

- Comprendre la notion d'apprentissage **NON supervisé**
- Le lien à la notion de **Découverte de Structures**
- Connaître des algorithmes de regroupement
  - Hiérarchiques et leur représentation graphique : **dendrogramme**
  - par Optimisation type **k-means, ISODATA**
- Comprendre que la notion de Similitude liée à la vaste notion mathématique de Distance est **subjective mais centrale** dans cette problématique
- Savoir **construire un espace** de mesure multi-dimensionnelle et définir une **mesure de similarité** dans cette espace
- Savoir **choisir l'algorithme** à utiliser en fonction des données en entrée
- Principes
  - Contexte non supervisé
  - "Révéler" l'organisation de motifs en groupes cohérents
  - Processus Subjectif
- Disciplines : Biologie, Zoologie, Psychiatrie, Sociologie, Géologie, Géographie...
- Synonymes : Apprentissage non supervisé, Taxonomie, Typologie, Partition

- Groupe ou Cluster : un ensemble d'objets ou d'individus
  - Semblables entre eux à l'intérieur d'un groupe
  - Différents d'un groupe à l'autre
- Segmentation ou Cluster analysis  
Classement des individus ou objets dans différents groupes ou segments
- Le clustering est une technique **non dirigée** (c-à-d. il n'y a pas de variable target )
- Quelques applications :
  - Reconnaissance de forme non supervisée
  - Taxonomie (biologie, zoologie)
  - Segmentation des marchés (marketing)
  - Geo-segmentation
  - WWW : classification des sites / des Weblog pour découvrir des profils d'accès semblables...

- Un bon algorithme de classification fera en sorte qu'il y aura une :
  - petite variabilité intra-classe (c-à-d petite distance entre les individus d'un même groupe)
  - grande variabilité inter-classe (c-à-d grande distance entre les individus de groupes différents)
- La qualité des résultats de la classification dépendra de la mesure de distance utilisée et de l'algorithme choisi pour l'implanter.

## • Hard Clustering

Soit  $X = \{x_1, \dots, x_n\}$

On appelle  $m$ -clustering de  $X$  la partition de  $X$  en  $m$  ensembles (clusters)

$C_1, \dots, C_m$  tels que :

- 1  $C_i \neq \emptyset, \quad i = 1, \dots, m$
- 2  $\bigcup_{i=1}^m C_i = X$
- 3  $C_i \cap C_j = \emptyset, \quad i \neq j, \quad i, j = 1, \dots, m$

## • Fuzzy Clustering - Zadeh (1965)

Soit  $X = \{x_1, \dots, x_n\}$

On appelle  $m$ -clustering flou de  $X$  en  $m$  clusters est caractérisé par  $m$

fonctions d'appartenance  $u_j$  avec :

- 1  $u_j : X \rightarrow [0, 1], \quad j = 1, \dots, m$
- 2  $\sum_{j=1}^m u_j(x_i) = 1, \quad i = 1, \dots, N$
- 3  $0 < \sum_{i=1}^N u_j(x_i) < N, \quad j = 1, \dots, m$

- **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
- **Caractéristiques** :
  - Extensibilité
  - Capacité à traiter différents types de données
  - Découverte de clusters de différents formes
  - Connaissances requises (paramètres de l'algorithme)
  - Capacité à traiter les données bruitées et isolées.

- 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

## $k$ -means

### Algorithme 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  $dist(O, M_i)$  est minimale
- 3 Recalculer  $M_i$  de chaque cluster (le barycentre)
- 4 Aller à l'étape 2 si on vient de faire une affectation

### Utilisation conseillée :

- 1 lorsque le nb de clusters est connu
- 2 pour le regroupement rapide de grands ensembles de données

## $k$ -medoids

### Algorithme 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  $dist(O, M_i)$  est minimale
- 3 Recalculer  $M_i$  de chaque cluster (le **point le plus proche du barycentre**)
- 4 Aller à l'étape 2 si on vient de faire une affectation

### Utilisation conseillée :

- 1 lorsque le nb de clusters est connu
- 2 pour le regroupement rapide de **données catégorielles**
- 3 échelle vers de grands ensembles de données

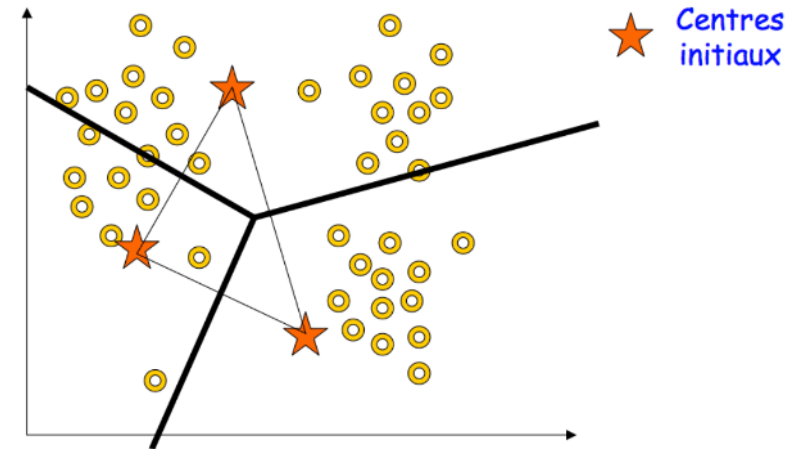
# k-means : Exemple

- AI for bio
- J-P Comet
- Introduction
- Trees
- Clustering
  - Objectifs
  - Partitionnement
  - k-means**
  - GMM
  - Hierarchiques
  - Algorithmes-densité
  - DBSCAN + OPTICS
  - Résumé
- kNN
- Bayes
- SVM
- NN
- Evaluation
- Data Mining
- Project

- $A = \{1, 2, 3, 6, 7, 8, 13, 15, 17\}$ . Créer 3 clusters à partir de  $A$
- On prend 3 objets au hasard : 1, 2 et 3.  
 $\Rightarrow C_1 = \{1\}, M_1 = 1, C_2 = \{2\}, M_2 = 2, C_3 = \{3\}$  et  $M_3 = 3$
- Chaque objet  $O$  est affecté au cluster au milieu duquel,  $O$  est le plus proche.  
 $\Rightarrow C_1 = \{1\}, M_1 = 1$   
 $\Rightarrow C_2 = \{2\}, M_2 = 2$   
 $\Rightarrow C_3 = \{3, 6, 7, 8, 13, 15, 17\}, M_3 = 69/7 = 9.86$
- $dist(3, M_2) < dist(3, M_3) \Rightarrow 3$  passe dans  $C_2$ .  
 Tous les autres objets ne bougent pas.  
 $C_1 = \{1\}, M_1 = 1, C_2 = \{2, 3\}, M_2 = 2.5,$   
 $C_3 = \{6, 7, 8, 13, 15, 17\}, M_3 = 66/6 = 11$
- $dist(6, M_2) < dist(6, M_3) \Rightarrow 6$  passe dans  $C_2$ .  
 Tous les autres objets ne bougent pas.  
 $C_1 = \{1\}, M_1 = 1, C_2 = \{2, 3, 6\}, M_2 = 11/3 = 3.67,$   
 $C_3 = \{7, 8, 13, 15, 17\}, M_3 = 12$
- $dist(2, M_1) < dist(2, M_2) \Rightarrow 2$  passe en  $C_1$ .  
 $dist(7, M_2) < dist(7, M_3) \Rightarrow 7$  passe en  $C_2$ . Les autres ne bougent pas.  
 $C_1 = \{1, 2\}, M_1 = 1.5, C_2 = \{3, 6, 7\}, M_2 = 5.34,$   
 $C_3 = \{8, 13, 15, 17\}, M_3 = 13.25$
- $dist(3, M_1) < dist(3, M_2) \Rightarrow 3$  passe en 1.  
 $dist(8, M_2) < dist(8, M_3) \Rightarrow 8$  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

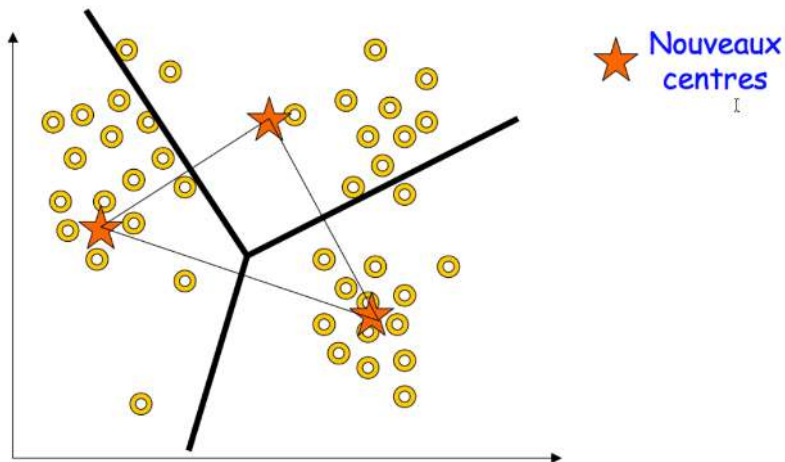
# k-means : illustration (1)

- AI for bio
- J-P Comet
- Introduction
- Trees
- Clustering
  - Objectifs
  - Partitionnement
  - k-means**
  - GMM
  - Hierarchiques
  - Algorithmes-densité
  - DBSCAN + OPTICS
  - Résumé
- kNN
- Bayes
- SVM
- NN
- Evaluation
- Data Mining
- Project



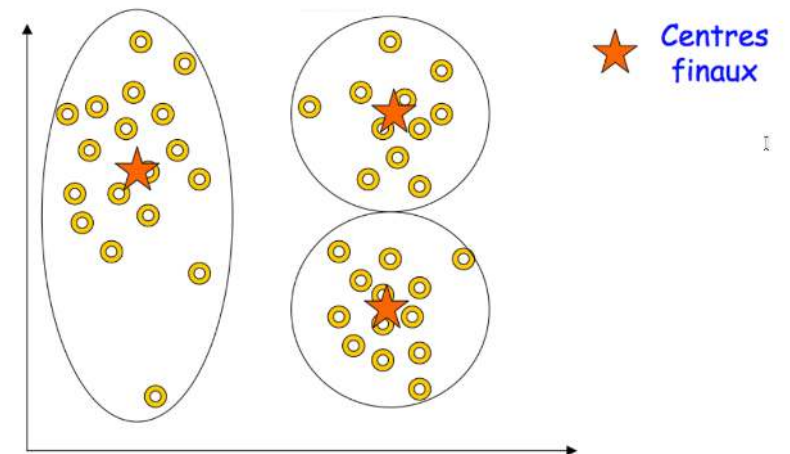
# k-means : illustration (2)

- AI for bio
- J-P Comet
- Introduction
- Trees
- Clustering
  - Objectifs
  - Partitionnement
  - k-means**
  - GMM
  - Hierarchiques
  - Algorithmes-densité
  - DBSCAN + OPTICS
  - Résumé
- kNN
- Bayes
- SVM
- NN
- Evaluation
- Data Mining
- Project



# k-means : illustration (3)

- AI for bio
- J-P Comet
- Introduction
- Trees
- Clustering
  - Objectifs
  - Partitionnement
  - k-means**
  - GMM
  - Hierarchiques
  - Algorithmes-densité
  - DBSCAN + OPTICS
  - Résumé
- kNN
- Bayes
- SVM
- NN
- Evaluation
- Data Mining
- Project



## Force

- Relativement extensible dans le traitement d'ensembles de taille importante
- Relativement efficace :  $O(t.k.n)$ , où  $n$  représente le nb objets,  $k$  le nb de clusters, et  $t$  le nb d'itérations. Normalement,  $k, t \ll n$ .

## Faiblesses

- N'est pas applicable en présence d'attributs où la moyenne n'est pas définie
- On doit spécifier  $k$  (nombre de clusters)
- Incapable de traiter des données bruitées
- Les clusters sont construits par rapports à des objets inexistantes (les moyennes)
- Ne peut pas découvrir les groupes non-convexes
- Les outliers sont mal gérés.

Choisir arbitrairement  $k$  medoïdes

Répéter

    affecter chaque objet restant au medoïde le plus proche

    Choisir aléatoirement un non-medoïde  $O_h$

    Pour chaque medoïde  $O_j$

        Calculer le coût  $TC_{jh}$  du remplacement de  $O_j$  par  $O_h$

        Si  $TC_{jh} < 0$  alors

            Remplacer  $O_j$  par  $O_h$

            Calculer les nouveaux clusters

        Finsi

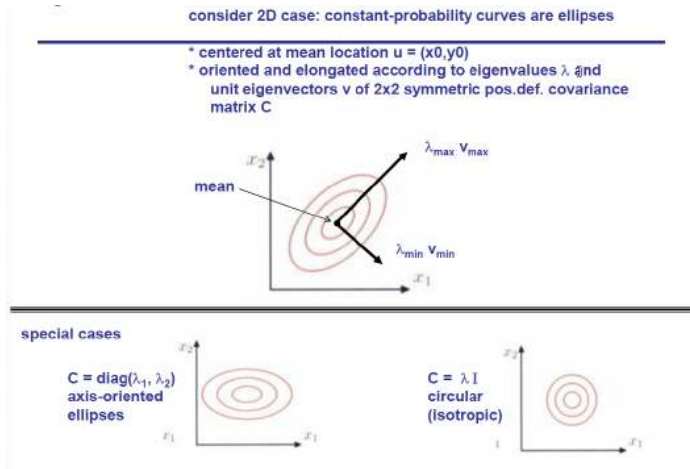
    FinPour

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

- $TC_{jh}$  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. autrement dit, les clusters seront plus compacts.

- Soit  $A = \{1, 3, 4, 5, 8, 9\}$ ,  $k = 2$  et  $M = \{1, 8\}$  ensemble des médoïdes
- $C_1 = \{1, 3, 4\}$  et  $C_2 = \{5, 8, 9\}$ .  
 $E_{\{1,8\}} = \text{dist}(3, 1)^2 + \text{dist}(4, 1)^2 + \text{dist}(5, 8)^2 + \text{dist}(9, 8)^2 = 23$
- Comparons 1 et 3 :  $M = \{3, 8\}$ ,  
 $C_1 = \{1, 3, 4, 5\}$  et  $C_2 = \{8, 9\}$   
 $E_{\{3,8\}} = \text{dist}(1, 3)^2 + \text{dist}(4, 3)^2 + \text{dist}(5, 3)^2 + \text{dist}(9, 8)^2 = 10$   
 $TC_{1,3} = E_{\{3,8\}} - E_{\{1,8\}} = -13 < 0$   
⇒ le remplacement est fait.
- Comparons 3 et 4 :  $M = \{4, 8\}$ ,  
 $C_1$  et  $C_2$  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 :  $M = \{5, 8\}$ ,  
 $C_1$  et  $C_2$  inchangés et  $E_{\{5,8\}} > E_{\{3,8\}}$

- Sélection des centres initiaux
- Calcul des similarités
- Calcul des centres ( $k$ -medoïdes : [Kaufman & Rousseeuw'87])
- GMM : Variantes de  $k$ -means basées sur les probabilités
- $k$ -modes : données catégorielles [Huang'98]
- $k$ -prototype : données mixtes (numériques et catégorielles)



- un GMM est un modèle probabiliste de  $p(X)$
- la vraisemblance de  $x$  étant donné une distribution Gaussienne :

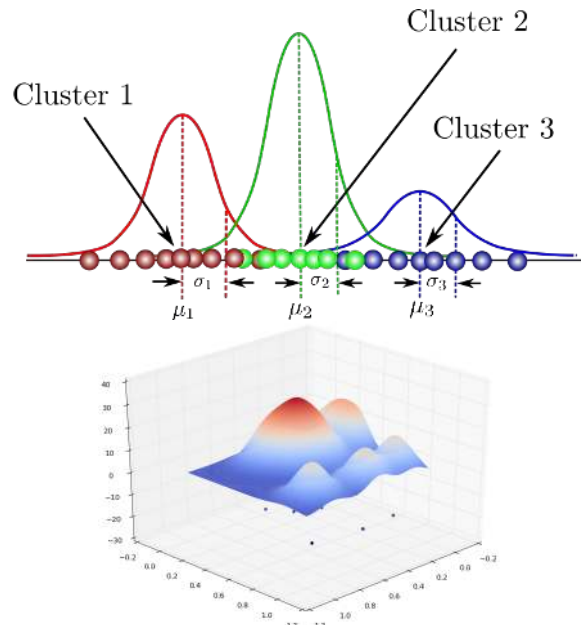
$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right)$$

- la vraisemblance de  $x$  étant donné un GMM :

$$p(x) = \sum_{k=1}^K w_k \cdot p(x|\mu_k, \Sigma_k) \Leftrightarrow p(x) = \sum_{k=1}^K P(k) \cdot p(x|k)$$

où  $K$  est le nombre de Gaussiennes et  $w_k$  est le poids associé à la Gaussienne  $k$ , avec :  $\sum_k w_k = 1, w_k \geq 0$

- Propriétés du GMM :
  - les GMM sont connus comme des approximateurs universels de densités
  - on utilise souvent une matrice de covariance  $\Sigma$  diagonale car le nombre de paramètres à optimiser est plus gérable : Gaussiennes sphériques :  $\Sigma = \sigma^2 \times Id$
  - on entraîne les GMM avec Expectation-Maximisation (E-M) : un algorithme efficace de type maximum de vraisemblance (ML) pour optimiser les paramètres d'un modèle probabiliste



Notations :

- $z_{nk}$  : variable aléatoire valant 1 si  $x_n$  provient de  $k$ , 0 sinon. probabilité que  $x_n$  provienne de  $k$  :  $p(z_{nk} = 1|x_n)$
- probabilité d'observer un point provenant de  $k$  :  $\pi_k = p(z_k = 1)$
- $z = (z_1, \dots, z_K)$  où une seule composante vaut 1
- $p(z) = p(z_1 = 1)^{z_1} p(z_2 = 1)^{z_2} \dots p(z_K = 1)^{z_K} = \prod_{k=1}^K \pi_k^{z_k}$

Calcul de la vraisemblance

- $p(x_n|z) = \prod_{k=1}^K \mathcal{N}(x_n|\mu_k, \Sigma_k)^{z_k}$
- $p(x_n, z) = p(x_n|z) \times p(z)$
- $p(x_n) = \sum_{k=1}^K p(x_n|z_k) p(z_k = 1) = \sum_{k=1}^K \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)$
- $p(x) = \prod_{n=1}^N p(x_n) = \prod_{n=1}^N \sum_{k=1}^K \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)$
- $\ln p(x) = \sum_{n=1}^N \ln \sum_{k=1}^K \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)$

Règles de Bayes :  $P(A|B) = P(A, B)/P(B)$

$$p(z_k = 1|x_n) = \frac{p(z_k = 1, x_n)}{p(x_n)} = \frac{p(x_n|z_k = 1) \times p(z_k = 1)}{p(x_n)}$$

$$p(z_k = 1|x_n) = \frac{p(x_n|z_k = 1) \times p(z_k = 1)}{\sum_{j=1}^K p(x_n|z_j = 1) p(z_j = 1)} = \frac{\pi_k \cdot \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \cdot \mathcal{N}(x_n|\mu_j, \Sigma_j)}$$

## Algorithme des $k$ -means :

- 1 initialiser les  $\mu_k$
- 2 étape Estimation : assigner chaque point à un cluster
- 3 étape Maximisation : étant donné les clusters, modifier les  $\mu_k$
- 4 alterner étapes 2 et 3 jusqu'à stabilisation

## Algorithme EM (Expectation-Maximisation) pour les GMM :

- 1 initialiser les  $\mu_k, \Sigma_k$  et les coeff  $\pi_k$
- 2 étape Estimation : assigner à chaque point  $X_i$  un vecteur de score  $\gamma_{i,k}$  ( $k$  parcourant les modèles gaussiens)
- 3 étape Maximisation : étant donné les scores, ajuster les  $\mu_k, \Sigma_k$  et  $\pi_k$
- 4 Calculer la vraisemblance puis alterner étapes 2 et 3 jusqu'à convergence de la vraisemblance

## Algorithme EM (Expectation-Maximisation) pour les GMM :

- 1 initialiser les  $\mu_k, \Sigma_k$  et les coeff  $\pi_k$
- 2 étape Estimation : assigner à chaque point  $X_i$  un vecteur de score  $\gamma_{i,k}$  ( $k$  parcourant les modèles gaussiens)

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}$$

- 3 étape Maximisation : étant donné les scores, ajuster les  $\mu_k, \Sigma_k$  et  $\pi_k$

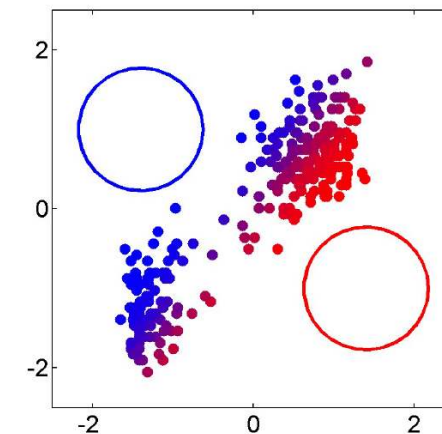
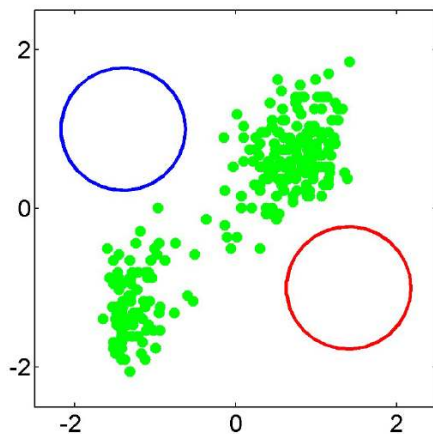
$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^N \gamma_{nk} x_n \quad \text{où } N_k = \sum_{n=1}^N \gamma_{nk}$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^N \gamma_{nk} (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N}$$

- 4 Calculer la vraisemblance puis alterner étapes 2 et 3 jusqu'à convergence de la vraisemblance

$$\ln p(X | \mu, \Sigma, \pi) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) \right\}$$



AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

**GMM**

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

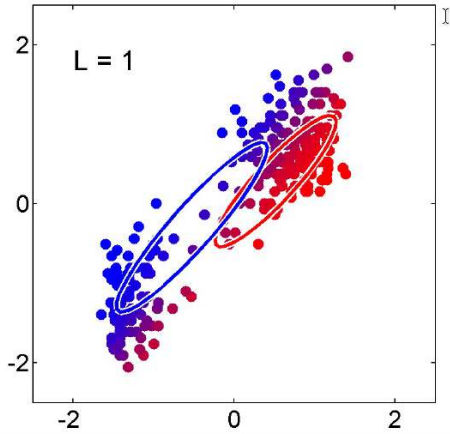
SVM

NN

Evaluation

Data Mining

Project



AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

**GMM**

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

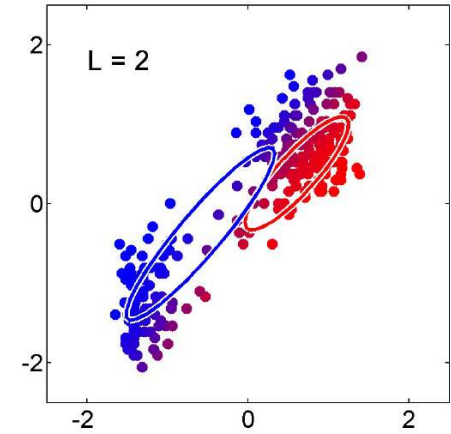
SVM

NN

Evaluation

Data Mining

Project



AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

**GMM**

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

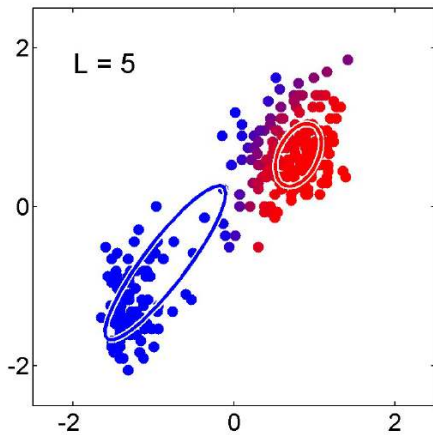
SVM

NN

Evaluation

Data Mining

Project



AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

**GMM**

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

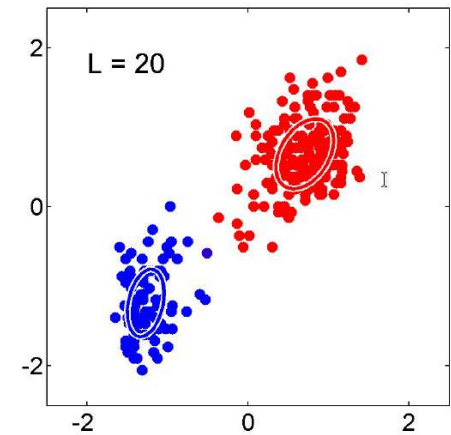
SVM

NN

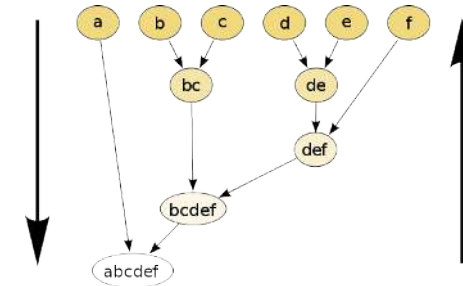
Evaluation

Data Mining

Project



- 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
- caractéristiques :
  - Une méthode hiérarchique : construit une hiérarchie de clusters, non seulement une partition unique des objets.
  - Le nombre de clusters  $k$  n'est pas exigé comme donnée
  - Utilise une matrice de distances comme critère de clustering
  - Une condition de terminaison peut être utilisée (ex. Nombre de clusters)



- Résultat : Graphe hiérarchique qui peut être coupé à un niveau de dissimilarité pour former une partition.
- La hiérarchie de clusters est représentée comme un arbre de clusters, appelé dendrogramme
- Les feuilles de l'arbre représentent les objets
- Les nœuds intermédiaires de l'arbre représentent les clusters

AGNES : AGglomerative NESTing    DIANA : Dlvivise ANALYSIS

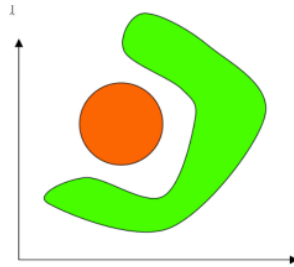
- Distance entre les centres des clusters (Centroid Method)
- Distance minimale entre toutes les paires de données des 2 clusters (Single Link Method)
 
$$d(i, j) = \min_{x \in C_i, y \in C_j} \{d(x, y)\}$$
- Distance maximale entre toutes les paires de données des 2 clusters (Complete Link Method)
 
$$d(i, j) = \max_{x \in C_i, y \in C_j} \{d(x, y)\}$$
- Distance moyenne entre toutes la paires d'enregistrements (Average Linkage)
 
$$d(i, j) = \text{avg}_{x \in C_i, y \in C_j} \{d(x, y)\}$$

- Avantages :
  - Conceptuellement simple
  - Propriétés théoriques sont bien connues
  - Quand les clusters sont groupés, la décision est définitive  $\Rightarrow$  le nombre d'alternatives différentes à examiner est réduit
- Inconvénients :
  - Groupement de clusters est définitif  $\Rightarrow$  décisions erronées sont impossibles à modifier ultérieurement
  - Méthodes non extensibles pour des ensembles de données de grandes tailles



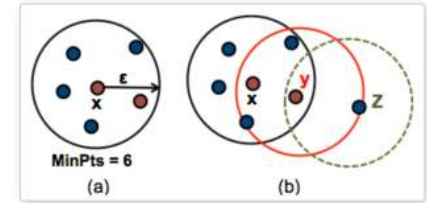
- Algorithmes de Partitionnement : Construire plusieurs partitions puis les évaluer selon certains critères
- Algorithmes hiérarchiques : Créer une décomposition hiérarchique des objets selon certains critères
- **Algorithmes basés sur la densité** : basés sur des notions de connectivité et de densité

- Pour ce types de problèmes, l'utilisation de mesures de similarité (distance) est moins efficace que l'utilisation de densité de voisinage.

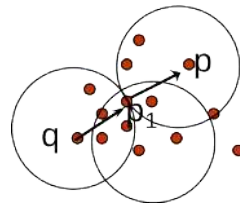


- Voit les clusters comme des régions denses séparées par des régions qui le sont moins (bruit)
- Deux paramètres :
  - $\epsilon$  : Rayon maximum du voisinage
  - $MinPts$  : Nombre minimum de points dans le  $\epsilon$ -voisinage d'un point
- Voisinage :  $V_\epsilon(p) = \{q \in D | dist(p, q) \leq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

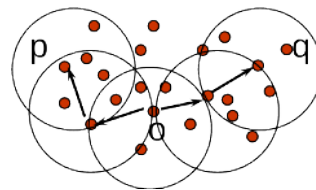
- $p \in V_\epsilon(q)$
- $|V_\epsilon(q)| \geq MinPts$



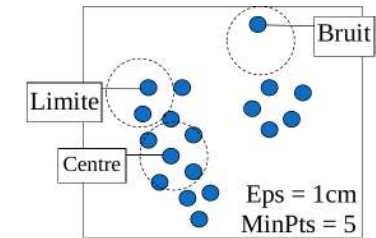
- **Accessibilité** :  
 $p$  est accessible à partir de  $q$  resp. à  $\epsilon, MinPts$  s'il existe  $p_1, \dots, p_n, p_1 = q, p_n = p$  t.q.  $p_{i+1}$  est directement densité-accessible à partir de  $p_i$



- **Connexité** :  
 $p$  est connecté à  $q$  resp. à  $\epsilon, MinPts$  s'il existe un point  $o$  t.q.  $p$  et  $q$  accessibles à partir de  $o$  resp. à  $\epsilon, MinPts$ .



- **Density Based Spatial Clustering of Applications with Noise**
- Un cluster est l'ensemble maximal de points connectés
- Découvre des clusters non nécessairement convexes
- **Algorithme**

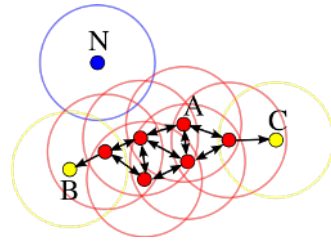


```

DBSCAN(D, eps, MinPts)
    C = 0
    pour chaque point P non visité des données D
        marquer P comme visité
        PtsVoisins = epsilonVoisinage(D, P, eps)
        si tailleDe(PtsVoisins) < MinPts
            marquer P comme BRUIT
        sinon
            C++
            etendreCluster(D, P, PtsVoisins, C, eps, MinPts)

etendreCluster(D, P, PtsVoisins, C, eps, MinPts)
    ajouter P au cluster C
    pour chaque point P' de PtsVoisins
        si P' n'a pas été visité
            marquer P' comme visité
            PtsVoisins' = epsilonVoisinage(D, P', eps)
            si tailleDe(PtsVoisins') >= MinPts
                PtsVoisins = PtsVoisins U PtsVoisins'
        si P' n'est membre d'aucun cluster
            ajouter P' au cluster C
    
```

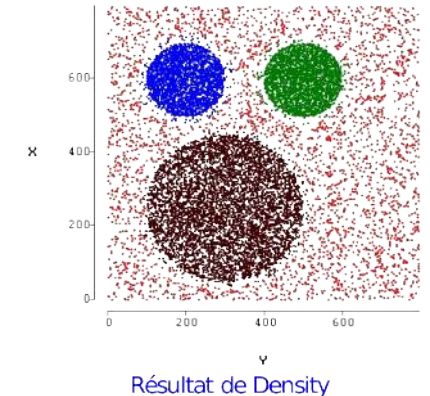
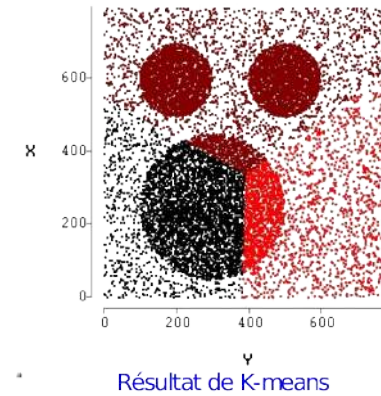
- Density Based Spatial Clustering of Applications with Noise
- Un cluster est l'ensemble maximal de points connectés
- Découvre des clusters non nécessairement convexes
- Algorithme



```

DBSCAN(D, eps, MinPts)
  C = 0
  pour chaque point P non visité des données D
    marquer P comme visité
    PtsVoisins = epsilonVoisinage(D, P, eps)
    si tailleDe(PtsVoisins) < MinPts
      marquer P comme BRUIT
    sinon
      C++
      etendreCluster(D, P, PtsVoisins, C, eps, MinPts)

  etendreCluster(D, P, PtsVoisins, C, eps, MinPts)
    ajouter P au cluster C
    pour chaque point P' de PtsVoisins
      si P' n'a pas été visité
        marquer P' comme visité
        PtsVoisins' = epsilonVoisinage(D, P', eps)
        si tailleDe(PtsVoisins') >= MinPts
          PtsVoisins = PtsVoisins U PtsVoisins'
      si P' n'est membre d'aucun cluster
        ajouter P' au cluster C
    
```



- Extension de DBSCAN
- $\epsilon$  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  $MinPts$ <sup>ième</sup> point le plus proche :

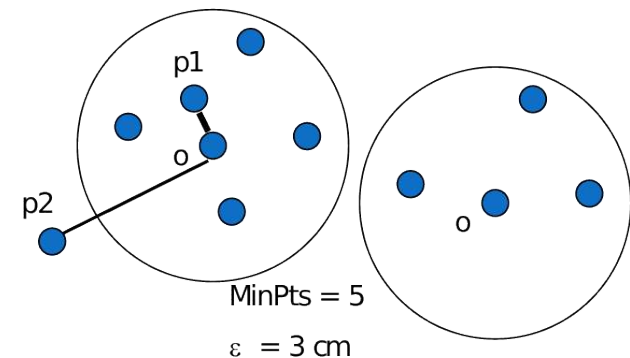
$$\text{core-distance}_{\epsilon, MinPts}(p) = \begin{cases} \text{Indéfini} & \text{si } |V_{\epsilon}(p)| < MinPts \\ \text{distance au } MinPts\text{-ième pt le + proche} & \text{sinon} \end{cases} \quad (1)$$

⇒ plus petite distance rendant le point central dense ( $min_{\epsilon}$ )

- La **reachability-distance** du point  $p$  à un autre point  $o$  est le max entre la distance entre  $o$  et  $p$ , et la core-distance de  $p$  :

$$\text{reachability-distance}_{\epsilon, MinPts}(o, p) = \begin{cases} \text{Indéfini} & \text{si } |V_{\epsilon}(p)| < MinPts \\ \max(\text{core-distance}_{\epsilon, MinPts}(p), \text{distance}(p, o)) & \text{sinon} \end{cases} \quad (2)$$

- La core-distance et la reachability-distance sont indéfinis si le groupe de points n'est pas suffisamment dense.



- La **reachability-distance** du point  $p$  à un autre point  $o$  est le max entre la distance entre  $o$  et  $p$ , et la core-distance de  $p$  :

$$\text{reachability-distance}_{\epsilon, MinPts}(o, p) = \begin{cases} \text{Indéfini} & \text{si } |V_{\epsilon}(p)| < MinPts \\ \max(\text{core-distance}_{\epsilon, MinPts}(p), \text{distance}(p, o)) & \text{sinon} \end{cases} \quad (1)$$

- La core-distance et la reachability-distance sont indéfinis si le groupe de points n'est pas suffisamment dense.

- OPTICS ne produit pas directement des clusters
- il renvoie un *ordre de clustering*
  - tous les individus sont dans une liste ordonné
  - cette liste représente la structure de classification basée sur la densité
- les individus dans un cluster plus dense sont listés plus proches les uns des autres
- Le seuil  $\epsilon$  n'est pas obligatoire
- Cet ordre sélectionne un individu qui est atteignable avec la plus faible valeur ( $\epsilon$ ) de telle manière que les clusters avec des hautes densités seront identifiés avant les autres
- pour chaque individu, on stocke la core-distance, et une reachability-distance
- On maintient une liste ordonnée des individus :
  - triée par la reachability-distance de l'individu-centre le plus proche
  - cad par la plus petite reachability-distance de chaque objet

- commencer avec un individu arbitraire  $p$
- Calculer  $V_\epsilon(p)$  et  $\text{core-distance}(p)$ ,  $\text{reachability-distance}(p) = \text{UNDEFINED}$
- sortir l'individu  $p$
- si  $p$  n'est pas un individu-centre :  
passer à l'individu d'après (soit dans Seeds, soit dans la BD)
- si  $p$  est un individu-centre :
  - pour chaque objet  $q$  de  $V_\epsilon(p)$  :
    - mettre à jour  $\text{reachability-distance}(q)$
    - insérer  $q$  dans Seeds si  $q$  n'a pas été traité
- L'itération continue tant que il reste des éléments dans BD et que Seeds n'est pas vide.

```

OPTICS(setOfObjects, e, MinPts, OrderedFile)
    OrderedFile.open()
    For i from 1 to SetOfObjects.size() Do:
        Obj = SetOfObjects.get(i)
        if not obj.processed then
            ExpandClusterOrder(SetOfObjects,
                               obj, e, MinPts, OrderedFile)
    OrderedFile.close()
    
```

```

ExpandClusterOrder(SetOfObjects, obj, e, MinPts, OrderedFile)
    neighbors = SetOfObjects.neighbors(obj, e)
    obj.processed = TRUE
    obj.reachDist = UNDEFINED
    obj.setCoreDistance(neighbors, e, MinPts)
    OrderedFile.write(obj)
    if obj.coreDist <> UNDEFINED then (obj est un "centre")
        Seeds.update(neighbors, obj)
        While not Seeds.empty() do:
            currentObj = Seeds.next()
            neighbors = SetOfObjects.neighbors(currentObj, e)
            currentObj.processed = TRUE
            currentObj.setCoreDistance(neighbors, e, MinPts)
            OrderedFile.write(currentObj)
            if currentObj.coreDist <> UNDEFINED then
                Seeds.update(neighbors, currentObj)
    
```

Obj est implement écrit dans le fichier OrderedFile avec ses core-distance et reachability-distance

AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

GMM

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

SVM

NN

Evaluation

Data Mining

Project

```

Seeds::update(neighbors, centerObj):
  c_dist = centerObj.coreDist
  for all obj from neighbors Do:
    if not obj.processed then:
      new_r_dist = max(c_dist, centerObj.dist(obj))
      if obj.reachDist = UNDEFINED then:
        obj.reachDist = new_r_dist
        insert(obj, new_r_dist)
      else: // obj est déjà dans Seeds
        if new_r_dist < obj.reachDist then
          obj.reachDist = new_r_dist
          decrease(obj, new_r_dist)
    
```

AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

GMM

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

SVM

NN

Evaluation

Data Mining

Project

- précondition  $\epsilon' \leq \epsilon$  (utilisé avant)
- on n'a pas encore trouvé de cluster, donc au début, ClusterID = NOISE
- Pour tout Obj de la liste triée (pris dans l'ordre) :
  - Si  $\text{Obj.reachDist} > \epsilon'$  : // UNDEF >  $\epsilon$ 
    - Si  $\text{Obj.coreDist} \leq \epsilon'$  :
      - ClusterID = nextID(ClusterID) // un cluster trouvé
      - Obj.clusterID = ClusterID
    - Sinon
      - Obj.clusterID = NOISE
  - Sinon
    - Obj.clusterID = ClusterID

AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

GMM

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

SVM

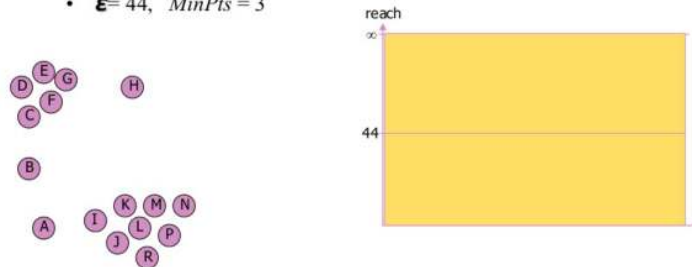
NN

Evaluation

Data Mining

Project

- Example Database (2-dimensional, 16 points)
- $\epsilon = 44$ ,  $\text{MinPts} = 3$



seedlist:

AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

GMM

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

SVM

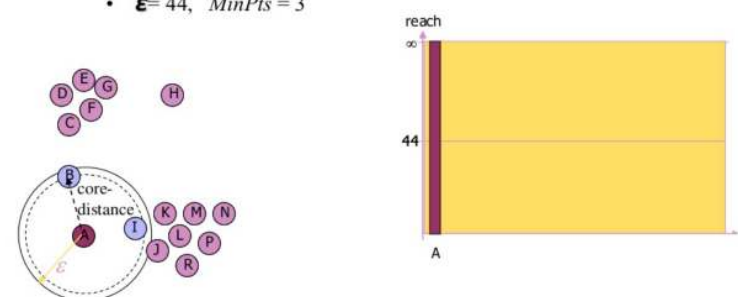
NN

Evaluation

Data Mining

Project

- Example Database (2-dimensional, 16 points)
- $\epsilon = 44$ ,  $\text{MinPts} = 3$



seedlist: (B,40) (I, 40)

AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

GMM

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

SVM

NN

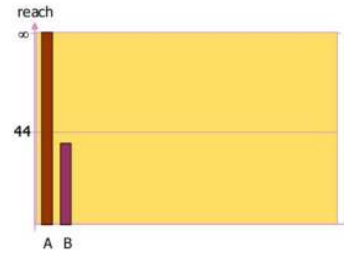
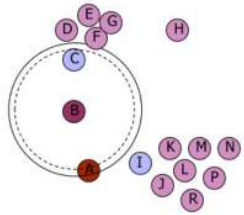
Evaluation

Data Mining

Project

• Example Database (2-dimensional, 16 points)

•  $\epsilon = 44$ ,  $MinPts = 3$



seedlist: (I, 40) (C, 40)

AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

GMM

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

SVM

NN

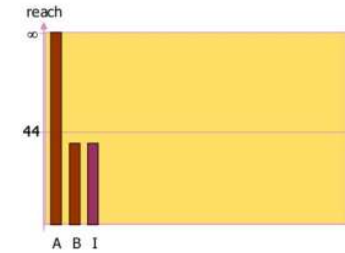
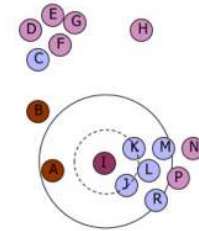
Evaluation

Data Mining

Project

• Example Database (2-dimensional, 16 points)

•  $\epsilon = 44$ ,  $MinPts = 3$



seedlist: (J, 20) (K, 20) (L, 31) (C, 40) (M, 40) (R, 43)

AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

GMM

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

SVM

NN

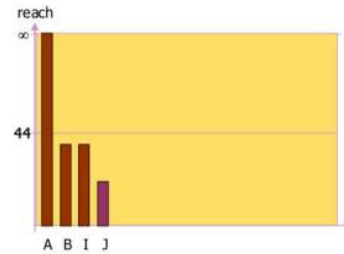
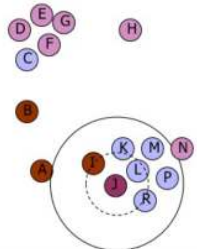
Evaluation

Data Mining

Project

• Example Database (2-dimensional, 16 points)

•  $\epsilon = 44$ ,  $MinPts = 3$



seedlist: (L, 19) (K, 20) (R, 21) (M, 30) (P, 31) (C, 40)

AI for bio

J-P Comet

Introduction

Trees

Clustering

Objectifs

Partitionnement

k-means

GMM

Hierarchiques

Algorithmes-densité

DBSCAN + OPTICS

Résumé

kNN

Bayes

SVM

NN

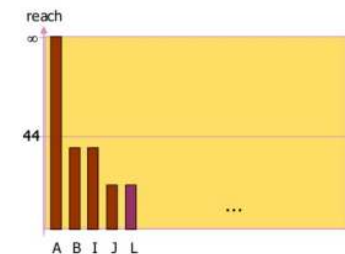
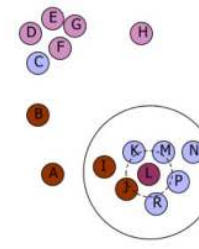
Evaluation

Data Mining

Project

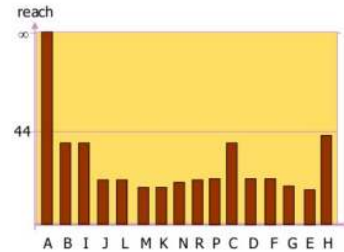
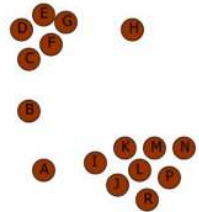
• Example Database (2-dimensional, 16 points)

•  $\epsilon = 44$ ,  $MinPts = 3$



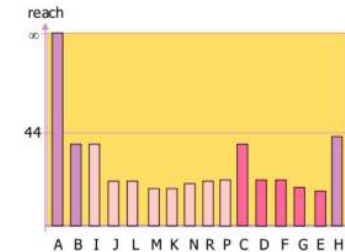
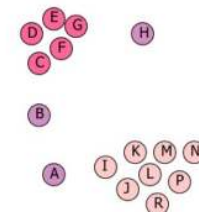
seedlist: (M, 18) (K, 18) (R, 20) (P, 21) (N, 35) (C, 40)

- Example Database (2-dimensional, 16 points)
- $\epsilon = 44$ ,  $MinPts = 3$

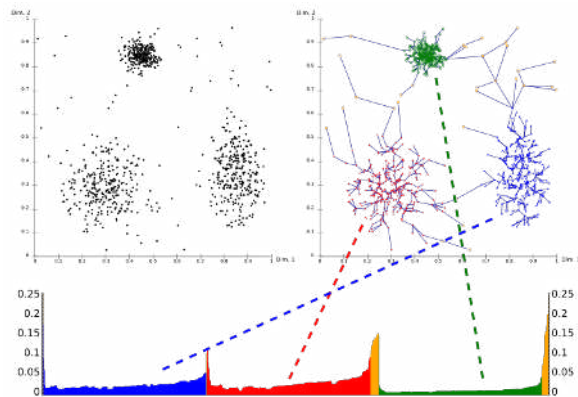


seedlist: -

- Example Database (2-dimensional, 16 points)
- $\epsilon = 44$ ,  $MinPts = 3$



seedlist: -



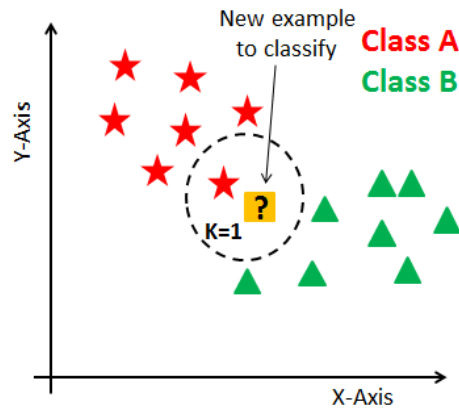
La sortie de l'algorithme : liste ordonnée de point associés à leur plus petite reachability-distance.

⇒ diagramme appelé reachability-plot :

- axe x : position d'un point dans la liste ordonnée
- axe y la reachability-distance associée

Les points d'un même cluster ont une reachability-distance assez basse, les vallées du diagramme représentent donc les différents clusters du jeu de données. Plus les vallées sont profondes, plus les clusters sont denses.

- Le clustering groupe des objets en se basant sur leurs **similarités**.
- Le clustering possède plusieurs applications.
- La mesure de similarité peut être calculée pour **différents types** de données.
- La sélection de la **mesure de similarité** dépend des données utilisées et le type de similarité recherchée.
- Les méthodes de clustering peuvent être classées en :
  - Méthodes de partitionnement,
  - Méthodes hiérarchiques,
  - Méthodes à densité de voisinage
  - Méthodes par grilles
  - Méthodes mixtes, ...
- Plusieurs travaux de recherche sur le clustering en cours...
  - Passage à l'échelle : échantillonnage, optimisation, indexation, mixages, ...
  - Prise en compte de contraintes : connaissances utilisateur ou données / objectifs utilisateur
- Applications en perspective : Génomique, Environnement, ... Tout...
- Évaluation des clusters : Que signifie un "bon" cluster ?
  - intra-clusters : haut degré de similarité
  - Utilisation de la variance<sup>1</sup> ⇒ les clusters avec la variance la plus petite sont bons.
  - La taille du cluster est aussi importante, donc une approche alternative est d'utiliser la variance pondérée<sup>2</sup>



- Principe intuitif de l'algorithme des plus proches voisins (*k* nearest neighbors) :
  - on stocke les exemples tels quels dans une table ;
  - pour prédire la classe d'une donnée, on détermine les exemples qui en sont le plus proche ;
  - de ces exemples, on déduit la classe ou on estime l'attribut manquant de la donnée considérée.
- Mesure de la dissimilarité entre deux données distantes. Mathématiquement parlant, une **distance** *d* est une application définie par :

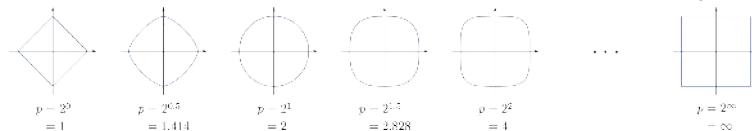
$$d : D \times D \rightarrow \mathbb{R}^+ \\ (x, y) \rightarrow d(x, y)$$

qui respecte les 3 propriétés suivantes  $\forall (x, y, z) \in D^3$  :

- non dégénérée :  $d(x, y) = 0 \Leftrightarrow x = y$ ,
  - symétrique :  $d(x, y) = d(y, x)$ ,
  - inégalité triangulaire :  $d(x, z) \leq d(x, y) + d(y, z)$ .
- Un espace de points muni d'une distance est un **espace métrique**.

- Les distances les plus classiques sont issues des normes de Minkowski. Elles concernent des données dont les attributs sont quantitatifs. Les plus classiques sont :

- distance euclidienne :  $\delta(x_i, x_j) = \sqrt{\sum_{k=1}^D (x_{i,k} - x_{j,k})^2}$   
sensible aux valeurs éloignées de la moyenne / continument dérivable.
- distance de Manhattan :  $\delta(x_i, x_j) = \sum_{k=1}^D |x_{i,k} - x_{j,k}|$   
Beaucoup moins sensible aux valeurs éloignées de la moyenne / pas dérivable en 0 ;
- la distance maximale :  $\delta(x_i, x_j) = \max_{k \in \{1, \dots, D\}} |x_{i,k} - x_{j,k}|$ .  
Cette mesure est simplement le plus grand écart entre  $x_i$  et  $x_j$ .



- Dissimilarité.**
  - Utiliser une distance n'est pas si simple que cela imposer l'inégalité triangulaire est souvent problématique.
  - On se contente alors d'une dissimilarité : application qui associe un réel positif à un couple de données et qui vérifie les deux propriétés de non dégénérescence et symétrie.
  - Ces deux propriétés sont assez faciles à respecter, même pour des attributs non quantitatifs.

- Attribut numérique.**
  - la dissimilarité introduite plus haut s'applique immédiatement.
  - PB si l'ordre de grandeur des attributs à combiner n'est pas le même pour tous : les attributs d'ordre de grandeur les plus grands vont dominer.  
⇒ mettre à l'échelle les attributs. Si l'attribut  $a_i$  prend sa valeur dans  $[\min(a_i), \max(a_i)]$ , on utilise l'attribut normalisé :

$$\hat{a}_i = \frac{a_i - \min(a_i)}{\max(a_i) - \min(a_i)} \in [0, 1]$$

- lorsque certains attributs sont plus importants que d'autres, on peut donc pondérer les attributs en fonction de leur importance.
- Attribut nominal et attribut ordinal.**
  - La dissimilarité est généralement définie par :
    - =0 si la valeur de l'attribut est la même,
    - =1 sinon.
  - cette définition n'est pas adaptée pour les attributs ordinaux.  
Ex : l'attribut "couleur"  $\in \{\text{rouge, jaune, rose, vert, bleu, turquoise}\}$ . la dissimilarité entre "bleu" et "rouge" est plus grande qu'entre "rouge" et "jaune".
- Valeur d'attribut manquante.**  
Si une valeur d'attribut est manquante dans une donnée, on considère généralement que la dissimilarité pour cet attribut est maximale.

**Nécessite:** 3 parameters : a set of exemples  $X$ , a new data  $x \in D$  and  $k \in \{1, \dots, N\}$

**Résultat :** the predicted class of  $x$

**for** each example  $x_i \in X$  **do**

  | compute the dissimilarity between  $x_i$  and  $x$  :  $\delta(x_i, x)$

**end**

**for**  $\kappa \in \{1, \dots, k\}$  **do**

  |  $kNN[\kappa] \leftarrow \operatorname{argmin}_{i \in \{1, \dots, N\}} \delta(x_i, x)$   
  |  $\delta(x_i, x) \leftarrow +\infty$

**end**

determine the class of  $x$  from the classes of exemples whose number is stocked in table  $kNN$

### Algorithm 1:

Prediction of the class associated to a a new data by the method «  $k$  Nearest Neighbours »

**Plusieurs stratégies sont envisageables.** On suppose un problème de classification binaire et on définit  $\text{classe}(x) = +1$  pour une donnée  $x$  positive,  $\text{classe}(x) = -1$  si elle est négative.

- la classe majoritaire parmi les  $k$  plus proches voisins, soit :

$$\text{classe}(x) \leftarrow \operatorname{sgn} \sum_{\kappa=1}^{\kappa=k} \text{classe}(kNN[\kappa]) \quad \text{avec} \quad \operatorname{sgn}(x) = \begin{cases} +1 & \text{si } x \leq 0 \\ -1 & \text{si } x < 0 \end{cases}$$

la somme est de signe positif s'il y a plus de positifs parmi les  $k$  plus proches voisins que de négatifs, et réciproquement.

- la classe majoritaire parmi les  $k$  plus proches en pondérant la contribution de chacun par l'inverse de sa dissimilarité à  $x$ , soit :

$$\text{classe}(x) \leftarrow \operatorname{sgn} \sum_{\kappa=1}^{\kappa=k} f(\delta(x_{kNN[\kappa]}, x)) \times \text{classe}(kNN[\kappa]).$$

$f$  peut être choisie de différentes manières :

- normale :  $f(\delta) = e^{-\delta}$ ;
- inverse :  $f(\delta) = \frac{1}{\delta}$ ;
- ...

Comme toujours en IA, toute autre solution est envisageable, à partir du moment où elle est cohérente avec les données et la définition de leurs attributs.

- Il n'y a pas de méthode particulière sinon qu'il faut choisir le  $k$  qui va bien pour les données que l'on a !
- on peut aussi prendre  $k = N$  dans le cas où l'influence de chaque exemple est pondérée par sa dissimilarité avec la donnée à classer.
- Prendre  $k$  petit (1, 2 ou 3 par exemple) est aussi une bonne stratégie dans la pratique, beaucoup moins coûteuse en temps de calcul quand le nombre de données est grand.

**Mesurer la similarité.** on peut mesurer leur similarité et inverser le raisonnement par rapport à une dissimilarité (en particulier, dans l'algorithme, il suffit de remplacer l'argmin par un argmax).

- Notons  $\bar{\delta}$  une fonction de similarité. En général, celle-ci doit respecter :
  - $\bar{\delta}(x, x)$  est maximal, voire infini ;
  - $\bar{\delta}(x, y) = \bar{\delta}(y, x), \forall (x, y)$
- Une fonction populaire respectant ces deux propriétés est la fonction normale :

$$\bar{\delta}(x, y) = e^{-\frac{\delta(x, y)}{2\sigma^2}}$$

où  $\delta(x, y)$  est une certaine distance ou dissimilarité entre  $x \in D$  et  $y \in D$ .  
 $\sigma$  permet de régler le rayon d'influence de chaque donnée : plus  $\sigma$  est grand, plus ce rayon est grand.  
 Cette fonction porte aussi le nom de "fonction à base radiale".