

# Approches de Clustering

- Algorithmes de Partitionnement: Construire plusieurs partitions puis les évaluer selon certains critères
- Algorithmes hiérarchiques: Créer une décomposition hiérarchique des objets selon certains critères
- Algorithmes basés sur la densité: basés sur des notions de connectivité et de densité
- Algorithmes de grille: basés sur un structure à multi-niveaux de granularité
- Algorithmes à modèles: Un modèle est supposé pour chaque cluster ensuite vérifier chaque modèle sur chaque groupe pour choisir le meilleur

# Algorithmes à partitionnement

- Construire une partition à  $k$  clusters d'une base  $D$  de  $n$  objets
- Les  $k$  clusters doivent optimiser le critère choisi
  - Global optimal: Considérer toutes les  $k$ -partitions
  - Heuristic methods: Algorithmes  $k$ -means et  $k$ -medoids
  - $k$ -means (MacQueen'67): Chaque cluster est représenté par son centre
  - $k$ -medoids or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Chaque cluster est représenté par un de ses objets

# La méthode des k-moyennes (*K-Means*)

- L'algorithme *k-means* est en 4 étapes :
  1. Choisir k objets formant ainsi k clusters
  2. (Ré)attribuer chaque objet O au cluster  $C_i$  de centre  $M_i$  tel que  $\text{dist}(O, M_i)$  est minimal
  3. Recalculer  $M_i$  de chaque cluster (le barycentre)
  4. Aller à l'étape 2 si on vient de faire une affectation

# K-Means :Exemple

- $A=\{1,2,3,6,7,8,13,15,17\}$ . Créer 3 clusters à partir de A
- On prend 3 objets au hasard. Supposons que c'est 1, 2 et 3. Ca donne  $C_1=\{1\}$ ,  $M_1=1$ ,  $C_2=\{2\}$ ,  $M_2=2$ ,  $C_3=\{3\}$  et  $M_3=3$
- Chaque objet O est affecté au cluster au milieu duquel, O est le plus proche. 6 est affecté à  $C_3$  car  $\text{dist}(M_3,6) < \text{dist}(M_2,6)$  et  $\text{dist}(M_3,6) < \text{dist}(M_1,6)$   
On a  $C_1=\{1\}$ ,  $M_1=1$ ,  
 $C_2=\{2\}$ ,  $M_2=2$   
 $C_3=\{3, 6,7,8,13,15,17\}$ ,  $M_3=69/7=9.86$

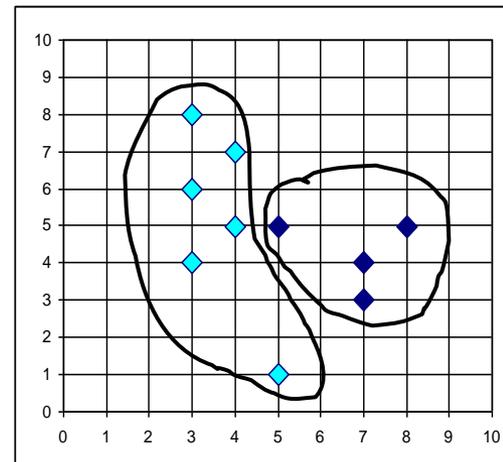
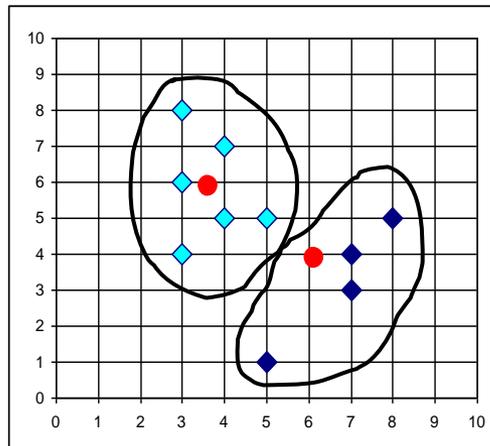
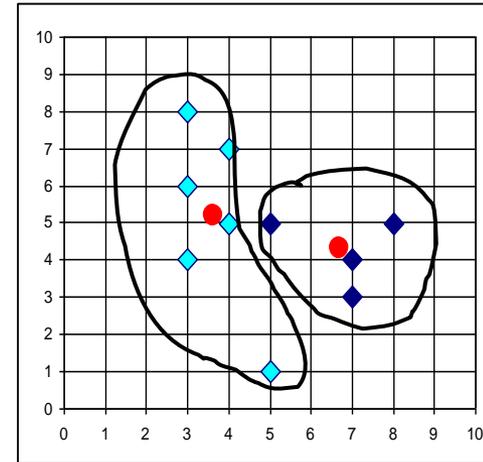
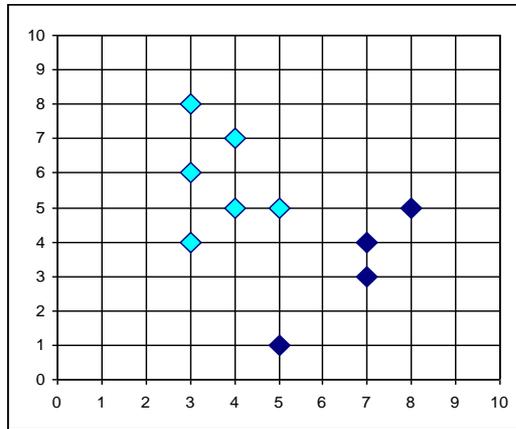
# K-Means :Exemple (suite)

- $\text{dist}(3, M_2) < \text{dist}(3, M_3) \rightarrow 3$  passe dans  $C_2$ . Tous les autres objets ne bougent pas.  $C_1 = \{1\}$ ,  $M_1 = 1$ ,  $C_2 = \{2, 3\}$ ,  $M_2 = 2.5$ ,  $C_3 = \{6, 7, 8, 13, 15, 17\}$  et  $M_3 = 66/6 = 11$
- $\text{dist}(6, M_2) < \text{dist}(6, M_3) \rightarrow 6$  passe dans  $C_2$ . Tous les autres objets ne bougent pas.  $C_1 = \{1\}$ ,  $M_1 = 1$ ,  $C_2 = \{2, 3, 6\}$ ,  $M_2 = 11/3 = 3.67$ ,  $C_3 = \{7, 8, 13, 15, 17\}$ ,  $M_3 = 12$
- $\text{dist}(2, M_1) < \text{dist}(2, M_2) \rightarrow 2$  passe en  $C_1$ .  $\text{dist}(7, M_2) < \text{dist}(7, M_3) \rightarrow 7$  passe en  $C_2$ . Les autres ne bougent pas.  $C_1 = \{1, 2\}$ ,  $M_1 = 1.5$ ,  $C_2 = \{3, 6, 7\}$ ,  $M_2 = 5.34$ ,  $C_3 = \{8, 13, 15, 17\}$ ,  $M_3 = 13.25$
- $\text{dist}(3, M_1) < \text{dist}(3, M_2) \rightarrow 3$  passe en 1.  $\text{dist}(8, M_2) < \text{dist}(8, M_3) \rightarrow 8$  passe en 2  
 $C_1 = \{1, 2, 3\}$ ,  $M_1 = 2$ ,  $C_2 = \{6, 7, 8\}$ ,  $M_2 = 7$ ,  $C_3 = \{13, 15, 17\}$ ,  $M_3 = 15$

Plus rien ne bouge

# Algorithme *K-Means*

- Exemple



# Commentaires sur la méthode des *K-Means*

- Force

- *Relativement efficace*:  $O(tkn)$ , où  $n$  est # objets,  $k$  est # clusters, et  $t$  est # itérations. Normalement,  $k, t \ll n$ .
- Tend à réduire

$$E = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2$$

- Faiblesses

- N'est pas applicable en présence d'attributs qui ne sont pas du type intervalle (moyenne=?)
- On doit spécifier  $k$  (nombre de clusters)
- Les clusters sont construits par rapports à des objets inexistantes (les milieux)
- Ne peut pas découvrir les groupes *non-convexes*

# La méthode des *K-Medoids* (*PAM*)

- Trouver des objets représentatifs (medoïdes) dans les clusters (au lieu de la moyenne)
- Principe
  - Commencer avec un ensemble de medoïdes puis itérativement remplacer un par un autre si ça permet de réduire la distance globale
  - Efficace pour des données de petite taille

# Algorithme des k-Medoides

Choisir arbitrairement k medoides

Répéter

    affecter chaque objet restant au medoide le plus proche

    Choisir aléatoirement un non-medoide  $O_r$

    Pour chaque medoide  $O_j$

        Calculer le coût TC du remplacement de  $O_j$  par  $O_r$

        Si  $TC < 0$  alors

            Remplacer  $O_j$  par  $O_r$

            Calculer les nouveaux clusters

        Finsi

    FinPour

Jusqu'à ce ce qu'il n'y ait plus de changement

# PAM (Partitioning Around Medoids) (1987)

Choisir arbitrairement  $k$  objets représentatifs

- Pour toute paire  $(h,j)$  d'objets t.q  $h$  est choisi et  $j$  non, calculer le coût  $TC_{jh}$  du remplacement de  $j$  par  $h$ 
  - Si  $TC_{jh} < 0$ ,  $j$  est remplacé par  $h$
  - Puis affecter chaque objet non sélectionné au medoïde qui lui est le plus similaire
- Répéter jusqu'à ne plus avoir de changements

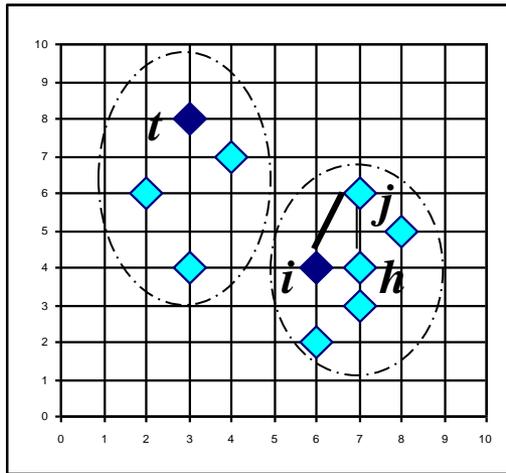
# La méthode des *K-Medoids*

- $TC_{jh}$  représente le gain en distance globale que l'on va avoir en remplaçant  $h$  par  $j$
- Si  $TC_{jh}$  est négatif alors on va perdre en distance. Ca veut dire que les clusters seront plus compacts.
- $TC_{jh} = \sum_i \text{dist}(j, h) - \text{dist}(j, i) = \sum_i C_{ijh}$

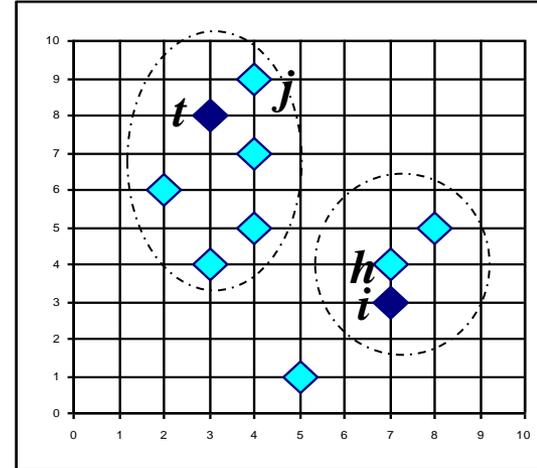
# La méthode des *K-Medoids*: Exemple

- Soit  $A=\{1,3,4,5,8,9\}$ ,  $k=2$  et  $M=\{1,8\}$  ensemble des medoides  
→  $C1=\{1,3,4\}$  et  $C2=\{5,8,9\}$   
 $E_{\{1,8\}} = \text{dist}(3,1)^2 + \text{dist}(4,1)^2 + \text{dist}(5,8)^2 + \text{dist}(9,8)^2 = 23$
- Comparons 1 et 3 →  $M=\{3,8\}$  →  $C1=\{1,3,4,5\}$  et  $C2=\{8,9\}$   
 $E_{\{3,8\}} = \text{dist}(1,3)^2 + \text{dist}(4,3)^2 + \text{dist}(5,3)^2 + \text{dist}(9,8)^2 = 10$   
 $E_{\{3,8\}} - E_{\{1,8\}} = -13 < 0$  donc le remplacement est fait.
- Comparons 3 et 4 →  $M=\{4,8\}$  →  $C1$  et  $C2$  inchangés et  
 $E_{\{4,8\}} = \text{dist}(1,4)^2 + \text{dist}(3,4)^2 + \text{dist}(5,4)^2 + \text{dist}(8,9)^2 = 12$  → 3 n'est pas remplacé par 4
- Comparons 3 et 5 →  $M=\{5,8\}$  →  $C1$  et  $C2$  inchangés et  $E_{\{5,8\}} > E_{\{3,8\}}$

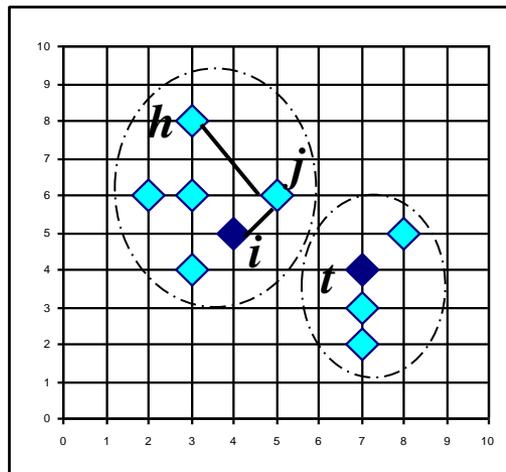
# PAM Clustering: $TC_{ih} = \sum_j C_{jih}$



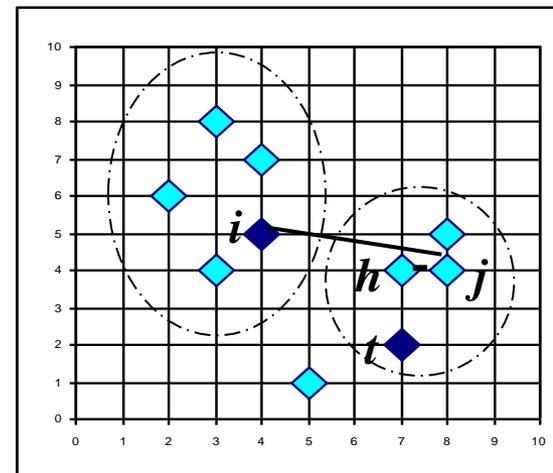
$$C_{jih} = d(j, h) - d(j, i)$$



$$C_{jih} = 0$$



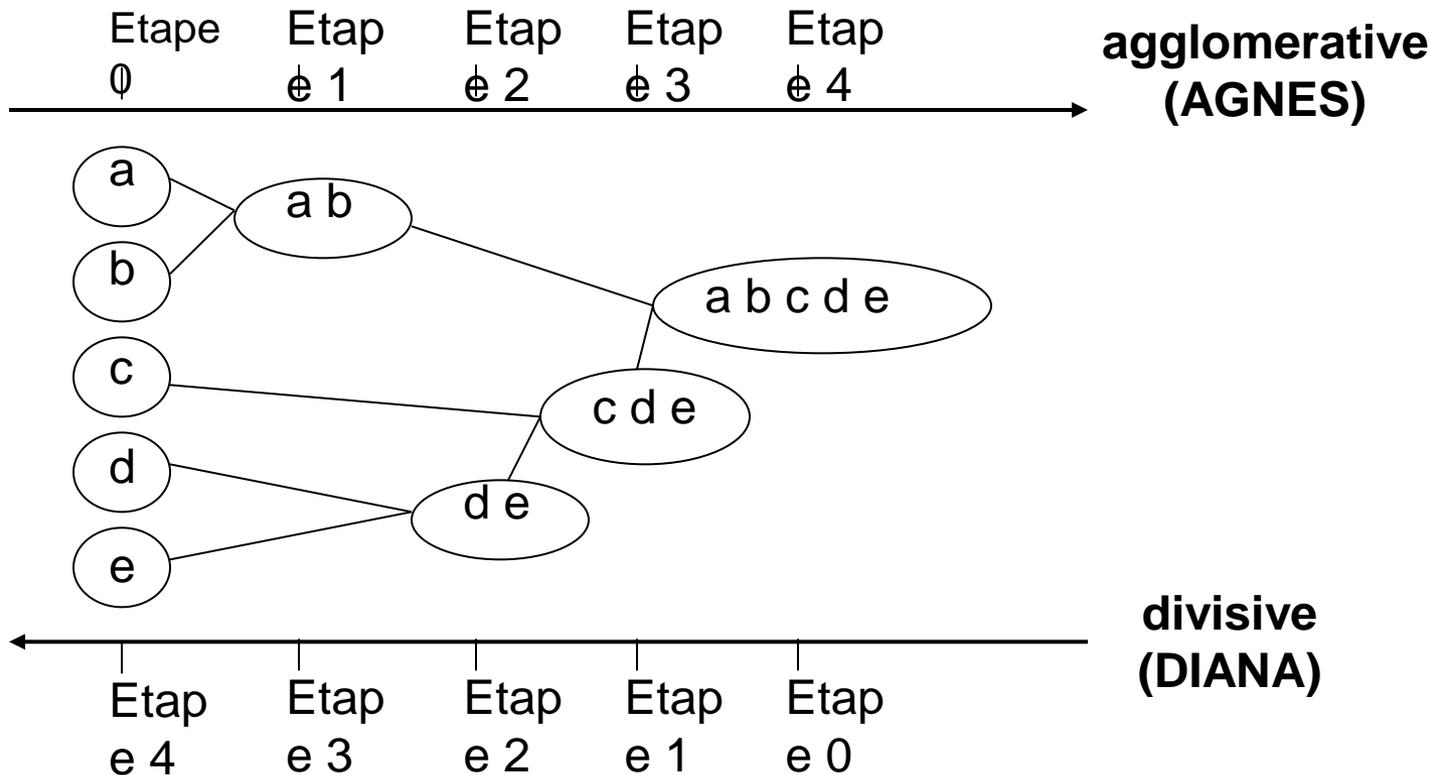
$$C_{jih} = d(j, h) - d(j, i)$$



$$C_{jih} = d(j, h) - d(j, t)$$

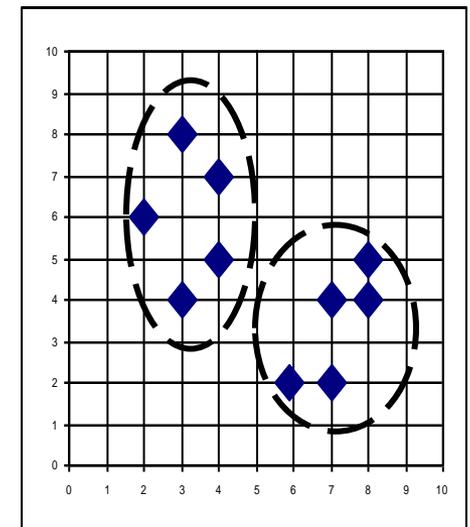
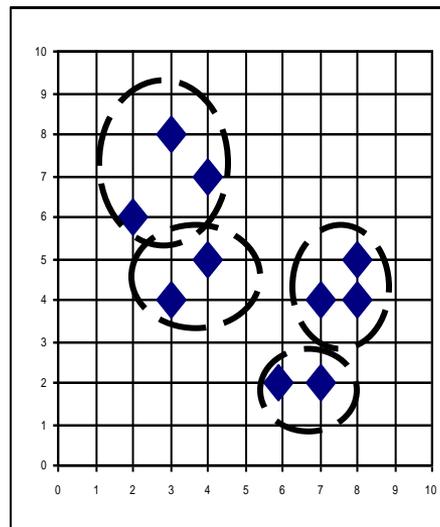
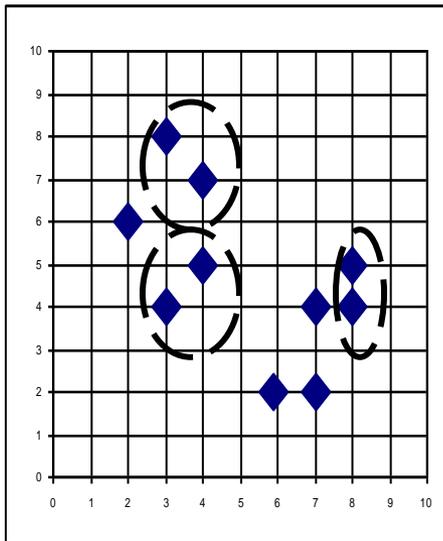
# Clustering Hiérarchique

- Utiliser la matrice de distances comme critère de regroupement.  $k$  n'a pas à être précisé, mais a besoin d'une condition d'arrêt



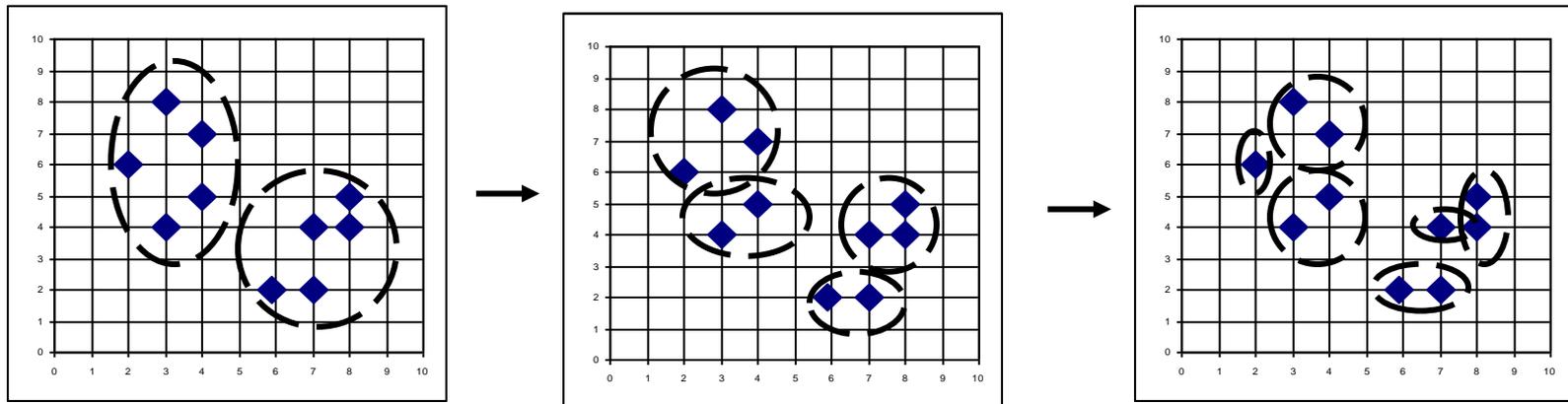
# AGNES (Agglomerative Nesting)

- Utilise la matrice de dissimilarité.
- Fusionne les nœuds qui ont la plus faible dissimilarité
- On peut se retrouver dans la situation où tous les nœuds sont dans le même groupe



# DIANA (Divisive Analysis)

- L'ordre inverse de celui d'AGNES
- Il se peut que chaque objet forme à lui seul un groupe



# Critères de fusion-éclatement

- Exemple: pour les méthodes agglomératives, C1 et C2 sont fusionnés si

- il existe  $o1 \in C1$  et  $o2 \in C2$  tels que  $dist(o1, o2) \leq$  seuil, ou
- il n'existe pas  $o1 \in C1$  et  $o2 \in C2$  tels que  $dist(o1, o2) \geq$  seuil, ou
- distance entre C1 et C2  $\leq$  seuil avec

$$dist(C_1, C_2) = \frac{1}{n1 * n2} \sum_{o1 \in C1, o2 \in C2} dist(o1, o2)$$

et  $n1 = |C1|$ .

- Ces techniques peuvent être adaptées pour les méthodes divisives

# BIRCH (1996)

- Birch: Balanced Iterative Reducing and Clustering using Hierarchies
- Construit incrémentalement un arbre (CF-tree : Clustering Feature), une structure hiérarchique où chaque niveau représente une phase de clustering
  - Phase 1: scanner la base pour construire le CF-tree dans la mémoire
  - Phase 2: utiliser n'importe quel algorithme de clustering sur les feuilles du CF-tree
- *Avantage*: trouve les clusters en une seule passe sur la BD
- *Inconvénient*: ne considère que les données numériques et est sensible à l'ordre des enregistrements

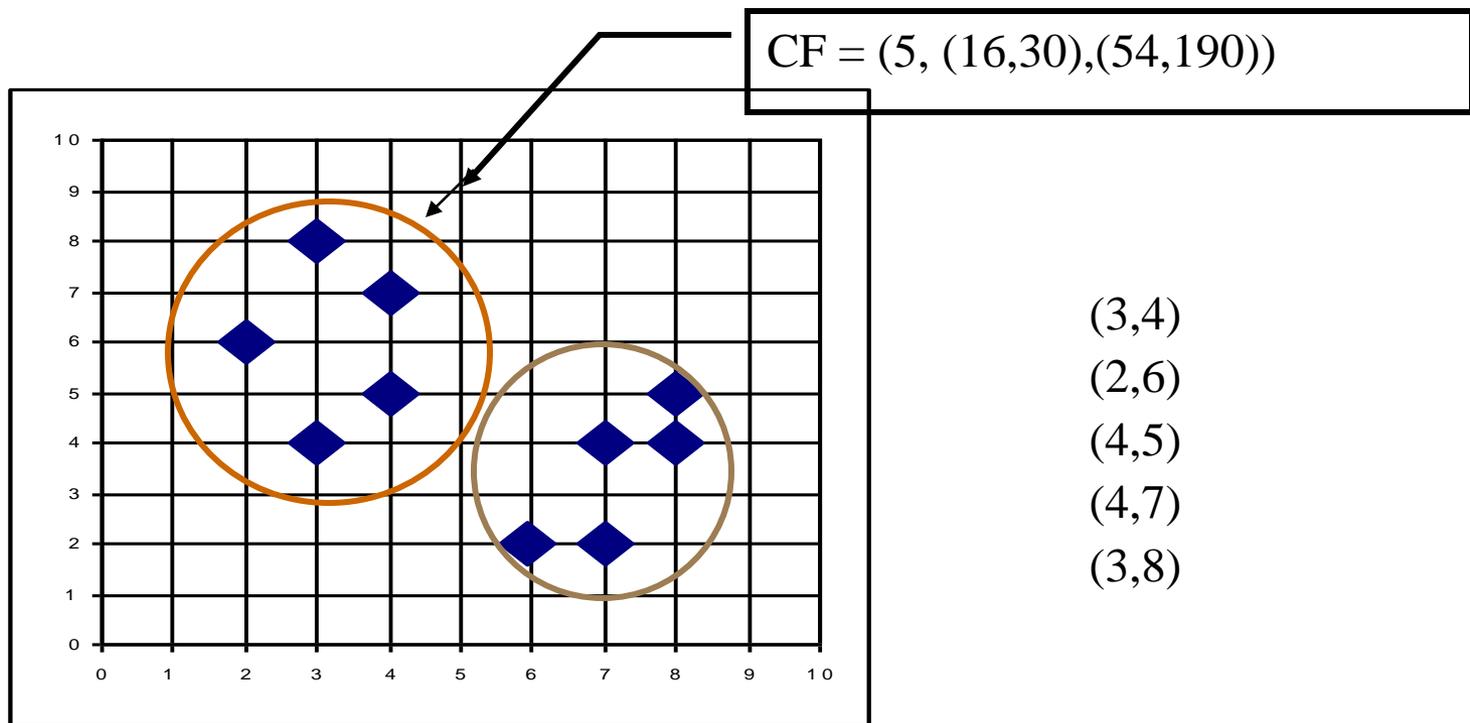
# Clustering Feature Vector

Clustering Feature:  $CF = (N, LS, SS)$

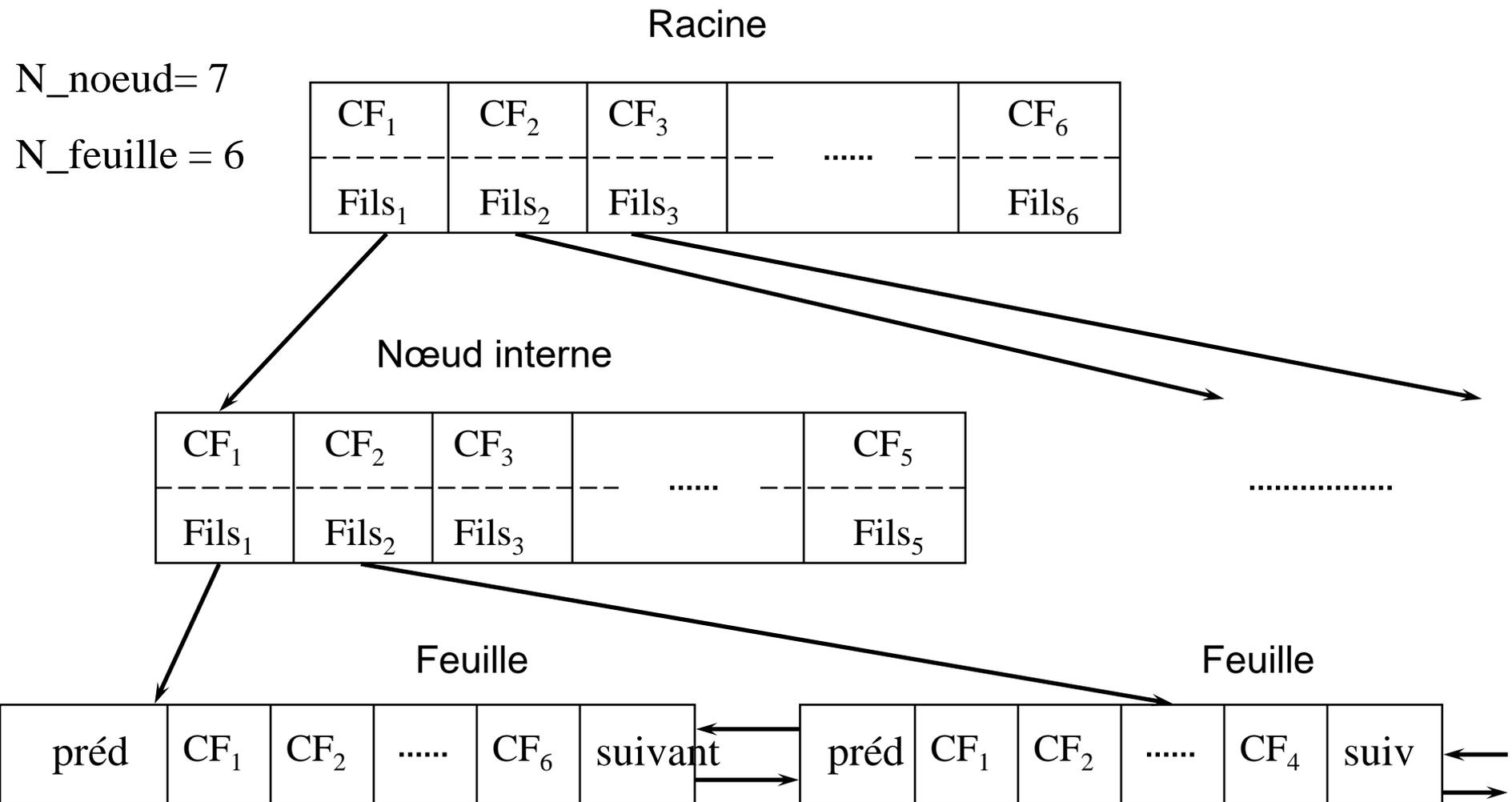
$N$ : Number of data points

$$LS: \sum_{i=1}^N \overrightarrow{X_i}$$

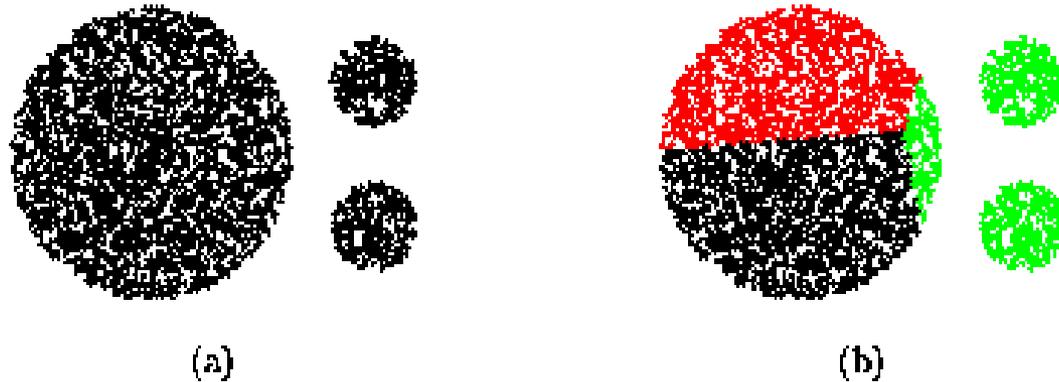
$$SS: \sum_{i=1}^N X_i^2$$



# CF Tree



# CURE (Clustering Using REpresentatives)



- Les méthodes précédentes donnent les groupes (b)
- CURE: (1998)
  - Arrête la création de clusters dès qu'on en a  $k$
  - Utilise plusieurs points représentatifs clusters

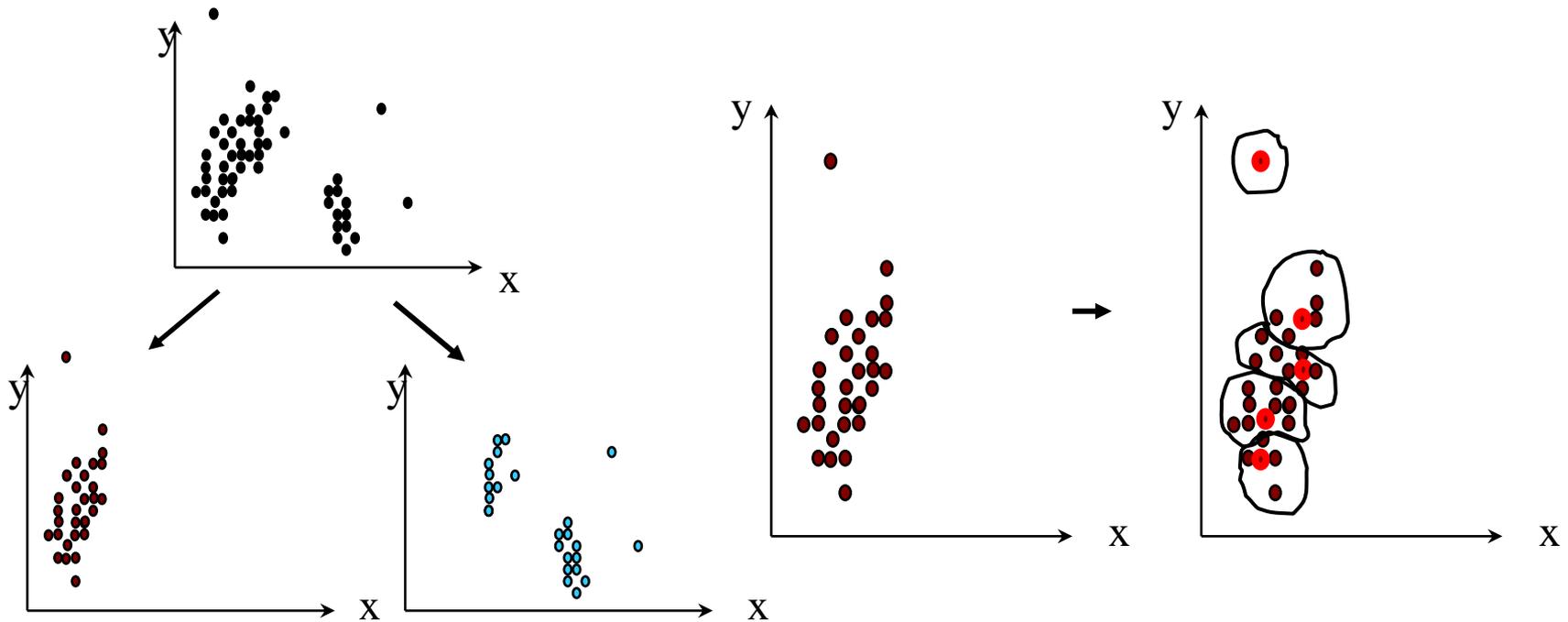
# Cure: l'algorithme

- Prendre un sous-ensemble  $s$
- Partitionner  $s$  en  $p$  partitions de taille  $s/p$
- Dans chaque partition, créer  $s/pq$  clusters
- Eliminer les exceptions (points aberrants)
- Regrouper les clusters partiels

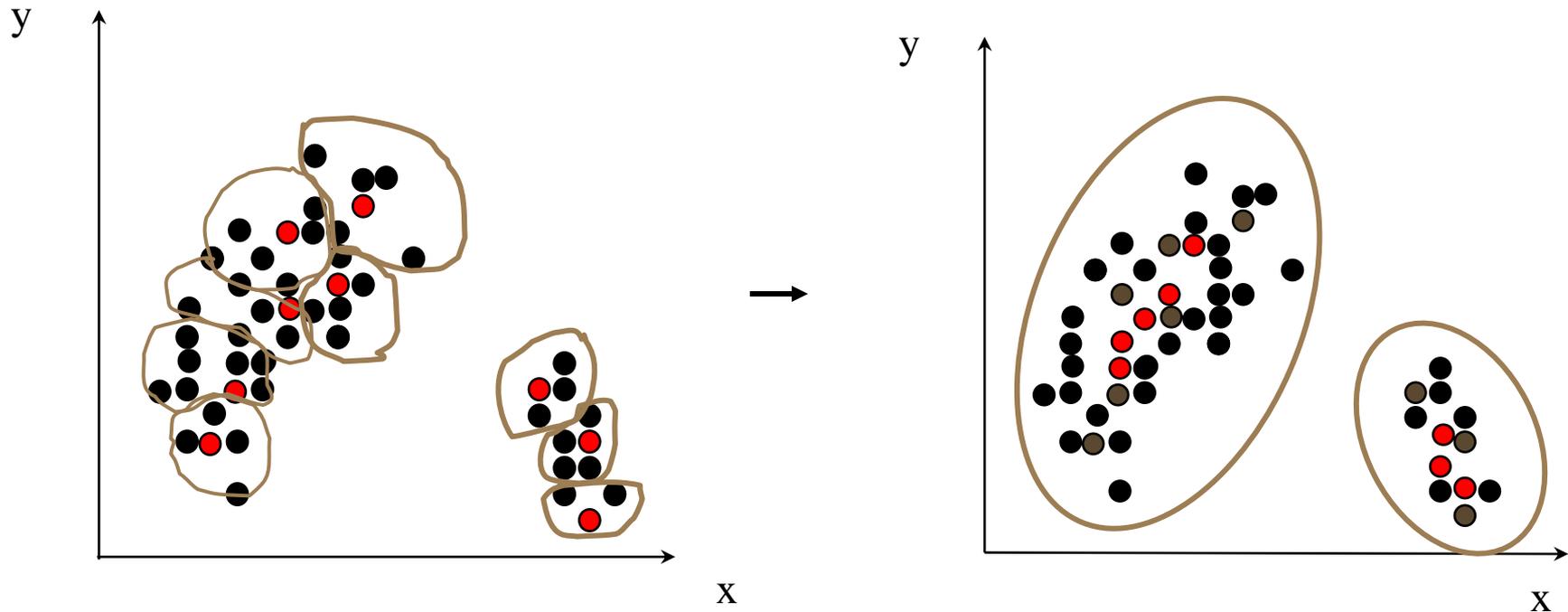
# Partitionnement et Clustering

- $s = 50$
- $p = 2$
- $s/p = 25$

■  $s/pq = 5$



# Cure: Rapprochement des points représentatifs



- Rapprocher les points représentatifs vers le centre de gravité par un facteur  $\alpha$ .
- Plusieurs points représentatifs permettent de figurer la forme du cluster

# Clustering de données Catégorielles

## ROCK

- ROCK: Robust Clustering using linkS
  - Utilise les liens pour mesurer la similarité/proximité
  - N'est pas basé sur la notion de distance
- Idée :
  - Fonction de similarité et voisins:

$$\text{Let } T_1 = \{1,2,3\}, T_2 = \{3,4,5\} \quad \text{Sim}(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

$$\text{Sim}(T_1, T_2) = \frac{|\{3\}|}{|\{1,2,3,4,5\}|} = \frac{1}{5} = 0.2$$

# Rock

- Considérons 4 transactions et 6 produits t.q

$$T1=\{1,2,3,5\} \quad T2=\{2,3,4,5\}$$

$$T3=\{1,4\} \text{ et } T4=\{6\}$$

- T1 peut être représentée par  $\{1,1,1,0,1,0\}$

$\text{dist}(T1,T2)=2$  qui est la plus petite distance entre 2 transactions  $\rightarrow$  T1 et T2 dans même cluster. La moyenne de  $C1=(0.5,1,1,0.5,1,0)$ .

$C2=\{T3,T4\}$  car  $\text{dist}(T3,T4)=3$ . Or T3 et T4 n'ont aucun produit en commun !

Idée : se baser sur le nombre d'éléments en commun

Ce n'est pas suffisant  $\{1,2\}$  est plus proche de  $\{1,2,3\}$  que de  $\{1,2,3,4,5,6\}$

# Rock: l'algorithme

- Liens: Le nombre de voisins communs de 2 points

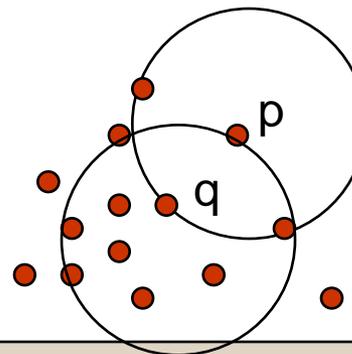
$\{1,2,3\}$ ,  $\{1,2,4\}$ ,  $\{1,2,5\}$ ,  $\{1,3,4\}$ ,  $\{1,3,5\}$   
 $\{1,4,5\}$ ,  $\{2,3,4\}$ ,  $\{2,3,5\}$ ,  $\{2,4,5\}$ ,  $\{3,4,5\}$



- Algorithme
  - Prendre un sous ensemble
  - Regrouper avec les liens

# Clustering basé sur la densité

- Voit les clusters comme des régions denses séparées par des régions qui le sont moins (bruit)
- Deux paramètres:
  - **Eps**: Rayon maximum du voisinage
  - **MinPts**: Nombre minimum de points dans le voisinage-Eps d'un point
- **Voisinage** :  $V_{Eps}(p)$ :  $\{q \in D \mid dist(p,q) \leq Eps\}$
- Un point **p** est directement densité-accessible à partir de **q** resp. à **Eps**, **MinPts** si
  - 1)  $p \in V_{Eps}(q)$
  - 2)  $|V_{Eps}(q)| \geq MinPts$



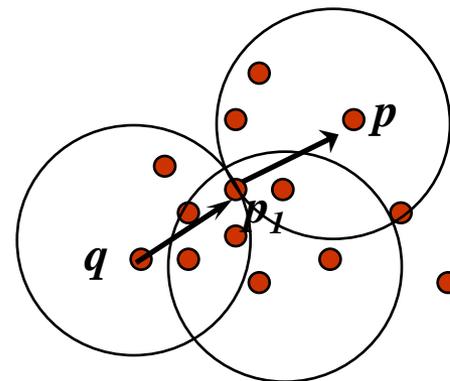
MinPts = 5

Eps = 1 cm

# Clustering basé sur la densité

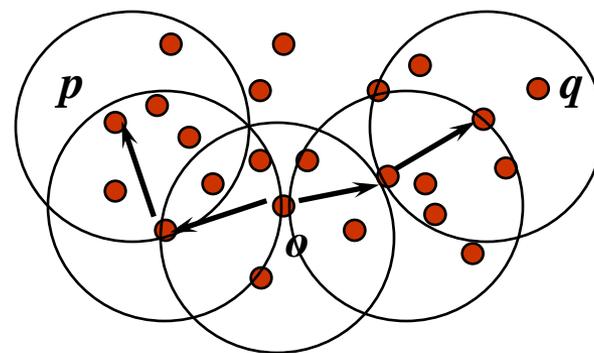
- Accessibilité:

- $p$  est accessible à partir de  $q$  resp. à  $Eps$ ,  $MinPts$  si il existe  $p_1, \dots, p_n$ ,  $p_1 = q$ ,  $p_n = p$  t.q  $p_{i+1}$  est directement densité accessible à partir de  $p_i$



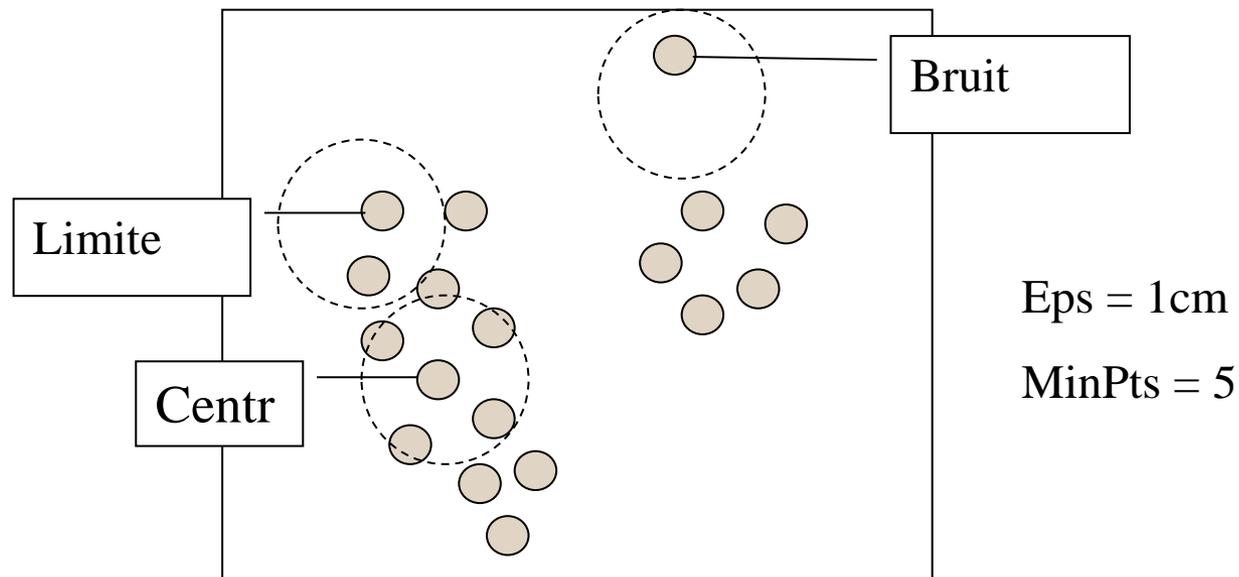
- Connexité

- $p$  est connecté à  $q$  resp. à  $Eps$ ,  $MinPts$  si il existe un point  $o$  t.q  $p$  et  $q$  accessibles à partir de  $o$  resp. à  $Eps$  et  $MinPts$ .



# DBSCAN: Density Based Spatial Clustering of Applications with Noise

- Un *cluster* est l'ensemble maximal de points connectés
- Découvre des clusters non nécessairement convexes



# DBSCAN: l'algorithme

- Choisir  $p$
- Récupérer tous les points accessibles à partir de  $p$  resp.  $Eps$  et  $MinPts$ .
- Si  $p$  est un centre, un cluster est formé.
- si  $p$  est une limite, alors il n'y a pas de points accessibles de  $p$  : passer à un autre point
- Répéter le processus jusqu'à épuiser tous les points.