Groupement (Clustering)
= Classification sans connaître les classes

- Objectifs
- Distances
- Algorithmes
  - Méthodes par partitionnement
  - Méthodes hiérarchiques
  - Méthodes par densité

1. Objectifs et Définitions

- Classification automatique d'objets
  - en se basant sur une mesure de similarité (ou distance) pour grouper les données
- Buts :
  - Maximiser la similarité intra-classes et minimiser la similarité inter-classes.
  - Mesurer la distance en assimilant un tuple à un point dans un espace à n dimensions.

Méthode non supervisée

- Méthode permettant de découvrir les groupes (clusters) d'individus similaires
- Application aux tables relationnelles
  - Formation de groupes de tuples (clusters) à partir d'une fonction de distance
  - On met ensemble les tuples les plus "proches"
- On ne connaît pas les classes à priori
  - elles sont à découvrir automatiquement
  - Il est parfois possible d'en fixer le nombre

Objectifs

- Passage à l'échelle
  - espaces de grande dimension supportés
  - nombreux objets lus une seule fois en base
- Robustesse
  - indépendance à l'ordre de parcours de la base
  - capacité à découvrir des groupes de toute forme
  - insensibilité aux points isolés (outliers)
  - insensibilité au bruit
- Lisibilité
  - comment donner un sens aux résultats ?
  - comment interpréter les classes ?

Similarité

- Problèmes :
  - La notion de distance et d'appartenance à un groupe varie selon le domaine
- Indice de similarité
  - S1) s est symétrique : $s(w, w') = s(w', w)$
  - S2) $s(w, w) = s(w', w') > s(w, w')$ C'est une constante supérieur : smax
- Indice de dissimilarité:
  - ne vérifie pas S2) mais S2') $\forall w \in \Omega, di(w, w) = 0$

Distance

- Distance : indice de dissimilarité qui vérifie également :
  - (D1) $d(w, w') = 0 \Rightarrow w = w'$
  - (D2) pour tout w, w',w" de Ω,

    $d(w, w') \leq d(w, w") + d(w", w')$
  - Indice de distance : indice de dissimilarité qui ne vérifie que (D1)
  - Ecart : indice de dissimilarité qui ne vérifie que (D2) /* inégalité triangulaire */

Distances (données numériques)

- **Distance de Minkowsky** :
  - $d(w_1, w_2) = [\ \Sigma_{j=1}^{p} |x_1^j - x_2^j|^{\lambda}\ ]^{1/\lambda}$
  - pour $\lambda = 1$, on a les valeurs absolues (distance de Manhattan)
  - pour $\lambda = 2$, on a la distance euclidienne simple (M = I)
  - pour $\lambda \rightarrow +\infty$, on obtient la distance de Chebyshev (ultramétrique):
    - $d(w_1, w_2) = \text{Max}_j\ |x_1^j - x_2^j|$

Distances (données binaires)

- On considère x1 et x2 deux vecteurs binaires :
  - Soit a le nombre de fois où x1j = x2j = 1
  - Soit b le nombre de fois où x1j = 0 et x2j = 1
  - Soit c le nombre de fois où x1j = 1 et x2j = 0
  - Soit d le nombre de fois où x1j = x2j = 0
- Exemple de similarités souvent utilisées :
  - D1(x1, x2) = a / (a+b+c+d)
  - D2(x1, x2) = a / (a+b+c)
  - D3(x1, x2) = 2a / (2a+b+c)
  - D4(x1, x2) = a / (a+2(b+c))
  - D5(x1, x2) = (a+d) / (a+b+c+d)

Distances (données qualitatives)

- Similarités entre individus :
  - codage disjonctif complet; permet de se ramener à un tableau de variables binaires -> utilise les notions des variables quantitatives (surtout $\chi 2$)
- Similarités entre variables :
  - considère le tableau de contingence associé à deux variables v1 et v2
  - permet de définir un similarité entre les variables.
  - Par exemple la valeur du $\chi 2$ de contingence peut-être utilisé comme similarité entre les variables. $\sum_{i=1}^{J} \frac{(n_i - n'_i)^2}{n'_i}$

Ressemblance : approche de Tversky

- Notion de similarité non basée sur la distance:
  - similarité n'est pas une relation symétrique
  - "A est similaire à B" : sujet A et référence B :
  - sujet peut ne pas ressembler autant à la référence que la référence au sujet

Notion de ressemblance et ensembles flous

- Ω : ensemble de référence
- $F(\Omega)$: ensemble des sous-ensembles flous de Ω
  - Un sous-ensemble flou A de $F(\Omega)$ est caractérisé par sa fonction d'appartenance fA définie sur $F(\Omega)$ à valeurs dans [0,1].
  - Cette fonction d'appartenance donne pour chaque x de Ω le degré fA(x) avec lequel il appartient au sous-ensemble A
- Ressemblance :
  - R(A, A) = 1
  - Si $(B \subseteq A)$ alors R(A, B) = 1
  - Si $(A \cap B) = \varnothing$ alors R(A, B) = 0

Principales Méthodes (1)

- Méthodes par partitionnement:
  - Construire k partitions et les corriger jusqu'à obtenir une similarité satisfaisante
  - k-means, k-medoids, CLARANS
- Méthodes hiérarchiques:
  - Créer une décomposition hiérarchique par agglomération ou division de groupes similaires ou dissimilaires
  - AGNES, DIANA, BIRCH, CURE, ROCK, …

Principales Méthodes (2)

- Méthodes par densité:
  - Grouper les objets tant que la densité de voisinage excède une certaine limite
  - DBSCAN, OPTICS, DENCLUE
- Méthodes par grille:
  - Diviser l'espace en cellules formant une grille multi-niveaux et grouper les cellules voisines en terme de distance
  - STING, WaveCluster, CLIQUE
- Méthodes par modèle:
  - Modéliser les groupes et utiliser le modèle pour classer les points
  - COBWEB, Neural Networks

2. Méthodes par Partition

- N objets sont classés en k-partitions
  - Construire k partitions et les corriger jusqu'à obtenir une similarité satisfaisante
  - Optimisation d'une fonction d'objectif
    - similarité inter-classe
  - k-means, k-medoids en mémoire
    - compression possible
  - CLARANS pour les BD

2.1 K-Means

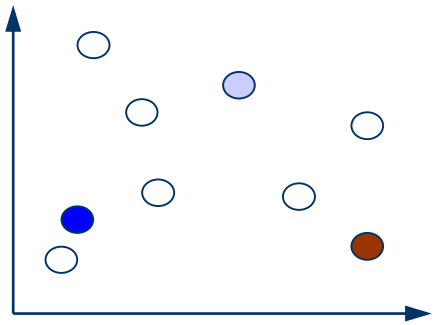
- Méthode des K-moyennes (*MacQueen'67)*
  - choisir K éléments initiaux "centres" des K groupes
  - placer les objets dans le groupe de centre le plus proche
  - recalculer le centre de gravité de chaque groupe
  - itérer l'algorithme jusqu'à ce que les objets ne changent plus de groupe
- Encore appelée méthode des centres mobiles

Algorithme

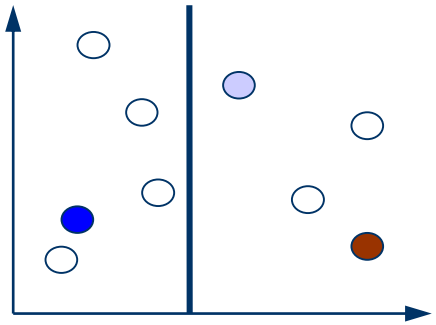
- Étapes:
  - fixer le nombre de clusters: k
  - choisir aléatoirement k tuples comme graines (centres)
  - assigner chaque tuple à la graine la plus proche
  - recalculer les k graines
  - tant que des tuples ont été changés
    - réassigner les tuples
    - recalculer les k graines

- C'est l'Algorithme le plus utilisé

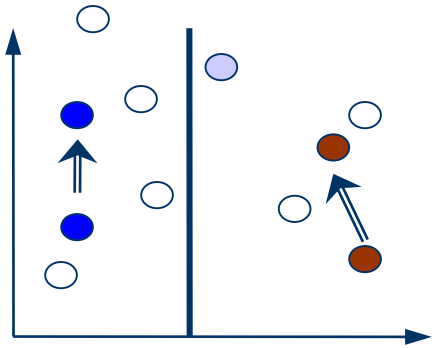
Exemple de K-Means (k=2)



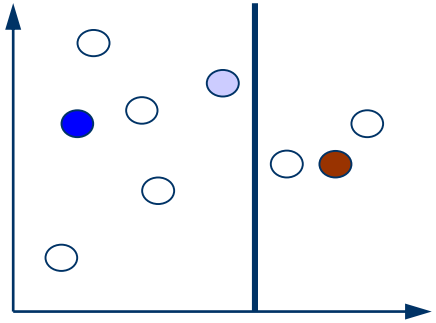
Choisir 2 graines



Assigner les tuples



Recalculer les centroïdes

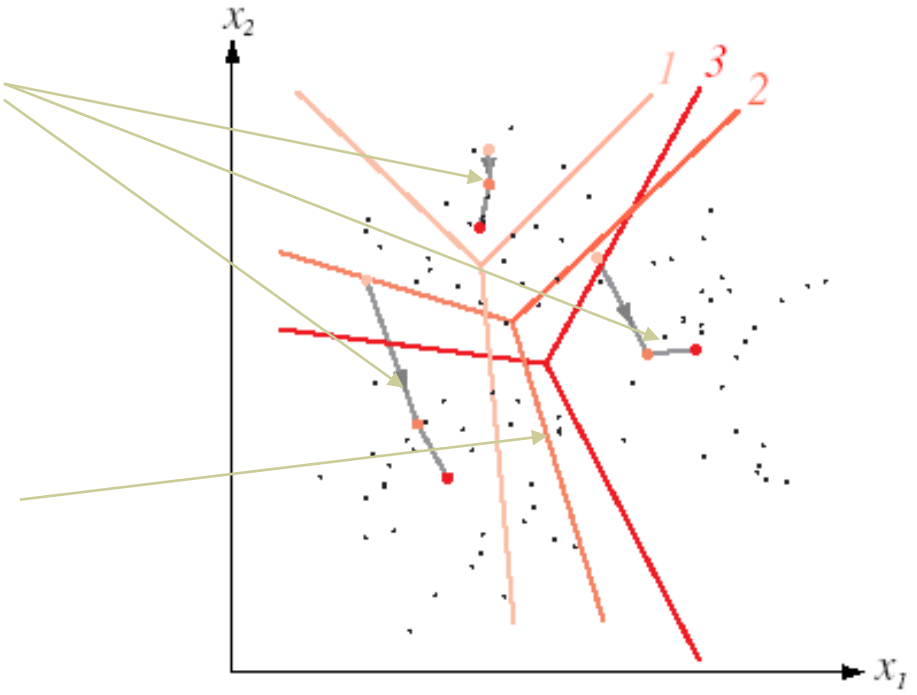


Réassigner les tuples

Trajectoire des centres

Trajectoires des 3 centres d'un nuage de points bidimensionnel

Les hyperplans séparateurs entre les classes



Autre exemple (1)

- 27-51-52-33-45-22-28-44-40-38-20-57
- K=3
- distance = différence/amplitude maximum

	27	51	52	33	45	22	28	44	40	38	20	57
Graine 27	0.00	0.65	0.68	0.16	0.49	0.14	0.03	0.46	0.35	0.30	0.19	0.81
Graine 51	0.65	0.00	0.03	0.49	0.16	0.78	0.62	0.19	0.30	0.35	0.84	0.16
Graine 52	0.68	0.03	0.00	0.51	0.19	0.81	0.65	0.22	0.32	0.38	0.86	0.14
Minimum	0	0	0	0.16	0.16	0.14	0.03	0.19	0.3	0.3	0.19	0.14
Affectation	1	2	3	1	2	1	1	2	2	1	1	3

- Cluster 1 : 27 - 33 - 22 - 28 - 38 - 20
- Cluster 2 : 51 - 45 - 44 - 40
- Cluster 3 : 52 - 57

Suite exemple (2)

	27	51	52	33	45	22	28	44	40	38	20	57
Graine 28	0.03	0.62	0.65	0.14	0.46	0.16	0	0.43	0.32	0.27	0.22	0.78
Graine 45	0.49	0.16	0.19	0.32	0	0.62	0.46	0.03	0.14	0.19	0.68	0.32
Graine 54.5	0.74	0.09	0.07	0.58	0.26	0.88	0.72	0.28	0.39	0.45	0.93	0.07
Minimum	0.03	0.09	0.07	0.14	0	0.16	0	0.03	0.14	0.19	0.22	0.07
Affectation	1	3	3	1	2	1	1	2	2	2	1	3

- Cluster 1:
  - 27 - 33 - 22 - 28 - 20 Jeunes majeurs - Centre = 26
- Cluster 2:
  - 45 - 44 - 40 - 38 Quadragénaires - Centre = 41.75
- Cluster 3:
  - 51 - 52 - 57 Quinquagénaires - Centre = 53.33

Faiblesse

- Mauvaise prise en compte des "outliers"
  - points extrêmes en dehors des groupes
  - fausses les moyennes et donc les centres
- Convergence plus ou moins rapide
- Amélioration:
  - utilisation de points centraux (médoïdes)

2.2 k-Médoïds

- Kaufman & Rousseeuw'87
- Les centres sont des points effectifs
  - recherche de centres approchés des groupes
  - calculés par substitution aléatoire:
    - choix aléatoire d'un nouveau centre
    - calcul de la différence en distance des points
    - substitution si la différence est négative
  - essai de tous les couples (x,y) de chaque groupe
    - l'un est centre, l'autre non

Forces et faiblesses

- Connue sous le nom PAM
  - Partitioning Around Medoids
- Beaucoup plus coûteuse que K-Means
  - Plus de calculs
- Plus robuste que k-means
  - Plus insensible aux "outliers"

2.3 CLARA et CLARANS

- CLARA (Clustering LARge Applications)
  - Kaufmann and Rousseeuw 1990
- Tire des petits échantillons successifs de la base
  - exemple: 5 de 40 points
- Applique PAM à chaque échantillon
  - isole les médoids
  - retourne le meilleur clustering
- La qualité d'un clustering est mesurée par la dissimilarité des objets de chaque partition sur toute la base

CLARANS (1)

- A Clustering Algorithm based on Randomized Search
  - Ng and Han'94
- Recherche d'un échantillon représentatif
  - Exploration d'un graphe où chaque nœud correspond à un ensemble de k médoids solution
- Deux nœuds sont voisins s'ils différent d'un seul médoid

CLARANS (2)

- Chaque nœud est affecté d'un coût mesurant la dissimilarité totale des points avec le médoid de leur cluster
- Il s'agit de rechercher le point de coût minimum du graphe
  - Algorithme de type "branch and bound"
  - A chaque étape, les voisins du nœuds courant sont évalués; celui correspondant à la descente maximum du coût est retenu comme solution suivante
- Plus efficace que CLARA et résultats meilleurs

3. Méthodes hiérarchiques

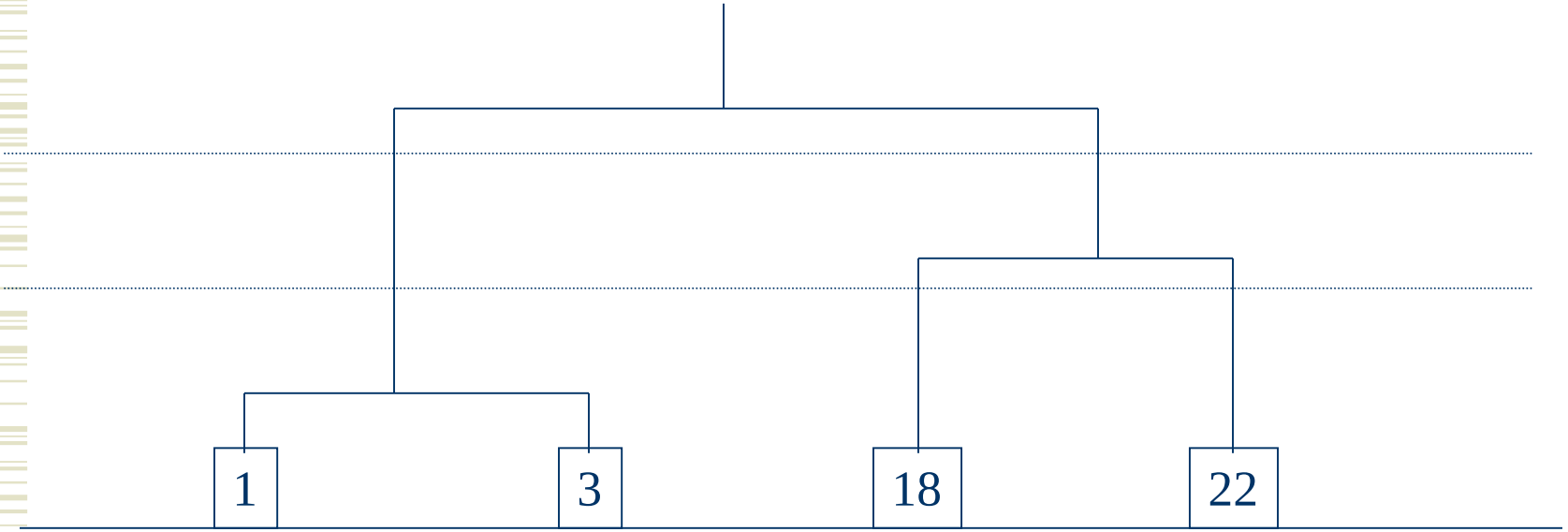
- Créer une décomposition hiérarchique en groupes (clusters)
- par agglomération de groupes similaires
- par division de groupes dissimilaires
- AGNES, DIANA, BIRCH, CURE, ROCK, Cameleon

3.1 Agglomération - Principe

- Etapes :
  - Chaque individu représente un groupe
  - Trouver les deux groupes les plus proches
  - Grouper ces deux groupes en un nouveau groupe
  - Itérer jusqu'à N groupes

Agglomération

- Exemple



Variante

- Calculer les distances de tous les points deux à deux.
- Associer tous les points dont la distance ne dépasse pas un seuil.
- Calculer le centre de chaque cluster.
- Répéter le processus avec les centres et un nouveau seuil jusqu ’à l’obtention du nombre de cluster souhaité.

Exemple (1)

- Points 27 - 51 - 52 - 33 - 45 - 22 - 28 - 44 - 40 - 38 - 20 - 57
- distance = |p1-p2| / E(pi) ; seuil = 10 %

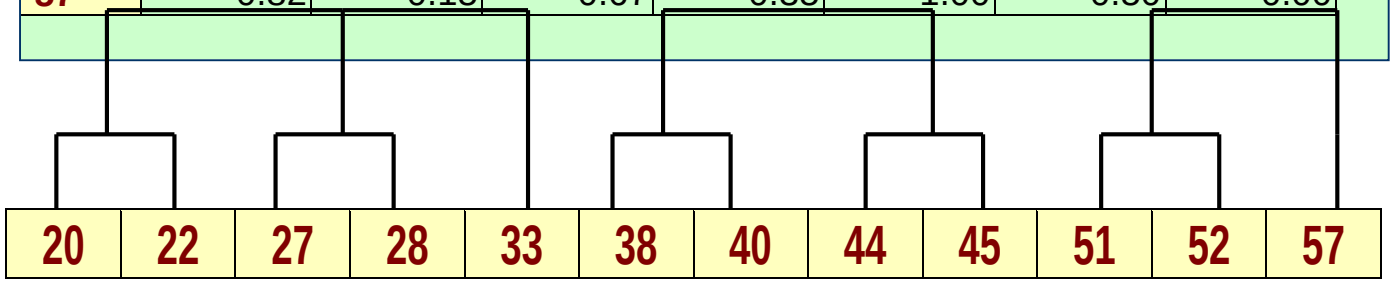
	27	51	52	33	45	22	28	44	40	38	20	57
27	0.00	0.65	0.68	0.16	0.49	0.14	0.03	0.46	0.35	0.30	0.19	0.81
51	0.65	0.00	0.03	0.49	0.16	0.78	0.62	0.19	0.30	0.35	0.84	0.16
52	0.68	0.03	0.00	0.51	0.19	0.81	0.65	0.22	0.32	0.38	0.86	0.14
33	0.16	0.49	0.51	0.00	0.32	0.30	0.14	0.30	0.19	0.14	0.35	0.65
45	0.49	0.16	0.19	0.32	0.00	0.62	0.46	0.03	0.14	0.19	0.68	0.32
22	0.14	0.78	0.81	0.30	0.62	0.00	0.16	0.59	0.49	0.43	0.05	0.95
28	0.03	0.62	0.65	0.14	0.46	0.16	0.00	0.43	0.32	0.27	0.22	0.78
44	0.46	0.19	0.22	0.30	0.03	0.59	0.43	0.00	0.11	0.16	0.65	0.35
40	0.35	0.30	0.32	0.19	0.14	0.49	0.32	0.11	0.00	0.05	0.54	0.46
38	0.30	0.35	0.38	0.14	0.19	0.43	0.27	0.16	0.05	0.00	0.49	0.51
20	0.19	0.84	0.86	0.35	0.68	0.05	0.22	0.65	0.54	0.49	0.00	1.00
57	0.81	0.16	0.14	0.65	0.32	0.95	0.78	0.35	0.46	0.51	1.00	0.00

51	52	27	28	44	45	38	40	20	22	33	57

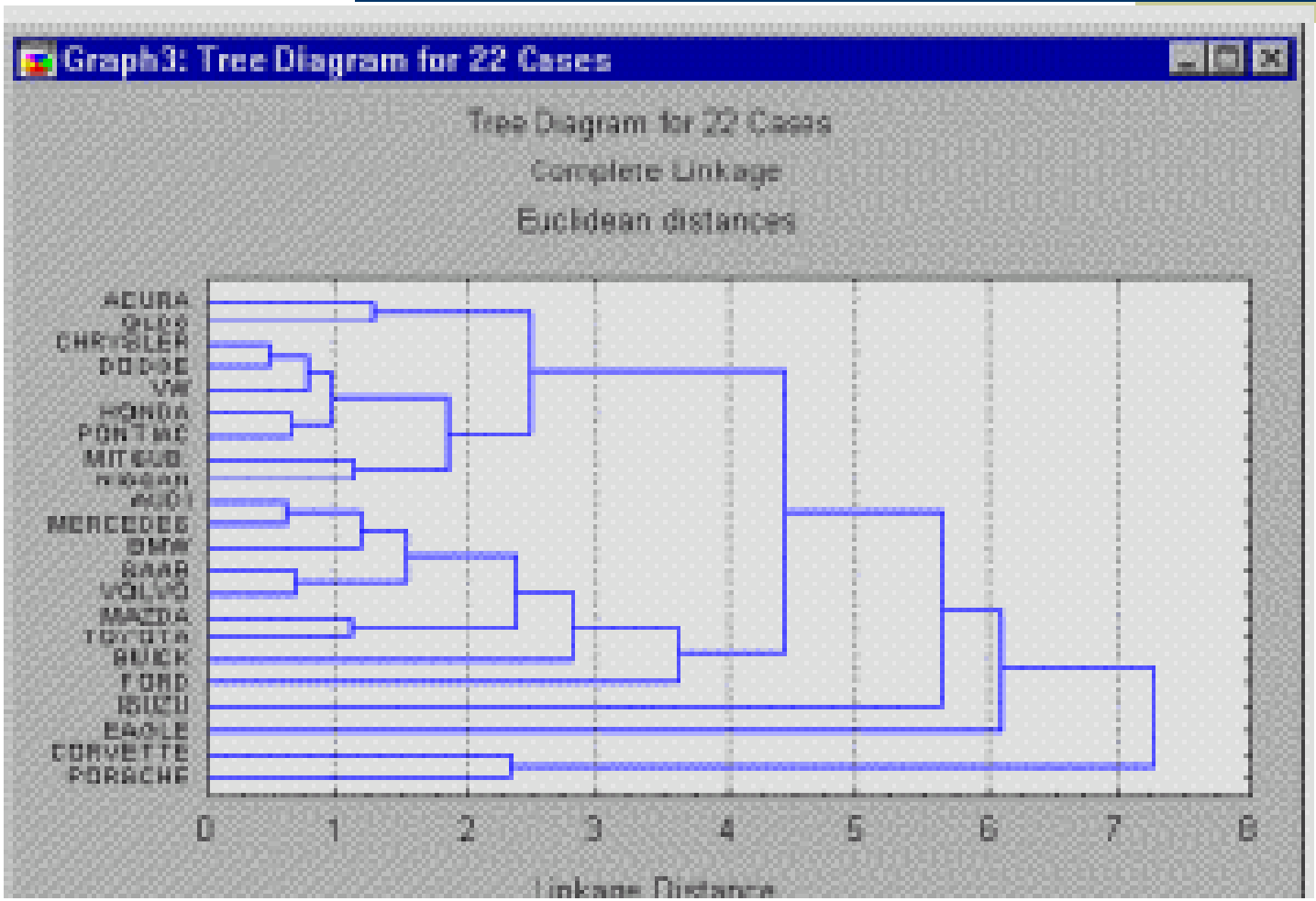
Exemple (2)

- Nouveaux centres
  - 27.5 - 51.5 - 33 - 44.5 - 21 - 39 - 57
- seuil = 20 %

	27.5	51.5	33	44.5	21	39	57
27.5	0.00	0.67	0.15	0.47	0.18	0.32	0.82
51.5	0.67	0.00	0.51	0.19	0.85	0.35	0.15
33	0.15	0.51	0.00	0.32	0.33	0.17	0.67
44.5	0.47	0.19	0.32	0.00	0.65	0.15	0.35
21	0.18	0.85	0.33	0.65	0.00	0.50	1.00
39	0.32	0.35	0.17	0.15	0.50	0.00	0.50
57	0.82	0.15	0.67	0.35	1.00	0.50	0.00



Un dendrogramme



Règles de liens

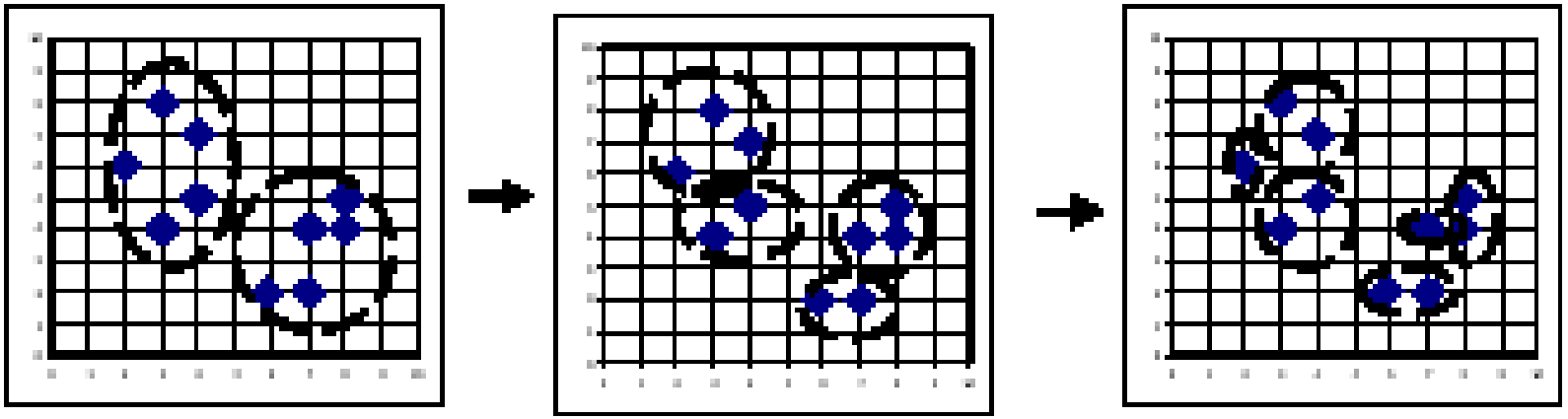
- Distance des objets les plus proches
  - tendance à former des chaînes
- Distance des objets les plus éloignés
  - bonne méthode pour des grappes
- Distance des centres
  - faible résistance aux "outliers"
- Distance moyenne
  - plus difficile à calculer

3.2 DIANA

- DIvide ANAlysis
  - Kaufmann and Rousseeuw (1990)
- Méthode par division récursive
- Tous les objets sont placés dans une cluster
- Divise de manière hiérarchique les clusters
  - selon un critère de dispersion des objets
  - e.g., celle(s) dont les objets les plus proches sont les plus éloignés
- Stoppe quand le nombre de clusters est atteint ou les clusters contiennent 1 seul objet

Exemple

- Possibilité d'utiliser un seuil de distance



3.3 BIRCH

- Balanced Iterative Reducing and Clustering using Hierarchies
  - Zhang, Ramakrishnan, Livny (SIGMOD'96)
- Par division, incrémentale en une passe
- Sauvegarde les informations de clustering dans un arbre balancé
- Chaque entrée de l'arbre décrit une cluster
- Les nouveaux nœuds sont insérés sous l'entrée la plus proche

Arbre de clustering

- CF (N,$\overrightarrow{LS}$,SS) = Clustering Feature
  - N: Nombre de points dans la cluster = Moment 0
  - $\overrightarrow{LS}$: Somme des points dans la cluster = Moment 1
  - SS: Somme des carrés des points dans la cluster = Mo. 2
- CF Tree
  - Arbre de recherche équilibré (proche B-tree)
  - Un noeud mémorise la somme des CF de chaque fils
  - Les feuilles représentent les clusters
    - mémorise les CF des clusters
    - possède un diamètre maximum (seuil)

Construction incrémentale

- Phase 1:
  - parcours de la base pour construire un CF-tree initial en mémoire
  - une feuille est éclatée quand le seuil de diamètre est dépassé
  - les moments sont alors remontés aux nœuds internes
- Permet d'obtenir un résumé comprimé de la base respectant les caractéristiques de groupage

Algorithme BIRCH

Input:
$D = \{t_1, t_2, ..., t_n\}$ // Set of elements
T // Threshold for CF tree construction.
Output:
K // Set of clusters.
BIRCH Clustering Algorithm:

```
for each t_i ∈ D do
    determine correct leaf node for t_i insertion;
    if threshold condition is not violated then
        add t_i to cluster and update CF triples;
    else
        if room to insert t_i then
            insert t_i as single cluster and update CF triples;
        else
            split leaf node and redistribute CF features;
```

Phase 2: Amélioration des clusters

- Si l'arbre ne tient pas en mémoire, augmenter le seuil de diamètre
- Etape 2 optionnelle :
  - Appliquer un algorithme de clustering en mémoire aux nœuds feuilles du CF tree, chaque nœud étant traité comme un point
  - Placer les points dans la cluster de centre le plus proche

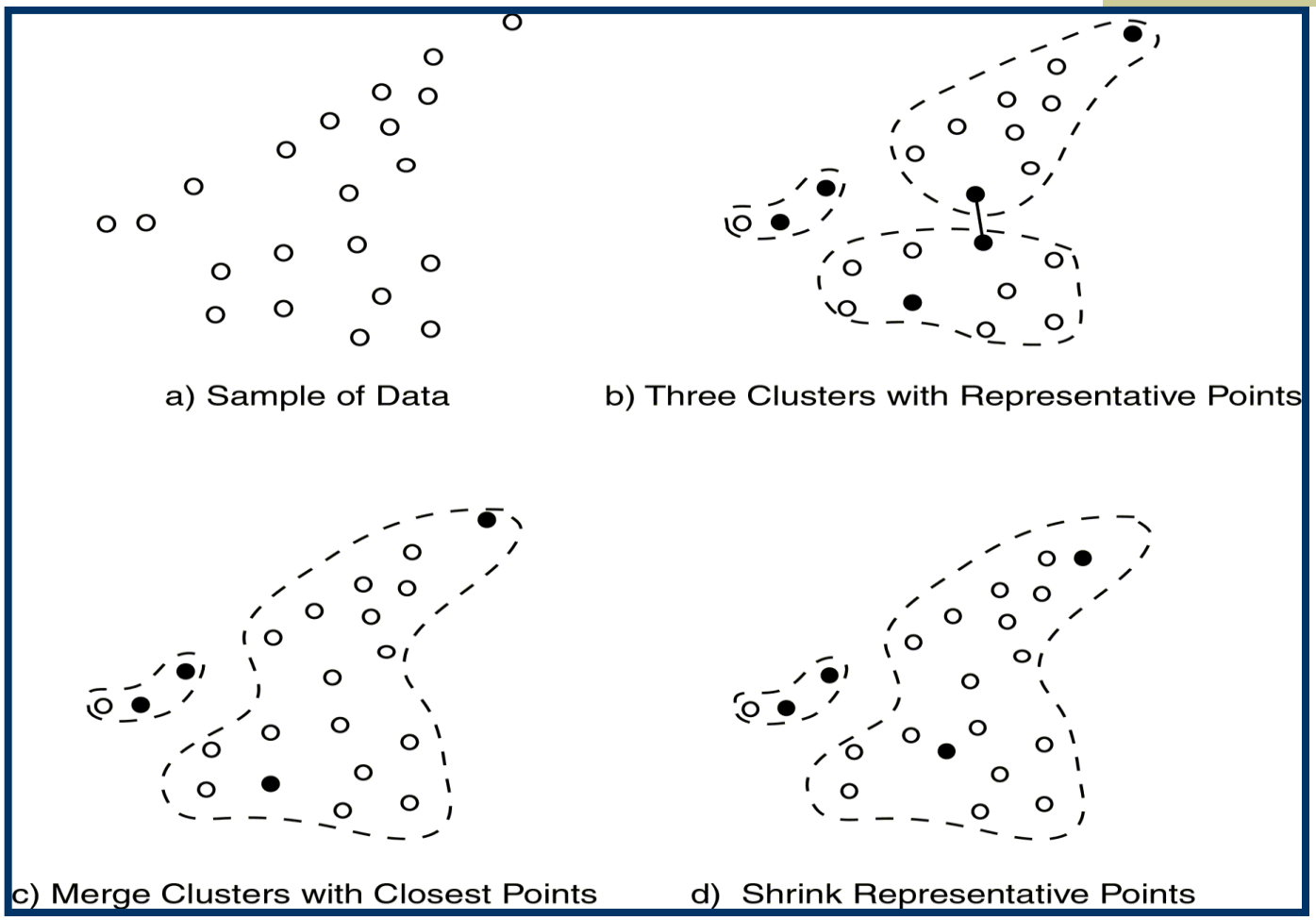
Forces et Faiblesses

- Passe à l'échelle avec un grand nombre d'objets
- Trouve un bon clustering en une passe et permet d'améliorer ensuite
- Marche mieux avec des groupes sphériques
  - utilisation du diamètre des clusters

3.4 CURE

- Groupe en utilisant des représentants
- Utilise un nombre fixe de points pour représenter un groupe
- Les points sont choisis:
  - 1) en choisissant des points écartés
  - 2) en les déplaçant vers le centre du groupe
- A chaque étape, les 2 groupes ayant les représentants les plus proches sont fusionnés

Principes de CURE



Algorithme CURE

Input:
$D = \{t_1, t_2, ..., t_n\}$ //Set of elements.
k // Desired number of clusters.
Output:
Q //Heap containing one entry for each cluster.
CURE Algorithm:
$T = build(D)$; // Put each point in Tree
$Q = heapify(D)$; // Initially build heap with one entry per item;
repeat
$u = min(Q)$;
$delete(Q, u.close)$;
$w = merge(u, v)$;
$delete(T, u)$;
$delete(T, v)$;
$insert(T, w)$;
for each $x \in Q$ **do**
x.close = find closest cluster to x;
if x **is closest to** w **then**
w.close = x;
$insert(Q, w)$;
until number of nodes in Q **is** k;

Application aux larges BD

- Tirer un échantillon
- Diviser l'échantillon en p partitions de q points
- Grouper les points partiellement dans chaque partition
- Enlever les outliers basés sur la taille des groupes
- Terminer le clustering de l'échantillon
- Grouper la base entière en classant chaque point dans le groupe de son représentant le plus proche

Cure : Forces et Faiblesses

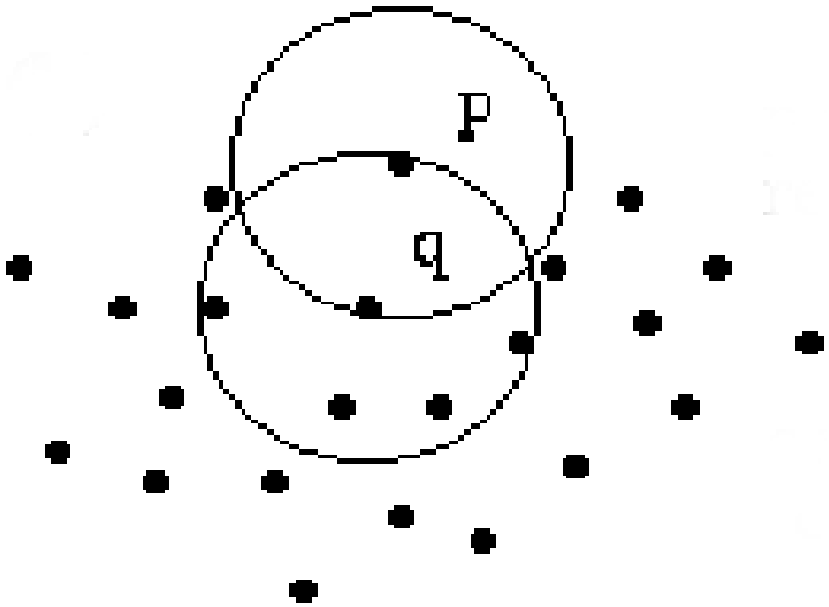
- S'adapte bien à la géométrie des clusters
  - représentant multiples
  - déplacement du bord vers le centre des représentants
- Philosophie pour passer à l'échelle

4. Méthodes par densité

- Principe
  - Utilisation de la densité à la place de la distance
  - Un point est voisin d'un autre point s'il est à une distance inférieure à une valeur fixée
  - Un point est dense si le nombre de ses voisins dépasse un certain seuil

Points denses

- q est dense, mais pas p



DBSCAN

- La découverte d'un groupe se déroule en 2 étapes
  - choisir aléatoirement un point dense
  - tous les points qui sont atteignables à partir de ce point, selon le seuil de densité, forment un groupe

Algorithme DBSCAN

Input:

$D = \{t_1, t_2, ..., t_n\}$ //Set of elements.

$MinPts$ // Number of points in cluster.

Eps // Maximum distance for density measure.

Output:

$K = \{K_1, K_2, ..., K_k\}$ //Set of clusters.

DBSCAN Algorithm:

$k = 0$; // Initially there are no clusters.

for $i = 1$ *to* n **do**

  if t_i **is not in a cluster then**

    $X = \{t_j \mid t_j$ **is density-reachable from** $t_i\}$;

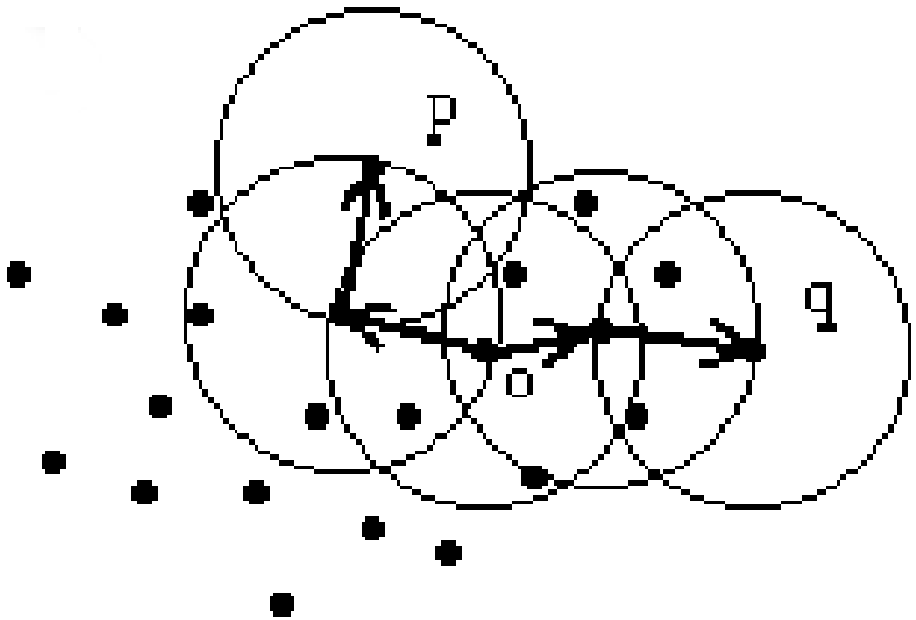
    if X **is a valid cluster then**

      $k = k + 1$;

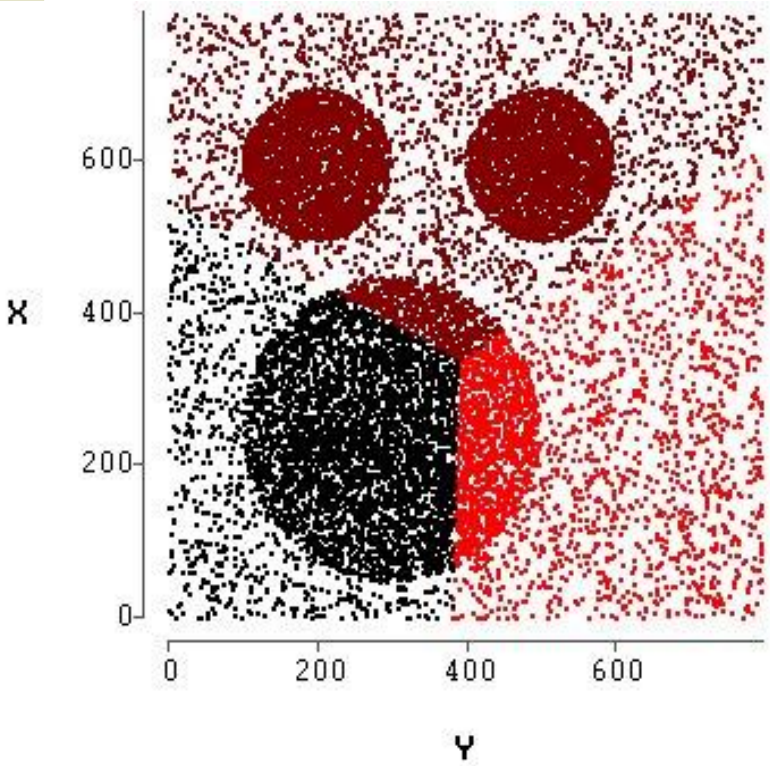
      $K_k = X$;

Exemple

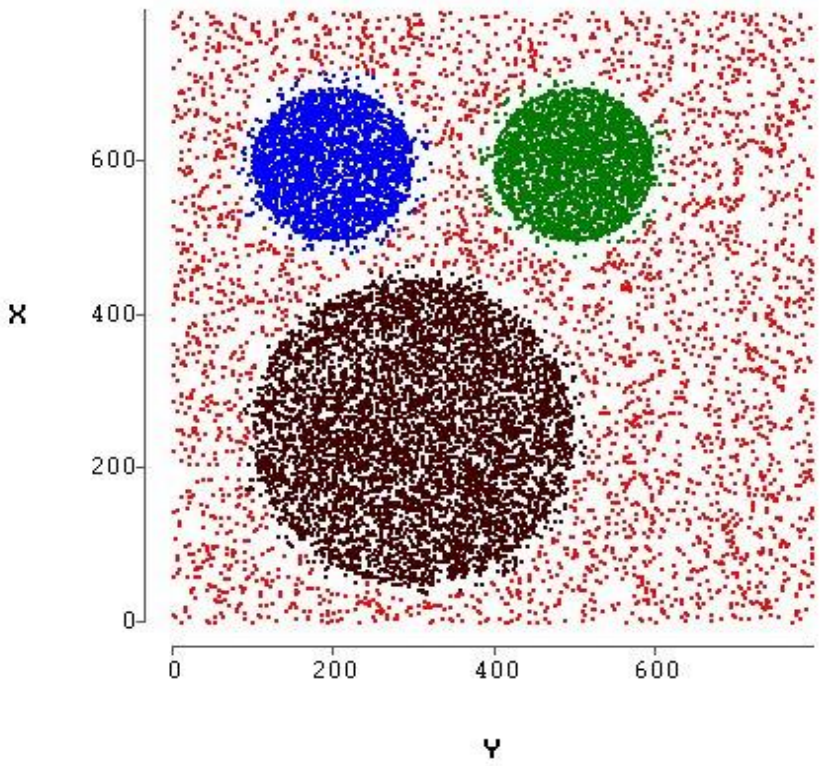
- on commence par o, ce cluster contient p, q, etc.



Comparaison K-means et DBSCAN



Résultat de K-means



Résultat de Density

Bilan

- Différents types de méthodes
  - Méthodes par partitionnement
  - Méthodes hiérarchique
  - Méthodes basées densités
  - Méthodes par grilles
  - Méthodes mixtes, Réseaux de neurones …
- Recherches en cours
  - Passage à l'échelle
    - échantillonnage, optimisation, indexation, mixages, …
  - Prise en compte de contraintes
    - connaissances utilisateur ou données
    - objectifs utilisateur