de classes.



Coefficient de Silhouette

Coefficient de Silhouette : Évalue le degré de compacité et de séparation des grappes. En utilisant le coefficient de Silhouette, nous pouvons choisir une valeur optimale pour le nombre de groupes.

- a est la moyenne des distances intra-cluster pour le point x_i

$$a(x_i) = \frac{\sum_{x_k \in C_j, k \neq i} D(x_i, x_k)}{|C_j| - 1}$$

où $|C_j|$ est le cardinal du cluster j.

- b est le minimum des distances inter-cluster pour le point x_i

$$b(x_i) = \min_{C_j : 1 \leq j \leq J, x_i \notin C_j} \frac{\sum_{x_k \in C_j, k \neq i} D(x_i, x_k)}{|C_j|}$$

où J est le nombre de classes.

Le coefficient de Silhouette

$$s(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i)}$$

est compris entre -1 et 1 ; plus il est grand, mieux c'est.

Exemple 4 classes

On simule 500 points en dimension 2, regroupés en 4 classes (voir CODE).
On fait le kmeans pour différents nombres de classes et on calcule 3 indices.
L'homogénéité et la compacité sont des scores supervisés (ie il faut connaitre les vrais labels pour les calculer).

Nb classes	Homogeneité [3]	Compacité [4]	Silhouette
2	0.50	1.00	0.52
3	0.74	0.98	0.68
4	1.00	1.00	0.75
5	1.00	0.89	0.65
6	1.00	0.80	0.53

Quelle est la meilleure partition ? Pourquoi ?

3. Entrophy de Shannon
4. score supervisé

Exemple 4 classes

On simule 500 points en dimension 2, regroupés en 4 classes (voir CODE).

Nb classes	Homogeneité	Compacité	Silhouette
2	0.50	**1.00**	0.52
3	0.74	0.98	0.68
4	**1.00**	**1.00**	**0.75**
5	**1.00**	0.89	0.65
6	**1.00**	0.80	0.53

Quelle est la meilleure partition est la partition à 4 classes : le coefficient silhouette est maximum pour 4 classes.

Dans le code, on montre un exemple à 4 classe un peu plus difficile pour lequel on compare les scores de la variance intra classe et le coefficient silhouette.

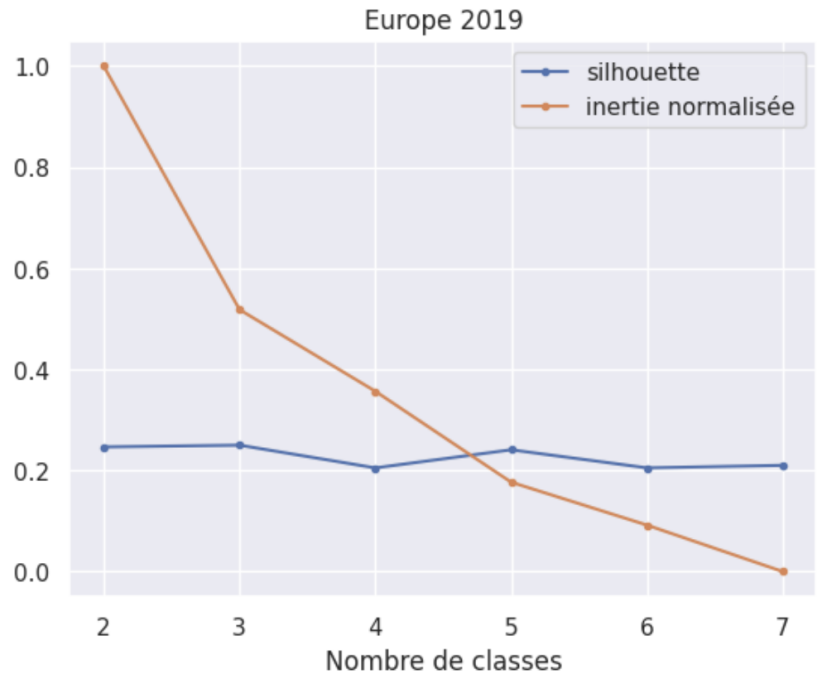
Outline

K-means
- Algorithme des kmeans
- Algorithme des kmeans++
- Exemple sur données simulées
- Choix du nombre de classes
- Un exemple de données réelles

Un exemple avec données réelles : Europe

On reprend l'exemple utilisé pour la CAH : données démographiques en Europe (2019).

- Choix du nombre de classes

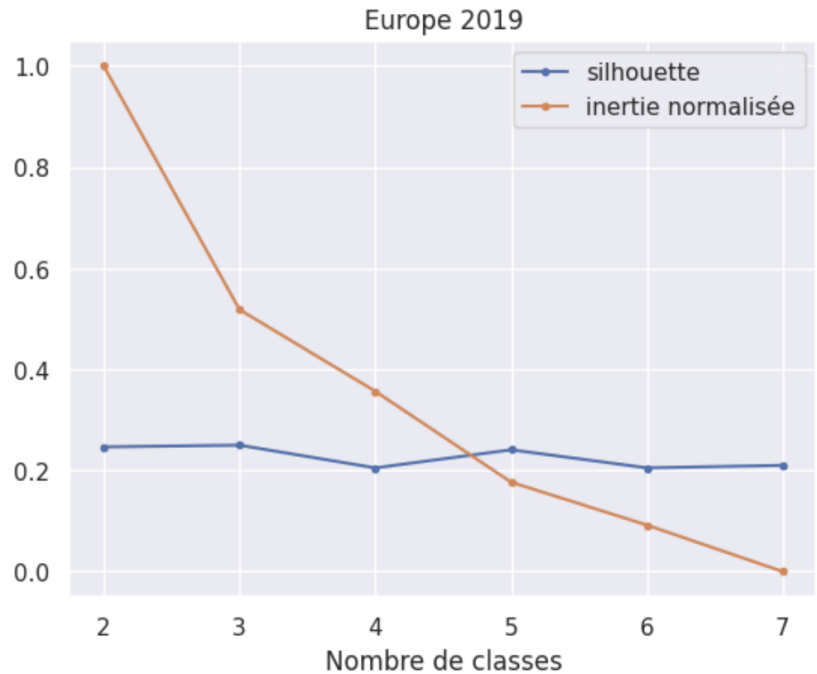


Qu'en pensez vous ?

Un exemple avec données réelles : Europe

On reprend l'exemple utilisé pour la CAH : données démographiques en Europe (2019).

- Choix du nombre de classes



On observe ci-dessous que le coefficient silhouette evolue peu. En appliquant la règle du coude, on pourrait sélectionner 3 ou 5 classes. 3 classes semble plus raisonnable au regard du nombre d'observations.
Cet exemple souligne bien la difficulté du choix du nombre de classes.

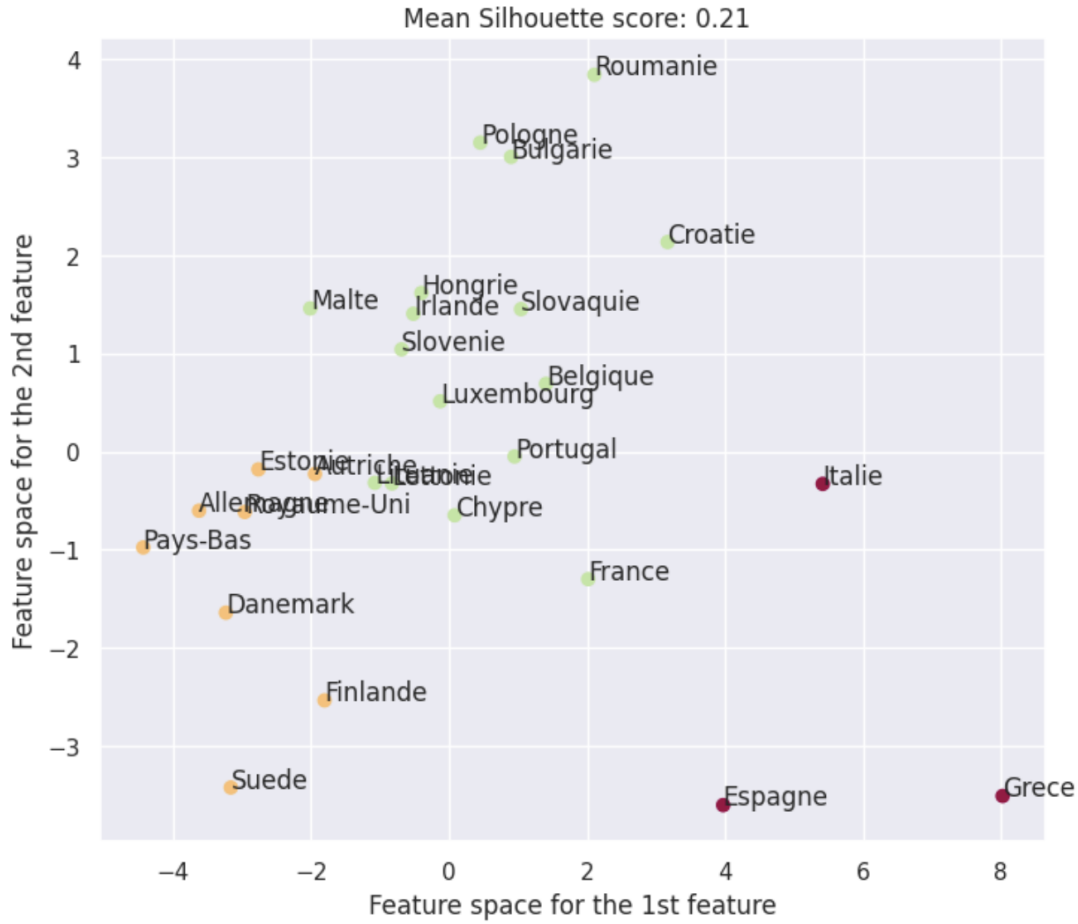
Un exemple avec données réelles : Europe

On voudrait maintenant vérifier qu'une classification non supervisée à 3 classes a du sens.

Des idées ?

Un exemple avec données réelles : Europe

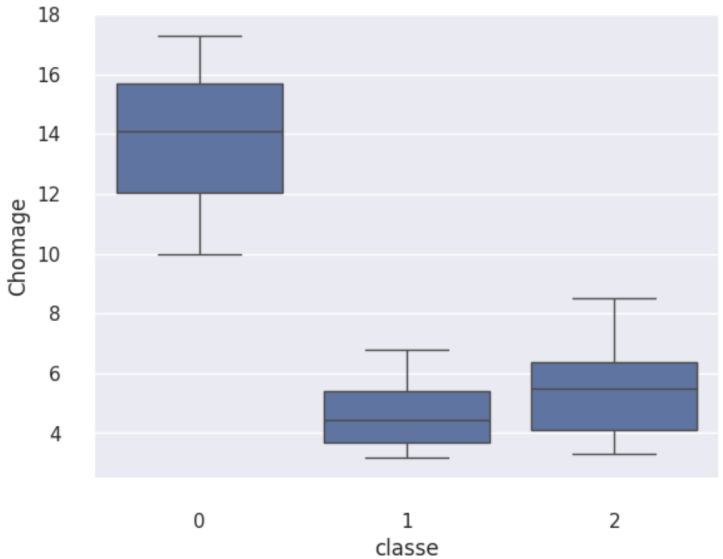
- La dimension des données est supérieure à 2, on peut utiliser une visualisation sur le 1er plan factoriel de l'ACP.

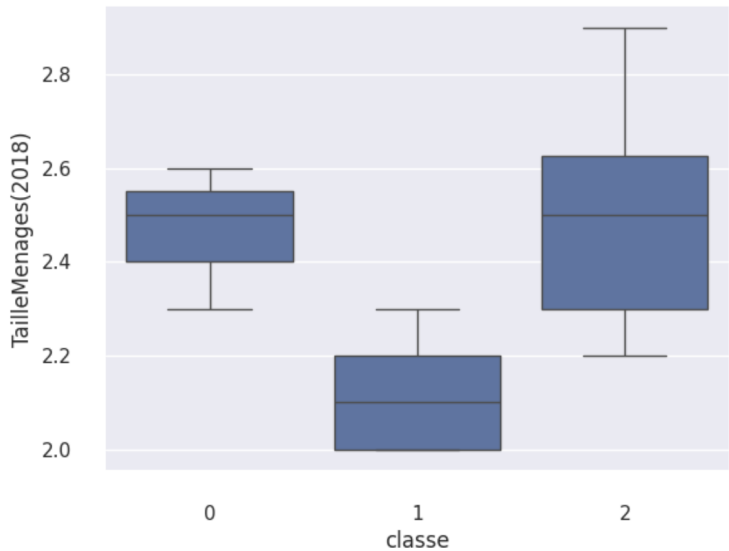


Les groupements de pays semblent être raisonnables.
Voir le code, pour l'interprétation des classes.

Un exemple avec données réelles : interprétation

- Il est aussi intéressant de visualiser la distribution de certaines variables [5] selon les classes.





Les boxplots sont utiles à l'interprétation des classes.

- Classe 0 : pays à fort taux de chomage et menages de taille supérieure à la moyenne.
- Classe 1 : pays à faible taux de chomage et ménages de taille plutôt inférieure à la moyenne.
- Classe 2 : pays à faible taux de chomage et ménages de taille supérieure à la moyenne (et très variable).

5. choisies d'après l'interprétation des axes de l'ACP

Kmeans, à retenir

- Les distances euclidiennes sont utilisées.
- Le centroïde est calculé à partir de la distance moyenne entre les membres de la classe.
- Les classes sont supposées isotropes et convexes (voir exemples interactifs)
- ✓ L'algorithme est simple et il converge rapidement vers un minimum local.
- ✗ Le nombre de classes doit être défini a priori.
- ✗ Algorithme stochastique - les résultats dépendent des critères d'initialisation (voir exemples interactifs)
  Et la solution est un optimuum local.
- ✗ Crée des groupes de variance égale (minimise l'inertie) (voir exemples interactifs)
- ✗ Peu robuste au bruit

Outline

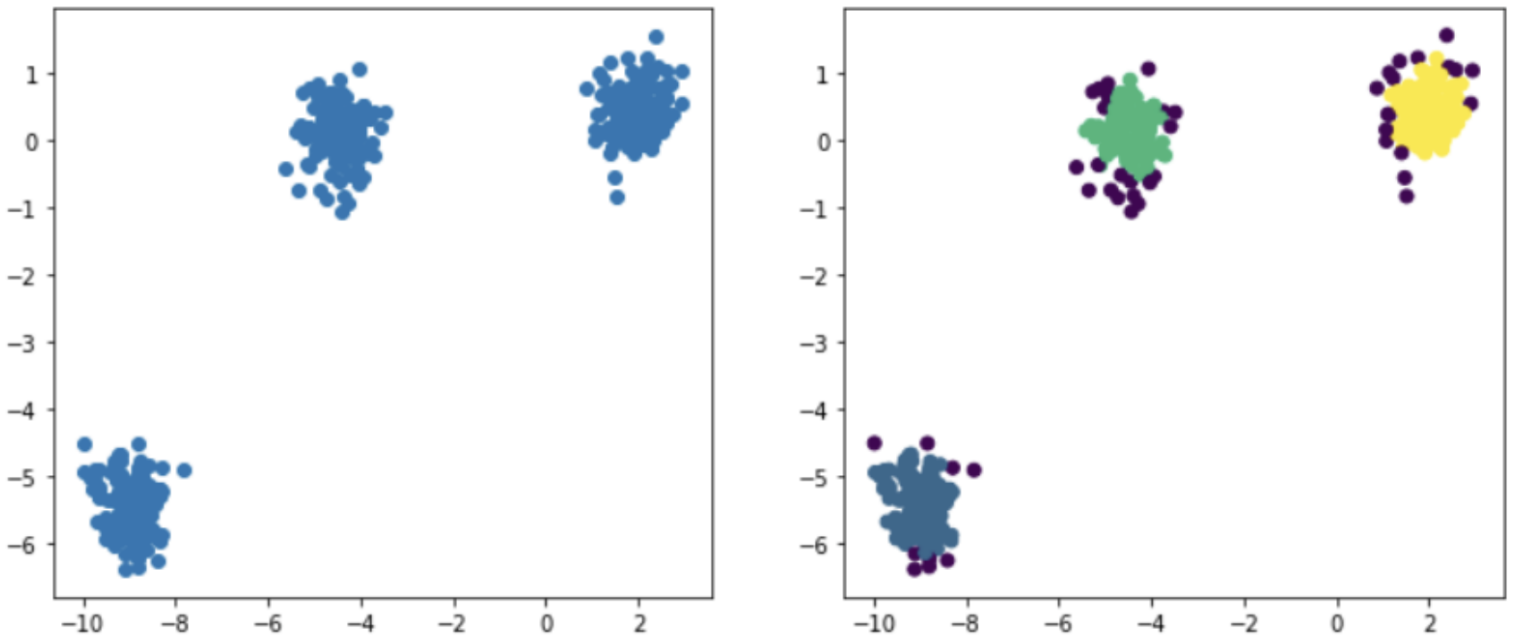
Introduction

K-means

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
Algorithme

Objectifs

Regrouper des individus (points de $\mathbb{R}^p$), en groupes (localement) denses d'individus.
Par exemple



A la différence du kmeans, DBSCAN identifie des outliers.

Outline

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
Algorithme

Les ingrédients

On a besoin de définir la notion de "densité locale".

- **ϵ-voisinage**
  Pour chaque observation on regarde le nombre de points à au plus une distance ϵ de celle-ci. On appelle cette zone le ϵ-voisinage de l'observation.

- **Nombre de voisins**
  Si une observation compte au moins un certain nombre de voisins, y compris elle-même, elle est considérée comme une observation cœur.

Le principe

principe de DBSCAN en animation

L'idée est que si un point particulier appartient à une classe, il doit être proche de nombreux autres points de cette classe.

Algorithme

1. Choisir un point x au hasard
2. Récupérer tous les points densément accessibles à partir de x étant données ϵ et MinPts.
3. Si x est un point central, une classe est formée.
4. Si x est un point frontière, aucun point n'est accessible en densité à partir de x. Le point suivant est alors visité.
5. Répétez le processus jusqu'à ce que tous les points de données aient été traités.

Exemple sur données réelles

Une des principales difficultés de DNSCAN consiste à choisir ϵ et MinPts.
L'idée est de calculer la moyenne des distances de chaque point à ses MinPts plus proches voisins. La valeur de k sera spécifiée par l'utilisateur et correspond à MinPts.
Ensuite, ces MinPts-distances sont représentées dans l'ordre croissant. L'objectif est de déterminer le "genou", qui correspond au paramètre epsilon optimal. Un coude correspond à un seuil où un changement brutal se produit le long de la courbe de la distance MinPts.

Exemple sur données réelles

Pour MinPts = 3,

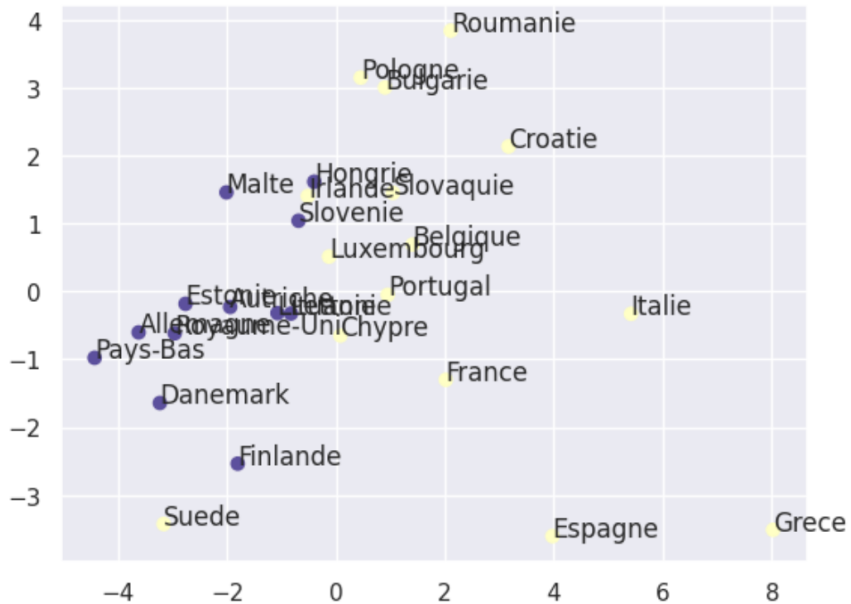
Figures/DBSCAN_sorted_ditances.png

on choisit *epsilon* = 3.

code

Exemple sur données réelles

On obtient 2 classes, qu'on peut représenter sur le plan factoriel.



on choisit *epsilon* = 3.

code

A retenir

- On choisit ϵ, le nombre minimum de points voisins et une distance.
  Le nombre minimum est aussi le nombre minimum de points pour former un cluster.
- ✓ L'algorithme est très simple et ne nécessite pas qu'on lui précise le nombre de clusters à trouver.
- ✓ Il est capable de gérer les données aberrantes en les éliminant du processus de partitionnement.
- ✓ Les clusters n'ont pas pour obligation d'être linéairement séparables (tout comme pour l'algorithme des k-moyennes par exemple).
- ✗ Cependant, il n'est pas capable de gérer des clusters de densités différentes.