Classification non supervisée

Guillaume Wisniewski

guillaume.wisniewsk@limsi.fr

LIMSI — UPS

février 2016

Classification supervisée

- $\mathcal{X}$ = ensemble des observations/exemples (en pratique $\mathcal{X} = \mathbb{R}^d$)
- $\mathcal{Y}$ = ensemble des étiquettes
- dépendance fonctionnelle f entre $\mathcal{X}$ et $\mathcal{Y}$
- un expert (oracle) peut étiqueter des données
- classification : retrouver f à partir d'un échantillon fini
- classification supervisée : on dispose d'un ensemble de données étiquetées

Classification non supervisée

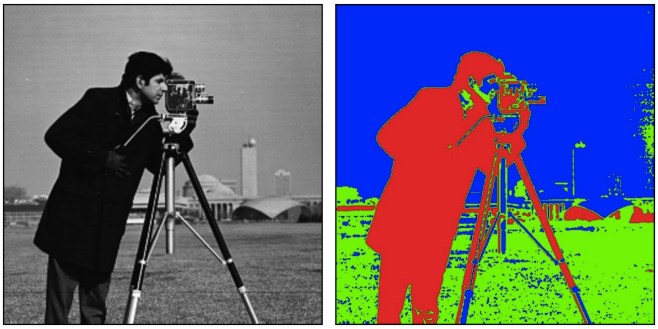
Définition

- classification non supervisée = pas d'information sur l'étiquette des données
  - étiquetage possible, mais coûteux
  - système évolutif (p.ex. détection de sujet dans une conversation ⇒ apparition de nouvelles classes)
  - analyse de données

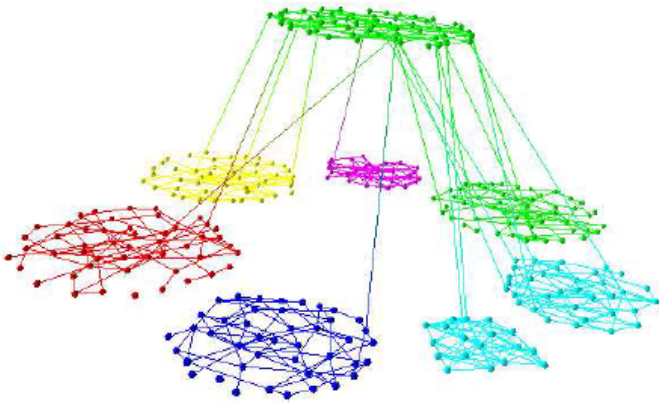
Hypothèse

- le nombre de classes k
- la probabilités à priori de chaque classe (?)
- la forme paramétrique de la densité de probabilité de chaque classe (? ?)

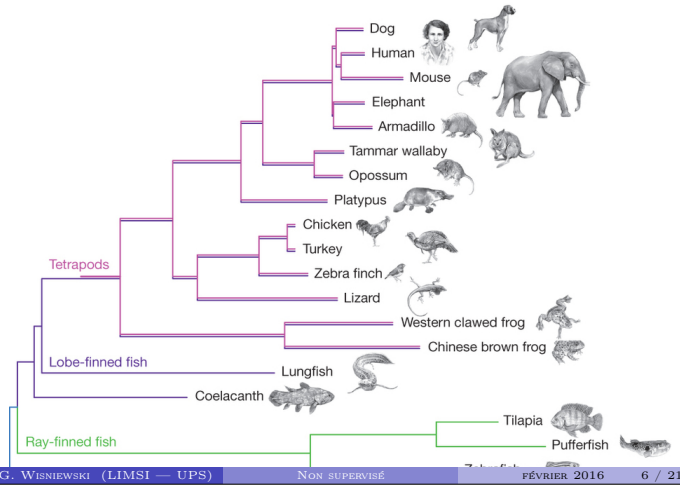
Exemple n° 1 : Segmentation d'images



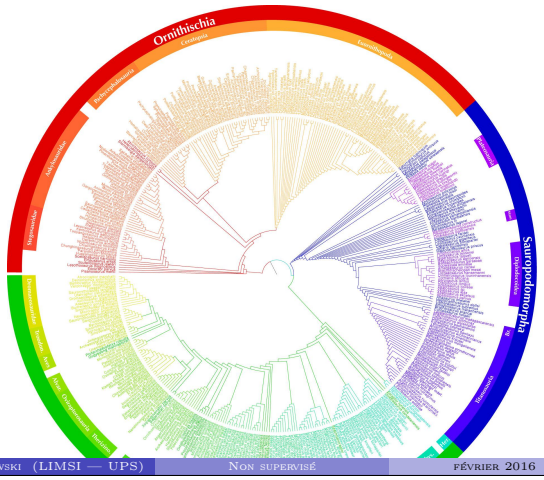
Exemple n° 2 : identification de communauté



Exemple n° 3 : Phylogenetic Tree



Exemple n° 3 : Phylogenetic Tree (vrai)



Exemple n° 4 : ciblage publicitaire

« The Natural History of Gmail Data Mining »

- Procès Texarkana : a obligé Google a dévoilé « plein » d'information sur le fonctionnement de son ciblage publicitaire
- chaque utilisateur fourni beaucoup de données :
  - formulaire
  - trace de navigation
  - contenu des échanges (p.ex. confirmation d'achat en ligne)
- Google est capable de les compléter (code postal/adresse $\Rightarrow$ revenu moyen)
- les utilisateurs sont regroupés en plusieurs millions de « bucket »

Remarque essentielle

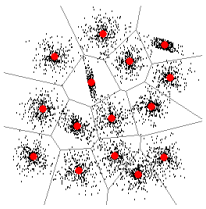
- dans tous les cas : on ne connait pas la bonne réponse ($\oplus$ qu'elle existe)
- qualité d'un clustering = arbitraire / subjectif
- pas de « vrai » critère d'observation

Première partie I

Méthode de clustering

Principes du clustering

- méthodes de coalescence *(clustering)* : séparer les données en paquets de points similaires
  - mesure de similarité/dissimilarité entres points ?
  - qualité de la partition des données entre paquets ?
- mesure de similarité : distance euclidienne + seuil de distance
  - problème : sensibilité aux changements d'échelle (x/y)
  - normalisation préalable des données : moyenne / variance ou analyse en composantes principales

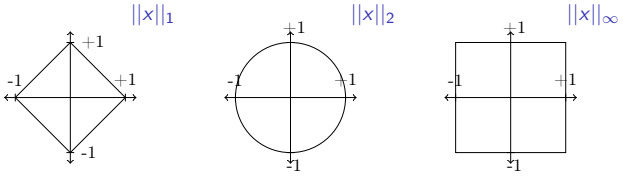


Rappel : distance

Norme dans $\mathbb{R}^n$

$$d_\lambda(x,y) = \left[\sum_{k=1}^{d} |x_k - y_k|^\lambda\right]^{\frac{1}{\lambda}}$$

- $\lambda = 1$ distance de Manhattan
- $\lambda = 2$ distance euclidienne
- $\lambda \to \infty$ distance de Chebyshev



Critère de qualité

- Critère de qualité
  - H = ensemble de n échantillons $\{x_1, ..., x_n\}$
  - H partionné en c paquets disjoints $H_1, ..., H_c$
  - qualité de la partition : $\mathcal{J}(H)$ (question de recherche ouverte)
- moindre carrés
  - soit μ_i la moyenne du paquet H_i : $\mu_i = \frac{1}{n_i} \sum_{x_j \in H_i} x_j$
  - alors la somme des erreurs au carré est $\mathcal{J} = \sum_{i=1}^{c} \sum_{x_j \in H_i} ||x_j - \mu_i||^2$
  - partition à variance minimum
    - ⋆ adapté pour les nuages de points compacts
    - ⋆ problème si le nombre de points des nuages est déséquilibré
- reformulation :

$$\mathcal{J} = \frac{1}{2} \sum_{i=1}^{c} n_i S_i, \text{ avec } S_i = \frac{1}{n_i^2} \sum_{x_i, x_j \in H_i} s(x_j, x_k)$$

Recherche de la partition

- recherche directe :
  - explosion combinatoire : $\sim \frac{c^n}{n!}$ possibilités
- recherche par optimisation itérative
  - partition initiale
  - modification de la partition en améliorant le critère de qualité
  - problème : atteinte d'un minimum local
- procédure des k-moyennes
- généralisation : nuée dynamique
- variante ISODATA
  - regroupement / division pour avoir des classes homogènes

k-moyennes : le cadre

Entrée

- (x_i) : ensemble de points de $\mathbb{R}^n$
- k : nombre de partitions à trouver
- d : distance / mesure de similarité

Sortie

- k points de $\mathbb{R}^n$: centre/représentation des clusters
- chaque x_i est assigné à un cluster

k-moyennes : le principe

- classifieur distance minimale
- si on connait les centres : trivial de partitionner les données
- si on connait le partitionnement : trivial de trouver les centres
- estimations successives jusqu'à convergence

Algorithme des k-moyennes

1. choisir des valeurs initiales $\hat{\mu}_1, ..., \hat{\mu}_c$
2. classifier les n échantillons dans la classe pour laquelle ils sont le plus proche de $\hat{\mu}_i$:

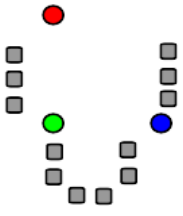
$$\omega(x_k) = \underset{i}{\text{argmin}} ||x_k - \hat{\mu}_i||$$

3. recalculer la moyenne à partir des points associés à la classe

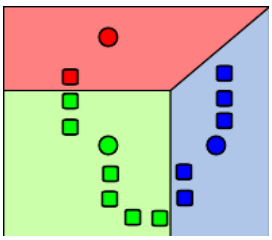
$$\hat{\mu}'_i = \frac{\sum_{\omega(x_k)=i} x_k}{\#\{x_k | \omega(x_k) = i\}}$$

4. reboucler à l'étape 2 tant que :
   - il existe i tel que $\hat{\mu}_i \neq \hat{\mu}'_i$
   - le nombre maximal d'itérations n'est pas atteint
   - le gain relatif du critère de qualité est trop faible

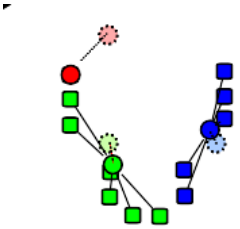
Principe des k-means



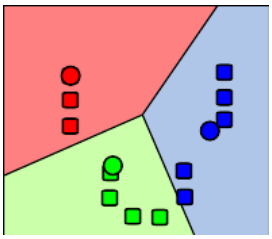
Principe des k-means



Principe des k-means

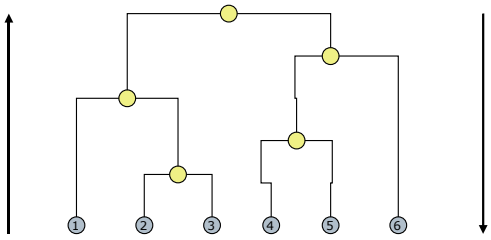


Principe des k-means



Classification hiérarchique

- au lieu d'une partition, on considère une séquence de partitions imbriqués :
  - niveau 1 : 1 paquet de n éléments
  - niveau n : n paquets de 1 éléments



- approches ascendantes (par agglomération) et descendantes (par division)

Méthode par agglomération

- principe
  1. initialement, un paquet par classe, $c = n$, $H_i = \{x_i\}$
  2. choisir les deux paquets les plus proches et les fusionner
  3. répéter l'étape 2 jusqu'à atteindre le nombre de classe désiré (ou c=1, ou un autre critère d'arrêt)
- distances inter-clusters

$$d_{\min}(H_i, H_j) = \min_{x \in H_i, y \in H_j} ||x - y|| \qquad d_{\max}(H_i, H_j) = \min_{x \in H_i, y \in H_j} ||x - y||$$

$$d_{\text{avg}}(H_i, H_j) = \frac{1}{n_i \cdot n_j} \sum_{x \in H_i, y \in H_j} ||x - y|| \qquad d_{\text{mean}(H_i, H_j)} = ||\mu_i - \mu_j||$$

- détecter quand le critère dérive de la distance inter-éléments (possible à partir de la matrice des distances)

Méthode par division

- principe
  1. initialement, tous les points dans une classe ($c = 1$)
  2. diviser un ou plusieurs paquets en sous-paquets
  3. répéter jusqu'à satisfaction du critère d'arrêt
- quantification vectorielle binaire (algorithme Linde, Buzo, Gray)
  - on part d'un centroïde
  - à chaque itération, le nombre de paquets est doublé en créant de nouveaux centroïdes par perturbation du centroïde initial de chaque classe

$$\begin{cases} \mu_i^+ = \mu_i(1+\epsilon) \\ \mu_i^- = \mu_i(1-\epsilon) \end{cases}$$

  - les centroïdes sont recalculés par les k-moyennes