Apprentissage non supervisé
Chapitre 2 : Méthodes par partitionnement

Master "Machine Learning for Data Science", Paris V

Allou Samé
allou.same@ifsttar.fr

2017/2018

Plan

Données

n individus décrits par p variables

$\mathbf{X} =$

	var 1	...	var j	...	var p
indiv $\mathbf{x}_1$	x_{11}	...	x_{1j}	...	x_{1p}
indiv $\mathbf{x}_i$	x_{i1}	...	x_{ij}	...	x_{ip}
indiv $\mathbf{x}_n$	x_{n1}	...	x_{nj}	...	x_{np}

Algorithme des centres mobiles ou k-means

Objectif

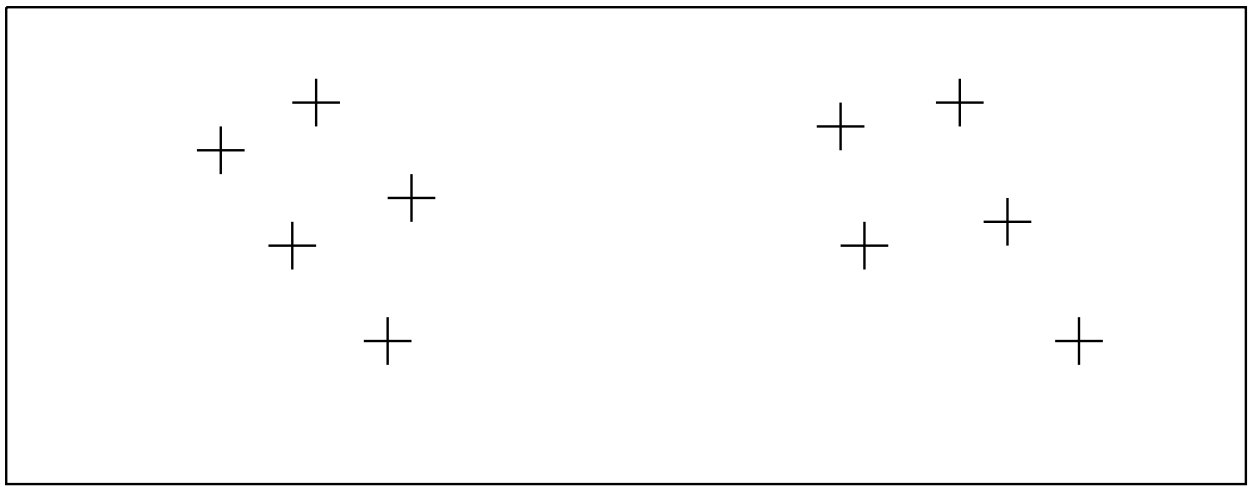
Partitionner l'ensemble E en K classes. On suppose ici que l'ensemble E est muni de la distance euclidienne.

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 de la partition qui vient d'être calculée ; ceux-ci deviennent les nouveaux centres

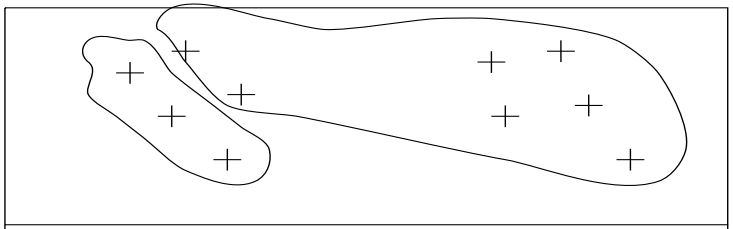
Exemple (1/2)

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

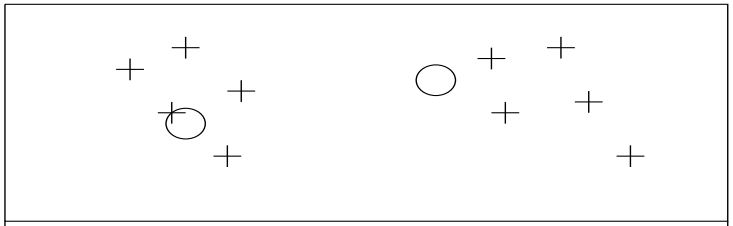


Exemple (2/2)

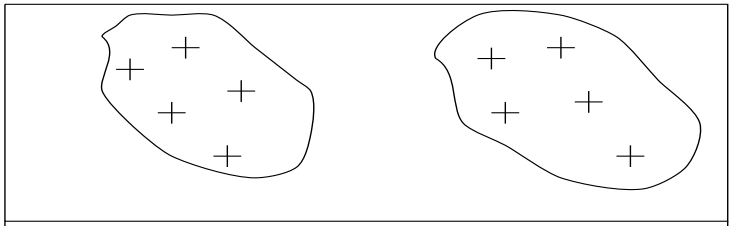
Etape 1 : choix de 2 points au hasard



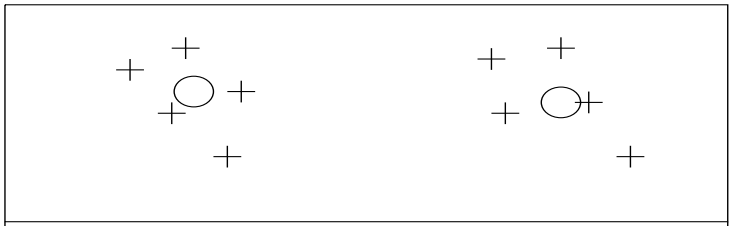
Etape 2 : affectation de chaque point au centre le plus proche



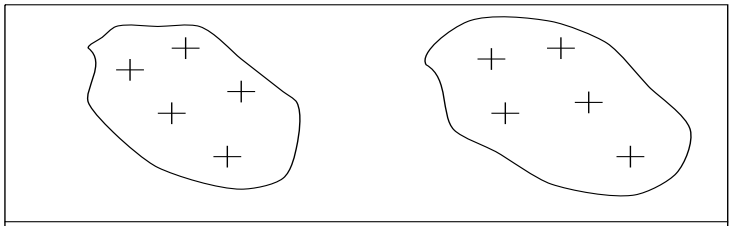
Etape 3 : calcul des centres de gravite



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

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

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 :
$\boldsymbol{\mu}^{(0)} \to P^{(1)} \to \boldsymbol{\mu}^{(1)} \to P^{(2)} \to \boldsymbol{\mu}^{(2)} \ldots \to \boldsymbol{\mu}^{(c)} \to P^{(c+1)} \to \boldsymbol{\mu}^{(c+1)} \ldots$

Proposition 1

Chaque étape de l'algorithme améliore le critère C :

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

Corollaire 1

La suite numérique $\left(C(P^{(c)}, \boldsymbol{\mu}^{(c)})\right)$ est stationnaire.

Proposition 2

La suite $(P^{(c)}, \boldsymbol{\mu}^{(c)})$ est stationnaire.

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

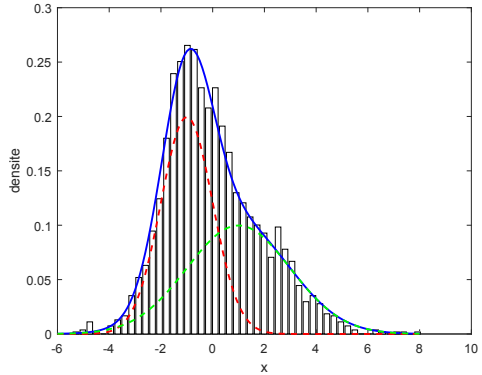
Remarques

- Le nombre de classes souhaité doit être spécifié au démarrage de l'algorithme, contrairement à la CAH.
- Choix de K : problème difficile

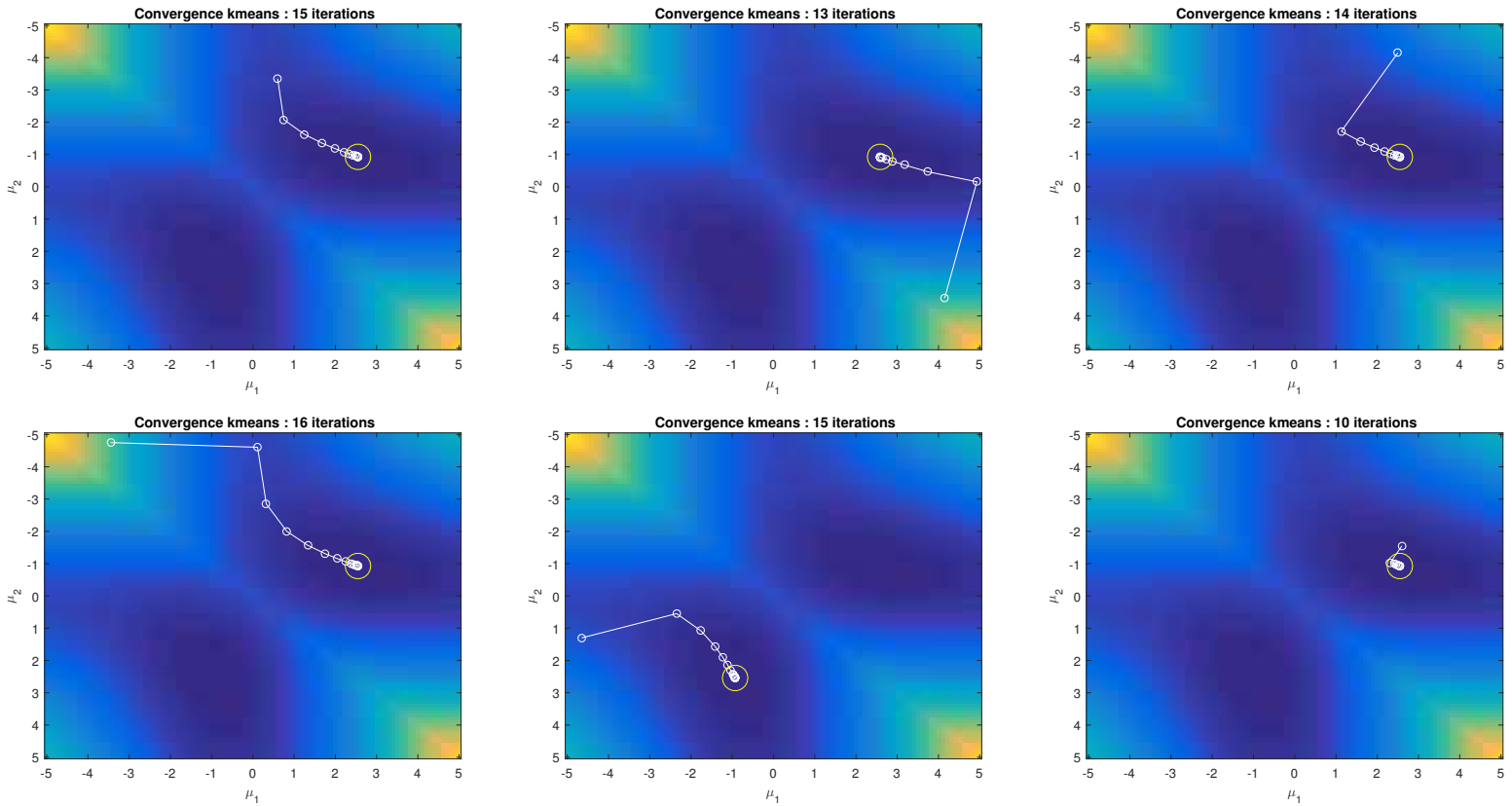
  Une solution consiste à exécuter l'algorithme pour $K = 1, \ldots, K_{max}$ puis à choisir le nombre de classes à partir duquel le critère ne décroît plus de manière significative (on parle de méthode du coude (elbow)).
- Il est possible d'utiliser des distances autres que la distance euclidienne (par exemple la distance L_1 sera moins sensible aux outliers).
- En pratique l'algorithme converge assez rapidement (dans certains cas 10 itérations suffisent).

Illustration du problème des minima locaux

Données : deux classes gaussiennes très mélangées



Comportement de l'algorithme suivant l'initialisation



Liens avec la méthode de Ward

- L'algorithme des k-means possède des connection 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

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 des flux de données

Algorithme

- Choix aléatoire de K centres $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
- Dès qu'un nouveau point $\mathbf{x}_i$ est disponible :
  - 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

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

avec

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

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)

# Résultats
summary (KM)

# Centres des classes
KM$centers

# Classe de chaque objet
KM$cluster

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

Parallélisation de l'algorithme des k-means

Motivations

- Traitement de données volumineuses (big-data)
- 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

- Modèle de programmation initié par Google et basé sur la parallélisation de calculs sur plusieurs machines
- Adaptation de l'algorithme des k-means :
  - 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 précédente

Parallélisation de l'algorithme des k-means

k-means MapReduce

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 $d(\mathbf{x}_i, \mathbf{x}_j)$

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
     - (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

- Méthode similaire à celle des k-means : k-médoïdes remplacés par k-moyennes
- Méthode plus robuste en présence d'outliers (critère s'appuyant sur une somme de distances au lieu d'une somme de carrés de distances)
- Inconvénient : temps de calcul pouvant être élevés si le nombre d'observations est grand
- Variantes permettant de réduite 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 $\sum_{k=1}^{K} z_{ik} = 1$ $\qquad$ $c_{ik} \in [0;1]$ et $\sum_{k=1}^{K} c_{ik} = 1$

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 (version “floue” des 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 :

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

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.

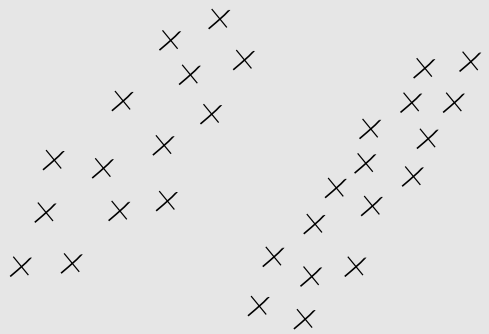
Méthode des nuées dynamiques

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.