Apprentissage non-supervisé : méthodes de partitionnement

Rym Guibadj

LISIC, EILCO

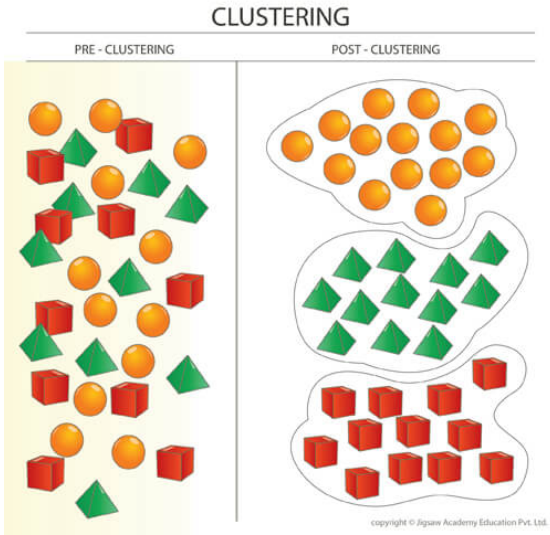
Types d'apprentissage :

- **Apprentissage supervisé**
  - Chaque exemple est associé à une étiquette
  - Objectif : prédire l'étiquette de chaque donnée
  - Le système apprend à classer les données
- **Apprentissage non supervisé**
  - Les exemples ne sont pas étiquetés
  - Objectif : trouver une structure aux données
  - Le système apprend une classification des données
- **Apprentissage par renforcement**
  - Les exemples sont (parfois) associés à une récompense ou une punition
  - Objectif : trouver les actions qui maximisent les récompenses
  - Le système apprend une politique de décision

Algorithmes de clustering :

Définition

Les algorithmes de clustering permettent de **partitionner** les données en sous-groupes, ou **clusters**, de manière non supervisée. Ces sous-groupes regroupent entre elles des observations **similaires**.



Intérêt du clustering :

- **Comprendre les données.** Par exemple identifier :
  - des clients qui ont des comportements similaires (segmentation de marché) ;
  - des utilisateurs qui ont des usages similaires d'un outil ;
  - des communautés dans des réseaux sociaux ;
  - des motifs récurrents dans des transactions financières.
- **Visualiser les données :** afficher uniquement un point représentatif par cluster.
- **Inférer des propriétés des données** lorsque c'est coûteux d'étiqueter les données. Ils s'agit par exemple d'identifier :
  - des images similaires, représentant le même objet, animal ou personne ;
  - des textes similaires, qui parlent du même sujet ;
  - dans une image, les points qui appartiennent au même objet (segmentation d'images).

Position du problème

Partitionnement

entrée :

Ensemble de n points / exemples / observations

$$P = \{p_1, p_2, \ldots, p_n\}$$

sortie :

Partition de P

$$C = \{C_1, C_2, \ldots, C_k\}$$

équivalent à une fonction $f : P \to \{1, \ldots, k\}$

Algorithmes de partitionnement

Différentes approches :

- Partitionnement hiérarchique :
  - Regroupement de clusters selon un critère
  - Dendrogramme
- Partitionnement controïde :
  - Utilisation de centres pour paramètrer les clusters
  - k-means
- Partitionnement fondé sur la densité :
  - Selon la densité locale de points, croissance du cluster
  - DBSCAN

Critères de performance

- Forme des clusters
- Stabilité des clusters
- Compatibilité avec des connaissances spécifiques au domaine

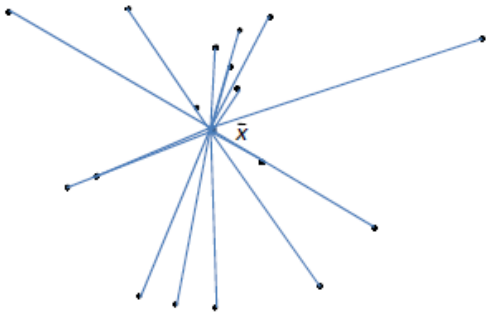
En conclusion

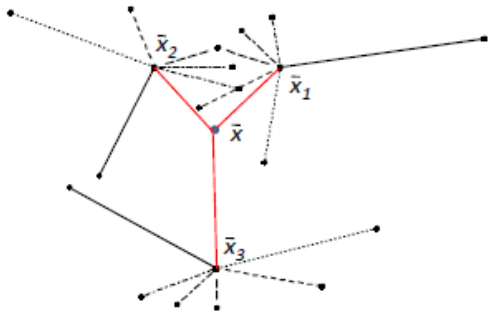
L'apprentissage non supervisé est très subjectif !

Critères de performance

Forme des clusters : On cherche à maximiser la similarité **intra-classe** et à minimiser la similarité **inter-classe**

- centroïde d'un cluster (barycentre) : $\bar{x_k} = \frac{1}{|C_k|} \sum_{x \in C_k} x$
- homogénéité d'un cluster (tightness) : $T_k = \frac{1}{|C_k|} \sum_{x \in C_k} d(x, (\bar{x_k}))$
- moyenne des homogénéités : $T = \frac{1}{K} \sum_{k=1}^{K} T_k$

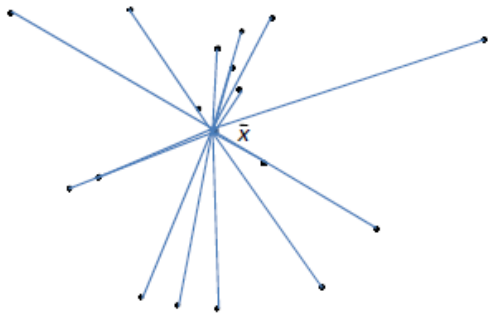


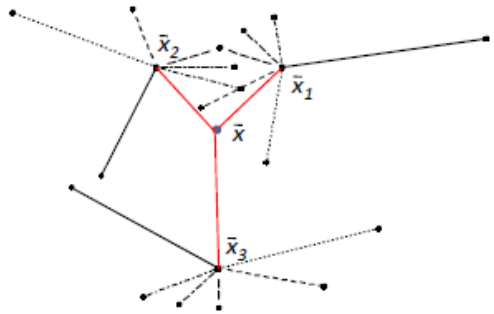


Critères de performance

Forme des clusters : On cherche à maximiser la similarité **intra-classe** et à minimiser la similarité **inter-classe**

- mesure de séparation entre deux clusters : $S_{k,l} = d(\bar{x_k}, \bar{x_l})$
- moyenne de séparation sur l'ensemble des paires de cluster : $S = \frac{2}{k(k-1)} \sum_{k=1}^{K} \sum_{l=k+1}^{K} S_{k,l}$

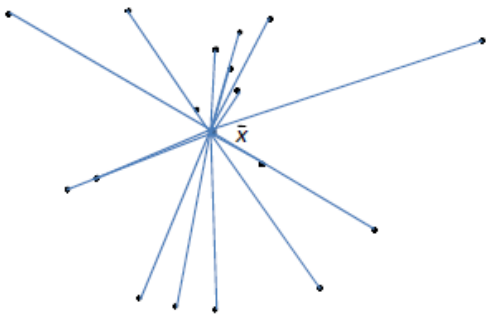


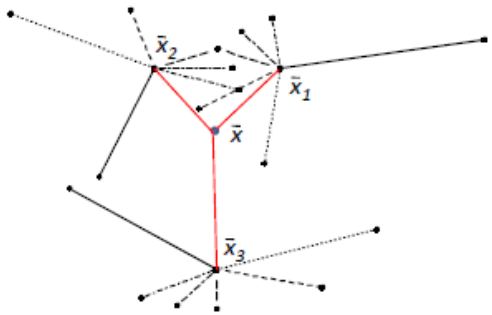


Critères de performance

Forme des clusters : On cherche à maximiser la similarité **intra-classe** et à minimiser la similarité **inter-classe**

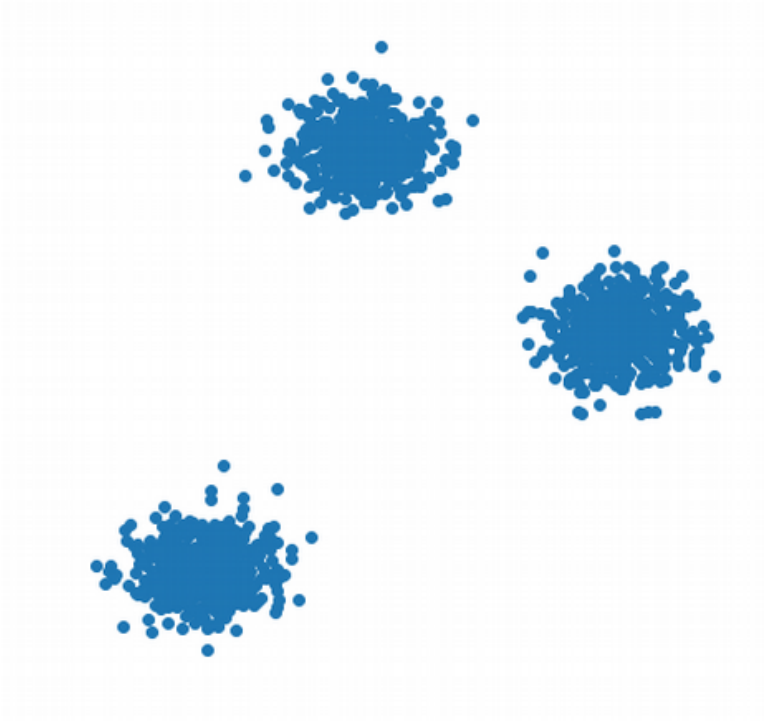
- indice de *Davies-Bouldin* pour comparer les distances **intra-cluster** (l'homogénéité), que l'on veut faibles, aux distances **inter-cluster** (la séparation), que l'on veut grandes. Pour un cluster k cet indice est donné par : $D_k = max_{l \neq k} \frac{T_l + T_k}{S_{k,l}}$
- l'indice de Davies-Bouldin global : $D = \frac{1}{K} \sum_{k=1}^{K} D_k$





Critères de performance

Stabilité des clusters : algorithme peu sensible aux conditions d'initialisation. Critère particulièrement pertinent pour choisir le nombre de clusters.



Critères de performance

Compatibilité avec des connaissances spécifiques au domaine : travailler sur un jeu de données sur lequel on connaît le partitionnement puis comparer cette partition avec celle retournée par notre algorithme de clustering.

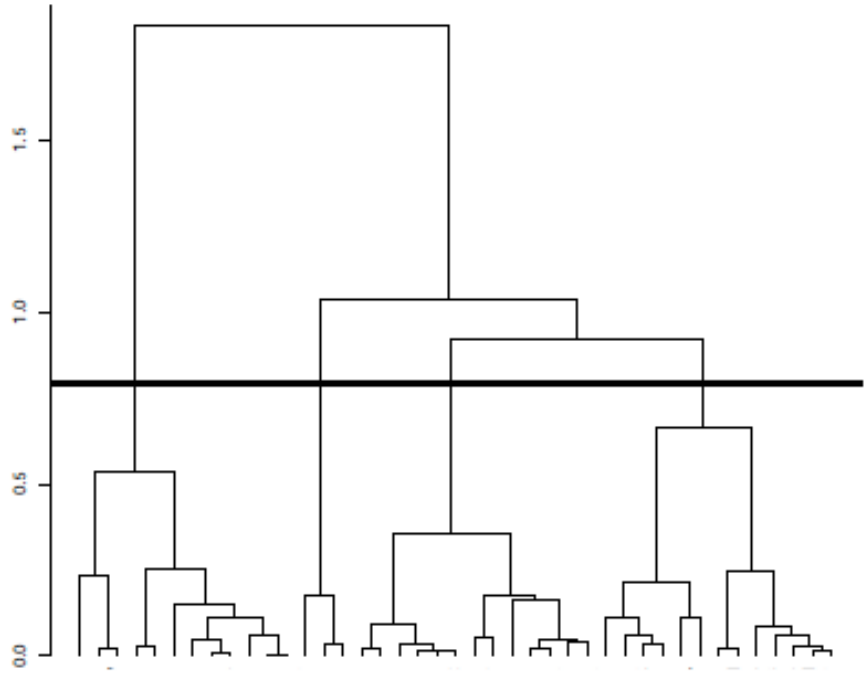
Critères de performance

Compatibilité avec des connaissances spécifiques au domaine : utiliser des mesures de performance spécifiques (l'indice de *Rand*)

Clustering hiérarchique ascendant

Principe :

- Considérer chaque point comme étant un cluster.
- On trouve les deux clusters les plus proches, et on les agglomère en un seul cluster.
- On réitère ces étapes jusqu'à ce que tous les points appartiennent à un seul cluster.



Clustering hiérarchique ascendant

Distances entre deux clusters :

- le **lien simple**, ou *single linkage* : la distance minimale entre deux points, l'un appartenant au premier cluster et l'autre au deuxième.
- le **lien complet**, ou *complete linkage* : la distance moyenne entre toutes les paires de points deux à deux.
- le **lien centroïdal** est aussi appelé UPGMC pour *Unweighted Paired Group Method with Centroid* : la distance entre les moyennes des points de chaque cluster, autrement dit la distance entre les deux centroïdes.

Clustering hiérarchique ascendant

Avantages :

- Il n'est pas nécessaire de définir le nombre de clusters à l'avance.
- Fournit une structure (préférable pour une analyse détaillée).
- Adapté aux échantillons contenant un faible nombre d'individus.

Inconvénients :

- Passage à l'échelle difficile : Complexité en $O(N^2)$.
- Pas de remise en question possible des divisions ou agglomérations : deux classes agglomérées à un niveau ne peuvent pas être séparées à un autre niveau.

k-moyennes

- Principe :
  - Une classe est représentée par son centre de gravité (barycentre).
  - Un objet appartient à la classe dont le centre de gravité lui est le plus proche.
- Algorithme :
  - Fixer le nombre de classes à priori K.
  - Initialiser aléatoirement les centres (K individus tirés au hasard comme représentants des classes).
  - Affecter chaque observation à la classe qui minimise la distance entre le centre et l'observation (centre le plus proche).
  - Recalculer les nouveaux centres (la moyenne).
  - Itérer l'opération jusqu'à stabilité.

Pseudo-code

k-means

Choisir (aléatoirement) k centres $\bar{x_1}, \ldots, \bar{x_k}$

repeat

Affecter chaque observations au centre le plus proche :

$$C_i = \{p_j \ : \ d(p_j, \bar{x_i}) \leq d(p_j, \bar{x_a}) \ \forall a = 1..k\}$$

Mettre à jour les centres (moyenne des clusters) :

$$\bar{x_i} = \frac{1}{|C_i|} \sum_{p_j \in C_i} p_j$$

until plus de changement (convergence)

Le k-means est un algorithme local : il fait décroître la somme des carrés des distances entre les points du cluster, mais rien n'assure que la répartition soit optimale. Il se peut que l'algorithme converge vers un minimum local.

Critère de qualité

Critère de la qualité du partitionnement :

$$U(C) = \sum_{i \in 1}^{k} w(C_i) = \sum_{j=1}^{k} \sum_{i \in C_j} d^2(p_i, \bar{x}_j)$$

Avantages / Inconvénients

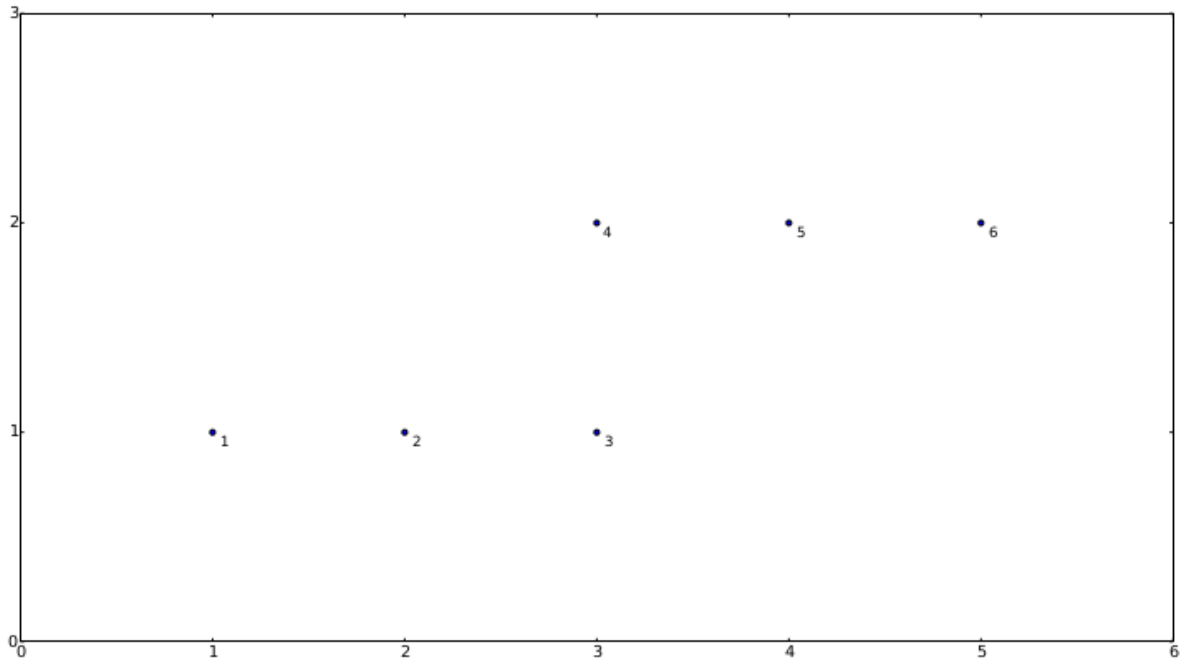
Avantages

- Simple à implémenter et à comprendre.
- Complexité polynomiale : $O(T.K.N)$ où T est le nombre d'itérations et N le nombre d'exemples.

Inconvénients

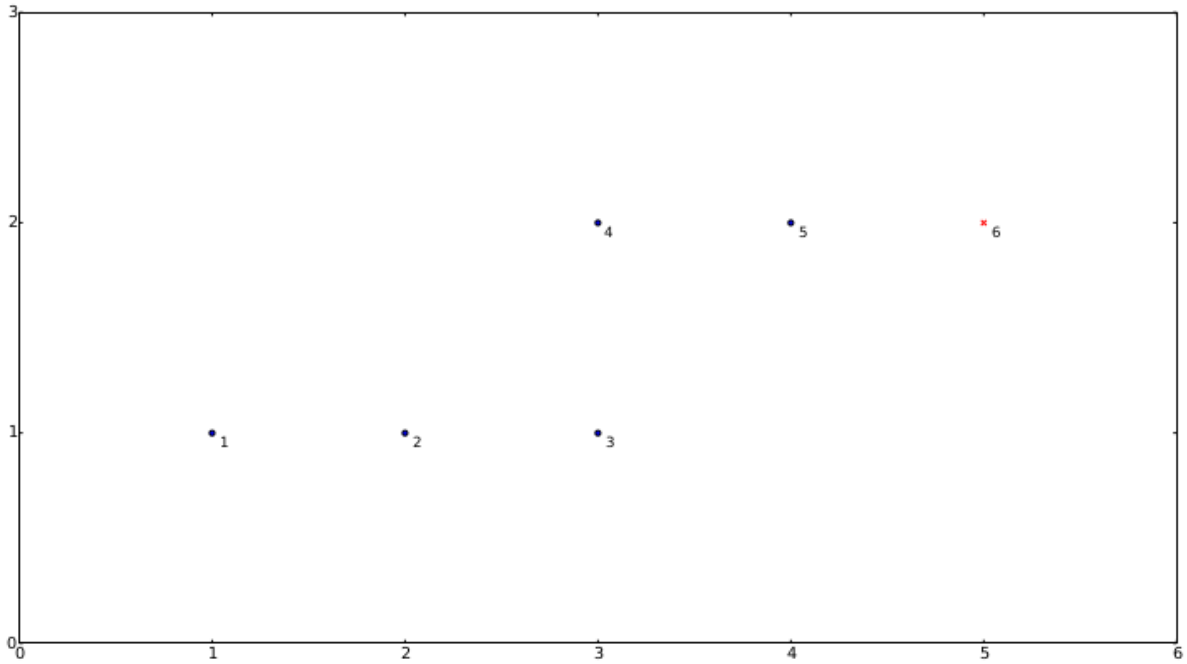
- Nécessite de spécifier le nombre de classes à l'avance.
- Utilisable seulement quand la notion de moyenne existe (donc pas sur des données nominales).
- Sensible au bruit et aux anomalies.
- Sensibilité aux conditions d'initialisation.
- La forme des clusters est supposée "sphérique" et les clusters séparables.

Exemple classique



- Nous avons 6 objets numérotés.
- On cherche à les regrouper en 2 classes.
- On commence par tirer deux points au hasard : par exemple les points 5 et 6.

Exemple classique

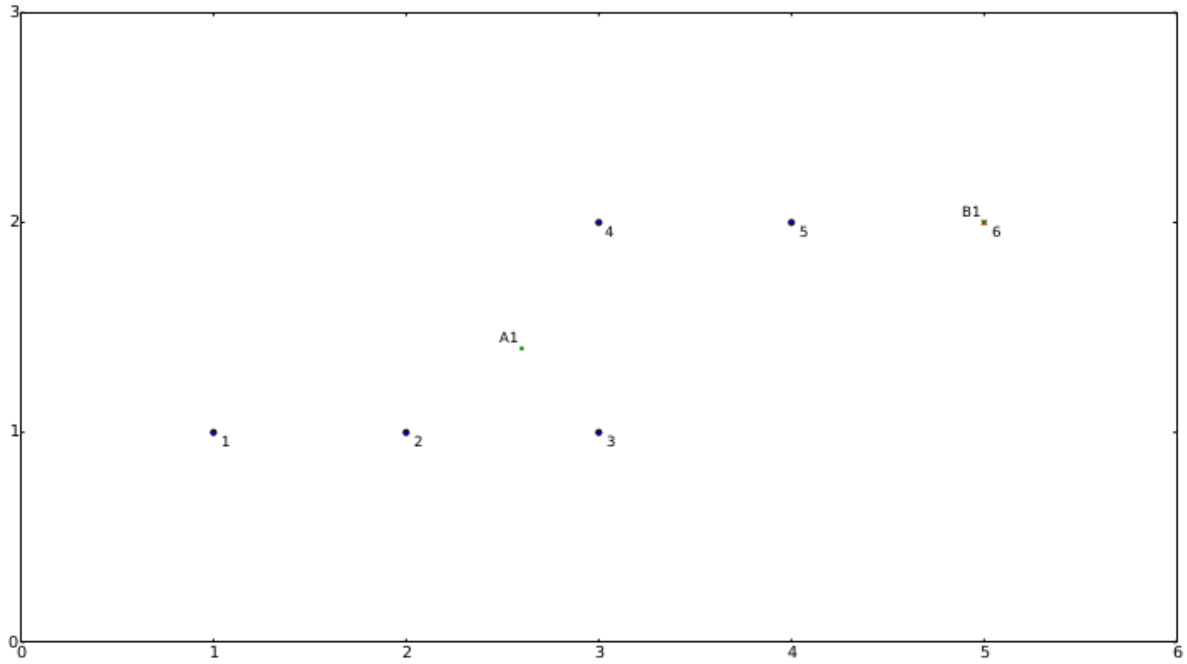


p_i	$d(p_i, \bar{x_1})$	$d(p_i, \bar{x_2})$
1	3.16	4.12
2	2.24	3.16
3	1.41	2.24
4	1.00	2.00
5	0.00	1.00
6	1.00	0.00

On obtient deux clusters :

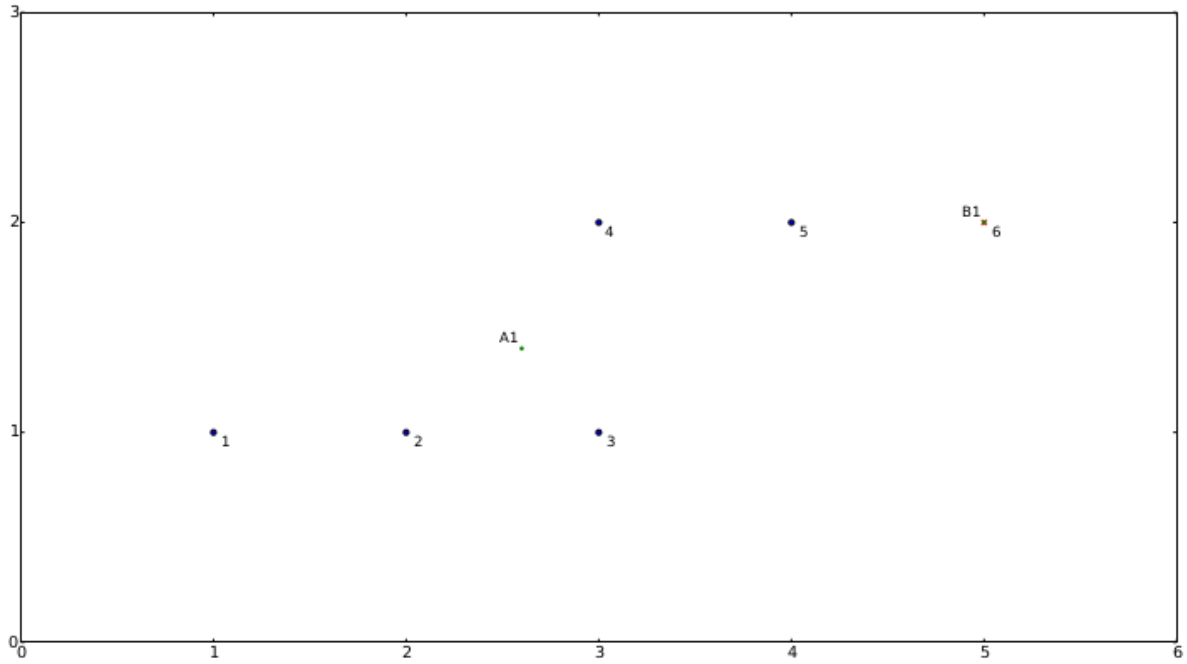
- Le premier $\{1, 2, 3, 4, 5\}$ a comme centre le point 5.
- Le deuxième $\{6\}$ a comme centre le point 6.

Exemple classique



- On calcule les nouveaux centres de chaque cluster.
  - $\bar{x_1} = \frac{1}{5}[(1,1) + (2,1) + (3,1) + (3,2) + (4,2)] = (2.6, 1.4)$
  - $\bar{x_2} = \frac{1}{1}(5,2) = (5,2)$
- On les nomme A_1 et B_1.
- Mesure de qualité $= \sum_{j=1}^{k} \sum_{i \in C_j} d^2(p_i - \bar{x_j}) = 6.4$

Exemple classique

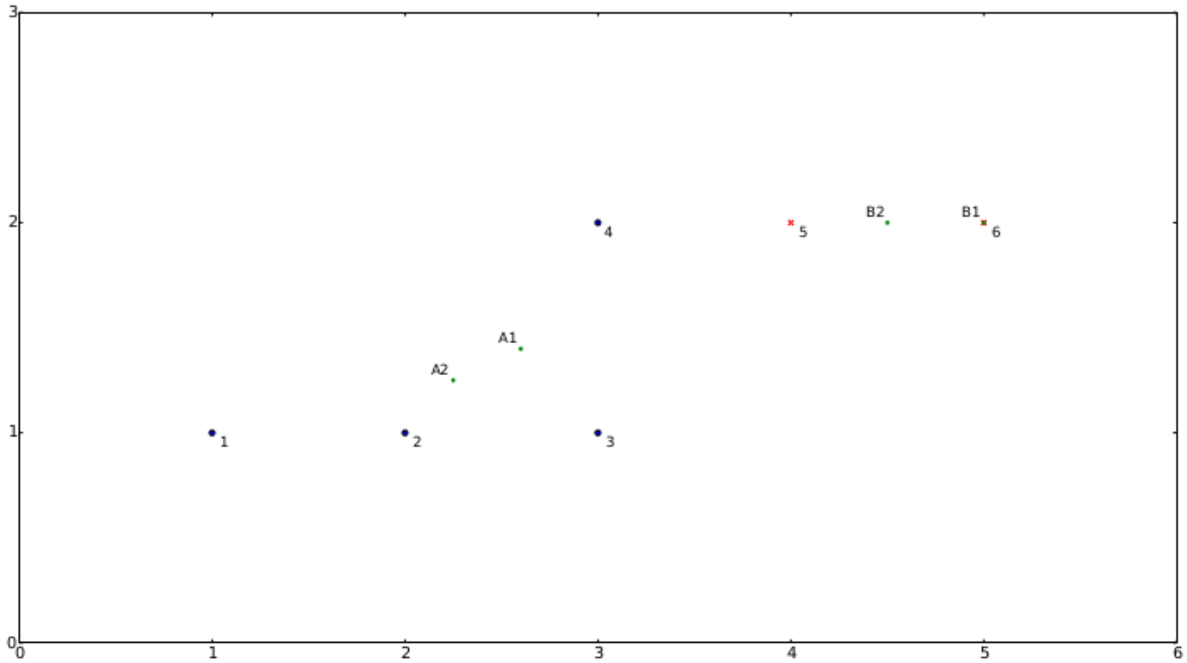


p_i	$d(p_i, \bar{x_1})$	$d(p_i, \bar{x_2})$
1	1.65	4.12
2	0.72	3.16
3	0.57	2.24
4	0.72	2.00
5	1.52	1.00
6	2.47	0.00

On obtient deux clusters :

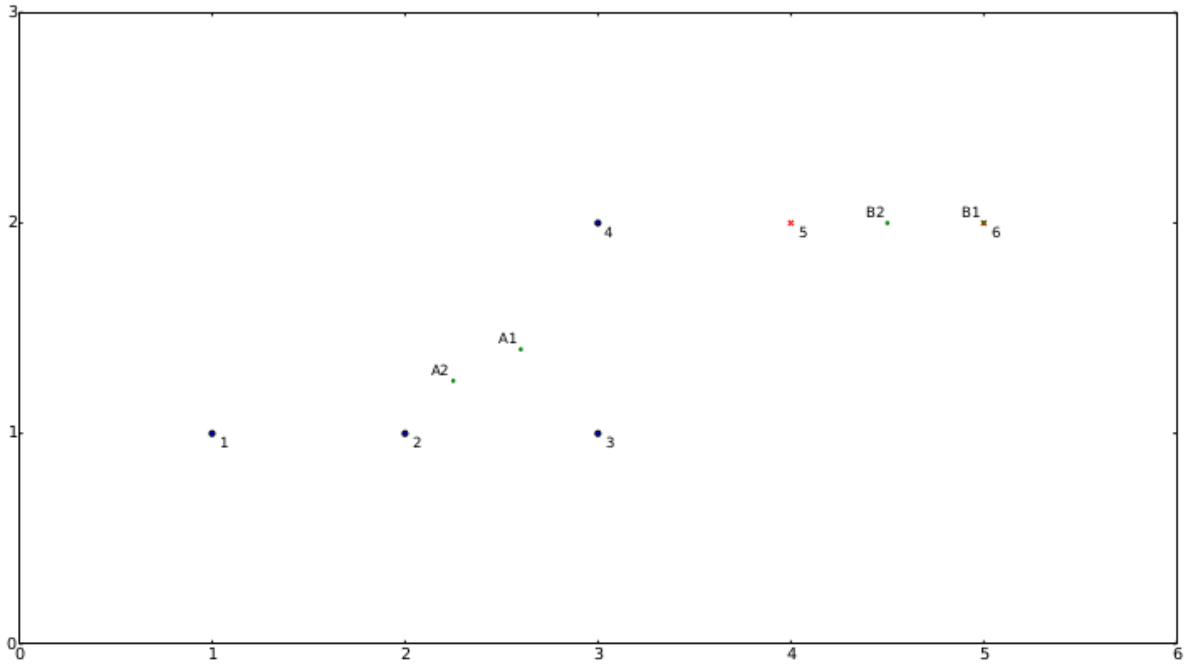
- Le premier $\{1, 2, 3, 4\}$ a comme centre le point A_1
- Le deuxième $\{5, 6\}$ a comme centre le point B_1.

Exemple classique



- On calcule les nouveaux centres de gravité de chaque cluster.
  - $\bar{x_1} = \frac{1}{4}[(1,1) + (2,1) + (3,1) + (3,2)] = (2.25, 1.25)$
  - $\bar{x_2} = \frac{1}{2}[(4,2) + (5,2)] = (4.5, 2)$
- On les nomme A_2 et B_2.
- Mesure de qualité $= \sum_{j=1}^{k} \sum_{i \in C_j} d^2(p_i - \bar{x_j}) = 4$

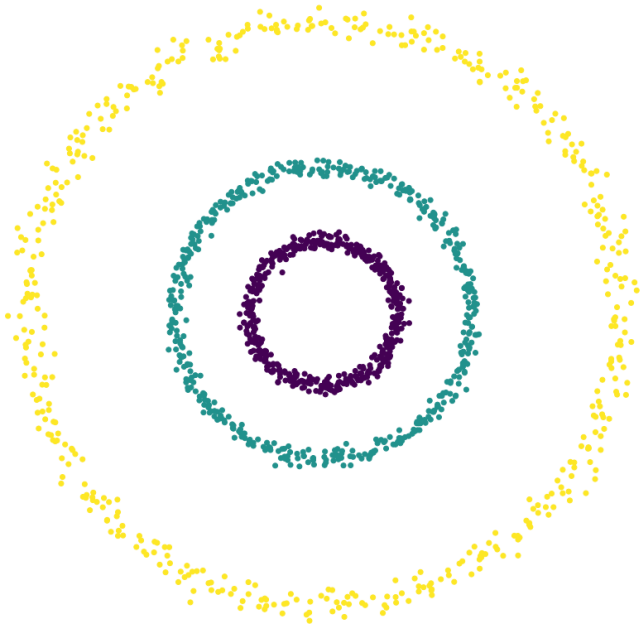
Exemple classique

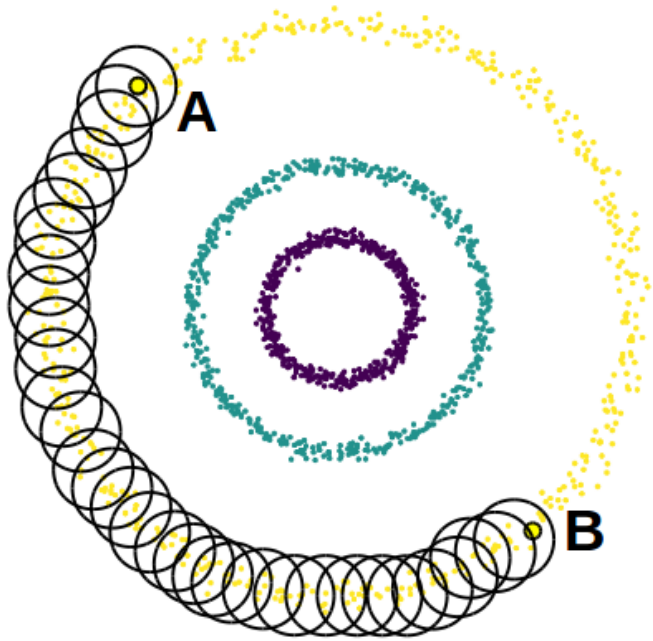


- On arrête le processus car la répartition des points ne change plus.

DBSCAN

L'algorithme DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) a été introduit en 1996 par Martin Ester. Il se base sur la densité estimée des clusters pour effectuer le partitionnement.





Vocabulaire :

- On appelle **epsilon-voisinage** d'un point x l'ensemble des points dont la distance à x est inférieure à ϵ
- On dit que x est un **point intérieur** (*core point* en anglais) si son ϵ-voisinage contient au moins *nmin* points : $|N_\epsilon(x)| \geq nmin$.
- Deux points u et x sont connectés par densité si l'on peut passer de l'un à l'autre par une suite d' ϵ-voisinages contenant chacun au moins *nmin* points.

DBSCAN (P, ϵ , $nmin$)

repeat
  Choisir un point $x \in P$ non déjà visité
  Construire N = voisinage(ϵ, x)
  if $|N| < nmin$ **then**
    Marquer x comme bruit
  else
    Initialiser $C = \{x\}$
    $etendreCluster(C, N, \epsilon, nmin)$
    Ajouter C à la liste des clusters
    Marquer tous les points de C comme visités
  end if
until tous les points ont été visités

etendreCluster (C, N , ϵ, $nmin$)

for $u \in N$ **do**

  if u n'a pas été visité **then**
    $N' = voisinage(\epsilon, u)$
    if $|N'| \geq nmin$ **then**
      $N = N \cup N'$
    end if
  end if
  if u n'appartient à aucun autre cluster **then**
    $C = C \cup \{u\}$
  end if
end for

Avantages / Inconvénients

Avantages

- Efficace en temps de calcul sans avoir besoin de prédéfinir le nombre de clusters.
- Permet de trouver des clusters de forme arbitraire.

Inconvénients

- Difficile à utiliser en très grande dimension.
- Le choix des paramètres ϵ et *nmin* peut être délicat.
- Difficulté de trouver des clusters de densités différentes.