

# Classification supervisée et clustering

Mihaela JUGANARU  
[juganaru@emse.fr](mailto:juganaru@emse.fr)

École Nationale Supérieure des Mines de St Etienne

2025-2026



## Bibliographie :

- Oded Maimon · Lior Rokach (Editors) - *Data Mining and Knowledge Discovery Handbook*, Second Edition, Springer 2010
- Jiawei Han · Micheline Kamber - *Data Mining : Concepts and Techniques*, Morgan Kaufmann Publication, 2006

# Plan

## 1 Clustering - définitions, généralités et algorithmes

- Définitions
- Fonctions de distance, similarité
- Caractérisation des solutions de clustering
- Algorithmes de  $K$ -moyennes et de  $K$ -médoïdes
- Méthodes hiérarchiques de clustering
- DBSCAN
- Méthodes mixtes et autres méthodes

## 2 Classification supervisée - validation

- Définitions
- Les 2 étapes du processus de classification
- Qualité d'un classifieur
- Méthode KNN
- Extensions k-NN

## Objectif

Départ : un ensemble de  $N$  observations réelles :

$$\mathcal{D} = \{x_i \in \mathbb{R}^d | i = 1, N\}$$

Une observation (tuple) est composée de  $d$  attributs.

Objectif du **clustering** (data segmentation ou **classification non-supervisée**) : trouver un ensemble de catégories ou **clusters** de  $\mathcal{D}$  :  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k$  auxquelles les données de  $\mathcal{D}$  appartiennent.

Les données d'un cluster doivent être les plus semblables / **similaires** possible.

Les clusters se doivent être homogènes.

## Objectif

Formellement :  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k \subset \mathcal{D}$  avec  $k \ll N$  et

$$\mathcal{C}_1 \cup \mathcal{C}_2 \cup \dots \cup \mathcal{C}_k = \mathcal{D}$$

Clustering dur (hard clustering) :

- $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ , pour tout  $i, j$

"Une observation appartient à un seul cluster."

Clustering flou (soft clustering) :

- $x_i \in \mathcal{C}_j$  avec une probabilité  $p_{ij}$  et :

$$\sum_{j=1}^k p_{ij} = 1, \text{ pour tout } i = 1, N$$

"Une observation appartient à plusieurs clusters."

# Classification versus clustering

Classe  $\equiv$  cluster

Dans la tâche DM (Data Mining) de classification les classes sont connues à l'avance, mais pas pour le clustering.

Classification : but prédictif, apprentissage supervisé.

Clustering : but descriptif, apprentissage non-supervisé.

## Caractéristiques possibles d'un procédé de clustering

- supporte le passage à l'échelle (attention : le clustering appliqué à un échantillon de  $\mathcal{D}$  peut fournir des résultats biaisés)
- supporte des attributs hétérogènes (rarement une observation a des attributs homogènes dpvd signification et type)
- découvrir des clusters avec des frontières arbitraires
- avoir un minimum d'information (paramètres) : nombre de clusters
- capacité à détecter le bruit
- travaille de manière incrémentale
- supporte un grand nombre de dimension ( $d$  - nombre d'attributs de l'espace d'observations)

## Fonction distance

A partir de la matrice qui caractérise  $\mathcal{D}$  ( $N \times d$ ) avec la représentation on peut calculer la matrice de distance entre les observations :

$$\begin{pmatrix} 0 & & & & \\ dist(2, 1) & 0 & & & \\ dist(3, 1) & dist(3, 2) & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ dist(N, 1) & dist(N, 2) & \dots & \dots & 0 \end{pmatrix}$$

$dist(i, j) = distance(x_i, x_j)$ , distance ou dissimilarité.

## Fonction distance

Une fonction  $dist$  distance métrique doit satisfaire :

- $dist(x, x) = 0$  pour tout  $x$
- $dist(x, y) = dist(y, x)$ , pour tout  $x, y$  (symétrie)
- $dist(x, y) \leq dist(x, z) + dist(z, y)$  (inégalité triangulaire)
- si  $dist(x, y) = 0$ , alors  $x = y$  (définitivité)

Une fonction de **dissimilitude** est une fonction distance métrique qui satisfait aussi :

- $dist(x, y) \leq max(dist(x, z), dist(z, y))$  (inégalité ultramétrique)
- si  $dist(x, y) = 0$ , alors  $dist(z, x) = dist(z, y)$  (régularité)

Plus la dissimilitude est grande, plus les points sont éloignés.  
La **similarité** est l'inverse de la dissimilitude.

## Distances entre observations numériques

Soient  $x, y \in \mathbb{R}^d$ ,  $x = (x_1, x_2, \dots, x_d)$ ,  $y = (y_1, y_2, \dots, y_d)$ .

- distance de Minkowski :

$$dist(x, y) = \left( \sum_{j=1}^d |x_j - y_j|^g \right)^{1/g}$$

- distance pondérée par  $w_j, j = 1, d$  :

$$dist(x, y) = \left( \sum_{j=1}^d w_j \times |x_j - y_j|^g \right)^{1/g}$$

## Distances

La distance de Minkowski devient :

- pour  $g = 1$  - distance de Manhattan :

$$dist(x, y) = \sum_{j=1}^d |x_j - y_j|$$

- pour  $g = 2$  - distance euclidienne :

$$dist(x, y) = \sqrt{\sum_{j=1}^d |x_j - y_j|^2}$$

- pour  $g \rightarrow \infty$  distance de Tchebychev :

$$dist(x, y) = \max_j |x_j - y_j|$$

## Distance entre observations discrètes (binaires ou non)

- distance de Hamming :

$$dist(x, y) = \text{card}(\{j | x_j \neq y_j\})$$

- distance de Levenshtein ou distance d'édition : le nombre minimum de pas pour transformer  $x$  en  $y$

## Distances entre observations binaires

Une observation binaire est formée de **variables binaires** (0 ou 1).

Une variable binaire est symétrique, si les deux valeurs ont une signification équivalente (*exemple fille/garçon dans des observations médicales*).

Il convient souvent de prendre en compte les variables asymétriques.

## Distances entre observations binaires

Une mesure de dissimilarité :

$$dist(x, y) = \frac{r + s}{q + t + r + s}$$

avec  $r$  et  $s$  le nombre de positions avec des valeurs différentes (0-1 et 1-0) et  $t$  et  $q$  le nombre de position avec la même valeur 0 ou 1

Si on ignore  $t$  (variable asymétrique), on peut prendre la distance de Jaccard :

$$dist(x, y) = \frac{r + s}{q + r + s}$$

## Distances entre attributs nominaux

- matching simple :

$$dist(x, y) = \frac{d - m}{d}$$

$m$  le nombre d'attributs à valeurs égale

- transformation en variables binaires en associant pour chaque attribut nominal avec  $v$  valeurs  $v$  attributs binaires

## Distances entre attributs mixtes

On calcule une pondération des distances entre les attributs (ou groupe d'attributs) de même type :

$$dist(x, y) = \frac{\sum_{j=1}^d \delta^{(j)} d^{(j)}(x, y)}{\sum_{j=1}^d \delta^{(j)}}$$

avec  $\delta^{(j)}$  un coefficient de pondération de l'attribut  $j$ .

## Indices de similarité

- cosinus :

$$\cos(x, y) = \frac{x \times y}{|x||y|}$$

- indice de Tanimoto (pour attributs binaires)

$$T(x, y) = \frac{x \times y}{|x|^2 + |y|^2 - x \times y}$$

$x \times y$  est le produit scalaire

- coefficient de Dice :

$$s(x, y) = \frac{2x \times y}{|x|^2 + |y|^2}$$

## Texte

Si les observations à examiner sont de type texte :

- prétraitement
- calcul des fréquences de mots dans chaque observation (chaque document) et dans la totalité des documents (corpus)
- sur la base des fréquences calculées construire un tableau de valeurs pour chaque document
- prendre des fonctions distance dédiées comme le cosinus, le produit scalaire ou Jaccard

## Texte (2)

Prétraitement :

- séparer en unités (mots)
- corriger les fautes d'orthographe ou remplacer les sigles
- filtrer : éliminer les mots vides (très fréquents et / ou non significatifs) ou ne garder que les mots pré-listés
- lemmatisation : mettre les mots à leur forme invariable
- autres traitements quant à leur classe grammaticale, un dictionnaire, etc.

Considérer l'ensemble des mots présents dans le corpus (termes) :  
 $t_1, t_2, \dots, t_m$

## Texte (3)

Calculer les fréquences des termes pour chaque document et dans le corpus :

- fréquence d'un terme dans un document  $d$  :  
 $tf(t, d)$  le nombre d'apparitions de  $t$  dans  $d$

Le plus souvent on normalise cette valeur par le max :

$$\frac{tf(t, d)}{\max_{d' \text{-document}} df(t, d')}$$

- fréquence d'un terme dans le corpus :

$$cf(t) = \sum_{d \text{ document}} tf(t, d)$$

- fréquence des documents par rapport à un terme :

$$df(t) = \text{card}(\{d\text{-document} | d \text{ contient } t\})$$

## Texte (4)

Pour représenter un document, il faut considérer pour chaque document soit le tableau

$$tf(d) = (tf(t_1, d), tf(t_2, d) \dots tf(t_m, d)))$$

soit le tableau

$$tf.idf(d) = tf(d) \times \log \frac{D}{df(t)}$$

(produit scalaire)

D est le nombre total des documents.

## Texte (5)

Remarque 1 : la matrice  $tf$  est (très) creuse.

Remarque 2 : la matrice  $tf$  n'est jamais une matrice carrée et  $m = K \times n^\alpha$ , loi de Heaps sur la taille du vocabulaire ( $K$  et  $\alpha$  constantes)

Remarque 3 : les fréquences des mots ne sont pas comparables : la fréquence d'un mot dans un corpus est inversement proportionnelle à son rang des fréquences, loi de Zipf (après avoir analysé "Ulysse" de James Joyce)

Remarque 4 : assez peu de lois de probabilité modélisent le nombre d'occurrences d'un terme dans un document (loi de Poisson, distribution de Dirichlet)

## Un clustering, deux clusterings ...

Si on fournit des clusters  $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k$  qui forment une partition de  $\mathcal{D}$ , on se pose la question de caractériser la solution autrement et aussi de donner des critères de choix entre deux solutions de clustering.

## Métriques de qualité du cluster

On peut calculer facilement pour un cluster  $\mathcal{C}_k$  :

- son centre :

$$\mu_k = \frac{\sum_{x_i \in \mathcal{C}_k} x_i}{N_k}$$

avec  $N_k = \text{card}(\mathcal{C}_k)$

- son diamètre :

$$\text{diam}(\mathcal{C}_k) = \max_{x, y \in \mathcal{C}_k} \text{dist}(x, y)$$

- inertie :

$$J_k = \sum_{x \in \mathcal{C}_k} \text{dist}(x, \mu_k)^2$$

## Métriques de qualité du inter-cluster

- distance inter-cluster :

$$dist(\mathcal{C}, \mathcal{C}') = dist(\mu(\mathcal{C}), \mu(\mathcal{C}'))$$

- distance entre les éléments :

$$dist(\mathcal{C}, \mathcal{C}') = \min_{x \in \mathcal{C}, y \in \mathcal{C}'} dist(x, y)$$

- intertie inter-cluster

$$JB = \sum_k N_k dist(\mu, \mu_k)^2$$

avec  $\mu$  le centre de gravité de  $\mathcal{D}$

- indice de Dunn, de Silhouette,  $\tau$  ...

## Classes d'algorithmes de clustering

- algorithmes de partitionnement
- algorithmes hiérarchiques (ascendants ou descendants)
- méthodes basées sur les techniques de grille
- méthodes basées sur une connaissance du modèle

## Clustering par partitionnement

Si on cherche un clustering fort avec  $K$  clusters, cela signifie trouver une partition de  $\mathcal{D}$ . Il est impossible de construire toutes les partitions et de choisir ensuite la meilleure.

Problème NP-difficile - espace exponentiel.

Idée : générer un partitionnement (pas trop mauvais) et l'affiner jusqu'à obtenir une certaine stabilité.  
(algorithme connu aussi comme "partitionnement par centres mobiles")

# Algorithme de $K$ -moyennes

## Algorithme de $K$ -moyennes

Idée : Hugo Steinhaus 1957, algorithme : James MacQueen 1967

L'algorithme de  $K$ -moyennes ( *$K$ -means*) : on cherche un clustering fort avec  $K$  clusters avec des centres de clusters mouvants. On s'arrête quand le partitionnement devient stable.

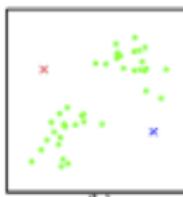
- pas 0 : choix aléatoire de  $K$  centres parmi les  $N$  éléments de  $\mathcal{D}$
- pas 1 : si on connaît le  $\mu_k$ ,  $k = 1, K$ , on affecte un élément  $x_i$  au cluster  $(C)_k$  le plus proche
- pas 2 : on calcule pour chaque  $(C)_k$  son nouveau centre  $\mu_k$
- on alterne pas 1 et pas 2 jusqu'à la convergence

## Exemple $K=2$

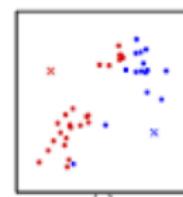
Données



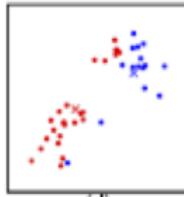
Initialisation



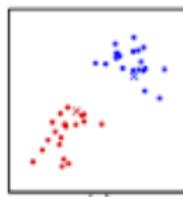
Itération 1



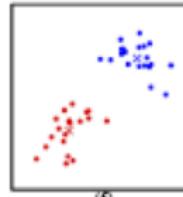
Itération 2



Itération 3



Itération 4



# Algorithme $K$ -moyennes

Avantages :

- efficace : complexité en  $\mathcal{O}(KNt)$  –  $K$  : nb clusters,  $N$  : nb objets,  $t$  : nb itérations – en général  $K \ll t \ll N$
- interprétation aisée des résultats
- le centroid (centre de gravité) caractérise le cluster

Désavantage :

- nécessité de fixer  $K$
- sensible au bruit et données hors-rang (outliers) → besoin éventuel de normalisation et de centrage
- clusters convexes uniquement
- valeurs numériques

Fixer  $k$  le nombre de clusters :

- $k$  connu (par les experts et donnée dans la définition du problème)
- méthode du coude : on représente sur un graphique les inerties intra-cluster (mieux - la moyenne des inerties pour 10 à 20 exécutions) et on trouve un point de fléchissement ( $W$  ou  $\log W$ )

Fixer  $k$  le nombre de clusters (suite) :

- méthode de Tibshirani (2001) : usage d'une courbe des statistiques de Gap

$$G(k) = \log W_k - E(k)$$

- $W_k$  la variance attendue pour un partage en  $k$  clusters d'une distribution uniforme
- $s_k = E(k)$  la variance observée pour l'inertie inter-cluster

La valeur optimale pour  $k$  :

$$k^* = \min\{k \mid G(k) \geq G(k+1) - s'_k\}$$

avec  $s'_k = s_k \sqrt{1 + 1/\alpha}$  une correction de  $s_k$  pour les  $\alpha$  exécutions

Version quant au choix des centres initiaux de cluster :

- choix aléatoire de  $k$  éléments parmi les  $k$  observations (version de Lloyd)
- choix aléatoire d'un cluster pour chaque élément et calcul des moyennes
- choix aléatoire des points quelconque dans l'espace de valeurs possibles
- version  $k++$  (2007 , D. Arthur et S. Vassilvitskii) : choix d'un premier centre, pour tout observation on calcule  $D(x)$  la distance au plus proche centre, on choisi un nouveau centre avec une probabilité proportionnelle à  $D(x)$

# Algorithme K-médoïdes

## PAM - Partitioning Around Medoids

Proposé par Kaufmann and Rousseeuw (1987)

- permet de traiter des valeurs non-numériques
- un cluster est représenté par un objet lui appartenant : un médoïde - qui a la propriété d'être à distance minimale des autres objets
- principe : remplacement itératif des médoïdes par un objet non-médoïde

## Algorithme $K$ -médoïdes

- pas 0 : choix aléatoire de  $K$  objets de  $\mathcal{D}$  comme médoïdes initiaux
- pas 1 : assigner chaque objet restant de  $\mathcal{D}$  au médoïde le plus proche
- pas 2 : choix aléatoire d'un objet  $x \in \mathcal{D}$  qui n'est pas médoïde
- pas 3 : pour chaque cluster calculer le coût de remplacement du médoïde de  $\mathcal{C}_k$  par  $x$  ( $k$  varie entre 1 et  $K$ )
- pas 4 : si le coût est meilleur : remplacer ce médoïde par  $x$
- on reprend pas 1 jusqu'au pas 4 jusqu'à la convergence

# Algorithme $K$ -médoides

Avantages :

- moins sensible au bruit (usage de la médiane au lieu de la moyenne)
- cluster représenté par un objet

Désavantage :

- temps plus long (pas 2 et 3 en plus)

## Méthodes hiérarchique de clustering

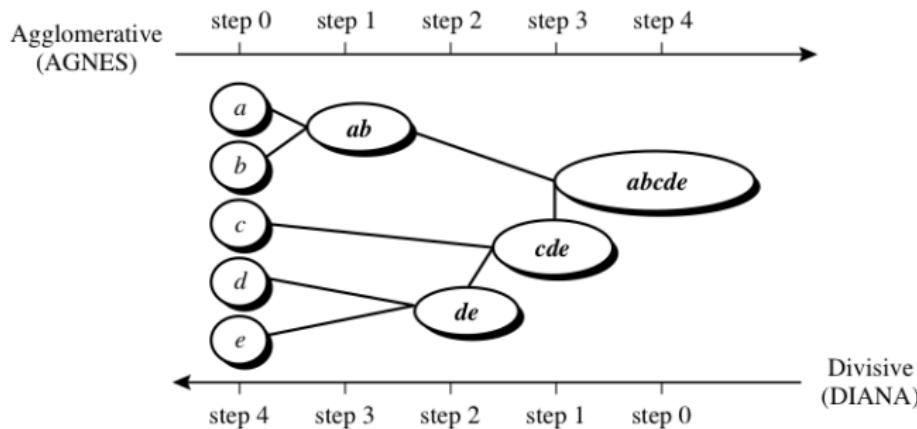
Deux classes de clustering hiérarchique :

- agglomerative (bottom - up)
  - chaque objet forme un cluster
  - on fusionne les clusters en clusters plus grands
  - jusqu'à obtenir un seul
- divisive (top-down)
  - on commence par considérer un unique cluster qui contient tous les objets
  - on divise en clusters plus petits
  - instable, difficilement applicable

## Méthodes hiérarchiques de clustering

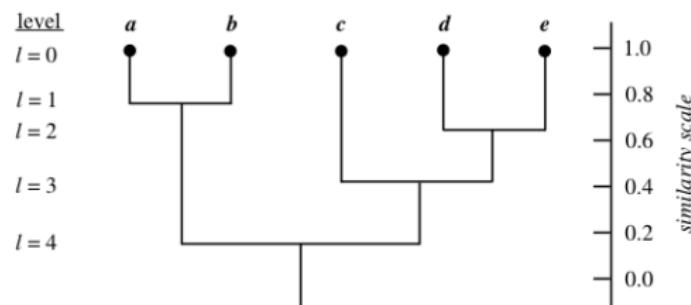
AGNES (AGglomerative NESting)

DIANA (DIvisive ANAlysis)



## Méthodes hiérarchique de clustering

On représente le résultat par un dendrogramme :



Pour un clustering avec  $K$  clusters on coupe à une certaine hauteur.

## Algorithme hiérarchique ascendant

- initialisation  $C_i \leftarrow x_i, i = 1, N$   
 $N$  clusters initiaux, formé chacun d'un seul élément
- répéter  $N - 1$  fois
- $\rightarrow$  parmi toutes les paires de clusters, choisir celle qui satisfait mieux un critère

## Méthodes hiérarchique de clustering

La fusion/division de cluster se décide selon des **stratégies** :

- distance minimale : saut minimal (single linkage)

$$d_{min}(C_j, C_k) = \min_{x \in C_j, x' \in C_k} dist(x, x')$$

- distance maximale : saut maximal (complete linkage)

$$d_{max}(C_j, C_k) = \max_{x \in C_j, x' \in C_k} dist(x, x')$$

- distance entre les barycentres :

$$d_G(C_j, C_k) = dist(centre(C_j), centre(C_k))$$

- distance moyenne :

$$d_{avg}(C_j, C_k) = \frac{1}{card(C_j)card(C_k)} \sum_{x \in C_j, x' \in C_k} dist(x, x')$$

## Méthode Ward pour clustering hiérarchique

Famille d'algorithmes de Lance–Williams :

Soient  $C_i$ ,  $C_j$  et  $C_k$  clusters qui existent et on envisage de fusionner  $C_i \cup C_j$ .

On note  $dist_{ij}$ ,  $dist_{jk}$  et  $dist_{ik}$  les distances entre les clusters et  $dist_{(ij)k}$  une distance qui permet de voir le bénéfice de former le cluster  $C_i \cup C_j$ .

$$dist_{(ij)k} = \alpha_i dist_{ik} + \alpha_j dist_{jk} + \beta dist_{ij} + \gamma |dist_{ik} - dist_{jk}|$$

avec  $\alpha_i$ ,  $\alpha_j$ ,  $\beta$ ,  $\gamma$  constantes ou dépendantes de la taille des clusters.

## Méthode Ward pour clustering hiérarchique (2)

- la **méthode de Ward** est un cas particulier de Lance-Williams :

$$\alpha_I = \frac{n_I + n_k}{n_i + n_j + n_k}, \beta = -\frac{n_k}{n_i + n_j + n_k} \text{ et } \gamma = 0$$

Pour une paire candidate on calcule toutes les distances par rapport aux autres clusters :

$$d_{(ij)k} = \frac{n_i + n_k}{n_i + n_j + n_k} d_{ik} + \frac{n_j + n_k}{n_i + n_j + n_k} d_{jk} - \frac{n_k}{n_i + n_j + n_k} d_{ij}$$

méthode Ward : une méthode des plus utilisées, car elle vise réduire l'inertie inter-cluster.

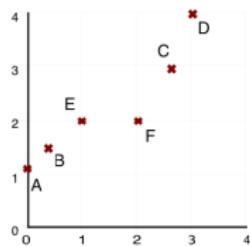
- On se résume parfois à minimiser uniquement un facteur calculé sur la base de clusters  $C_i$  et  $C_j$  et de leurs barycentres  $\mu_i$ ,  $\mu_j$  :

$$\delta(C_i, C_j) = \frac{n_i n_j}{n_i + n_j} d^2(\mu_i, \mu_j)$$

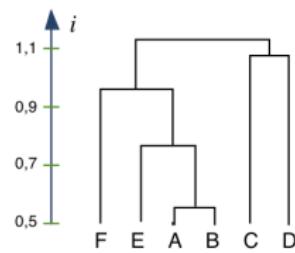
## Méthodes hiérarchique de clustering

Selon la stratégie utilisée le résultat est différent :

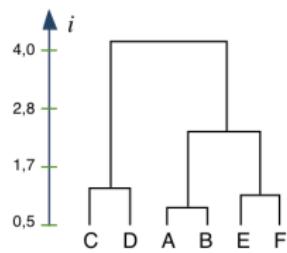
Données (métrique : dist. Eucl.)



Saut minimal



Saut maximal



## Méthodes hiérarchique de clustering

Avantage :

- pas besoin de fixer  $K$
- peut s'appliquer (certaines stratégies) à des données non-numériques

Désavantages :

- mauvaise complexité : au moins  $O(N^2)$
- aucune remise en cause d'une solution partielle

Il y a beaucoup d'algorithmes de clustering hiérarchique : BIRCH (1996), CURE (1998), CHAMELEON (1999)

## DBSCAN

### DBSCAN - Density-Based Spatial Clustering of Applications with Noise

Algorithme proposé en 1996 par Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiaowei Xu.

On se base sur une notion de densité de cluster exprimée par deux valeurs :  $MinPt$  le nombre minimum de points dans un cluster et  $\varepsilon$  un rayon à l'intérieur les points sont considérés comme "proches" et faisant partie d'une même cluster.

Principe : on visite tous les points de l'ensemble des données initial  $\mathcal{D}$ .

Si le point a peu de voisins (moins que  $MinPt$ ) dans un rayon de  $\varepsilon$ , on considère ce point comme étant du bruit et on ne construit pas de cluster.

Si le voisinage est assez important, on explore récursivement les points de ce voisinage en augmentant le voisinage à explorer avec les voisins des voisins.

$\text{DBSCAN}(\mathcal{D}, \text{MinPts}, \varepsilon)$

Pour chaque point  $p$  pas encore visité :

- $\text{Voisinage} \leftarrow \text{Voisin}(p, \varepsilon)$  (y compris  $p$ )
- si  $\text{card}(\text{Voisinage}) < \text{MinPts}$  alors  $p$  est bruit, ignorer
- sinon  $C$  nouveau numéro de cluster et  
 $\text{Expand}(\mathcal{D}, \text{MinPts}, \varepsilon, p, C, \text{voisinage})$

$Expand(\mathcal{D}, MinPts, \varepsilon, p, C, Voisinage)$  est une fonction qui parcourt au plus loin possible l'ensemble de données ( $\mathcal{D}$ ) à partir de  $p$  et  $Voisinage$  :

- ajouter  $p$  au cluster  $C$
- pour chaque point  $p' \in Voisinage$ 
  - si  $p'$  n'est pas encore visité
    - marquer  $p'$  comme visité
    - $BVoisinage \leftarrow Voisin(p', \varepsilon)$
    - si  $card(Bvoisinage) \geq MinPts$ , alors  $Voisinage \leftarrow Voisinage \cup BVoisinage$
  - si  $p'$  n'est pas encore dans un cluster, ajouter  $p'$  à  $C$
  - éliminer  $p'$  de  $Voisinage$

Avantages :

- capable de détecter des clusters de toute forme, y compris concave
- capable d'éliminer les outliers
- pas besoin de savoir le nombre de clusters

Désavantages :

- temps de calcul important et espace mémoire non-négligeable
- on doit avoir des informations sur la densité.

## Méthodes mixtes

Applicable pour des gros volumes de données.

Trois étapes :

- appliquer l'algorithme de K-moyennes pour K suffisamment grand
- clustering ascendant
- coupure à un certain niveau (consolidation selon un indice de qualité - inerties, indice de Dunn, etc)

## Autres méthodes

- algorithme bio-inspiré (algorithmes génétiques)
- cartes de Kohonen (Self Organizing Map ou SOM)

# Plan

## 1 Clustering - définitions, généralités et algorithmes

- Définitions
- Fonctions de distance, similarité
- Caractérisation des solutions de clustering
- Algorithmes de  $K$ -moyennes et de  $K$ -médoides
- Méthodes hiérarchiques de clustering
- DBSCAN
- Méthodes mixtes et autres méthodes

## 2 Classification supervisée - validation

- Définitions
- Les 2 étapes du processus de classification
- Qualité d'un classifieur
- Méthode KNN
- Extensions k-NN

# Classification

La **classification** (appelée aussi **classification supervisée**) consiste à découvrir (apprendre) un modèle sur la base d'un ensemble d'exemples déjà classés, le modèle appris servira à classer des nouvelles observations.

On dispose donc d'un ensemble initial  $\mathcal{D}$  (sous ensemble fini de  $\mathcal{X}$ , univers possible d'observations) et pour chaque  $x \in \mathcal{D}$  on connaît sa classe d'appartenance  $y$ ,  $y \in \mathcal{Y}$  où  $\mathcal{Y}$  est un ensemble fini de classes possibles.

La classification comporte généralement deux phases :

- l'apprentissage du modèle qui consiste à déterminer :
  - soit une caractérisation de chaque classe  $y_i \in \mathcal{Y}$
  - soit une fonction :

$$h : \mathcal{X} \rightarrow \mathcal{Y}, h(x) = y, \forall x \text{ (idéalement)}$$

La fonction  $h$  (le modèle) s'appelle **classifieur**.

- une phase de **prédiction** = traitement des nouvelles observations = détecter la classe d'un nouvel objet :  
pour n'importe quel  $x \in \mathcal{X}$ , trouver  $\hat{y} \in \mathcal{Y}$  tel que  $h(x) = \hat{y}$

Propriétés de  $h(x)$  :

- $\forall i (x_i, y_i) \in \mathcal{D}$ , on veut que  $h(x_i)$  prédise la bonne valeur de  $y_i$
- $h$  doit pouvoir prédire les bonnes sorties pour des exemples futurs  $x_j \notin \mathcal{D}$

## Qualités (possibles) des méthodes de classification / prédition

- exactitude : capacité de étiqueter correctement des nouveaux objets
- vitesse de traitement (coûts)
- robustesse : faire des prédictions correctes sur des données fausses ou bruitées
- passage à l'échelle
- interprétabilité : capacité de rendre le résultat de la phase d'apprentissage dans une forme intelligible

## Qualité et exactitude d'une classification

Erreurs de classement :

- prédire un exemple dans  $C$  alors qu'il appartient à  $\text{non}(C)$
- prédire un exemple dans  $\text{not}(C)$ , alors qu'il appartient à  $C$

Calcul du taux d'erreur :

- Vrai Positif (True Positive) : exemple prédict dans  $C$  et appartient à  $C$
- Vrai Négatif (True Negative) : exemple prédict dans  $\text{not}(C)$  et appartient à  $\text{not}(C)$
- Faux Positif (False Positive) : exemple prédict dans  $C$  et appartient à  $\text{non}(C)$
- Faux Négatif (False Negative) : exemple prédict dans  $\text{not}(C)$  et appartient à  $C$

## Qualité et exactitude d'une classification

La matrice de confusion est une matrice  $K \times K$  ( $K$  nb de classes).

Pour  $K = 2$  :

		Predicted class	
		$C_1$	$C_2$
Actual class	$C_1$	true positives	false negatives
	$C_2$	false positives	true negatives

En R on utilise peut utiliser la fonction `table`

## Qualité et exactitude d'une classification

Sur la base de la matrice de confusion on peut calculer diverses mesures :

- sensibilité (rappel) :  $sens = \frac{t\_pos}{pos}$
- spécificité :  $spec = \frac{t\_neg}{neg}$
- précision :  $prec = \frac{t\_pos}{t\_pos+f\_pos}$
- taux d'erreur :  $\tau = \frac{f\_pos+f\_neg}{pos+neg}$
- exactitude :  $exact = \frac{t\_pos+t\_neg}{pos+neg}$

## Méthode $k$ -plus proches voisins

Depuis 1950 : *k-nearest neighbor (kNN)*

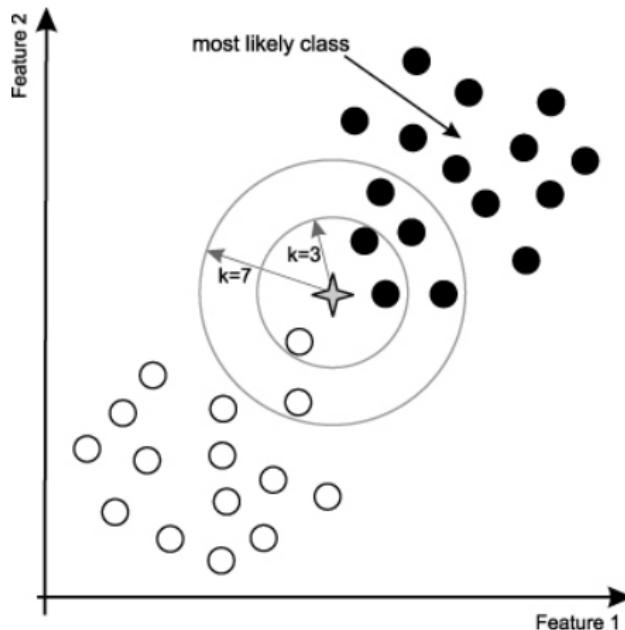
La phase d'apprentissage n'est pas explicitement utilisée (sauf si normalisation).

Pour un élément inconnu  $x$

- on calcule les distances par rapport à tous les éléments de l'ensemble d'apprentissage  $x_i$ ,  $i = 1, N'$
- on trie ces valeurs
- on choisit la classe majoritaire parmi les  $k$  premiers.

Donc parmi les  $k$  plus proches voisins on choisit la classe la plus représentée.

$k$  est un paramètre qu'on fixe selon le problème à traiter (taille du problème et signification)



## Méthode $k$ -plus proches voisins

On prend, généralement, la distance euclidienne ou Manhattan et  $k$  relativement petit.

La méthode s'implémente facilement dans un langage de programmation.

## Méthode $k$ -plus proches voisins - extensions

Si l'ensemble initial d'apprentissage est très grand, pour chaque classe on peut prendre un nombre  $R$  (relativement petit) de points caractéristiques et ensuite comparer l'élément  $x$  avec le  $K \times R$  points et appliquer le principe de la classe majoritaire parmi le  $k$  premiers.

## Extension k-means

Calculer pour chaque classe  $\mathcal{C}_i$  un clustering de type  $k$ -means (stable?) avec  $k = R$ ; on prend en considération les centres des clusters. On obtiens alors  $R \times K$  prototypes, chaque prototype a une étiquette.

## LVQ - Learning Vector Quantization

- choisir au hasard  $R$  points dans chaque classe  $\mathcal{C}_i$ , ces points sont nommés prototypes.
- répéter pour un  $\varepsilon$  décroissant :
  - choisir un point  $x$  au hasard. Soit  $j$  sa classe.
  - déterminer le prototype  $r$  le plus proche.  $r$  appartient à une classe  $l$
  - si  $j = l$  déplacer le prototype vers  $x$

$$r \leftarrow r + \varepsilon(x - r)$$

- sinon éloigner le prototype de  $x$

$$r \leftarrow r - \varepsilon(x - r)$$