Machine Learning 4DS

Techniques de Segmentation

Mohamed Heny SELMI

medheny.selmi@esprit.tn

Enseignant et Responsable option Data Science à ESPRIT

Deux familles de techniques

Apprentissage Non Supervisé

Analyse descriptive

- ✓ Fournissent directement des résultats : à interpréter et à utiliser !
- ✓ visent à mettre en évidence des connaissances présentes mais cachées par le volume des données
- ✓ réduisent, résument, synthétisent les masses de données
- ✓ pas de variable « cible »

Apprentissage supervisé

Analyse prédictive

- ✓ Fournissent un modèle (et non pas des résultats), créé à partir d'un entrepôt d'apprentissage, testé et validé sur un entrepôt de test, et utilisé dans les problèmes de prise de décision sur des entrepôts de travail
- ✓ visent à découvrir de nouvelles informations à partir des informations présentes : connaissances, décisions
- ✓ expliquent mieux les données
- ✓ Une(des) variable(s) « cible(s)»

Segmentation – Définition

Ensemble de méthodes d'apprentissage non supervisé, qui visent à diviser une ensemble de données en différents sous-ensembles homogènes, de telle façon les données de chaque partition partagent des caractéristiques communes, qui correspondent le plus souvent à des critères de proximité (similarité au sens informatique) que l'on définit en introduisant des mesure de distance entre individus et/ou classes.

Utilité de la segmentation

- Banque / Assurance
  - Catégoriser la clientèle : chercher un profil qui représente les membres de chaque classe
  - Regrouper les clients selon des critères et caractéristiques communs : cibler « les mailing »
- Médecine
  - Déterminer des segments de patients susceptibles d'être soumis à des protocoles thérapeutiques déterminés, chaque segment regroupant tous les patients réagissant identiquement
  - Retrouver les différents comportements similaires
- Biologie – Zoologie – Ethologie – Sciences humaines
  - Expliquer les relations entre espèces, races, genres, familles,
  - Retrouver de nouvelles répartitions

- Profiling
  - Analyse sémantique, sentimentale,
  - Analyse et mesure de la tonalité d'un contenu textuel
  - Catégorisation des concepts ou des entités nommés
  - Construction d'agrégateur synthétique à partir des flux d'actualités

Problématique – Données purement ***quantitatives***

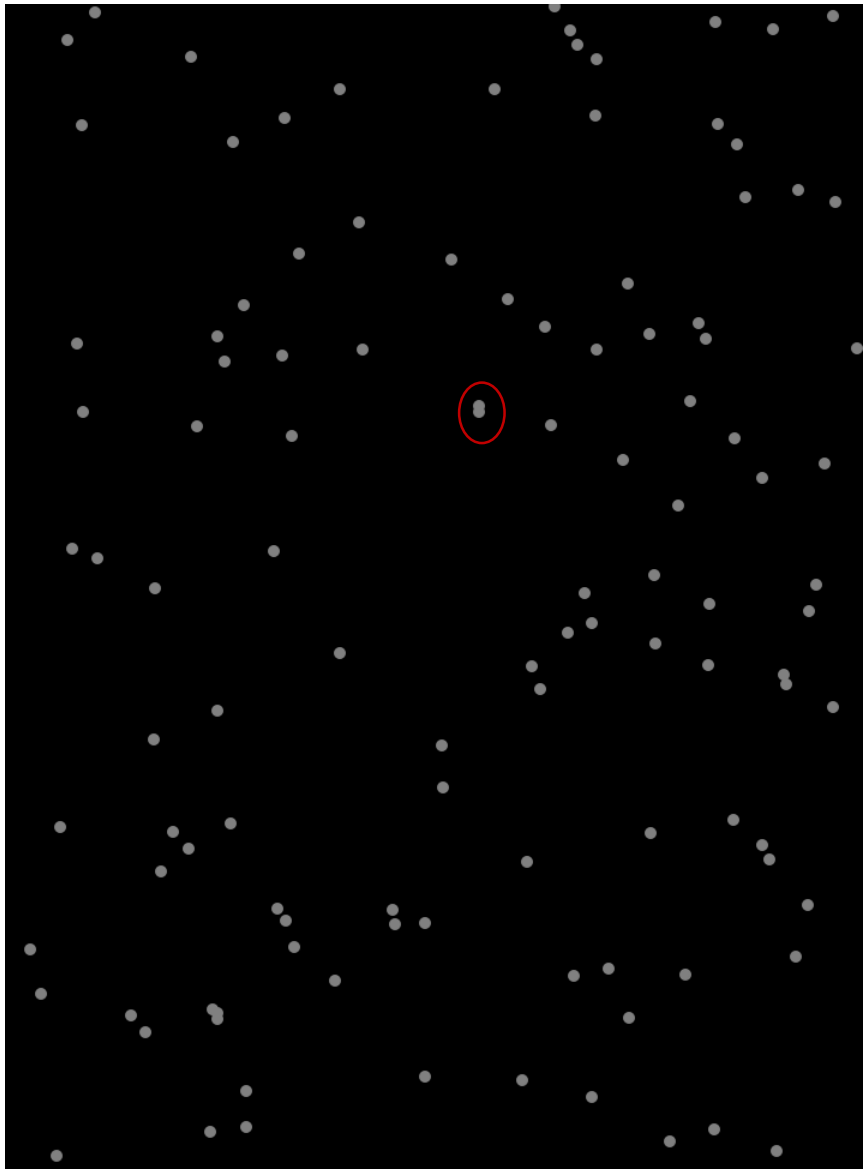
How many clusters?

Six Clusters

Two Clusters

Four Clusters

Idée de base de la segmentation



Deux individus se ressemblent le plus

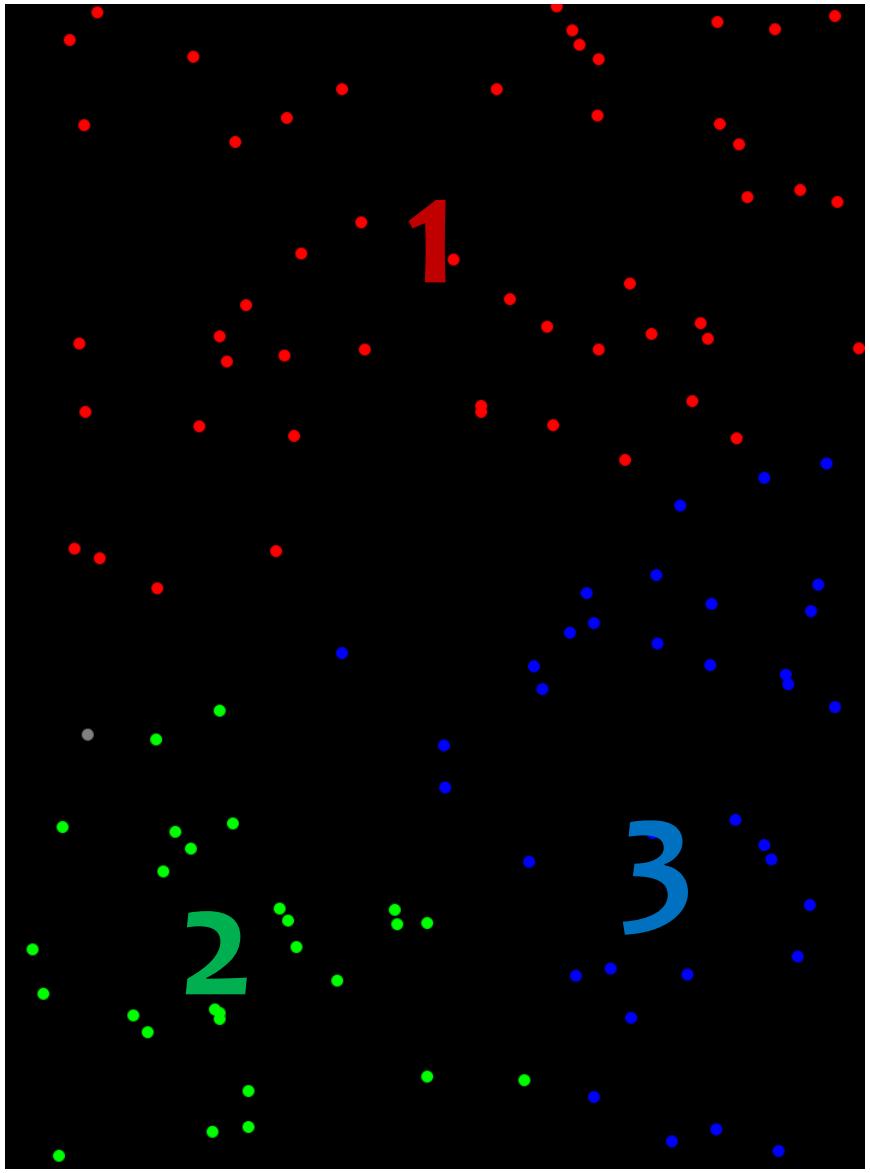
SI

les points qui les représentent dans le nuage sont les plus proches

Nécessité d'une métrique de la distance

Distance Euclidienne
Distance de Mahalanobis
Distance de Manhattan
Distance de Ward
….

Idée de base de la segmentation



Deux individus se ressemblent le plus

SI

les points qui les représentent dans le nuage sont les plus proches

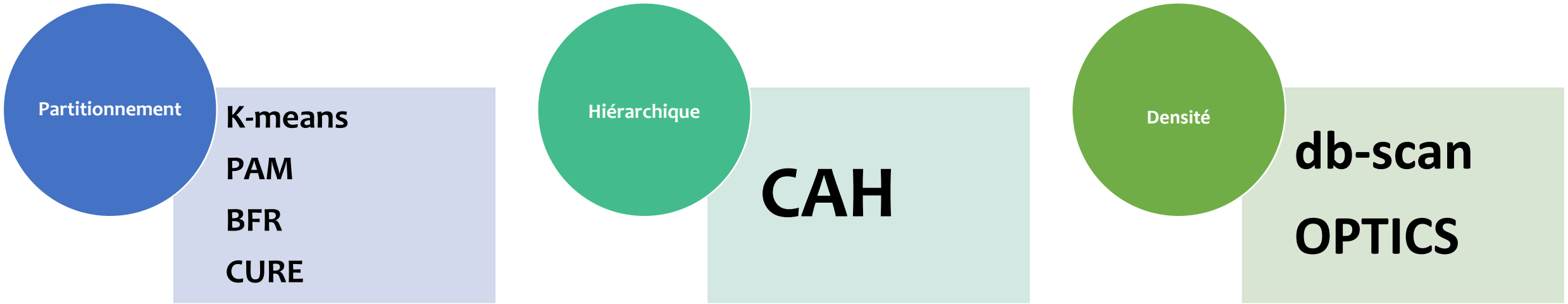
Nécessité d'une métrique de la distance

Distance Euclidienne
Distance de Mahalanobis
Distance de Manhattan
Distance de Ward
……

Un critère d'évaluation d'une segmentation

$$I_{tot} = I_{inter} + I_{intra}$$

Familles de méthodes



K-means

Méthode des Centres Mobiles

Med Heny SELMI

Principe algorithmique

Algorithme K-Means

Entrée : *k le nombre de groupes cherchés*

DEBUT

Choisir aléatoirement les centres des groupes

REPETER

i. ***Affecter chaque cas au groupe dont il est le plus proche au son centre***

ii. ***Recalculer le centre de chaque groupe***

JUSQU'A *(stabilisation des centres)*

OU *(nombre d'itérations =t)*

OU *(stabilisation de l'inertie totale de la population)*

FIN

stabilisation de l'inertie totale de la population

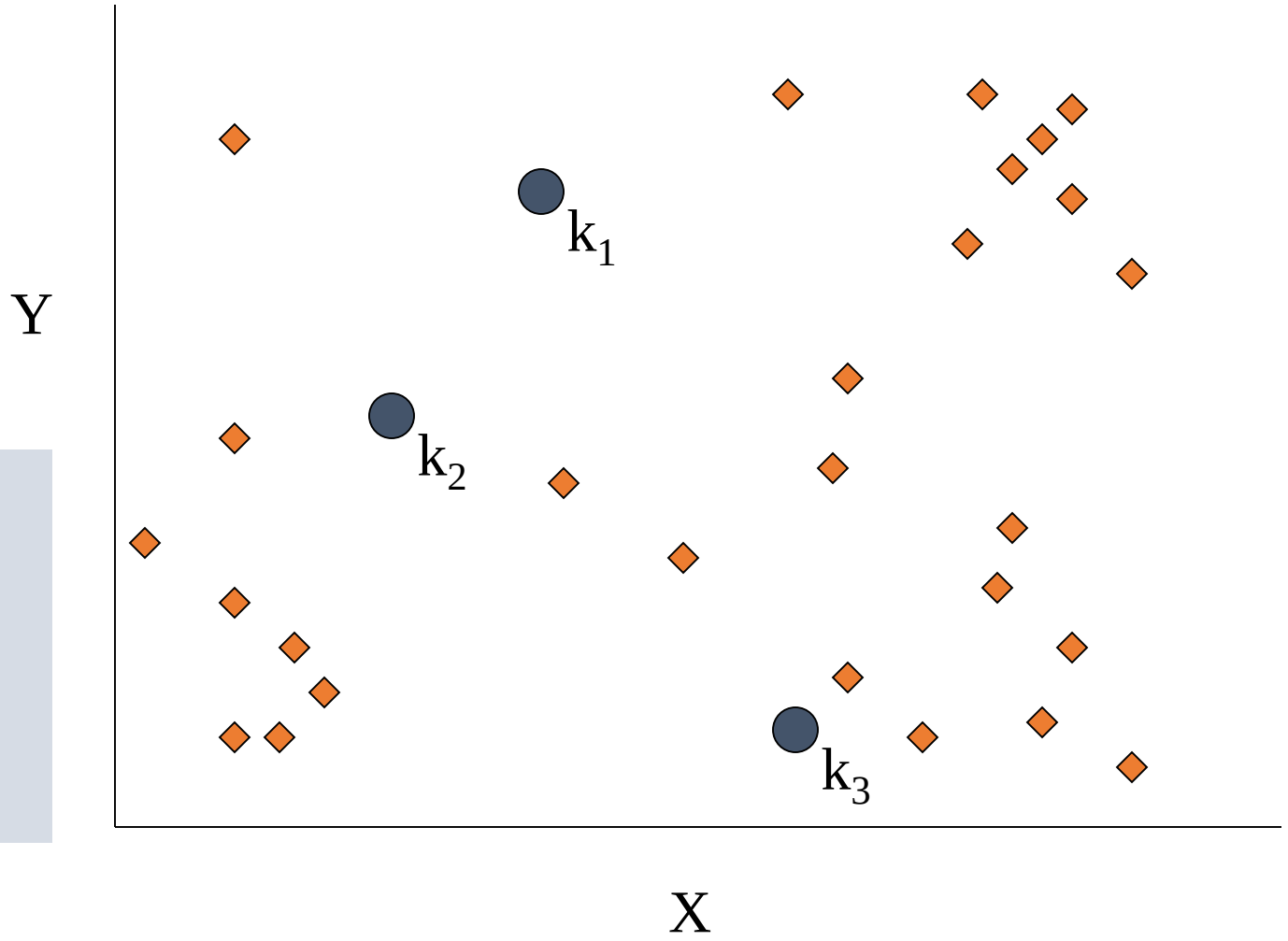
- Inertie totale I_{tot} : somme de l'inertie intraclasse I_A et de l'inertie interclasse I_c

$$I_{tot} = I_A + I_c$$

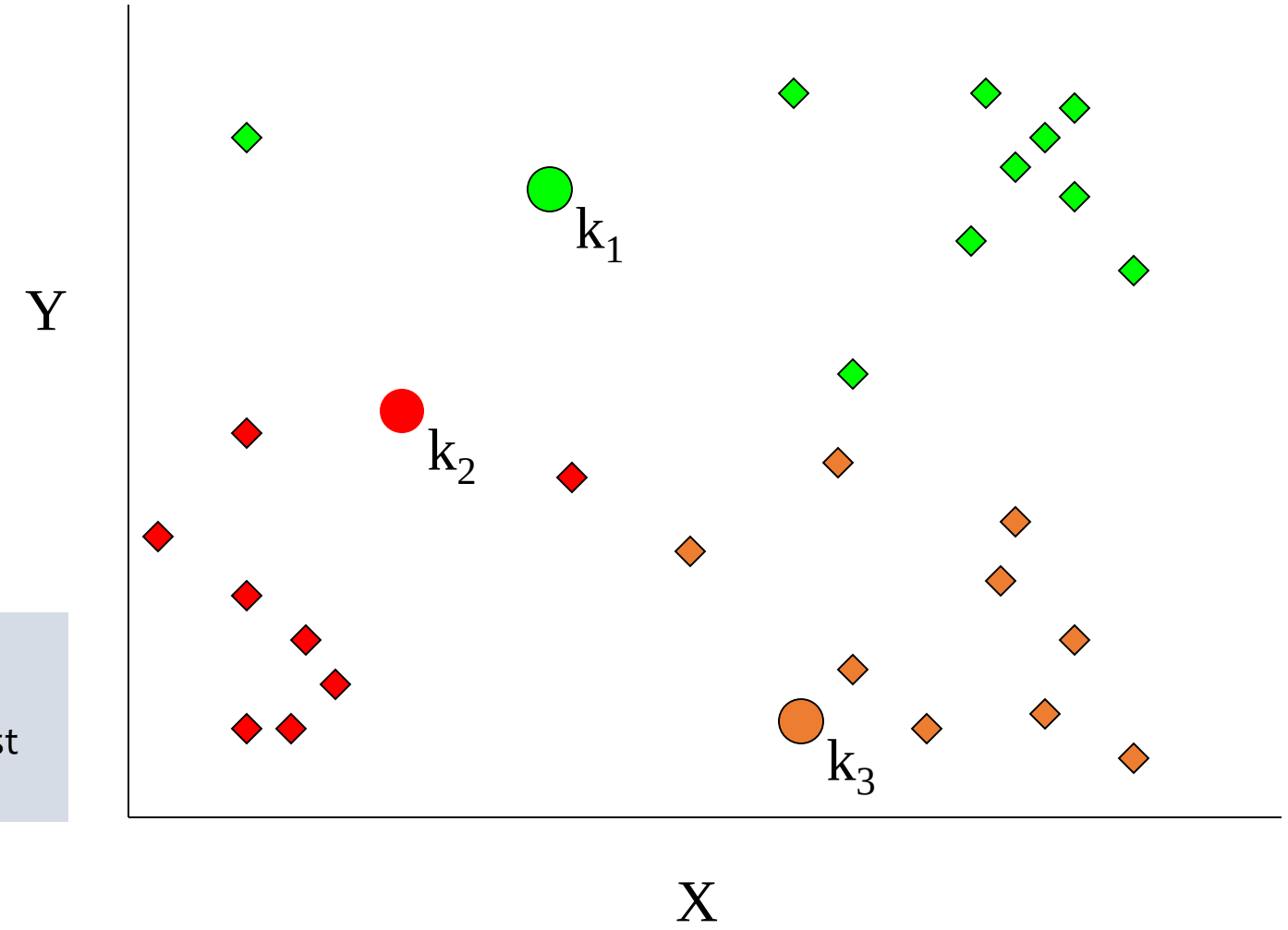
- Inertie intraclasse I_A : somme des inerties totales de chaque classe
- Inertie interclasse I_c : moyenne (pondérée par la somme des poids de chaque classe) des carrés des distances des barycentres de chaque classe au barycentre global

Simulation du K-Means 1

Choisir **3** Centres de classes (au hasard)

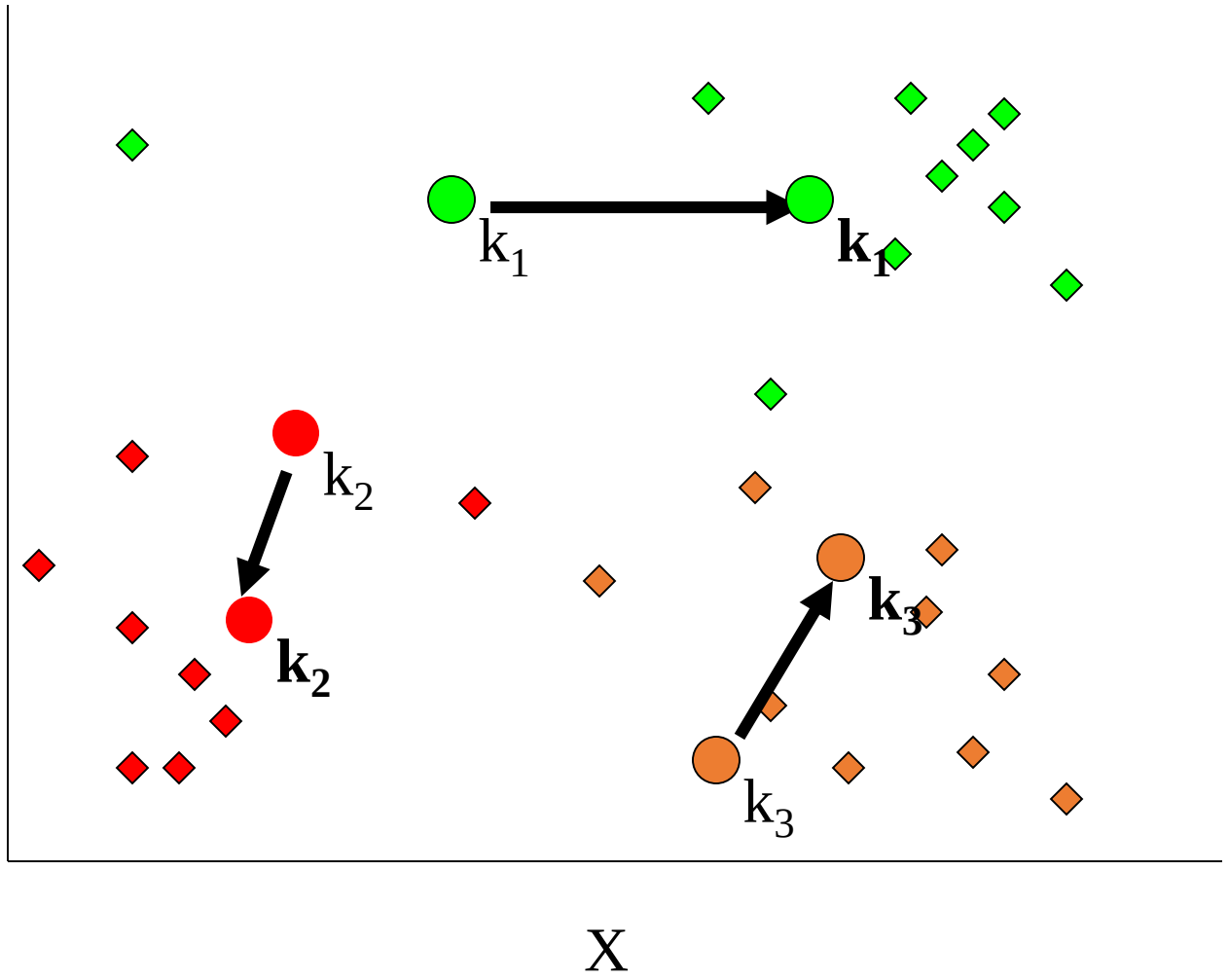


Simulation du K-Means 2



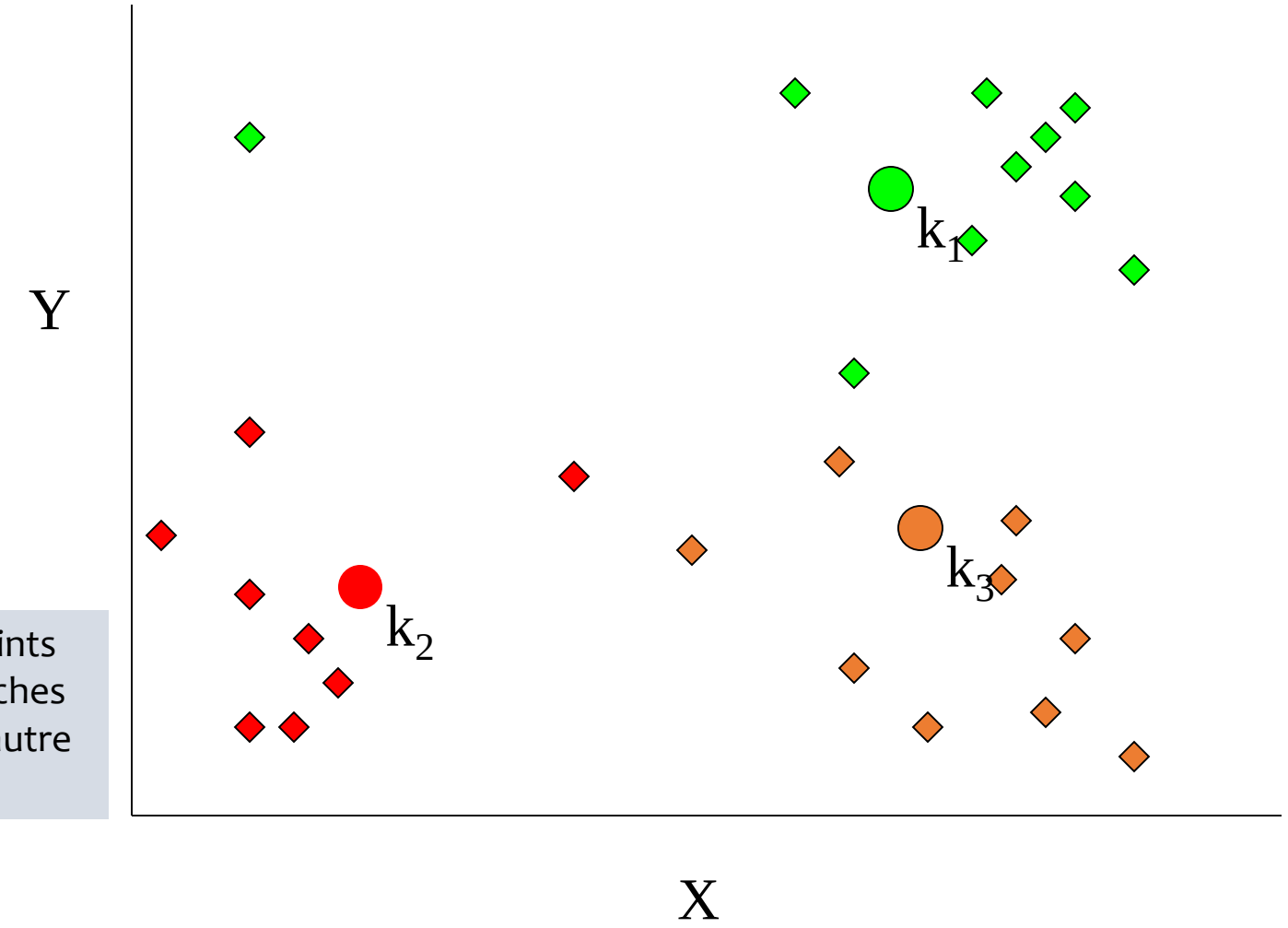
Affecter chaque point à la classe dont le centre est le plus proche

Simulation du K-Means 3



Déplacer chaque centre de classe vers la moyenne de chaque classe

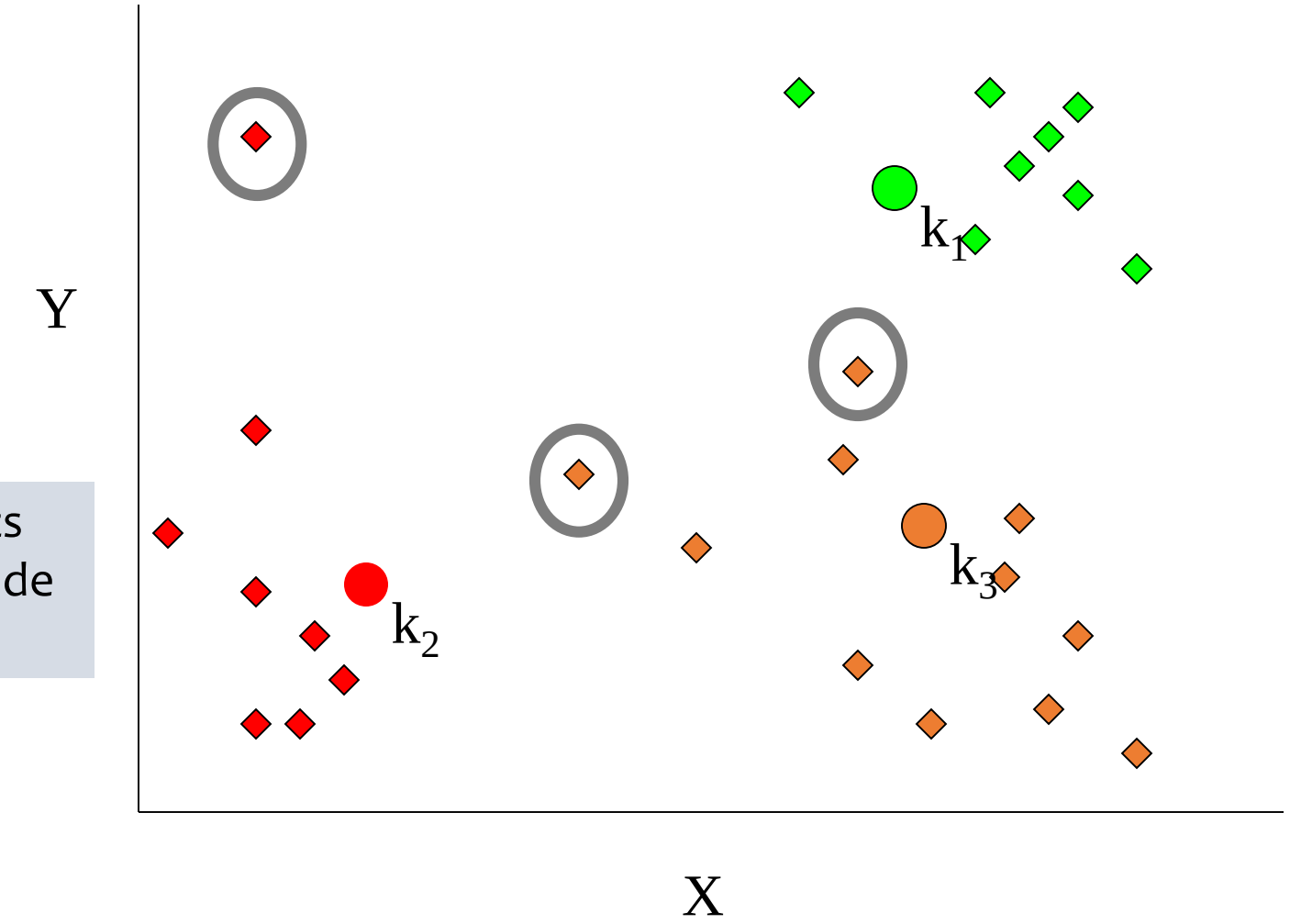
Simulation du K-Means 4



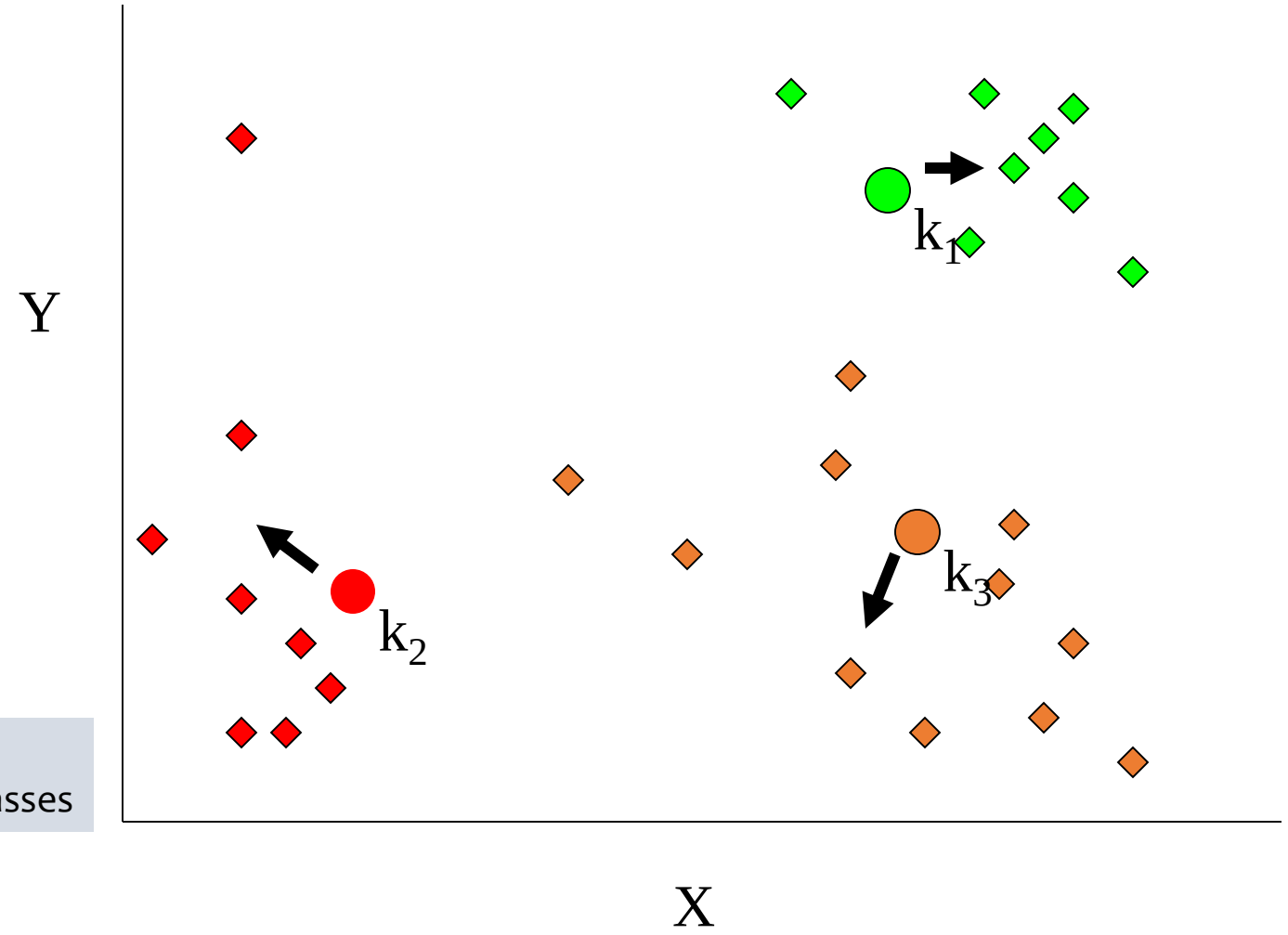
Réaffecter les points qui sont plus proches du centre d'une autre classe

Simulation du K-Means 5

les trois points qui changent de classe

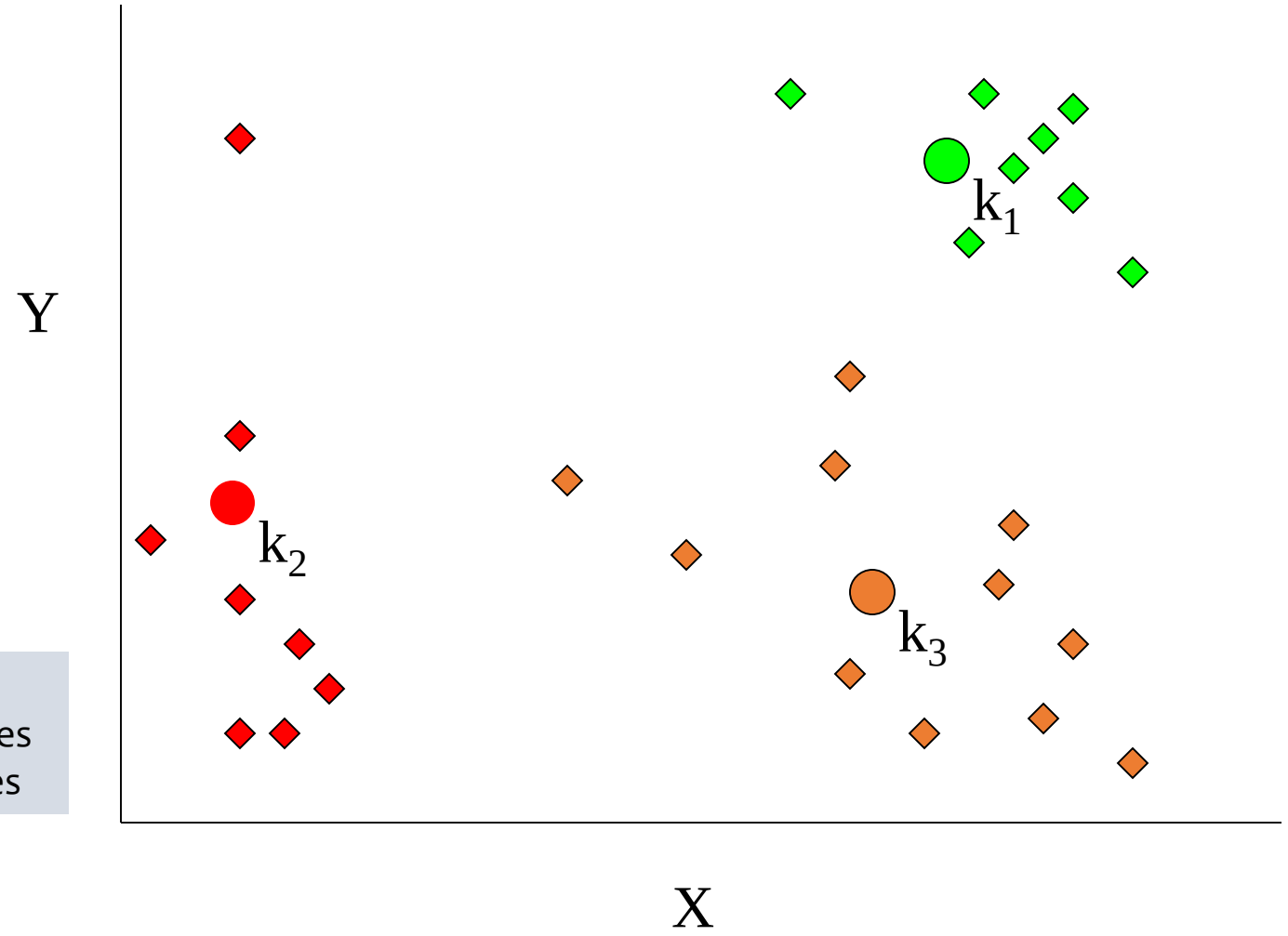


Simulation du K-Means 6



Re-calculer les moyennes des classes

Simulation du K-Means 7



Déplacer les centres des classes vers les moyennes

Points Faibles de K-means

- Le choix du nombre de groupes est subjectif dans le cas où le nombre de classes est inconnu au sein de l'échantillon.
- L'algorithme du K-Means ne trouve pas nécessairement la configuration la plus optimale correspondant à la fonction objective minimale.
- Les résultats de l'algorithme du K-Means sont sensibles à l'initialisation aléatoires des centres.

PAM

Partitioning Around Medoids

Med Heny SELMI

Principe du PAM

- similaire à K-means, car les deux sont des algorithmes qui consiste à segmenter l'ensemble de données en groupes.
- utilise les Medoids, qui sont des entités représentant le groupe dans lequel ils sont insérés, à la différence de K-means qui travaille avec centroïdes.
- PAM partitionne l'ensemble de données de n objets en k clusters, (k est une entrée de l'algorithme).
- fonctionne avec une matrice de dissemblance, dont le but est de réduire au minimum la dissemblance (dissimilarité) globale entre les représentants de chaque groupe et de ses membres.

Algorithme PAM – Phase de ***construction***

1. Choisissez les entités k pour devenir des medoids.

2. Calculer la **matrice de dissemblance**.

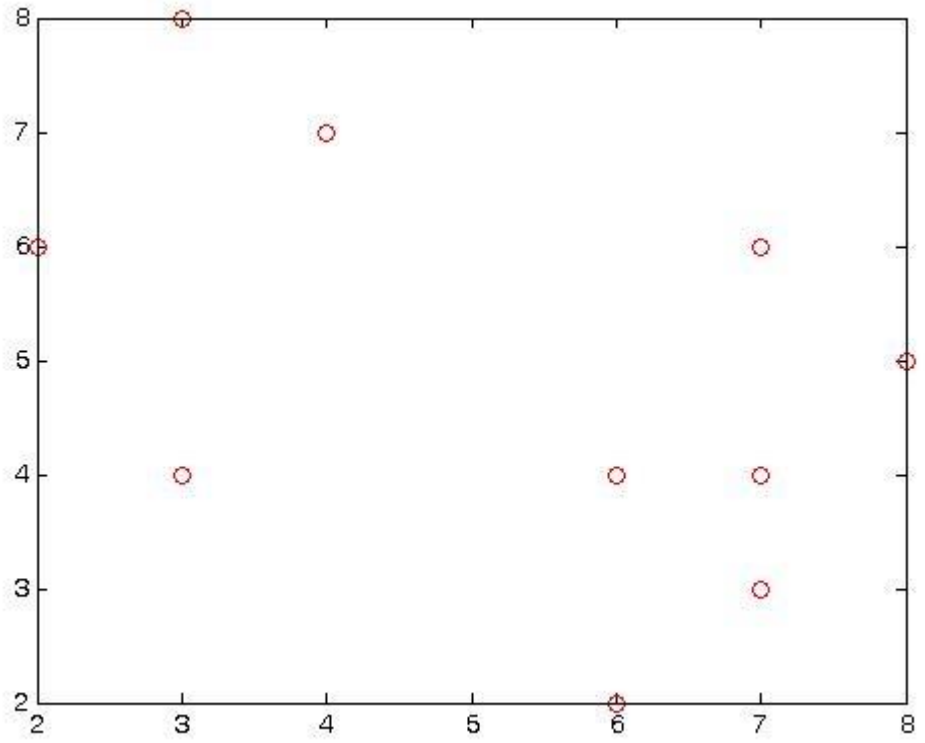
3. Attribuez chaque entité à son medoid le plus proche

Algorithme PAM – Phase de ***swap***

4. Pour chaque recherche de cluster , si l'une des entités de la cluster diminue le coefficient de dissimilarité moyen, si c’est le cas, sélectionnez l'entité qui abaisse ce coefficient le plus, en tant que medoid pour cette cluster .
5. Si au moins une medoide a changé aller à (3), sinon mettre fin à l'algorithme.

Exemple d'exécution PAM

X_1	2	6
X_2	3	4
X_3	3	8
X_4	4	7
X_5	6	2
X_6	6	4
X_7	7	3
X_8	7	4
X_9	8	5
X_{10}	7	6



Exemple d'exécution PAM – itération1

Supposons que x2 et x8 sont sélectionnés en tant que medoid, donc les centres sont c1 = (3,4) et c2 = (7,4)

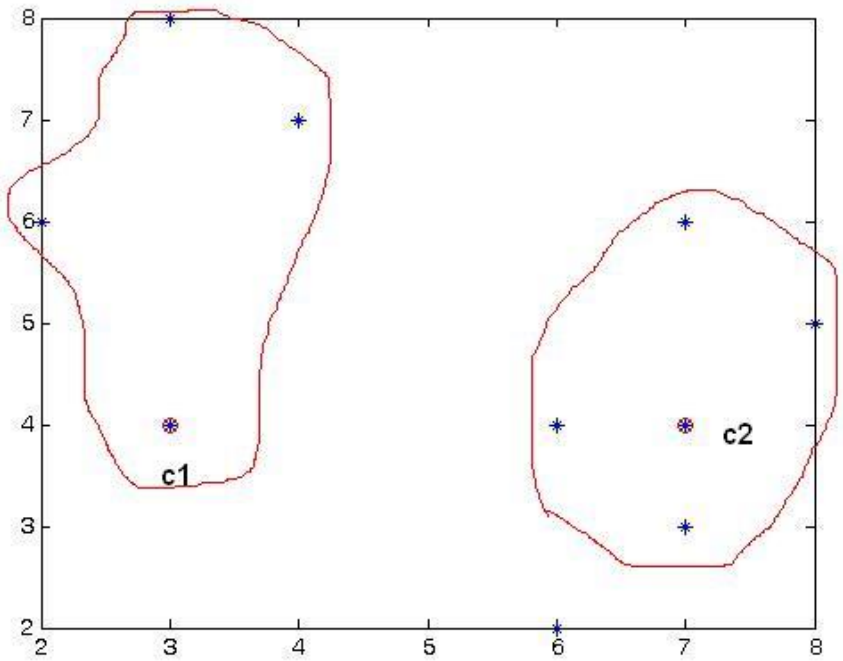
Cost (distance) to c_1					
i	c_1		Data objects (X_i)		Cost (distance)
1	3	4	2	6	**3**
3	3	4	3	8	**4**
4	3	4	4	7	**4**
5	3	4	6	2	5
6	3	4	6	4	3
7	3	4	7	3	5
9	3	4	8	5	6
10	3	4	7	6	6

Cost (distance) to c_2					
i	c_2		Data objects (X_i)		Cost (distance)
1	7	4	2	6	7
3	7	4	3	8	8
4	7	4	4	7	6
5	7	4	6	2	**3**
6	7	4	6	4	**1**
7	7	4	7	3	**1**
9	7	4	8	5	**2**
10	7	4	7	6	**2**

$Cluster_1$ = {(3,4)(2,6)(3,8)(4,7)}

$Cluster_2$ = {(7,4)(6,2)(6,4)(7,3)(8,5)(7,6)}

Exemple d'exécution PAM – itération1



$$\begin{aligned}\text{total cost} &= \{\text{cost}((3,4),(2,6)) + \text{cost}((3,4),(3,8)) + \text{cost}((3,4),(4,7))\} \\ &+ \{\text{cost}((7,4),(6,2)) + \text{cost}((7,4),(6,4)) + \text{cost}((7,4),(7,3)) \\ &+ \text{cost}((7,4),(8,5)) + \text{cost}((7,4),(7,6))\} \\ &= (3+4+4) + (3+1+1+2+2) \\ &= 20\end{aligned}$$

Exemple d’exécution PAM – itération2

Sélectionnez l'un des non-medoide.

Supposons que O '= (7,3), (x7) .

Alors maintenant les medoides sont c1 (3,4) et O '(7,3).

i	c_1		Data objects (X_i)		Cost (distance)
1	3	4	2	6	**3**
3	3	4	3	8	**4**
4	3	4	4	7	**4**
5	3	4	6	2	5
6	3	4	6	4	3
8	3	4	7	4	4
9	3	4	8	5	6
10	3	4	7	6	6

i	O'		Data objects (X_i)		Cost (distance)
1	7	3	2	6	8
3	7	3	3	8	9
4	7	3	4	7	7
5	7	3	6	2	**2**
6	7	3	6	4	**2**
8	7	3	7	4	**1**
9	7	3	8	5	**3**
10	7	3	7	6	**3**

Exemple d’exécution PAM – itération2

Calculer le coût total

$$\begin{aligned}\text{total cost} &= 3+4+4+2+2+1+3+3\\ &= 22\end{aligned}$$

Coût de l'échange de medoid de c2 à O ‘ est

$$\begin{aligned}S &= \text{current total cost} - \text{past total cost}\\ &= 22-20\\ &= 2>0.\end{aligned}$$

Alors passer à O‘ serait une mauvaise idée, le choix précédent était bon.

Donc, nous essayons d'autres non-medoide et on constate que notre premier choix était le meilleur.

La configuration ne change pas et l'algorithme se termine ici.

CAH

Classification Ascendante Hiérarchique

Principe algorithmique

i. Créer à chaque étape une partition en agrégeant 2 à 2 les éléments les plus proches !

   Eléments : individus ou groupe d'individus

ii. L'algorithme fournit une hiérarchie de partitions : arbre contenant l'historique de la classification et permettant de retrouver n-1 partitions : Dendrogramme

iii. Nécessité de se munir d'une métrique (distance euclidienne, chi2, Ward…)

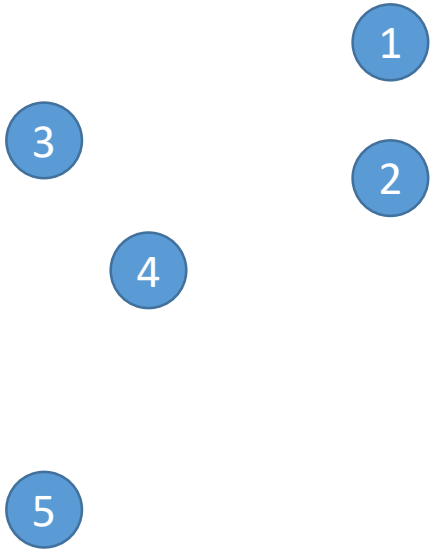
iv. Nécessité de fixer une règle pour agréger un individu et un groupe d'individus (ou bien 2 groupes d'individus)

Le dendrogramme

- Durant les étapes d'un algorithmes de classification hiérarchique, on est en train de construire un dendrogramme.
- Le dendrogramme indique les objets et classes qui ont été fusionnées à chaque phase.
- Le dendrogramme indique aussi la valeur du critère choisi pour chaque partition rencontrée
- Il donne un résumé de la classification hiérarchique
- Chaque palier correspond à une fusion de classes
- Le niveau d'un palier donne une indication sur la qualité de la fusion correspondante
- **Toute coupure horizontale correspond à une partition**

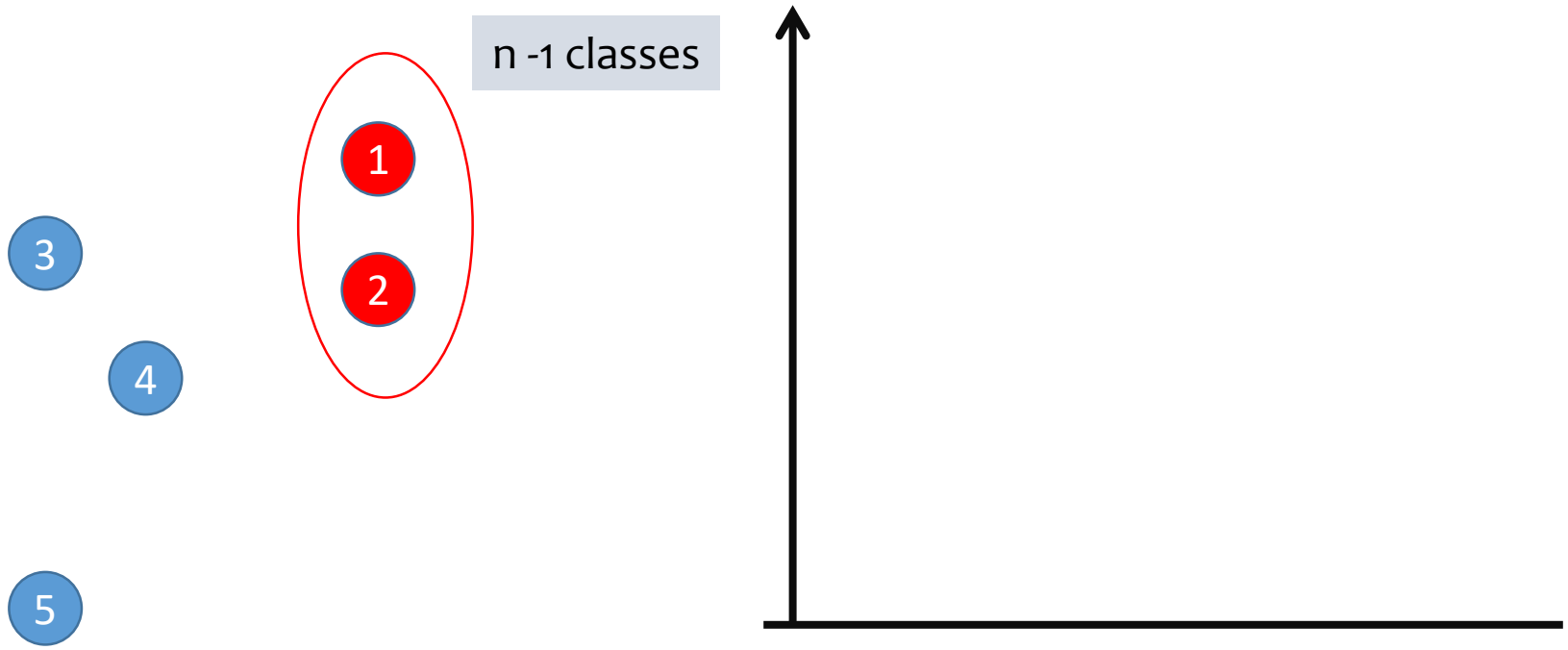
Simulation du CAH 1

n individus / n classes

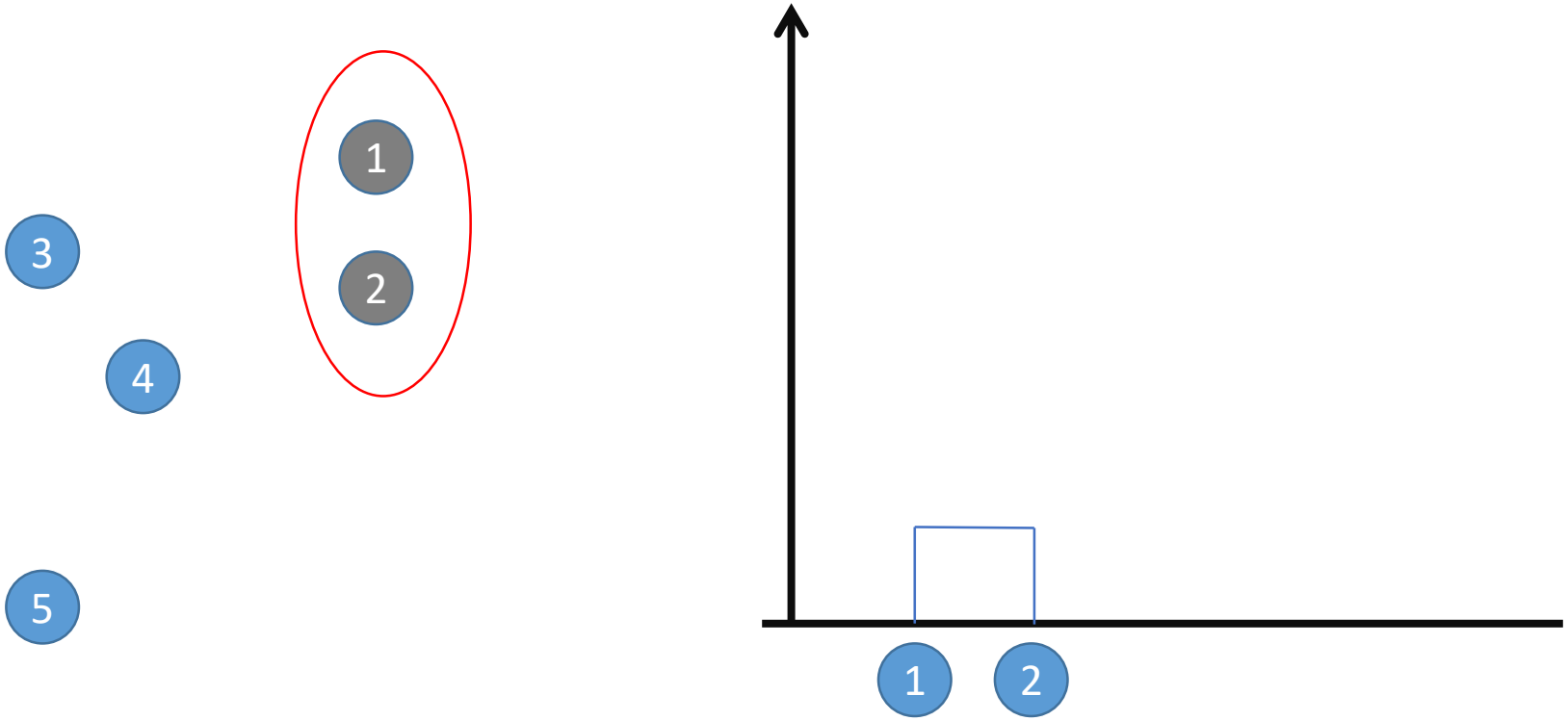


On construit la matrice de distance entre les n éléments
et on regroupe les 2 éléments les plus proches

SIMULATION DU CAH 2



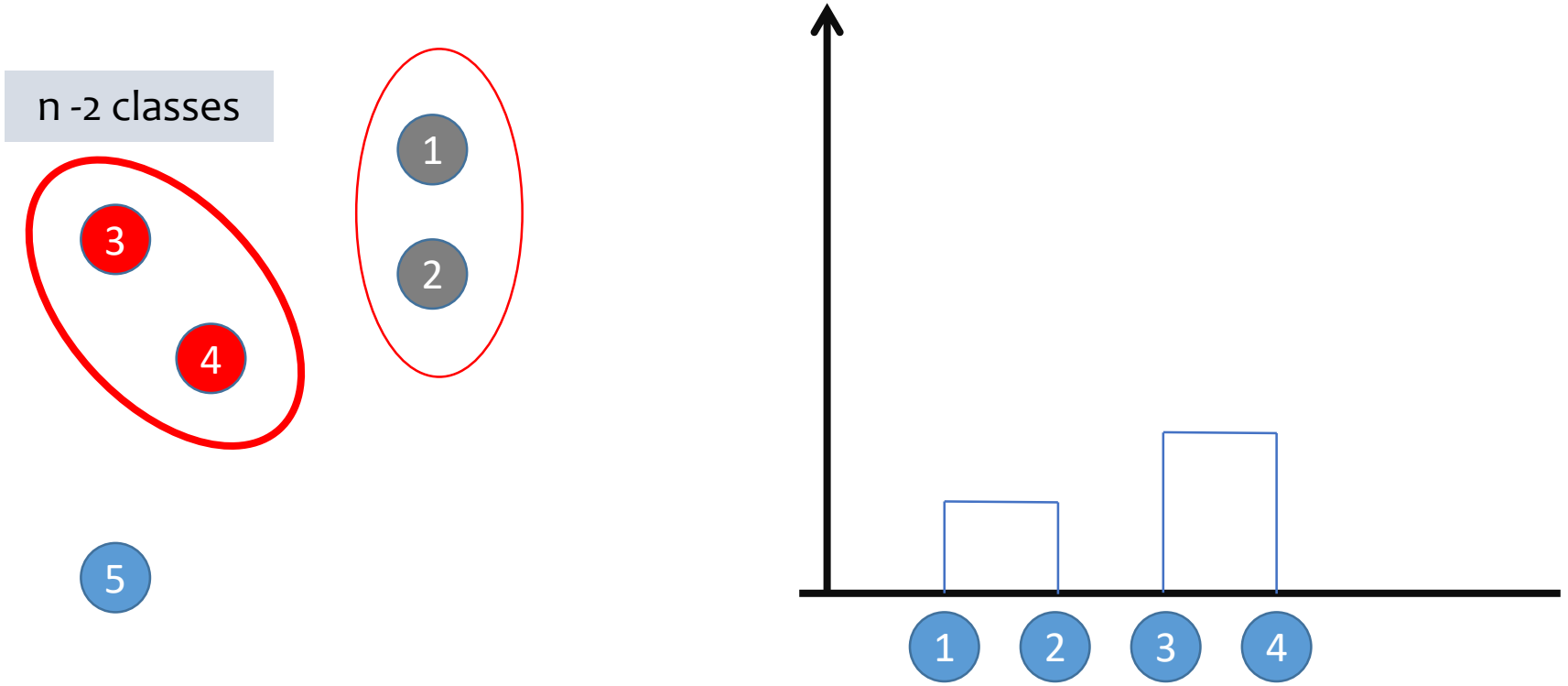
SIMULATION DU CAH 3



Comment mesurer la distance entre une classe et un élément individuel ?

Critères des centres de gravité, de la distance minimale, maximale, critère de Ward…

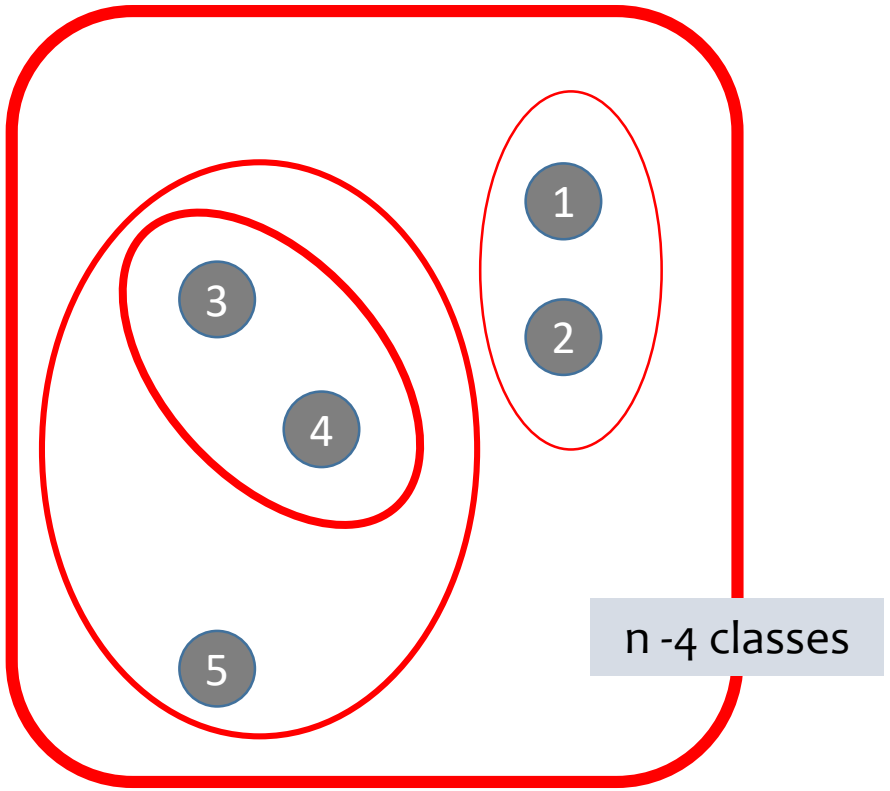
SIMULATION DU CAH 4

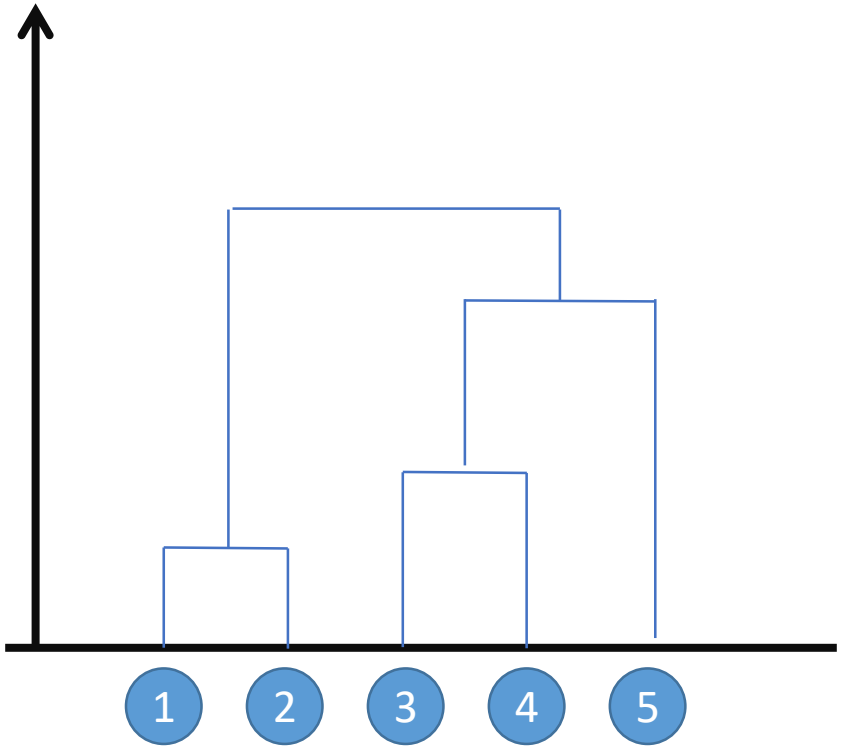


Comment mesurer la distance entre une classe et un élément individuel ?

Critères des centres de gravité, de la distance minimale, maximale, critère de Ward…

SIMULATION DU CAH 6

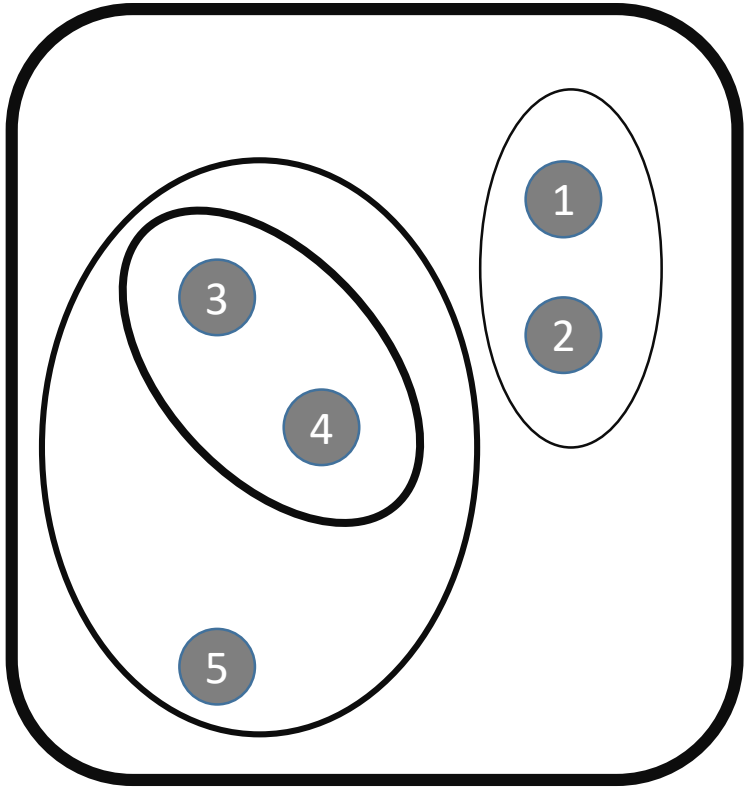


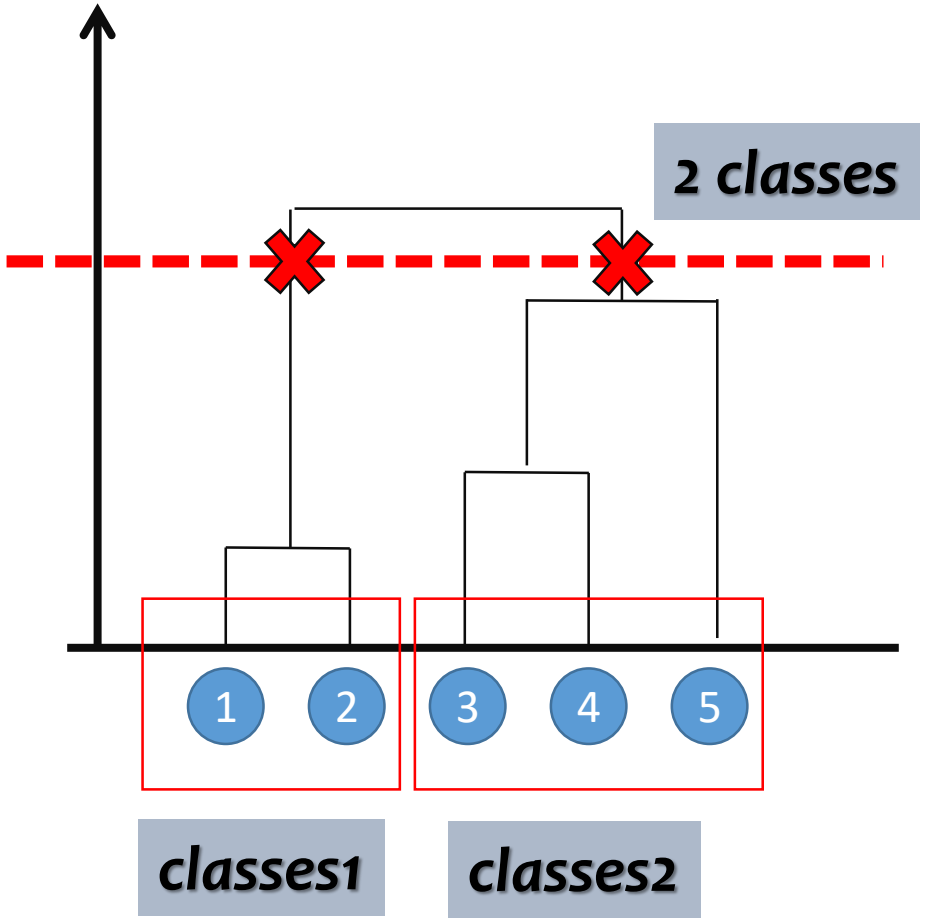


Comment mesurer la distance entre une classe et un élément individuel ?

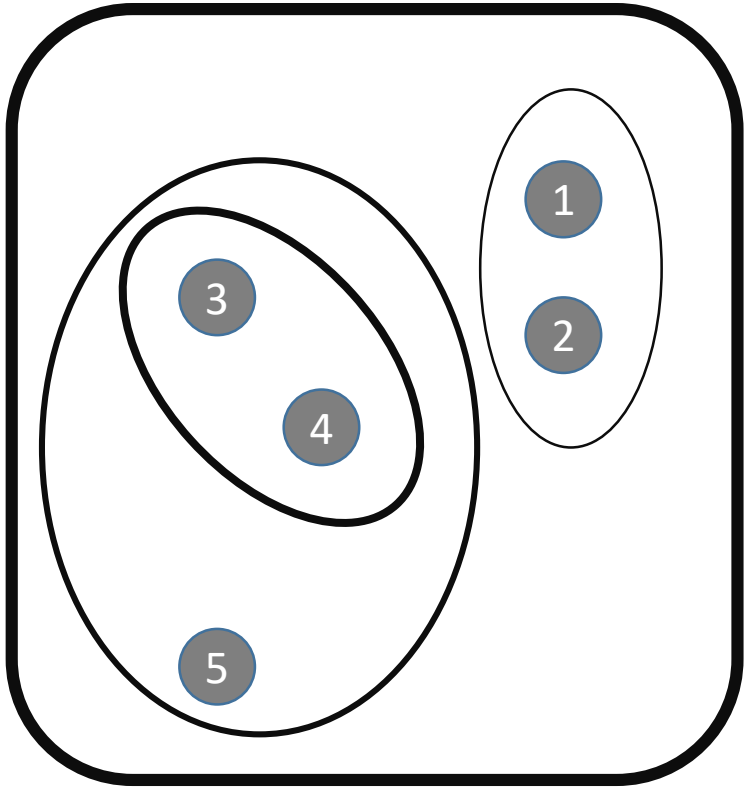
Critères des centres de gravité, de la distance minimale, maximale, critère de Ward…

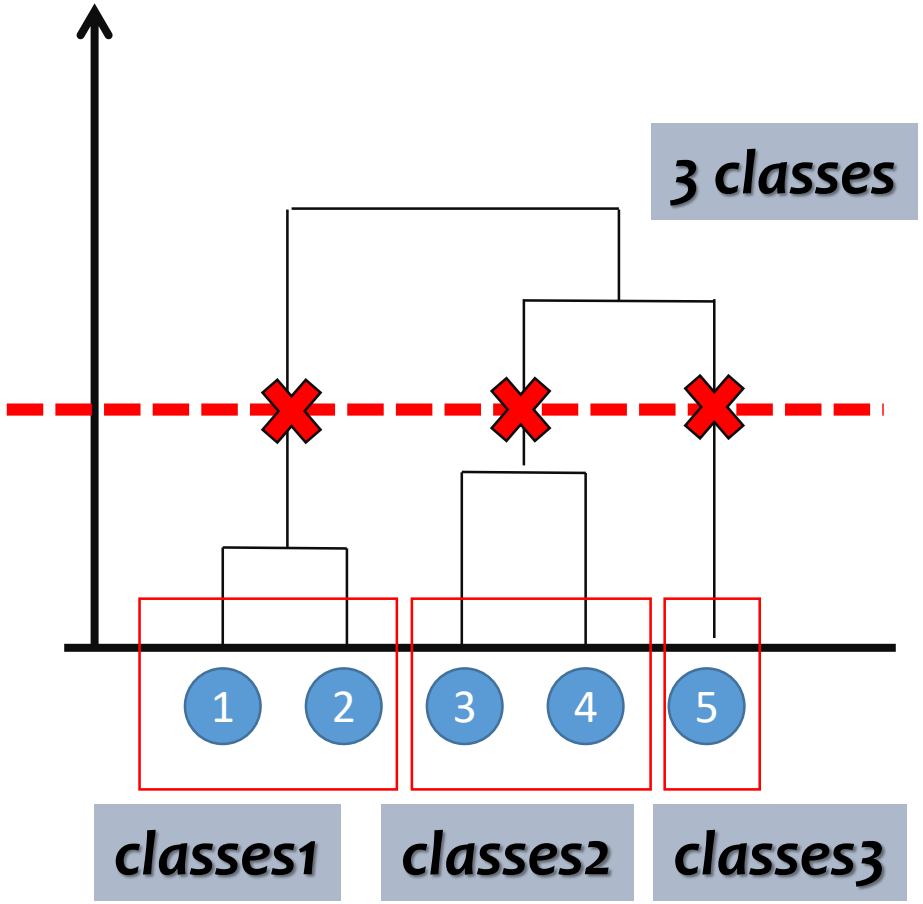
SIMULATION DES CLASSES



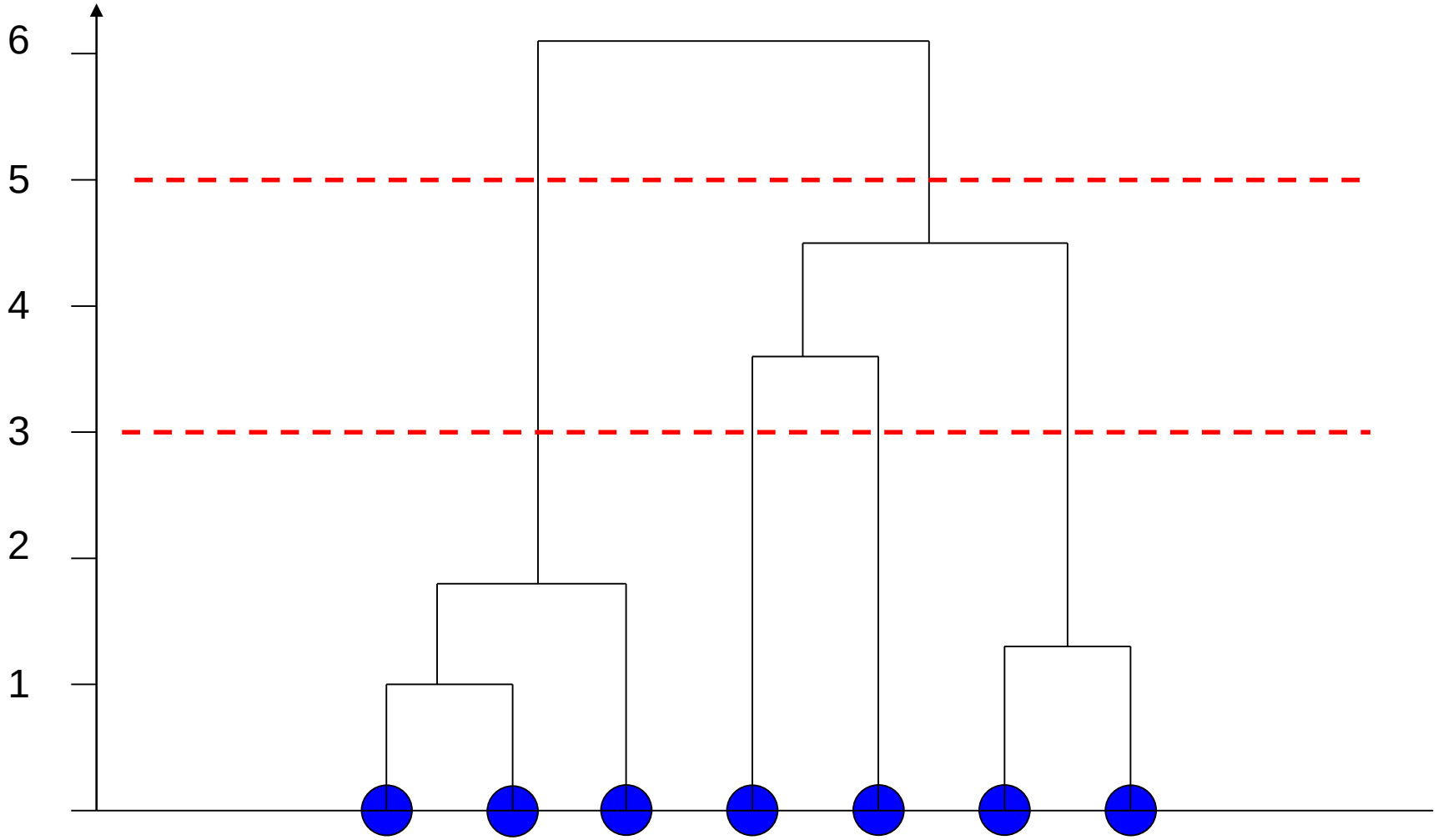


SIMULATION DES CLASSES

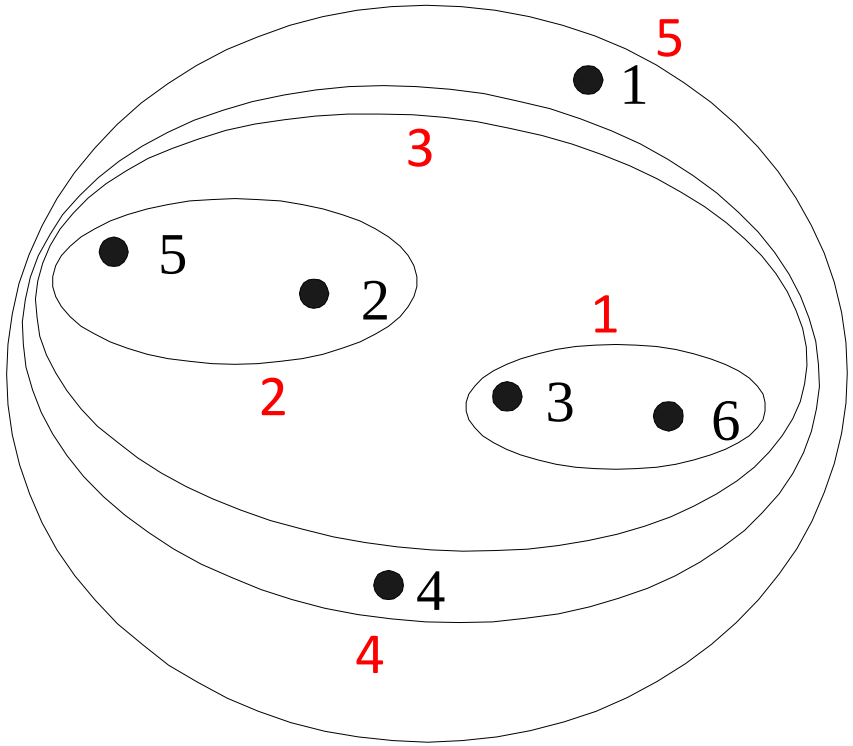




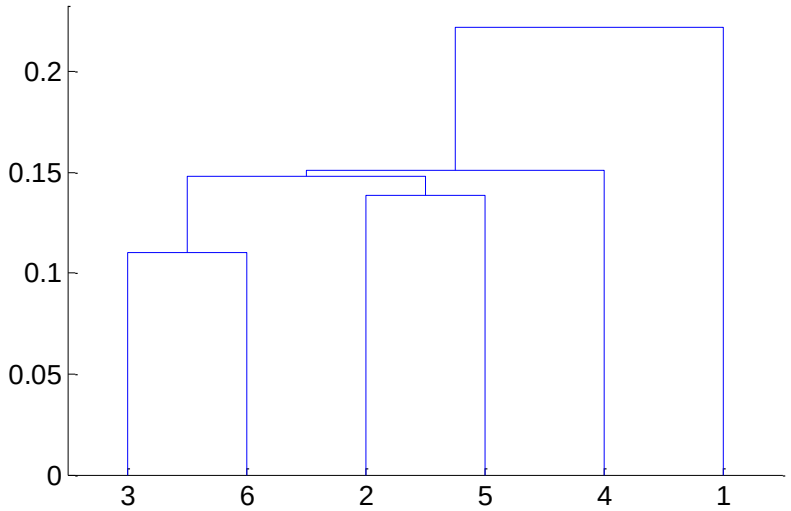
Exemple de dendrogramme



Dendrogramme -> clusters ?



Nested Clusters



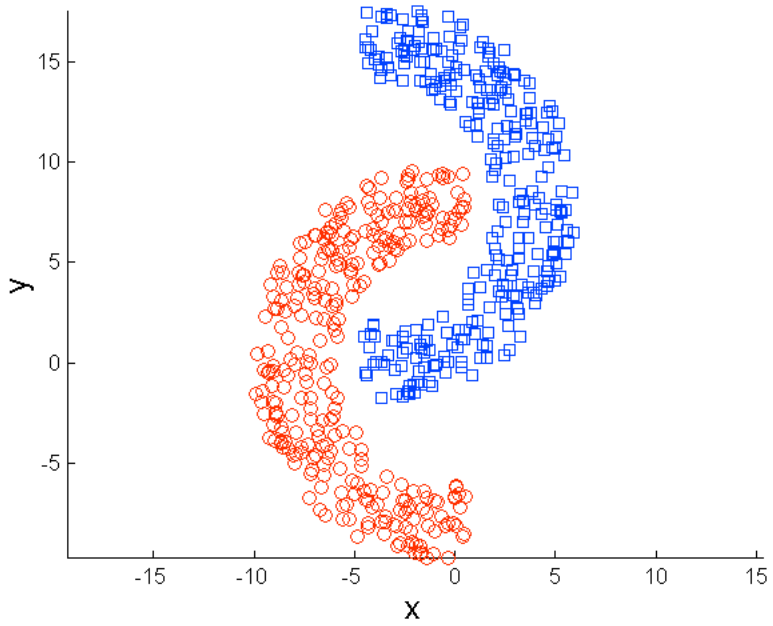
Dendrogram

db-scan

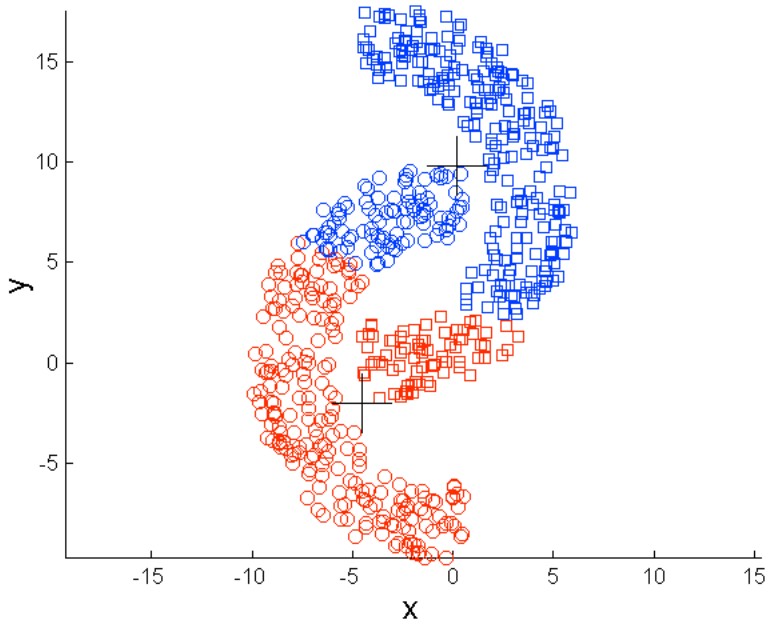
Density-Based Spatial Clustering of Applications with Noise

Med Heny SELMI

Problématique

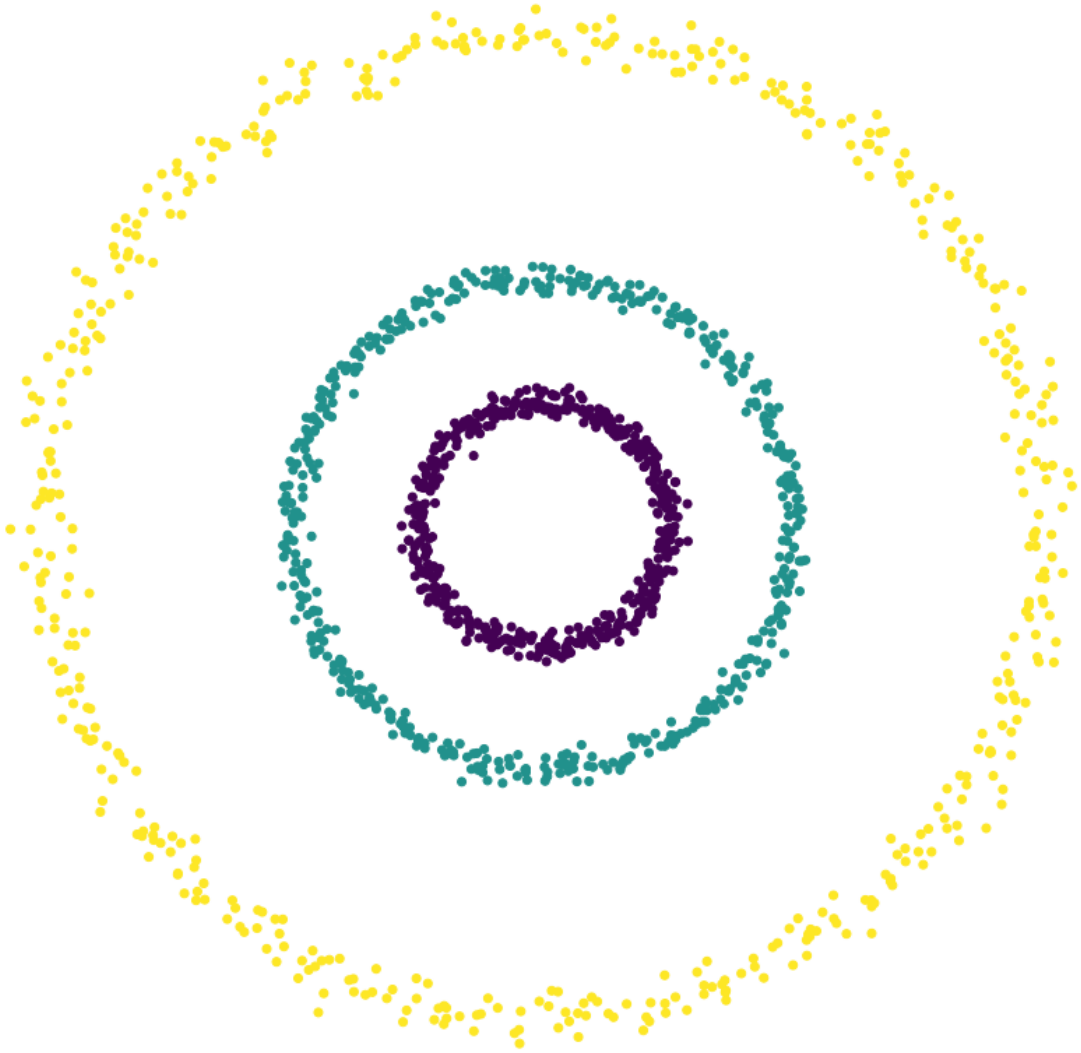


Original Points



K-means (2 Clusters)

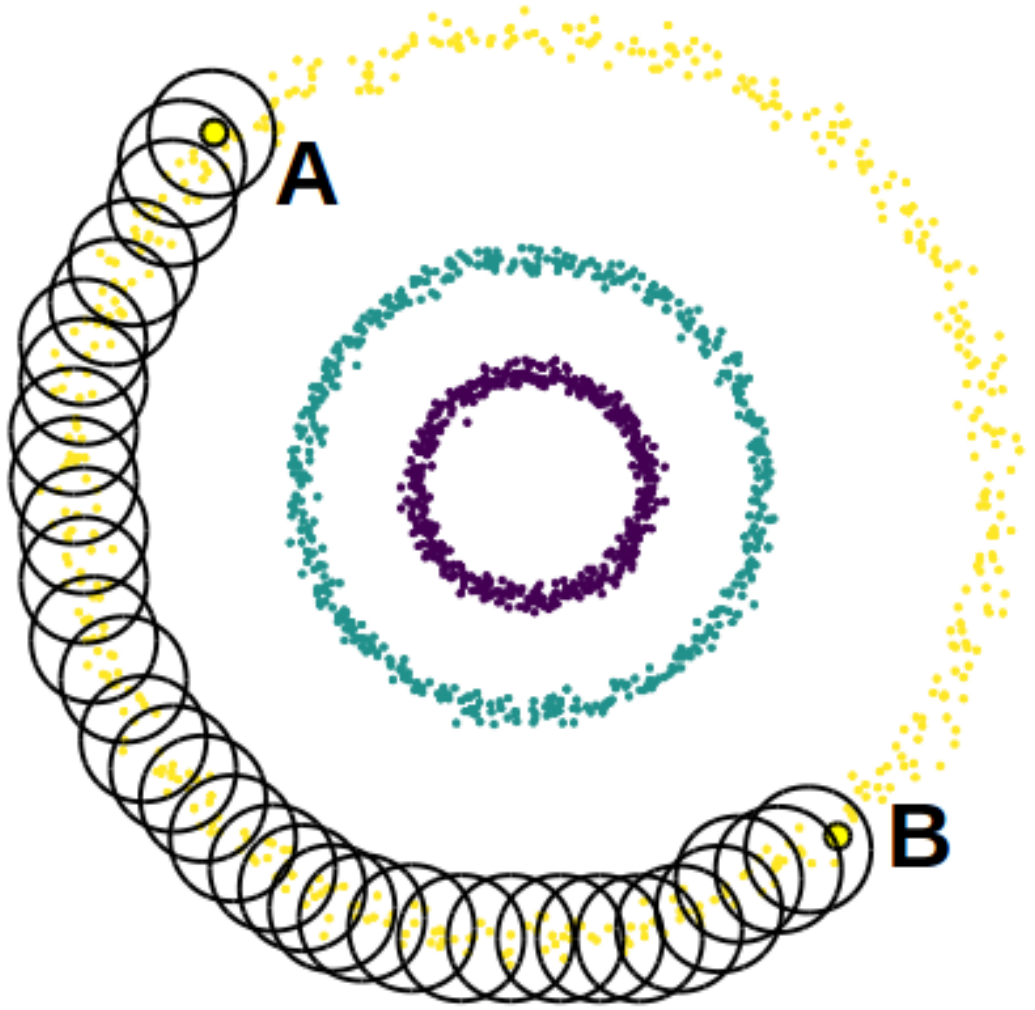
Comment séparer des cercles imbriqués les uns dans l’autre ?



Comment séparer des cercles imbriqués les uns dans l'autre ?

Les méthodes de **clustering par densité** sont basées sur l'observation suivante :

- bien que les deux points A et B ne puissent pas être liés l'un à l'autre par un segment sans intersecté les deux autres clusters,
- on peut créer un chemin pour passer de l'un à l'autre de proche en proche ***en restant à l'intérieur du même cluster*.**
- On peut connecter A à B par des **petits voisinages** (les cercles) contenant uniquement des points du cluster extérieur.



Principe algorithmique

1. Prendre un point x qui n'a pas été visité

2. Construire N = voisinage(eps, x)

3. If |N| < n_min:

Marquer x comme bruit

Else:

Initaliser C = {x}

agrandir_cluster(C, N, eps, n_min)

Ajouter C à la liste des clusters

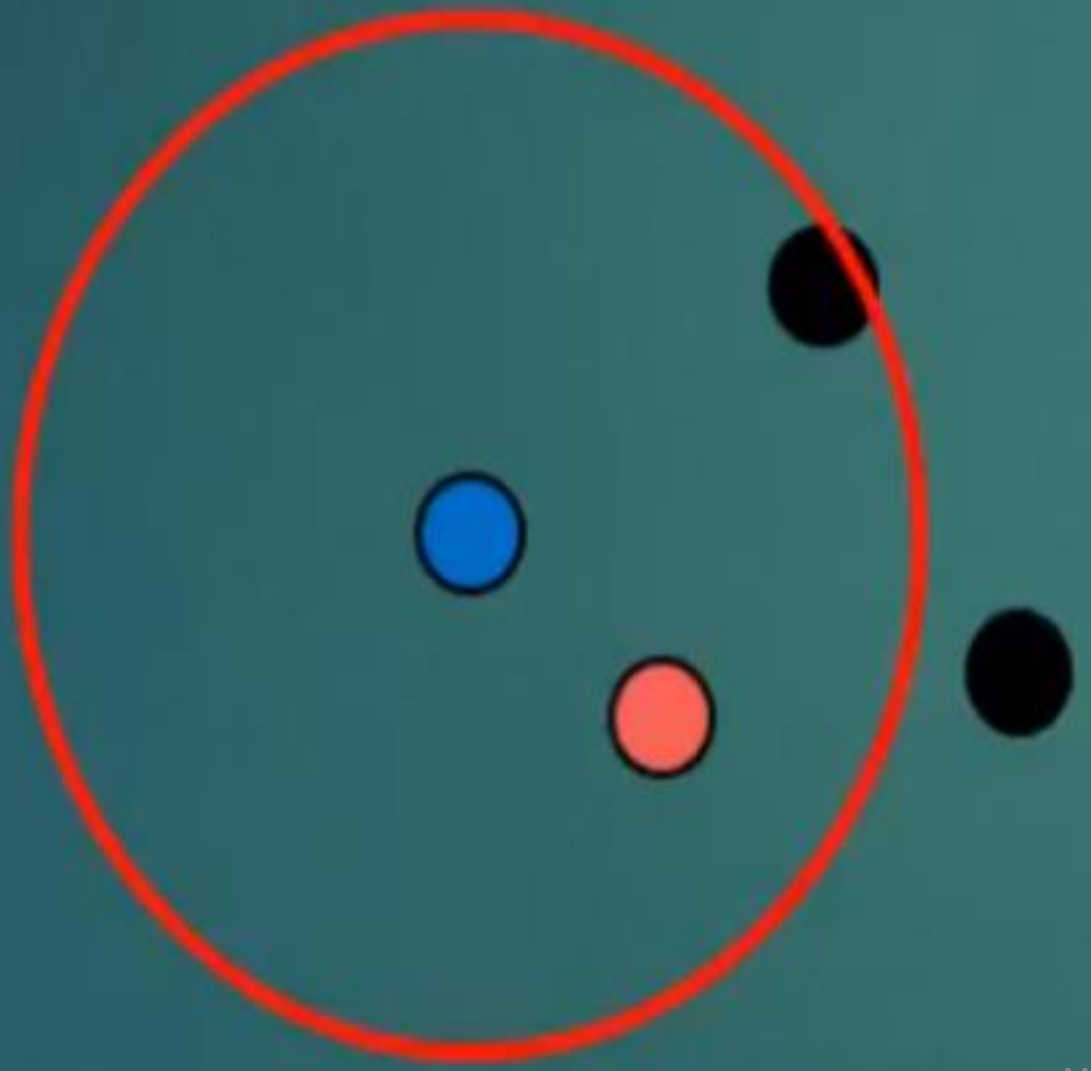
Marquer tous les points de C comme visités

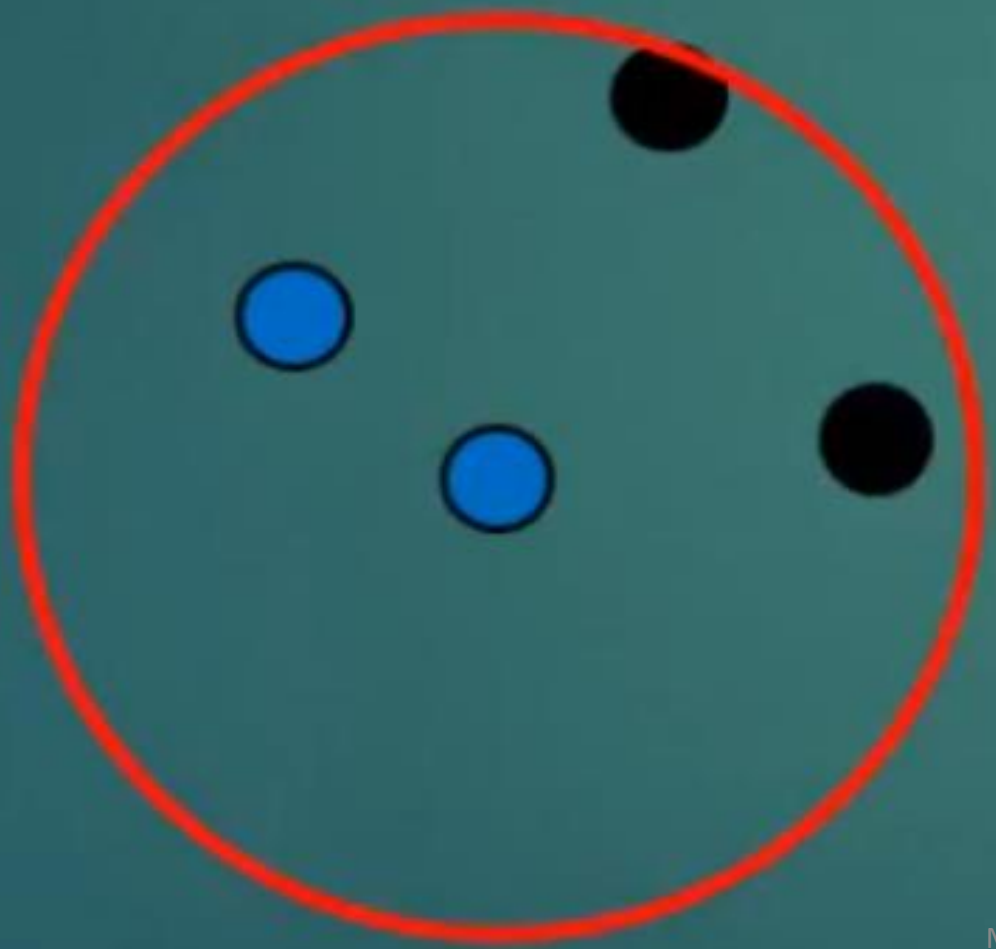
4. Repeat 1-3 until tous les points ont été visités

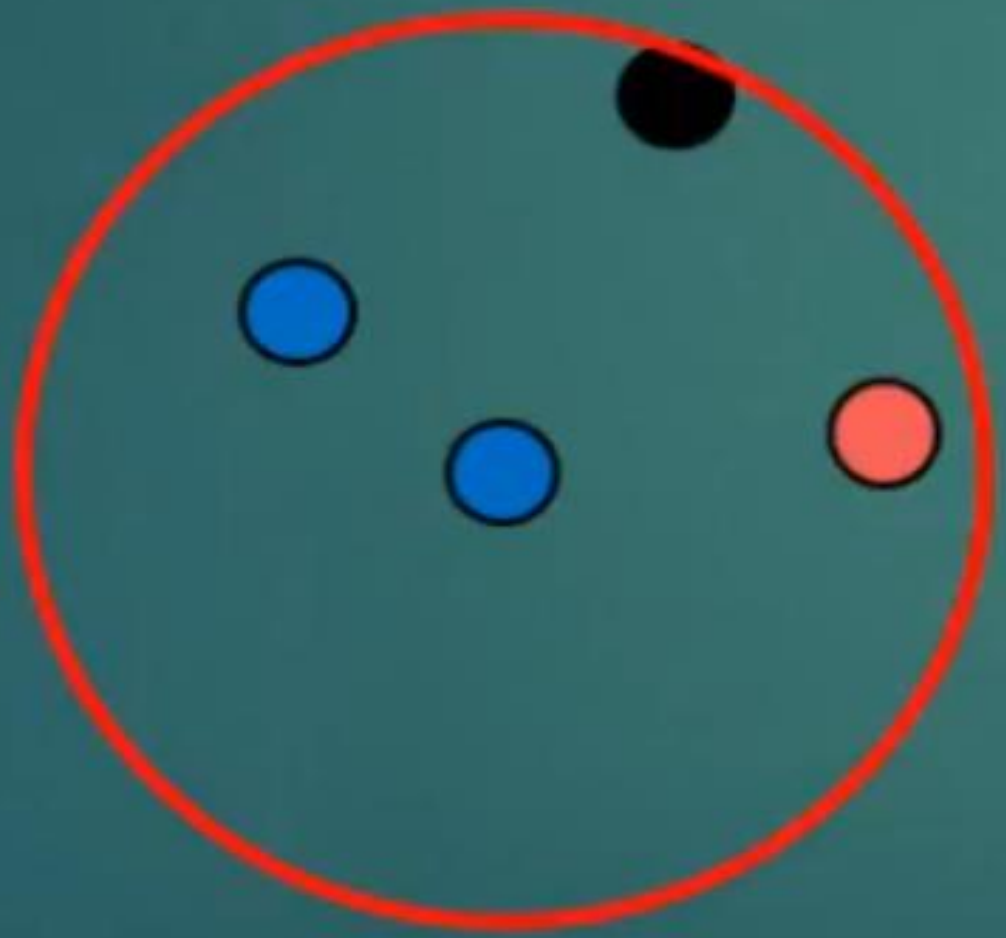
Construire une partition

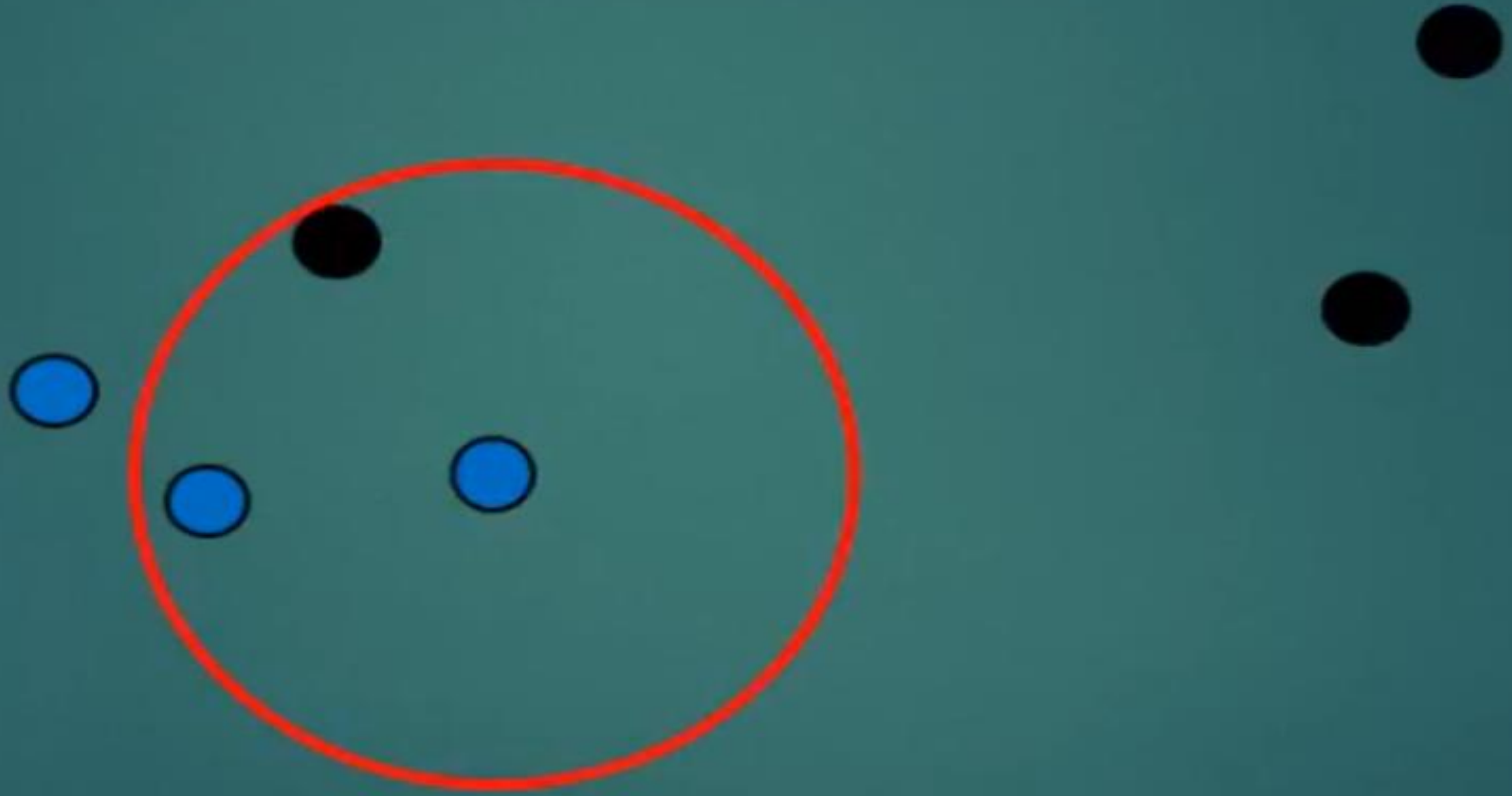
```
agrandir_cluster(C, N, eps, n_min):
        For u in N:
                If u n'a pas été visité:
                        N' = N(eps, u)
                If |N'| >= n_min:
                        N = union(N, N')
                If u n'appartient à aucun autre cluster:
                        Ajouter u à C
```

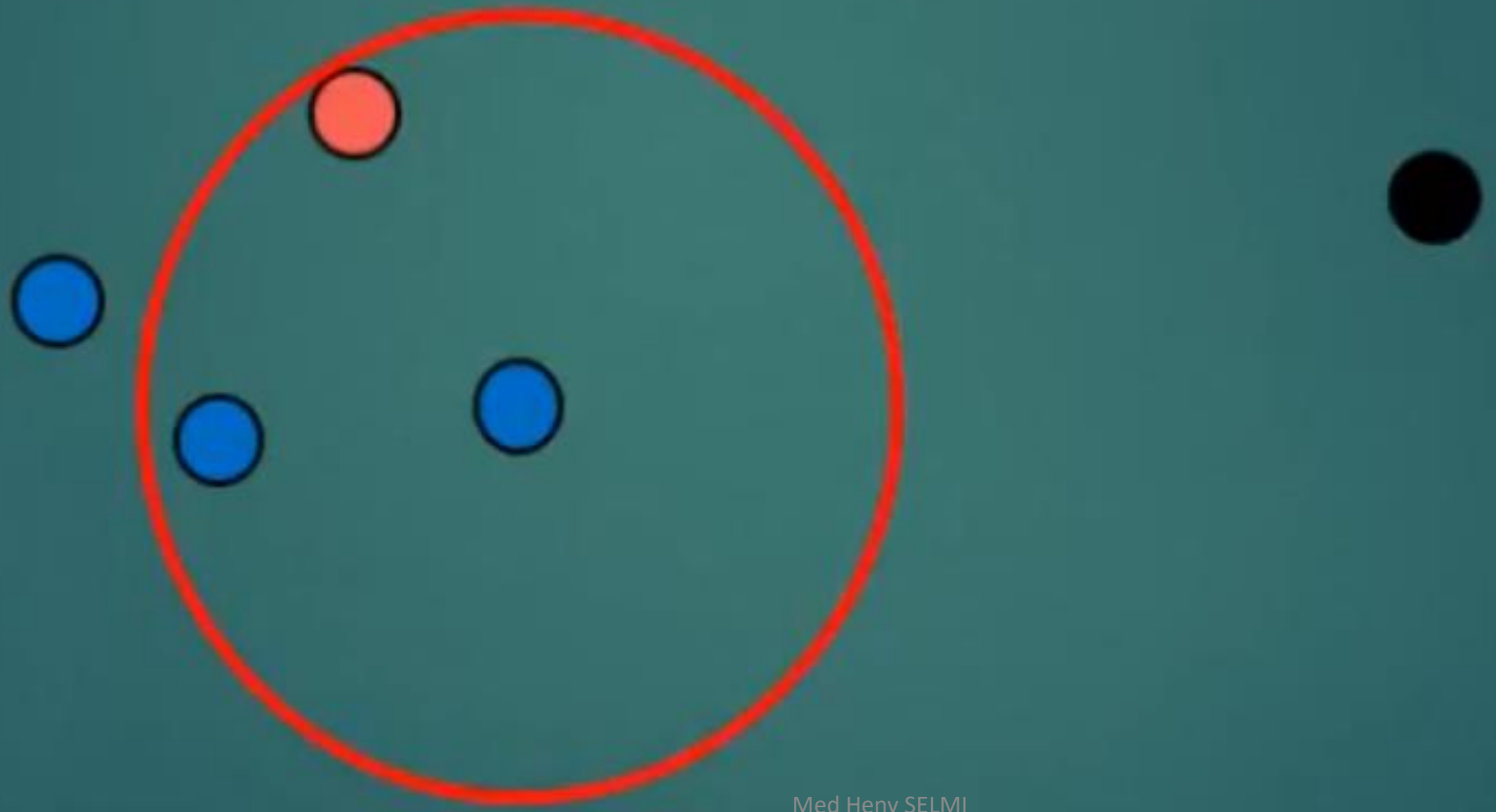

epsilon

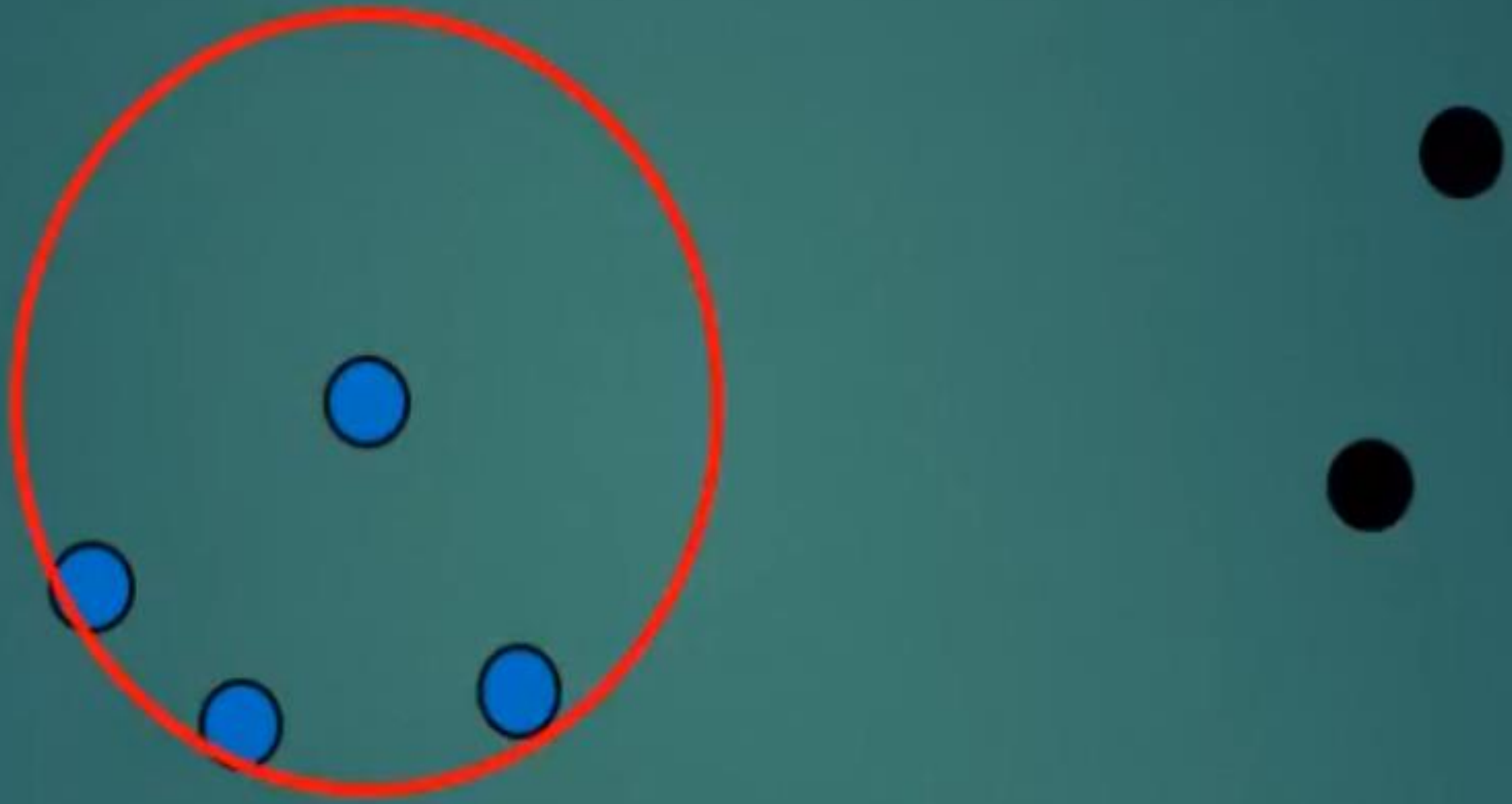












Points forts de db-scan

- Capable de construire des partitions non linéairement séparées.
- On est pas obligé de fournir le nombre de groupes à construire.
- Robuste par rapport à la présence des points aberrants.
- La solution finale n’est pas forcément affectée par le point de départ.

Tolérance aux outliers

- Un point initialement considéré comme du bruit par DBSCAN peut être ultérieurement rattaché à un autre cluster
- il peut appartenir au voisinage d'un autre point, qui lui contient plus de n_min points

- si n_{min}=4, le point orange n'est pas un point intérieur car son epsilon-voisinage ne contient pas suffisamment de points.
- Par contre, il appartient à l'epsilon-voisinage du point rose, qui lui contient plus de n_min points ;
- Ainsi le point orange sera assigné au même cluster que le point rose plutôt que d'être considéré comme du bruit.

Points faibles de db-scan

- Les points de frontière qui sont accessibles à partir de plus d'un cluster peuvent faire partie de l'un ou l'autre cluster en fonction de l'ordre dans lequel les données sont traitées.
- Repose fortement sur le choix de la mesure de distance.
- Ce n’est pas évident de trouver la meilleure valeur de ε dans le cadre des dimensions très importantes.
- **DBSCAN ne pourra pas trouver des clusters de densités différentes => OPTICS**

OPTICS

Ordering Points To Identify the Clustering Structure

Med Heny SELMI

Principe

- Comme DBSCAN, OPTICS demande deux paramètres : ε définissant un rayon maximum à considérer, et MinPts définissant un nombre de points minimum.
- Ces 2 paramètres définissent une densité minimale pour constituer un groupe d'individus.
- Un point p appartient à un groupe si au moins MinPts points existent dans son ε-voisinage $N_\varepsilon(p)$.
- à l'inverse de DBSCAN, le paramètre ε est optionnel : s'il est omis, il sera alors considéré comme infini.
- L'algorithme définit pour chaque point une distance, appelée *core distance*, qui décrit la distance avec le MinPts^{ème} point le plus proche.

$$\text{core-distance}_{\varepsilon,MinPts}(p) = \begin{cases} \text{Indéfini} & \text{si } |N_\varepsilon(p)| < MinPts \\ \text{distance au } MinPts\text{-ème point le plus proche} & \text{sinon} \end{cases}$$

reachability-distance

- La *reachability-distance* du point p à un autre point o est la distance entre o et p, ou la core-distance de p

$$\text{reachability-distance}_{\varepsilon,MinPts}(o,p) = \begin{cases} \text{Indéfini} & \text{si } |N_\varepsilon(p)| < MinPts \\ \max(\text{core-distance}_{\varepsilon,MinPts}(p), \text{distance}(p,o)) & \text{sinon} \end{cases}$$

- La core-distance et la reachability-distance sont non définis si le groupe de points n'est pas suffisamment dense.
- Si ε est suffisamment grand, cela n'arrive jamais, mais toutes les requêtes d'ε-voisinage retourneront l'ensemble des points.