Classification automatique

Chapitre 3 : Méthodes par partitionnement

Allou Samé
allou.same@ifsttar.fr

Plan

Données

n individus décrits par p variables

$$\mathbf{X} = \begin{pmatrix} x_1^1 & \dots & x_1^j & \dots & x_1^p \\ x_i^1 & \dots & x_i^j & \dots & x_i^p \\ x_{n1} & \dots & x_{nj} & \dots & x_{np} \end{pmatrix}$$

Algorithme des centres mobiles ou k-means

On suppose ici que le tableau X est formé de variables quantitatives
On considère la distance euclidienne

Objectif

Partitionner l'ensemble des individus $E = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ en K classes

Algorithme

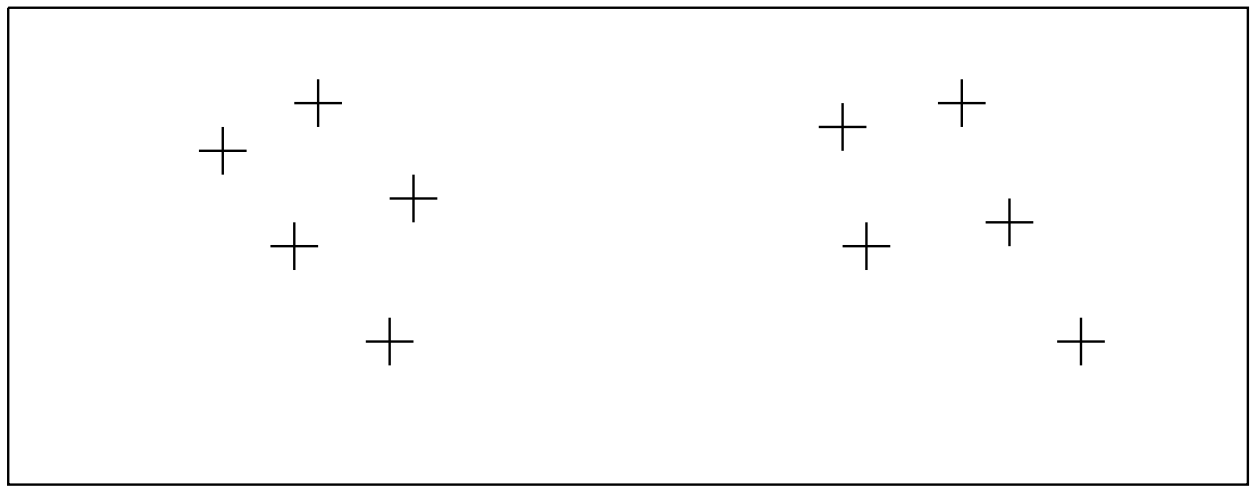
1. ***Initialisation*** : tirage au hasard de K points de E qui forment les centres initiaux des K classes
2. ***Tant que les classes ne sont pas stabilisées*** *(où que le critère d'inertie intra-classe ne décroît plus de manière significative)*

   (a) Construction d'une partition en affectant chaque point de E à la classe dont il est le plus près du centre de gravité

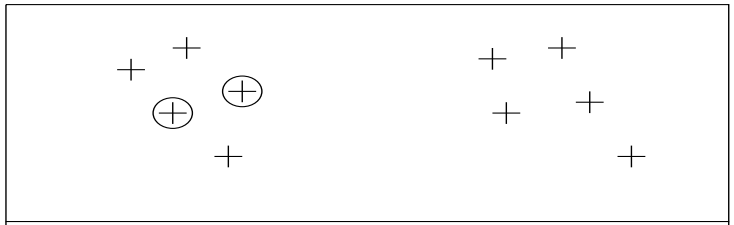
   (b) Calcul des centres de gravité des classes qui viennent d'être calculées ; ceux-ci deviennent les nouveaux centres

Exemple (1/2)

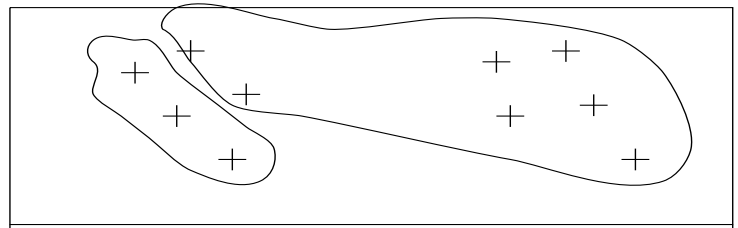
$n = 10$ points de $\mathbb{R}^2$ à partitionner en $K = 2$ classes



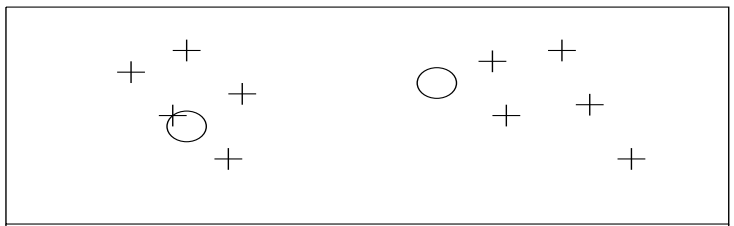
Exemple (2/2)



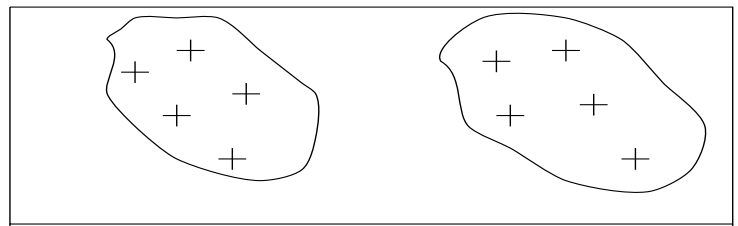
Etape 1 : choix de 2 points au hasard



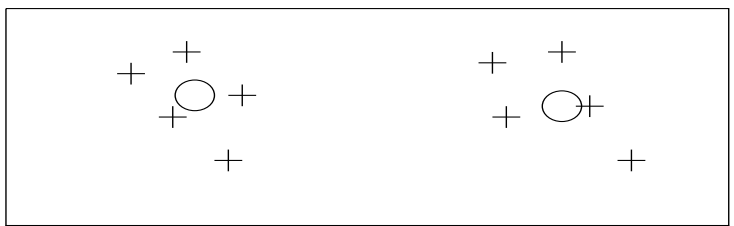
Etape 2 : affectation de chaque point au centre le plus proche



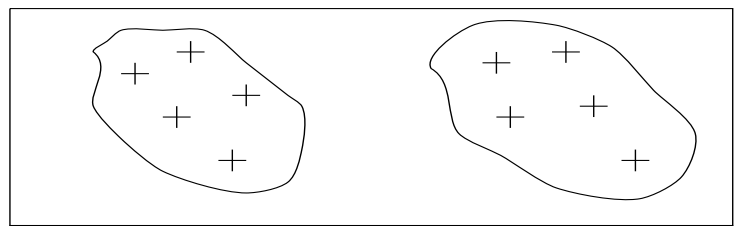
Etape 3 : calcul des centres de gravite



Etape 4 : affectation de chaque point au centre le plus proche



Etape 5 : calcul des centres de gravite



Etape 6 : affectation de chaque point au centre le plus proche

Critère optimisé par l'algorithme des centres mobiles

Inertie intra-classe

L'algorithme des centres mobiles permet de trouver la partition $P = (P_1, \ldots, P_K)$ de E qui minimise le critère d'inertie intra-classe

$$I_W(P) = \frac{1}{n} \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in P_k} \| \mathbf{x}_i - \mathbf{g}_k \|^2$$

avec

$$\mathbf{g}_k = \frac{1}{n_k} \sum_{\mathbf{x}_i \in P_k} \mathbf{x}_i \quad \text{(centre de gravité de la classe } P_k\text{)}$$

$$n_k = \text{nombre d'éléments de la classe } P_k$$

Critère optimisé par l'algorithme des centres mobiles

Formulations équivalentes

La minimisation par rapport à P de l'inertie intra-classe revient à la minimisation par rapport à la partition $P = (P_1, \ldots, P_K)$ et aux centres des classes $\boldsymbol{\mu} = (\boldsymbol{\mu}_1 \ldots, \boldsymbol{\mu}_K)$ du critère

$$C(P, \boldsymbol{\mu}) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in P_k} \| \mathbf{x}_i - \boldsymbol{\mu}_k \|^2$$

qui peut également s'écrire

$$\begin{aligned} C(\mathbf{z}, \boldsymbol{\mu}) &= \sum_{k=1}^{K} \sum_{i=1}^{n} z_{ik} \| \mathbf{x}_i - \boldsymbol{\mu}_k \|^2 \\ &= \sum_{i=1}^{n} \| \mathbf{x}_i - \boldsymbol{\mu}_{z_i} \|^2 \end{aligned}$$

où $\mathbf{z} = (z_{ik})$ est la matrice binaire de classification $\left(z_{ik} \in \{0; 1\} \right)$

Convergence de l'algorithme des centres mobiles

L'algorithme des centres mobiles construit une séquence de centres et de partitions jusqu'à la convergence :
$\boldsymbol{\mu}^{(0)} \to P^{(1)} \to \boldsymbol{\mu}^{(1)} \to P^{(2)} \to \boldsymbol{\mu}^{(2)} \ldots \to \boldsymbol{\mu}^{(c)} \to P^{(c+1)} \to \boldsymbol{\mu}^{(c+1)} \ldots$

Proposition

Chaque étape de l'algorithme améliore (fait décroître) le critère C :

$$\begin{aligned} C(P^{(c+1)}, \boldsymbol{\mu}^{(c+1)}) &\leq C(P^{(c+1)}, \boldsymbol{\mu}^{(c)}) \quad \text{(calcul des centres)} \\ C(P^{(c+1)}, \boldsymbol{\mu}^{(c)}) &\leq C(P^{(c)}, \boldsymbol{\mu}^{(c)}) \quad \text{(calcul de la partition)} \end{aligned}$$

Corollaire

La suite numérique $\left(C(P^{(c)}, \boldsymbol{\mu}^{(c)})\right)$ converge en un nombre fini d'itérations.

Remarques

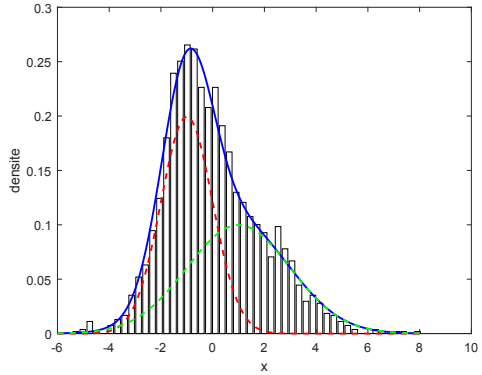
- L'algorithme des centres mobiles conduit à une suite décroissante du critère d'inertie intra-classes ; la partition obtenue dépend de l'initialisation ; on obtient donc un minimum local de l'inertie intra-classe.
- Compte tenu de la dépendance à l'initialisation, plusieurs exécutions à partir d'initialisations différentes sont recommandées (ex. initialiser les centres en tirant au hasard les centres parmi les données).
- Le critère optimisé est associé à un nombre de classes fixé par l'utilisateur ; ce critère ne peut pas être minimisé par rapport au nombre de classes, sinon la partition en n classes où chaque classe est un singleton serait la meilleure

Remarques

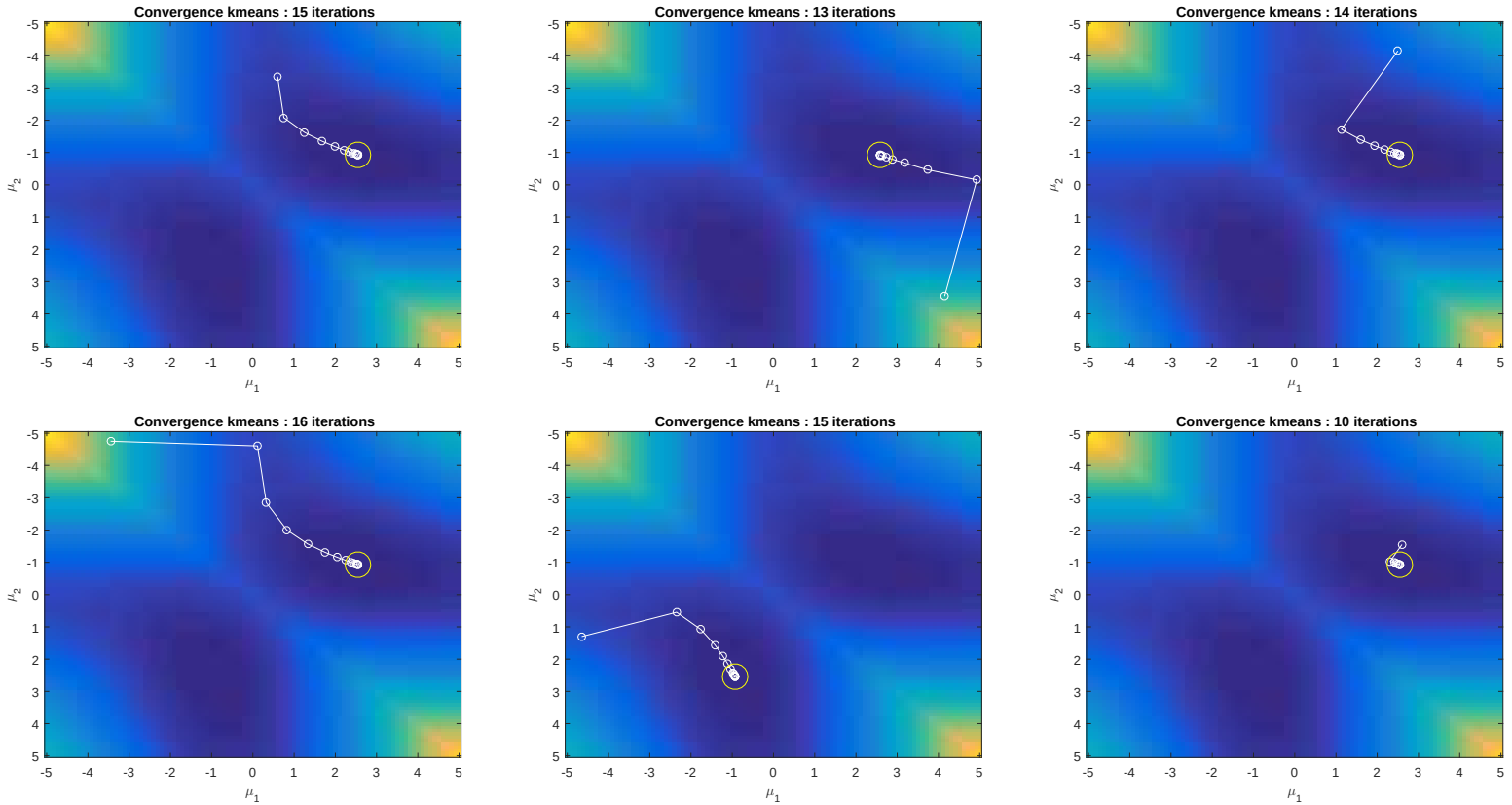
- Le nombre de classes souhaité doit être spécifié au démarrage de l'algorithme, contrairement à la CAH.
- Choix de K : problème difficile
- Il est possible d'utiliser des distances autres que la distance euclidienne (par exemple la distance L_1 sera moins sensible aux outliers).
  $\rightarrow$ Voir plus loin l'algorithme des k-médoïdes
- En pratique l'algorithme converge assez rapidement (dans certains cas, une dizaine d'itérations suffit).

Illustration (dans l'espace des centres de classes)

Données : deux classes qui se chevauchent



Comportement de l'algorithme suivant l'initialisation



Liens avec la méthode de Ward

- L'algorithme des k-means possède des connections avec la méthode de Ward dans le sens où toutes les deux méthodes optimisent à leur manière l'inertie intra-classe.
- Plusieurs techniques ont été proposées pour combiner les deux algorithmes :
  - exécuter l'algorithme des K-means avec un nombre de classes d'environ 10% de n
  - appliquer la méthode de Ward aux centres des classes obtenues et déterminer le nombre de classes adéquat
  - exécuter l'algorithme des K-means en partant du nombre de classes et des moyennes de classes obtenues à l'étape précédente

Détermination du nombre de classes

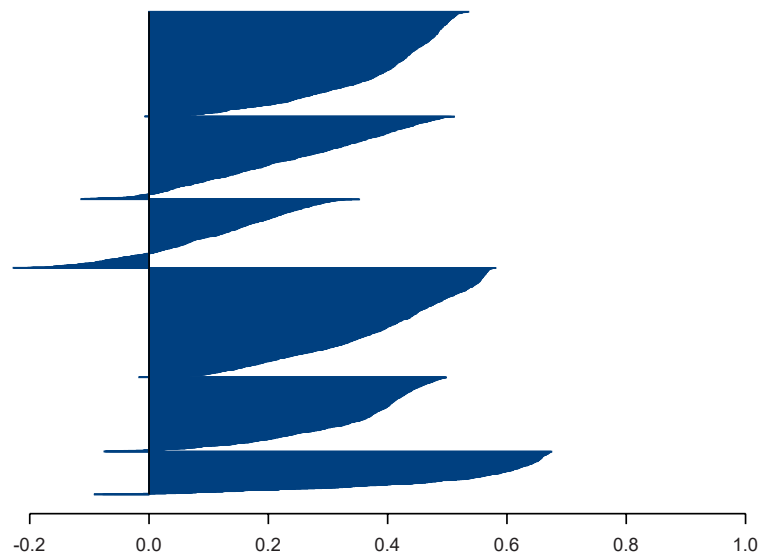
Méthode du coude (elbow)

Exécuter l'algorithme des k-means pour $K = 1, \ldots, K_{max}$, afficher le critère d'inertie intra-classe pour chaque nombre de classes, puis choisir le nombre K^* à partir duquel le critère d'inertie ne décroît plus de manière significative

Méthode de la silhouette

$$s_i(K) = \frac{b_i - a_i}{\max(a_i, b_i)}$$

- a_i : dissimilarité moyenne entre $\mathbf{x}_i$ et les autres points appartenant à la même classe que $\mathbf{x}_i$
- b_i : plus faible dissimilarité moyenne de $\mathbf{x}_i$ à chaque classe autre que celle à laquelle $\mathbf{x}_i$ appartient
- s_i proche de 1 $\Leftrightarrow$ $\mathbf{x}_i$ bien classifié
- La moyenne des s_i constitue un indicateur global pour le choix du nombre de classes



Détermination du nombre de classes

Minimisation du critère d'inertie intra-classe pénalisé

$$\mathcal{P}(K) = \underbrace{np\left(1 + 2\pi + \log\left(\frac{I_W}{np}\right)\right)}_{\text{log-vraisemblance}} + \underbrace{(Kp)\log(n)}_{\text{pénalité}}$$

avec

n et p : nombre de lignes et de colonnes du tableau de données

K le nombre de classe, et I_W : le critère d'inertie intra-classe

Version séquentielle des k-means (online k-means)

Méthode permettant de classifier les données de manière séquentielle

- utile lorsque les données ne sont pas toutes disponibles en même temps
- utile pour traiter rapidement de grandes quantités de données

Algorithme

- Choix aléatoire de K centres $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
- Pour chaque nouveau point $\mathbf{x}_i$
  - calculer le centre le plus proche de $\mathbf{x}_i$
    soit $\boldsymbol{\mu}_k$ ce centre et n_k l'effectif de sa classe associée
  - mettre à jour le centre $\boldsymbol{\mu}_k$ et l'effectif n_k

$$\begin{aligned} \boldsymbol{\mu}_k &\leftarrow \boldsymbol{\mu}_k + \frac{1}{n_k + 1}(\mathbf{x}_i - \boldsymbol{\mu}_k) \\ n_k &\leftarrow n_k + 1 \end{aligned}$$

k-means séquentiel ou online k-means

L'algorithme online k-means est un algorithme dit de gradient stochastique dont la forme canonique est

$$\boldsymbol{\mu}_k^{(i)} = \boldsymbol{\mu}_k^{(i-1)} + \varepsilon_i(\mathbf{x}_i - \boldsymbol{\mu}_k^{(i-1)})$$

avec $\boldsymbol{\mu}_k^{(i)}$ le centre de la classes obtenu à l'itération i et ε_i le pas vérifiant

$$\sum_{i=1}^{\infty} \varepsilon_i = \infty \quad \sum_{i=1}^{\infty} \varepsilon_i^2 < \infty$$

Par exemple la suite (ε_i) telle que $\varepsilon_i = \frac{1}{i}$ vérifie cette propriété

La convergence de cet algorithme est assurée quand le nombre de données n devient très grand ($i \to \infty$).

Mise en œuvre des k-means sous R

```
# Lancer des k-means pour K=2 classes
KM <- kmeans(data, 2, nstart = 20)

# Résultats
summary(KM)

# Centres des classes
KM$centers

# Classe de chaque objet
KM$cluster

# Inertie intra-classe
KM$tot.withinss
```

Version parallèle/distribuée de l'algorithme des k-means

Motivations

- L'algorithme des k-means nécessite de calculer, à chaque itération, $n \times K$ distances entre les données $\mathbf{x}_i$ et les centres $\boldsymbol{\mu}_k$
- Les distances à calculer sont indépendantes les unes des autres ; leur calcul peut donc être effectué en parallèle

Approche MapReduce

- Découpage des données en blocs
- Map : exécution, en parallèle sur les blocs, de l'étape de partitionnement
- Reduce : calcul des centres à partir des partitions locales obtenues dans l'étape Map

Version parallèle/distribuée de l'algorithme des k-means

Algorithme

1. Découpage de l'ensemble E en B blocs $(E_1, \ldots, E_b, \ldots, E_B)$
2. ***Initialisation*** : tirage au hasard de K points de E qui forment les centres initiaux des K classes
3. ***Tant que les classes ne sont pas stabilisées***
   (a) Map : pour chaque bloc E_b, construction d'une partition $(E_{1b}, \ldots, E_{kb}, \ldots, E_{Kb})$ en affectant chaque point de E_b à la classe dont il est le plus près du centre de gravité

$$\begin{aligned} S_{kb} &= \sum_{\mathbf{x}_i \in E_{kb}} \mathbf{x}_i \qquad \forall k, b \\ N_{kb} &= card(E_{kb}) \qquad \forall k, b \end{aligned}$$

   (b) Reduce : calcul des centres de gravité

$$\boldsymbol{\mu}_k = \frac{\sum_{b=1}^{B} S_{kb}}{\sum_{b=1}^{B} N_{kb}} \qquad \forall k = 1, \ldots, K$$

Méthode des k-médoïdes

Principe

- Appliquer l'algorithme des k-means en remplaçant les centres de gravité par des médoïdes
- Le médoïde $\mathbf{x}_{i_k}$ de la classe P_k est l'élément le plus central de la classe, défini par

$$\mathbf{x}_{i_k} = \arg \min_{\mathbf{x}_i \in P_k} \sum_{\mathbf{x}_j \in P_k} \| \mathbf{x}_i - \mathbf{x}_j \|$$

Remarque : la norme $\| \mathbf{x}_i - \mathbf{x}_j \|$ peut être remplacée par une dissimilarité

Critère optimisé

$$C(P, \widetilde{\mathbf{x}}) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in P_k} \| \mathbf{x}_i - \mathbf{x}_{i_k} \|,$$

où $\widetilde{\mathbf{x}} = (\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_K})$ est l'ensemble des médoïdes des classes.

Méthode des k-médoïdes

Algorithme partitioning around medoids (PAM)

1. ***Initialisation*** : tirage aléatoire de K points de E qui forment les médoïdes initiaux
2. ***Tant que les classes ne sont pas stabilisées***
   - (a) Calcul de la partition : affectation de chaque point non médoïde de E à la classe dont il est le plus proche du médoïde
   - (b) Mise à jour des médoïdes pour chaque classe
     - (1) tirage aléatoire d'un point non médoïde dans la classe
     - (2) permutation de ce point avec le médoïde le plus proche si cela fait décroître le critère C

Méthode des k-médoïdes

Remarques sur la méthode des k-médoïdes

- Algorithme similaire à celui des k-means : médoïdes remplacés par les moyennes
- Méthode plus robuste en présence d'outliers (critère s'appuyant sur une distance L_1)
- Inconvénient : temps de calcul pouvant être élevés si le nombre d'observations est grand
- Variantes permettant de réduire le temps de calcul
  - CLARA (Clustering LARge Applications)
  - CLARANS (Clustering LARge applications upon RANdomized Search)

Mise en œuvre de l'algorithme PAM sous R

```
library(cluster)
data(iris)
quant = iris[,1:4]
species = as.numeric(iris[,5])

# Lancer PAM pour K=2 classes
quant.pam = pam(quant,2)

# Résultats
summary(quant.pam)

# Médoïdes des classes
quant.pam$medoids

# Classes
quant.pam$clustering
```

Fuzzy k-means (version « floue » des k-means)

Rappel sur la notion de classification floue

Les degrés d'appartenance binaires sont remplacés par des degrés d'appartenance flous

$$\mathbf{z} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{c} = \begin{pmatrix} 0.3 & 0.6 & 0.1 \\ 0.8 & 0.1 & 0.1 \\ 0 & 0 & 1 \\ 0.4 & 0.5 & 0.1 \\ 0.2 & 0.1 & 0.7 \end{pmatrix}$$

$z_{ik} \in \{0;1\}$ et $\sum_{k=1}^{K} z_{ik} = 1$ $\qquad$ $c_{ik} \in [0;1]$, $\sum_{k=1}^{K} c_{ik} = 1$ et $\sum_{i=1}^{n} c_{ik} > 0$

Critère

L'algorithme fuzzy k-means détermine une partition floue $\mathbf{c}$ qui minimise le critère :

$$C(\mathbf{c}) = \sum_{k=1}^{K} \sum_{i=1}^{n} c_{ik}^{\alpha} \parallel \mathbf{x}_i - \mathbf{g}_k \parallel^2 \quad \text{avec} \quad \mathbf{g}_k = \frac{1}{\sum_{i=1}^{n} c_{ik}^{\alpha}} \sum_{i=1}^{n} c_{ik}^{\alpha} \mathbf{x}_i$$

où $\alpha > 1$ est un coefficient qui règle le degré de flou

Fuzzy k-means

Algorithme

1. Initialiser la matrice de partition floue $\mathbf{c}$
2. Itérer les étapes suivantes jusqu'à la convergence :
   - Calcul des centres :

$$\forall k \quad \mathbf{g}_k = \frac{1}{\sum_{i=1}^{n} c_{ik}^{\alpha}} \sum_{i=1}^{n} c_{ik}^{\alpha} \mathbf{x}_i$$

   - Calcul de la partition floue :

$$\forall i,k \quad c_{ik} = \frac{\left(\frac{1}{\|\mathbf{x}_i - \mathbf{g}_k\|^2}\right)^{\frac{1}{\alpha-1}}}{\sum_{\ell=1}^{K} \left(\frac{1}{\|\mathbf{x}_i - \mathbf{g}_\ell\|^2}\right)^{\frac{1}{\alpha-1}}}$$

Mise en œuvre de Fuzzy k-means sous R

```
library(cluster)

x = rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5)),
          cbind(rnorm(3,3.2,0.5),rnorm(3,3.2,0.5)),
          cbind(rnorm(15,5,0.5),rnorm(15,5,0.5)))

clust_flou = fanny(x,2,memb.exp=2)

clust_flou

plot(x,col=clust_flou$clustering)

# Degrés d'appartenance des données aux classes
matplot(clust_flou$membership,type="l",lty=1)
```

Méthode des nuées dynamiques (généralisation des k-means)

Principe

On remplace les centres $\boldsymbol{\mu}_k$ qui étaient des points de $\mathbb{R}^p$ dans l'algorithme des k-means par d'autres formes de représentants de classes appelées aussi noyaux. Ces noyaux peuvent être de natures variées selon le problème à résoudre.

Critère optimisé

Si l'on note $\mathcal{L}$ l'ensemble des noyaux et $D : E \times \mathcal{L} \to \mathbb{R}^+$ une mesure de ressemblance, l'objectif est alors de trouver la partition P qui minimise

$$C(P, L) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in P_k} D(\mathbf{x}_i, \lambda_k)$$

où $L = (\lambda_1, \ldots, \lambda_K)$ est un ensemble de K noyaux, avec $\lambda_k \in \mathcal{L}$.

Méthode des nuées dynamiques

Comme pour les centres mobiles, on utilise un algorithme d'optimisation alternée qui définit la suite :

$$L^{(0)} \to P^{(1)} \to L^{(1)} \to P^{(2)} \to L^{(2)} \ldots \to L^{(n)} \to P^{(n+1)} \to L^{(n+1)} \ldots$$

Algorithme

1. Calcul de la partition $P^{(n+1)}$ qui minimise $C(P, L^{(n)})$ par rapport à P
2. Calcul des noyaux $L^{(n+1)}$ qui minimisent $C(P^{(n+1)}, L)$ par rapport à L

La première étape est identique à celle de l'algorithme k-means

L'existence de cet algorithme ne dépendra que de celle de la seconde étape.

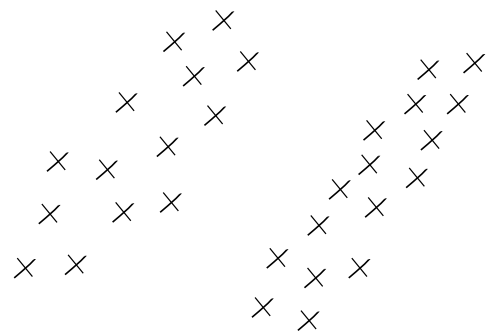
Méthode des nuées dynamiques

Exemple de noyau

$$D\Big(\mathbf{x}_i, \underbrace{(\boldsymbol{\mu}_k, M_k)}_{\lambda_k}\Big) = (\mathbf{x}_i - \boldsymbol{\mu}_k)^T M_k (\mathbf{x}_i - \boldsymbol{\mu}_k) - \log(|M_k|)$$

où M_k est une matrice symétrique définie positive et $|M_k|$ est son déterminant.

Ce noyau, associé aux distances quadratiques sur $\mathbb{R}^p$, permet de prendre en compte différentes configurations de classes



Si on impose $M_k = I$, on retrouve l'algorithme des k-means.

Classification de données qualitatives

Méthode des k-modes

- Procédure similaire aux k-means
- Utilisation de la distance de Hamming

$$d_H(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{p} \delta(x^j, y^j)$$

- Avec cette distance, les moyennes des classes deviennent leurs modes

Mode d'une classe

$$\begin{aligned} \mathbf{m}_k &= \arg\min_{\mathbf{a}} \sum_{\mathbf{x}_i \in P_k} d_H(\mathbf{x}_i, \mathbf{a}) \\ &= \text{vecteur des modalités les plus fréquentes de chaque variable} \end{aligned}$$

Exemple : mode$\big([a,\ b],\ [a,\ c],\ [c,\ b]\big) = [a,\ b]$

Algorithme des k-modes

Critère minimisé

$$C(P, \mathbf{m}) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in P_k} d_H(\mathbf{x}_i, \mathbf{m}_k)$$

avec $P = (P_1, \ldots, P_K)$ et $\mathbf{m} = (\mathbf{m}_1, \ldots, \mathbf{m}_K)$

Algorithme

- Initialisation : choix des K modes initiaux
- Tant que les classes ne sont pas stabilisées
  - (a) Construction d'une partition en affectant chaque point au mode le plus proche selon la distance de Hamming
  - (b) Mise à jour des modes

Classification de variables

Objectifs

Déterminer un nombre réduit de variables permettant d'expliquer l'ensemble des variables initiales

- Données textuelles : classification de documents à partir d'une matrice terme-document
- Données génomiques : classification de gènes décrits sous différentes conditions

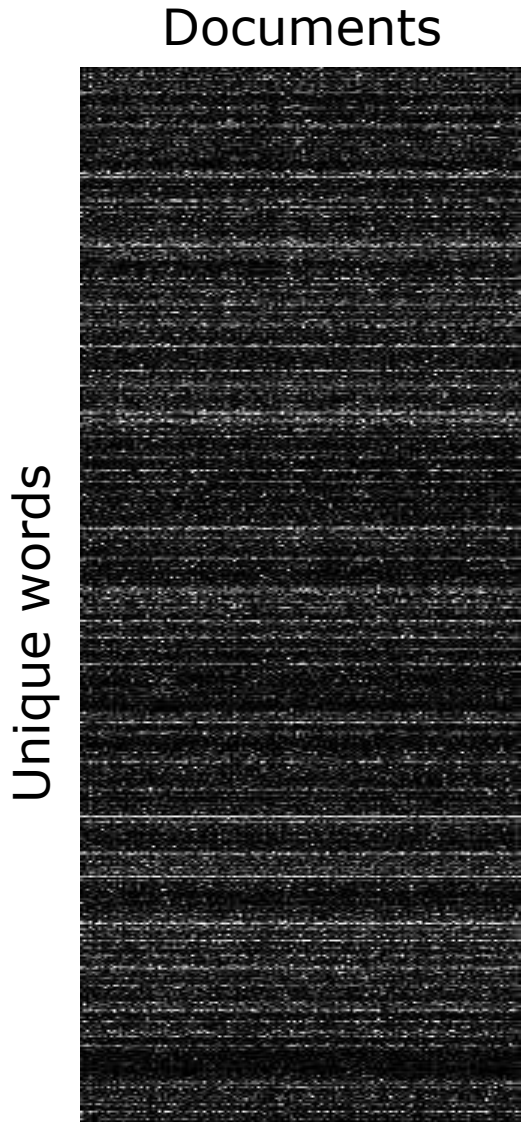
Méthode

Rechercher des clusters constitués de **variables corrélées**

1) Choix d'une mesure de similarité/dissimilarité
2) Mise en œuvre d'un algorithme de classification automatique (ex. CAH, k-means)

Exemple

Données : fréquence de 9275 mots dans 2431 documents (Dhillon,2001)



Classification de variables

On suppose que les variables sont centrées, normées et que $x_i^j \geq 0$

Mesure de similarité

Coefficient de corrélation entre les variables $\mathbf{x}^j$ et $\mathbf{x}^{j'}$

$$s(\mathbf{x}^j, \mathbf{x}^{j'}) = \sum_{i=1}^{n} x_i^j x_i^{j'}$$

Remarque

- $s(\mathbf{x}^j, \mathbf{x}^{j'}) =$ produit scalaire entre $\mathbf{x}^j$ et $\mathbf{x}^{j'}$
- $s(\mathbf{x}^j, \mathbf{x}^{j'}) = \cos(\mathbf{x}^j, \mathbf{x}^{j'})$

Mesure de dissimilarité

$$D(\mathbf{x}^j, \mathbf{x}^{j'}) = 1 - s(\mathbf{x}^j, \mathbf{x}^{j'})$$

Remarque

- $D(\mathbf{x}^j, \mathbf{x}^{j'}) = \frac{1}{2} \sum_i (\mathbf{x}_i^j - \mathbf{x}_i^{j'})^2$ (D est donc une distance euclidienne)

Classification de variables

Cas général de données numériques avec $x_i^j \in \mathbb{R}$

Mesure de similarité

Valeur absolue du coefficient de corrélation entre les variables

$$s(\mathbf{x}^j, \mathbf{x}^{j'}) = \left| \sum_{i=1}^{n} x_i^j x_i^{j'} \right|$$

Mesure de dissimilarité

- $D(\mathbf{x}^j, \mathbf{x}^{j'}) = 1 - s(\mathbf{x}^j, \mathbf{x}^{j'})$
- $D(\mathbf{x}^j, \mathbf{x}^{j'}) = \arccos(s(\mathbf{x}^j, \mathbf{x}^{j'}))$

Algorithme spherical k-means

Objectifs

Recherche d'une partition $\mathbf{Q} = (Q_1, \ldots, Q_L)$ d'un ensemble de p variables $(\mathbf{x}^1, \ldots, \mathbf{x}^j, \ldots, \mathbf{x}^p)$ en L classes

Critère maximisé

$$\mathcal{C}(\mathbf{Q}, \mathbf{c}) = \sum_{\ell=1}^{L} \sum_{\mathbf{x}^j \in Q_\ell} {\mathbf{x}^j}^T \mathbf{c}_\ell \quad \text{ou encore} \quad \mathcal{C}(\mathbf{w}, \mathbf{c}) = \sum_{j=1}^{p} {\mathbf{x}^j}^T \mathbf{c}_{w_j}$$

avec

$$\mathbf{w} = (w_j)_{j=1,\ldots,p} \quad \text{où} \quad w_j \in \{1, \ldots, L\}$$

et

$$\mathbf{c}_\ell = \frac{\boldsymbol{\mu}_\ell}{\|\boldsymbol{\mu}_\ell\|} \quad \text{où} \quad \boldsymbol{\mu}_\ell = \frac{\sum_{\mathbf{x}^j \in Q_\ell} \mathbf{x}^j}{\#Q_\ell}$$

(La norme $\|.\|$ est ici associée à la distance euclidienne)

Algorithme spherical k-means

1. **Initialisation**
   - Choix aléatoire d'une partition $\mathbf{Q}^{(0)} = (Q_1^{(0)}, \ldots, Q_L^{(0)})$
   - Calculer les centres (normalisés) des classes $\mathbf{c}^{(0)} = (\mathbf{c}_\ell^{(0)})$
2. ***Tant que l'algorithme n'a pas convergé***

   (a) Mise à jour de la partition : affecter chaque variable à la classe dont elle est la plus corrélée du centre

$$\mathbf{x}^j \in Q_\ell^{(t+1)} \quad \Longleftrightarrow \quad \ell = \arg \max_{1 \leq h \leq L} \left({\mathbf{x}^j}^T \mathbf{c}_h^{(t)} \right)$$

   (b) Mise à jour des centres :

$$\begin{aligned} \boldsymbol{\mu}_\ell^{(t+1)} &= \frac{\sum_{\mathbf{x}^j \in Q_\ell^{(t+1)}} \mathbf{x}^j}{\#Q_\ell^{(t+1)}} \\ \mathbf{c}_\ell^{(t+1)} &= \boldsymbol{\mu}_\ell^{(t+1)} / \| \boldsymbol{\mu}_\ell^{(t+1)} \| \end{aligned}$$

Convergence de l'algorithme spherical k-means

Proposition 1

Chaque étape de l'algorithme fait croître le critère $\mathcal{C}$:

(i) $\mathcal{C}(Q^{(t+1)}, \mathbf{c}^{(t+1)}) \geq \mathcal{C}(Q^{(t+1)}, \mathbf{c}^{(t)})$

(ii) $\mathcal{C}(Q^{(t+1)}, \mathbf{c}^{(t)}) \geq \mathcal{C}(Q^{(t)}, \mathbf{c}^{(t)})$

Proposition 2

La suite des critères $(\mathcal{C}(Q^{(t)}, \mathbf{c}^{(t)}))_{t \geq 0}$ converge en un nombre fini d'itérations.

Classification croisée ou co-clustering

Objectifs

- Trouver simultanément une partition des individus et une partition des variables
- Résumer un tableau de données par des blocs homogènes

Données

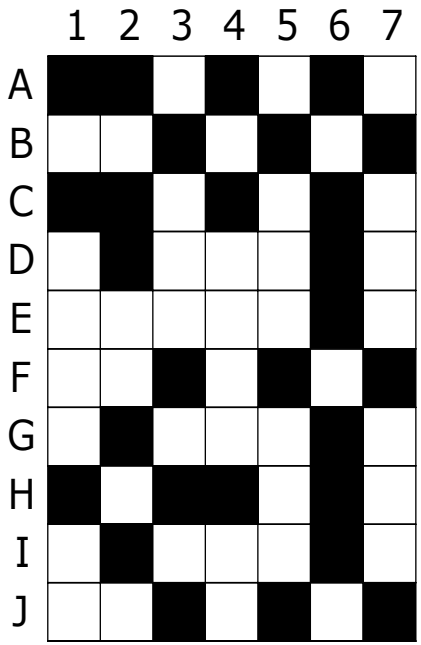
$$\mathbf{X} = \{x_i^j \ ; \ i = 1, \ldots, n, \ j = 1, \ldots, p\}$$

i : indice associé aux lignes

j : indice associé aux colonnes

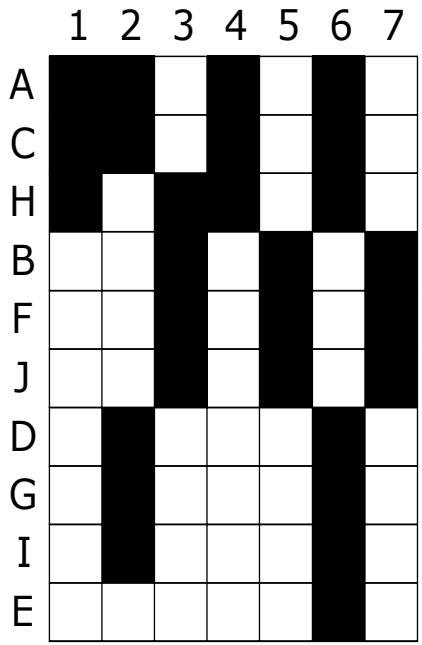
Exemple

Données



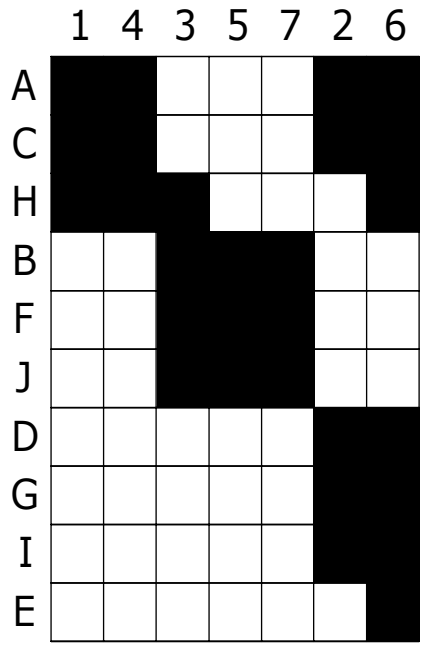
(1)

Réorganisation des lignes



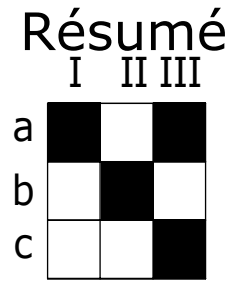
(2)

Réorganisation des colonnes



(3)

Résumé



(4)

Classification croisée de données continues

$$\mathbf{P} = (P_1, \ldots, P_K) : \quad \text{Partition des lignes}$$
$$\mathbf{Q} = (Q_1, \ldots, Q_L) : \quad \text{Partition des colonnes}$$
$$\mathbf{A} = (a_{k\ell})_{k\ell} : \quad \text{prototypes (centres) des blocs}$$

Critère optimisé

$$\mathcal{C}(\mathbf{P}, \mathbf{Q}, \mathbf{A}) = \sum_{k=1}^{K} \sum_{\ell=1}^{L} \sum_{i \in P_k} \sum_{j \in Q_\ell} (x_i^j - a_{k\ell})^2$$

Algorithme - CroEuc (Govaert, 1983 ; Govaert et Nadif, 2013)

1. *Initialisation* : $\mathbf{P}^{(0)}$, $\mathbf{Q}^{(0)}$, $\mathbf{A}^{(0)}$
2. *Répéter jusqu'à la convergence*
   (a) $(\mathbf{P}^{(t+1)}, \tilde{\mathbf{A}}^{(t+1)})$: Classification avec l'algorithme k-means des lignes du tableau $\mathbf{U} = (u_i^\ell)$ avec $u_i^\ell = \sum_{j \in Q_\ell^{(t)}} x_i^j / \#Q_\ell^{(t)}$
   (b) $(\mathbf{Q}^{(t+1)}, \mathbf{A}^{(t+1)})$: Classification avec l'algorithme k-means des colonnes du tableau $\mathbf{V} = (v_k^j)$ avec $v_k^j = \sum_{i \in P_k^{(t+1)}} x_i^j / \#P_k^{(t+1)}$

Exemple d'application du co-clustering

Tableau terme-document 9275×2431 (fréquences de mots) classifié en 2×2 blocs ; sujets abordés dans les documents : médecine et aéronautique