Méthodes de classifications

Dhafer Malouche

ESSAI-U2S-ENIT
http://essai.academia.edu/DhaferMalouche
dhafer.malouche@me.com

Juin-2013
ANR−Do Well Be, Saint Nectaire

Plan

Définition

- Algorithmes qui permettent de classer des objets observés dans des classes (appelées *clusters*).
- Les objets d'une même classe doivent être "similaires" et les objets de deux classes différentes doivent être "distincts".
- Différents algorithmes de classification
  - Méthodes Hiérarchiques (*classification hiérarchiques*...)
  - Partitionnement (*k-means*..)
  - Méthodes Probabilistes (*EM-algorithme*...) et encore d'autres...

Les objets

- $E = \{w_1, \ldots, w_n\}$ un ensemble de n objets (individus) pour lesquels on a observé d "variables" (qualitatives et/ou quantitatives) $x^1, \ldots, x^d$ où

$$x_i^j = x^j(w_i)$$

est la i$^{\text{ème}}$ observation de w_i selon x^j.

- On considère $w_i = (x_{i,t})_{t \in T_i}$ $i = 1, \ldots, n$ des signaux ou séries temporelles (ST) .

- Une matrice $D = (d_{ii'})$, $n \times n$ (n est le nombre d'objets à classer), où $d_{ii'}$ est une mesure de similarité entre deux objets w_i et $w_{i'}$,

Etape 1 : calcul des similarités

Cas de variables quantitatives.

- Distance *Euclidienne*

$$d(w_i, w_{i'}) = \sqrt{\sum_{j=1}^{d} (x_i^j - x_{i'}^j)^2}.$$

- ou la distance de *Minkowski* pour tout $p > 0$

$$d(w_i, w_{i'}) = \left(\sum_{j=1}^{d} (x_i^j - x_{i'}^j)^p \right)^{1/p}.$$

- encore d'autres exemples de distances : *Canberra*, *Manhattan*...
- Sous `R` : `dist`, il faut spécifier
  - une matrice de données quantitative
  - le type de distance

Distance entre signaux ou ST

- Distance Euclidienne ou l'une des distances déjà définies (Séries sont toutes de même "longueurs").
- *Dynamic time wrapping* ou Déformation temporelle dynamique (par Keogh & Pazzani, 2001 [1])
  - Permet de comparer deux séquences de ST pas forcément de même longueur.
  - Dilatation ou compression des séquence pour obtenir le meilleur alignement possible
  - Sous `R`, package `dtw`.

[1] Keogh, E. J. and Pazzani, M. J. (2001). Derivative dynamic time warping. In the 1st SIAM Int. Conf. on Data Mining (SDM-2001), Chicago, IL, USA.

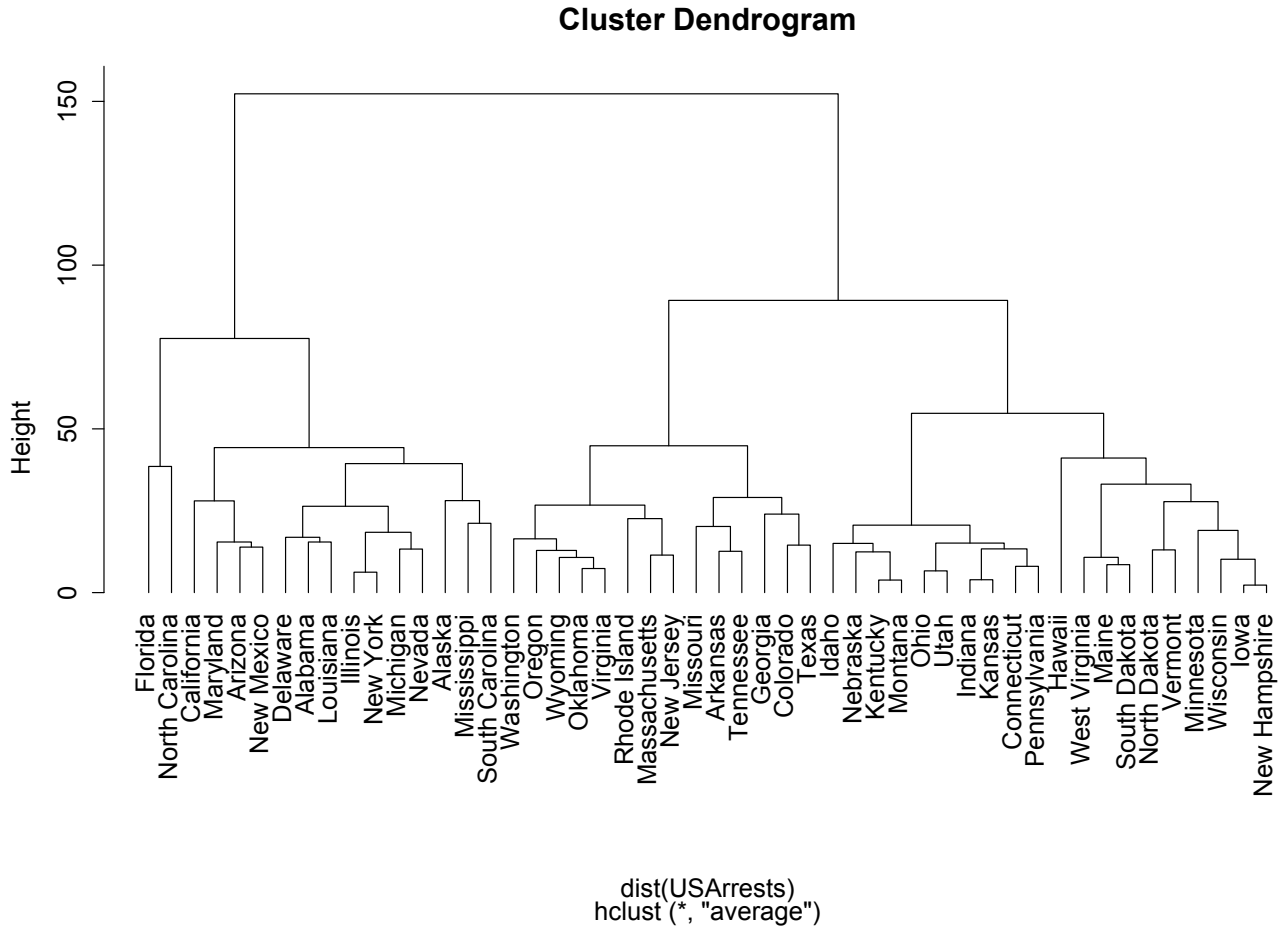
Algorithme ascendant (agrégatif) hiérarchique

1. $k = n$ le nombre de classes
2. Chaque classes contient une observation : n classes $C^1, \ldots, C^n$.
3. Calculer la matrice de distances entre classes (phase agrégative de l'algorithme).
4. Considérer les deux classes C^i et C^j telles que

$$d(C^i, C^j) = \min_{(i',j')} d(C^{i'}, C^{j'})$$

.

5. Agréger C^i et C^j dans une seule classe C^{ij}. Donc $k = k - 1$.
6. $k = 1$? Si oui STOP, sinon, aller à 3.


Cluster Dendrogram
Height
0
50
100
150
Florida
North Carolina
California
Maryland
Arizona
New Mexico
Delaware
Alabama
Louisiana
Illinois
New York
Michigan
Nevada
Alaska
Mississippi
South Carolina
Washington
Oregon
Wyoming
Oklahoma
Virginia
Rhode Island
Massachusetts
New Jersey
Missouri
Arkansas
Tennessee
Georgia
Colorado
Texas
Idaho
Nebraska
Kentucky
Montana
Ohio
Utah
Indiana
Kansas
Connecticut
Pennsylvania
Hawaii
West Virginia
Maine
South Dakota
North Dakota
Vermont
Minnesota
Wisconsin
Iowa
New Hampshire
dist(USArrests)
hclust (*, "average")

Méthodes d'agrégations

$$d(C^l, C^{ij}) = \alpha_i d(C^l, C^i) + \alpha_j d(C^l, C^j) + \beta d(C^i, C^j) + \gamma \left| d(C^l, C^i) - d(C^l, C^j) \right|$$

Algorithme de classification	α_i	α_j	β	γ
Single Linkage	$1/2$	$1/2$	0	$-1/2$
Complete Linkage	$1/2$	$1/2$	0	$1/2$
Centroïd Linkage	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$-\frac{n_i n_j}{n_i+n_j}$	0
Méthode de Ward	$\frac{n_i+n_l}{n_i+n_j+n_l}$	$\frac{n_j+n_l}{n_i+n_j+n_l}$	$-\frac{n_l}{n_i+n_j+n_l}$	0

Méthodes d'agrégations

- *Single Linkage* : $d(C^l, C^{ij}) = \min(d(C^l, C^i), d(C^l, C^j))$
- *Complete Linkage* : $d(C^l, C^{ij}) = \max(d(C^l, C^i), d(C^l, C^j))$
- *Centroïd Linkage* : $d(C^l, C^{ij}) = d(m_l, m_{ij})$ où m_l et m_{ij} sont resp. les centres de gravité de C^l et C^{ij}.
- *Méthode de Ward* (après une ACP [2]) : elle a pour objectif de minimiser l'accroissement dans la variance intra-classe :

$$E = \sum_{l=1}^{k} \sum_{x_i \in C^l} d(x_i, m_l)^2.$$

[2] Analyse en composantes principales

Algorithme descendant hiérarchique (DIANA).

1. $k = 1$, Toutes sont dans la même classe C_1
2. Considérer C_i la classe de diamètre maximale $(d(C_i) = \max_{w,w' \in C_i} d(w, w'))$.
3. Introduire $C_j = \emptyset$.
   - 3.1 Pour chaque $w \in C_i$, calculer

$$d\left(w, C_i \setminus \{w\}\right) = \frac{1}{|C_i| - 1} \sum_{w' \in C_i \setminus \{w\}} d(w, w').$$

   - 3.2 Calculer la différence $\Delta(w, C_i, C_j) = d\left(w, C_i \setminus \{w\}\right) - d(w, C_j)$
4. Si $w_m = \text{argmax}_{w \in C_i} d\left(w, C_i \setminus \{w\}\right)$ et $\Delta(w_m, C_i, C_j) > 0$ alors $w_m \ni C_j$
5. si $k < n$ alors $k = k + 1$ revient à 1/ sinon STOP.

- **Objectif** A partir d'une fonction critère J donnée, partitionner $w^1, \ldots, w_n$ en K classes (K est fixé d'avance) en minimisant ou maximisant la fonction J
- Méthode exhaustive : calculer pour toute les partitions possible C, le critère $J(C)$ et la solution est alors

$$C_{sol} = \text{argmax} J(C).$$

- Solution impossible si n devient grand. En effet le nombre de partitions possible est égal à

$$P(n, K) = \frac{1}{K!} \frac{m=1}{K} (-1)^{K-m} C_K^m m^N.$$

Pour $n = 30$, et $K = 3$, $P(n, K)$ est de l'ordre de 2×10^{14}.

$K-$means

- Critère à minimiser : la somme des carrées des erreurs (SSE) :

$$J(C) = \sum_{c \in C} \sum_{w \in c} d(w, m(c))$$

où $m(c)$ est le centre de gravité de c.

- Algorithme :
  1. Initialise $K-$ centres $m_1, \ldots, m_K$ et
  2. Affecter chaque individu vers la classe la plus proche $C = \{c_1, \ldots, c_K\}$:

$$w \in c_l \text{ si } d(w, m_l) = \min_{l'=1,\ldots,K} d(w, m_{l'}).$$

  3. Re-calculer les centres de gravité des nouvelles classes,
  4. Répéter 2 et 3 jusqu'à convergence.

Mesure de la validité d'une classification

Deux types de mesures :

- Indices de validité interne : à partir d'un résultat de classification et de l'information intrinsèque dans la base de données on mesure la qualité du résultat :
  - indice de connectivité
  - indice de silhouette
  - indice de *Dunn*
- Indice de validité externe : comparer la classification obtenue avec une autre classification (soit une partition déjà connue ou une autre classification obtenue par une autre méthode)

Notations

Soit

- n le nombre des observations,
- d le nombre des variables,
- $x_1, \ldots, x_n$ les vecteurs d'observations à classer,
- $\mathcal{C} = \{C_1, \ldots, C_k\}$ une classification à tester,
- $C(i)$ est la classe contenant x_i, i.e., $C(i) \ni x_i$.

Pour tout x_i, et $j \in \{1, \ldots, n\} \setminus \{i\}$: $x_i^{(j)}$ est le $j^{\text{ième}}$ voisin le plus proche de x_i :

$$d(x_i, x_{(1)}) \leq d(x_i, x_{(2)}) \leq \ldots d(x_i, x_{(j)}) \leq \ldots \leq d(x_i, x_{(n-1)})$$

On pose pour tout $i, j \in \{1, \ldots, n\}$

$$\delta(x_i, x_i^{(j)}) = \begin{cases} 0 & \text{si } x_i,\ x_i^{(j)} \in C(i) \\ 1/j & \text{sinon} \end{cases}$$

Indices de Connectivité

Pour une classification $\mathcal{C}$,

$$\text{Conn}(\mathcal{C}) = \sum_{i,j=1,\ldots,n} \delta(x_i, x_i^{(j)})$$

$\text{Conn}(\mathcal{C})$ prend ses valeurs dans $[0, +\infty[$ et c'est un cœfficient qui doit être minimisée :

$\text{Conn}(\mathcal{C})$ proche de zéro $\iff$ une bonne connectivité à l'intérieure des classes.

`R : clValid package, connectivity`

Largeur de la silhouette

- C'est la moyenne de toutes les valeurs de la silhouette.
- Valeur de la silhouette pour une observation x_i :

$$S(i) = \frac{b_i - a_i}{\max(b_i, a_i)}$$

où

- a_i est la distance moyenne entre x_i et tous les autres $x_j \in C(i)$.

$$a_i = \frac{1}{n(C(i))} \sum_{x \in C(i)} d(x_i, x).$$

- b_i est la distance moyenne entre x_i et les observations dans la classe la pus proche de x_i :

$$b_i = \min_{C \in \mathcal{C} \setminus C(i)} \frac{1}{n(C)} \sum_{x \in C} d(x_i, x).$$

Largeur de la silhouette

- Définie par

$$S(\mathcal{C}) = \frac{1}{n} \sum_i S(i)$$

- La largeur de la silhouette est un nombre qui est toujours compris entre -1 et $+1$.
- S doit être maximisée.
- une valeur de $S(i)$ négative indique que cet individu n'est pas dans "la bonne classe" et il pourrait être déplacé vers la classe la plus proche.
- `R, package cluster, silhouette`

Indices externes

- $\mathcal{P}$ une partition existante de l'ensemble des individus E, $\mathcal{P} = \{G_1, \ldots, G_r\}$.
- $\mathcal{C} = \{C_1, \ldots, C_k\}$ une classification obtenue à l'aide d'un algorithme de classification (*hiérarchique*, *kmeans*...)
- Objectif : comparer la classification $\mathcal{C}$ et la partition $\mathcal{P}$
- Définition de quelques indices de validité externes : package `clv` sour `R`

Calcul des indices externes

Soient deux individus x_i et $x_{i'}$. Quatre cas sont possibles

Cas 1 $\exists C \in \mathcal{C}$ et $\exists G \in \mathcal{P}$ tel que

$$x_i, x_{i'} \in C \text{ et } x_i, x_{i'} \in G$$

Cas 2 $\exists C \in \mathcal{C}$ et $\exists G \neq G' \in \mathcal{P}$ tel que

$$x_i, x_{i'} \in C \text{ et } x_i \in G, x_{i'} \in G'$$

Cas 3 $\exists C \neq C' \in \mathcal{C}$ et $\exists G \in \mathcal{P}$ tel que

$$x_i \in C, x_{i'} \in C' \text{ et } x_i, x_{i'} \in G$$

Cas 4 $\exists C \neq C' \in \mathcal{C}$ et $\exists G \neq G' \in \mathcal{P}$ tel que

$$x_i \in C, x_{i'} \in C' \text{ et } x_i \in G, x_{i'} \in G'$$

Calcul des indices externes

Notons par $\mathcal{A}_l$ l'ensemble des paires $(x_i, x_{i'})$ vérifiant le cas l et par $a_l = |\mathcal{A}_l|$. Trois indices :

- Indice de Rand

$$\text{Rand} = \frac{a_1 + a_4}{a_1 + a_2 + a_3 + a_4}$$

- Indice de Jaccard

$$\text{Jaccard} = \frac{a_1}{a_1 + a_2 + a_3}.$$

- Indice de Folkes et Mallows

$$\text{FM} = \sqrt{\frac{a_1}{a_1 + a_3} \frac{a_1}{a_1 + a_2}}$$

l'EM-algorithme

- Soit $\underline{X} = (X_1, \ldots, X_n) \sim g(x \mid \theta)$ un $n-$échantillon de vecteurs aléatoire dans $\mathbb{R}^d$. On suppose

$$g(x \mid \theta) = \int f(x, z \mid \theta) dz$$

- Objectif : calculer $\widehat{\theta} = \text{argmax}_\theta L(\underline{x} \mid \theta) = \text{argmax}_\theta \prod_{i=1}^{n} g(x_i \mid \theta)$ où $\underline{x} = (x_1, \ldots, x_n)$ est une réalisation de $\underline{X}$.
- On pose $(X, Z) \sim f(x, z \mid \theta)$, et la densité de $Z \mid \theta, X = x$ est

$$k(z \mid \theta, x) = \frac{f(x, z \mid \theta)}{g(x \mid \theta)}.$$

l'EM-algorithme

- Donc

$$\log g(x \mid \theta) = \log f(x, z \mid \theta) - \log k(z \mid \theta, x)$$

- Pour un θ_0 fixé,

$$\begin{aligned} \log g(x \mid \theta) k(z \mid \theta_0, x) &= \log f(x, z \mid \theta) k(z \mid \theta_0, x) \\ &\quad - \log k(z \mid \theta, x) k(z \mid \theta_0, x) \quad (1) \end{aligned}$$

- En intégrant (1) par rapport à z, on obtient

$$\log g(x \mid \theta) = \mathbb{E}_{\theta_0}(\log f(x, Z \mid \theta)) - \mathbb{E}_{\theta_0}(\log k(Z \mid x, \theta))$$

l'EM-algorithme

- La fonction vraisemblance s'écrit alors

$$\begin{aligned}\log L(\underline{x} \mid \theta) &= \sum_{i=1}^{n} \log g(x_i \mid \theta) \\ &= \underbrace{\mathbb{E}_{\theta_0}(\log L^c(\underline{x}, Z \mid \theta))}_{Q(\theta|\underline{x},\theta_0)} - \sum_{i=1}^{n} \mathbb{E}_{\theta_0}(\log k(Z \mid x_i, \theta))\end{aligned}$$

  où L^c est appelée la vraisemblance *complète*.
- Ainsi, dans la recherche du maximum de $\log L(\underline{x} \mid \theta)$, dans l'EM-algorithme on maximize, d'une façon itérative, $Q(\theta \mid \underline{x}, \theta_0)$.

l'EM-algorithme

1. Considérons une valeur initiale $\widehat{\theta}_{(0)}$
2. Calculer l'espérance *(E−step)*

$$Q(\theta \mid \underline{x}, \widehat{\theta}_{(m)}) = \mathbb{E}_{\widehat{\theta}_{(m)}}(\log L^c(\underline{x}, Z \mid \theta))$$

où la moyenne a été calculée par rapport à $k(z \mid \widehat{\theta}_{(m)}, x)$ et $m = 0$

3. Maximiser la fonction $Q(\theta \mid \underline{x}, \widehat{\theta}_{(m)})$ en θ *(M−step)* :

$$\widehat{\theta}_{(m+1)} = \text{argmax}_\theta Q(\theta \mid \underline{x}, \widehat{\theta}_{(m)})$$

et $m = m + 1$.

4. Répéter les étapes 2-3 jusqu'à le point fix soit atteint : $\widehat{\theta}_{(m+1)} = \widehat{\theta}_{(m)}$

l'EM-algorithme, dans la classification

- On suppose que les données $\underline{x} = (x_1, \ldots, x_n) \in (\mathbb{R}^d)^n$ est un $n-$échantillon qu'on suppose issu d'une loi de probabilité

$$g(x \mid \theta) = \sum_{l=1}^{k} \alpha_l g_l(x \mid \theta_l)$$

où $\alpha_l \in]0,1[$, $l = 1, \ldots, k$ et $\sum_{l=1}^{k} \alpha_l = 1$ et $g_l(x \mid \theta_l)$ est la densité d'une loi Gaussienne multivariée :

$$f_l(x \mid \theta_l) = \frac{1}{(2\pi)^{d/2}|\Sigma_l|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_l)'\Sigma_l^{-1}(x-\mu_l)\right)$$

- Donc $X \sim g(x \mid \theta)$ et Z est la variable à valeurs dans $\{1, \ldots, k\}$, où k est la nombre de classes et

$$\mathbb{P}(Z = l) = \alpha_l, \quad \forall\, l = 1, \ldots, k.$$

Mise en œuvre sous `R` : le package `mclust`

- Description détaillé du package dans
  - C. Fraley and A. E. Raftery (2006). MCLUST Version 3 for R : Normal Mixture Modeling and Model-Based Clustering, Technical Report no. 504, Department of Statistics, University of Washington
  - C. Fraley and A. E. Raftery (2002). Model-based clustering, discriminant analysis, and density estimation. Journal of the American Statistical Association 97 :611-631
- Utiliser la commande `Mclust`,
- `Mclust` applique l'EM-algorithme (avec des différentes paramétrisations), et utilise le BIC comme critère de sélection du modèle : chercher le modèle M_k correspondant au BIC maximal.

$$\text{BIC} = 2 \log \text{Vraisemblance} - \text{Nbre paramètres} \times \log(n)$$