Mathématiques Numériques MN1

Chapitre 4: Introduction à la classification non-hiérarchique

Bruno Pinçon /J.-F. Scheid

Télécom Nancy 1A

2025-2026

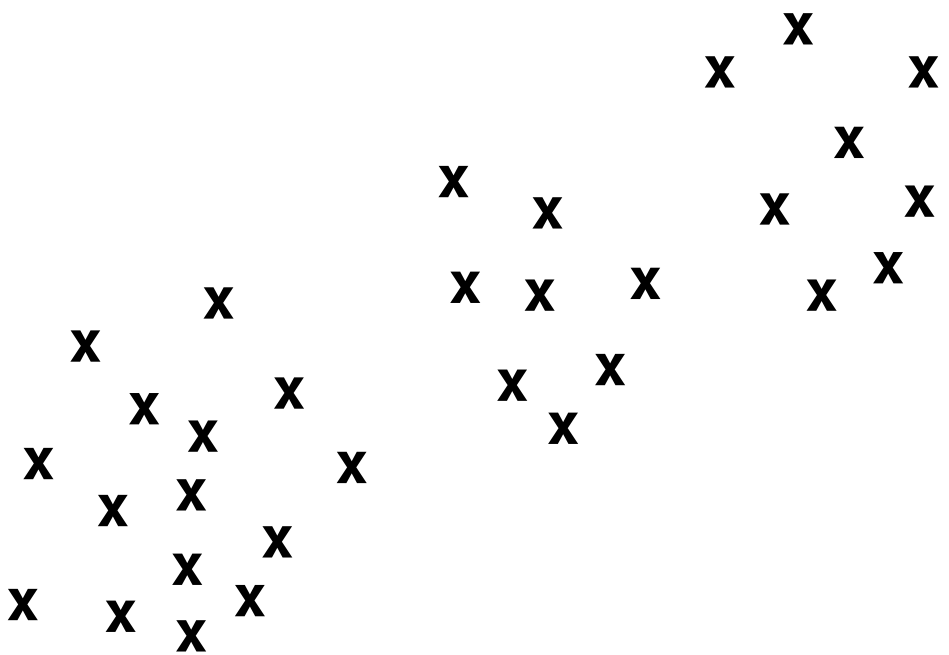
Version du 2/4/2025

I. Introduction et notations

Objectif des méthodes de classification: construire une partition (non-hiérarchique) ou une suite de partitions emboitées (hiérarchique) d'un ensemble d'objets (individus).

Problème de classification non-hiérarchique

Partitionner n objets en K classes/groupes les plus homogènes possibles ($K \ll n$).

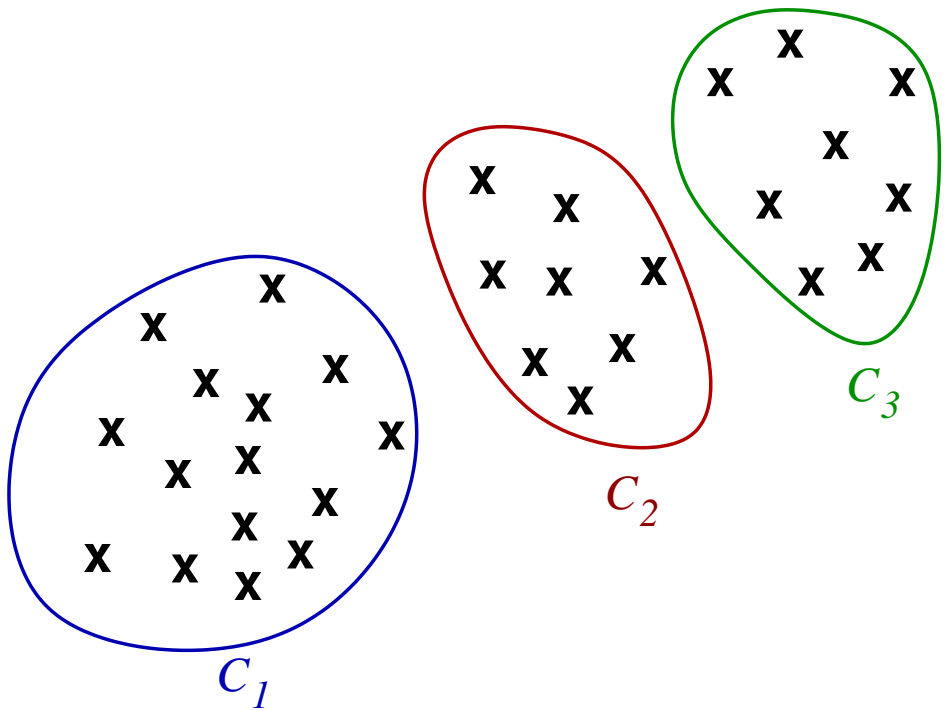


I. Introduction et notations

Objectif des méthodes de classification: construire une partition (non-hiérarchique) ou une suite de partitions emboitées (hiérarchique) d'un ensemble d'objets (individus).

Problème de classification non-hiérarchique

Partitionner n objets en K classes/groupes les plus homogènes possibles ($K \ll n$).



- **Objet/individu**. Il y a *p caractères* par individu. Un individu est donc caractérisé par un vecteur de $\mathbb{R}^p$. On notera

$$x_i \in \mathbb{R}^p, \quad i = 1, \cdots, n$$

les n objets à partitionner. Ainsi,

x_i^j correspond au caractère j de l'individu i.

☛ On stocke généralement les p caractères des n individus dans un tableau (matrice) X de taille $n \times p$.

- **Partition** : on notera

$$\Pi = \{C_1, C_2, \ldots, C_K\}$$

une partition des n objets en K classes (sous-ensembles) C_1, C_2, $\ldots$, C_K. Les classes sont aussi appelées **cluster**.

Remarque. Le nombre de classes K est ici **connu**. Il existe des techniques pour déterminer le nombre (pertinent) de classes (cf. dernier paragraphe).

II. Préparation des données

En général, il est nécessaire d'harmoniser les valeurs numériques de chaque caractère de sorte qu'elles soient comparables.

Par exemple, prenons ces données :

	caractères	
individus	c_1	c_2
x_1	1007	0.1
x_2	798	0.23
x_3	810	0.6
x_4	1030	0.65
$\bar{c}_j$	911.25	0.395
$\hat{\sigma}_j$	124.3	0.27

$d(x_i, x_j)$	x_1	x_2	x_3	x_4
x_1	0	209	197	23
x_2	209	0	12	232
x_3	197	12	0	220
x_4	23	232	220	0

☛*Il est clair que la classification se fera essentiellement sur le caractère c_1 puisque ses valeurs sont beaucoup plus grandes que celle du caractère c_2. Pourtant il y a des variations importantes dans le caractère c_2 !*

Une possibilité est de **centrer et réduire les données** : pour chaque colonne (c'est à dire pour chaque variable/caractère c_j) :

1. On calcule la moyenne et l'écart type :

$$\bar{c}_j = \frac{1}{n}\sum_{i=1}^{n} x_i^j, \quad \hat{\sigma}_j = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i^j - \bar{c}_j)^2}$$

2. On "centre et réduit" :

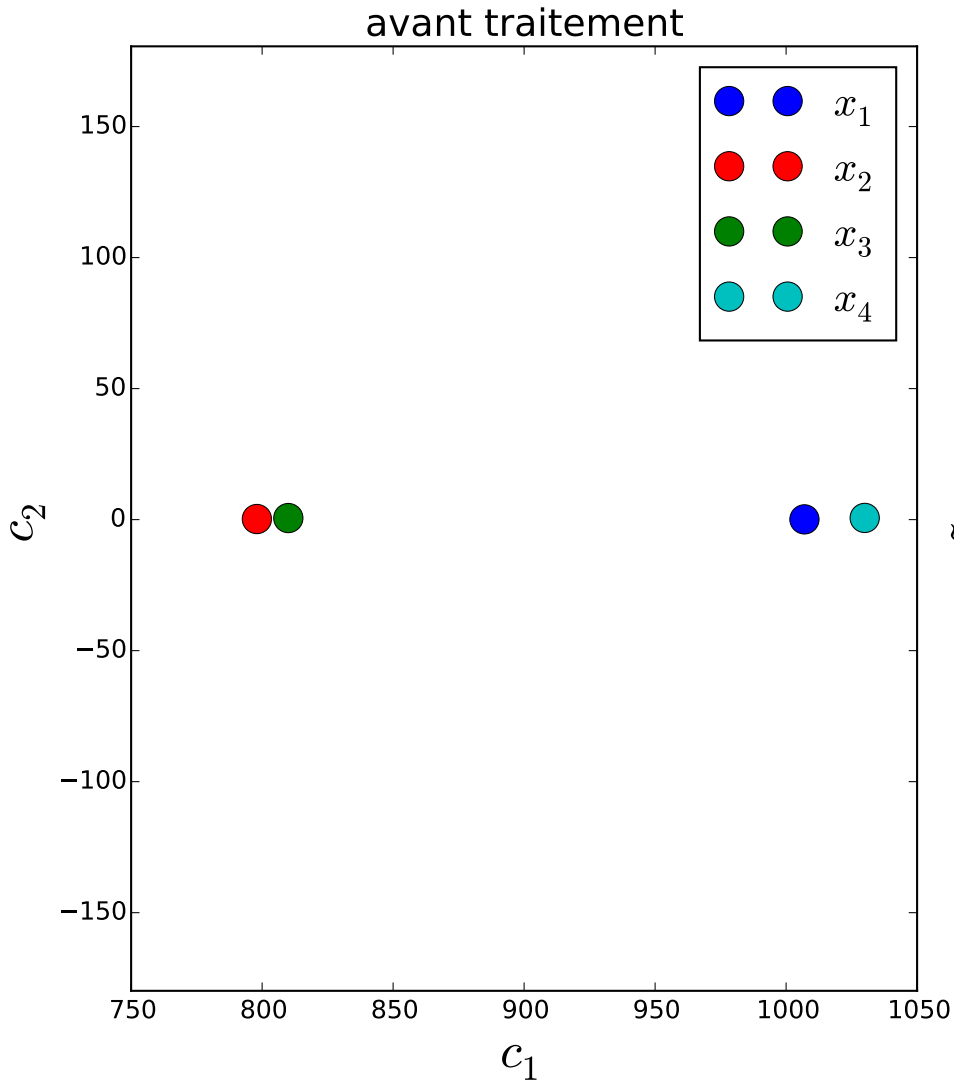
$$\tilde{x}_i^j = \frac{x_i^j - \bar{c}_j}{\hat{\sigma}_j}, i \in [\![1, n]\!]$$

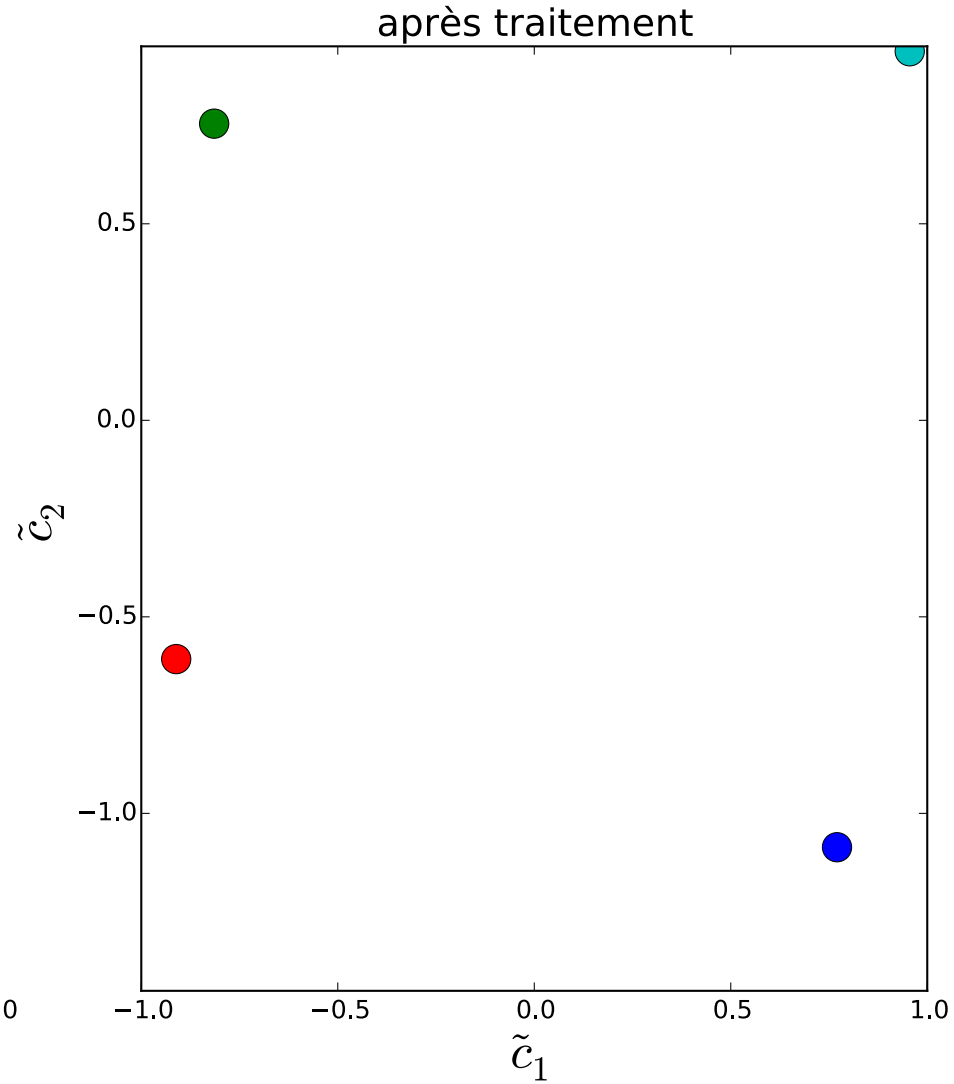
Sur les données précédentes, on obtient :

	$\tilde{c}_1$	$\tilde{c}_2$
$\tilde{x}_1$	0.770	−1.086
$\tilde{x}_2$	−0.911	−0.607
$\tilde{x}_3$	−0.814	0.755
$\tilde{x}_4$	0.955	0.939

$d(\tilde{x}_i, \tilde{x}_j)$	$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$	$\tilde{x}_4$
$\tilde{x}_1$	0	1.75	2.42	2.03
$\tilde{x}_2$	1.75	0	1.37	2.42
$\tilde{x}_3$	2.42	1.37	0	1.78
$\tilde{x}_4$	2.03	2.42	1.78	0

Graphiquement :





III. Barycentre, Inertie(s)

Plusieurs quantités importantes interviennent en classification.

- **Distance** entre individus x_i et x_k:

$$d(x_i, x_k) = \|x_i - x_k\| = \sqrt{\sum_{j=1}^{p} \left(x_i^j - x_k^j\right)^2} \tag{1}$$

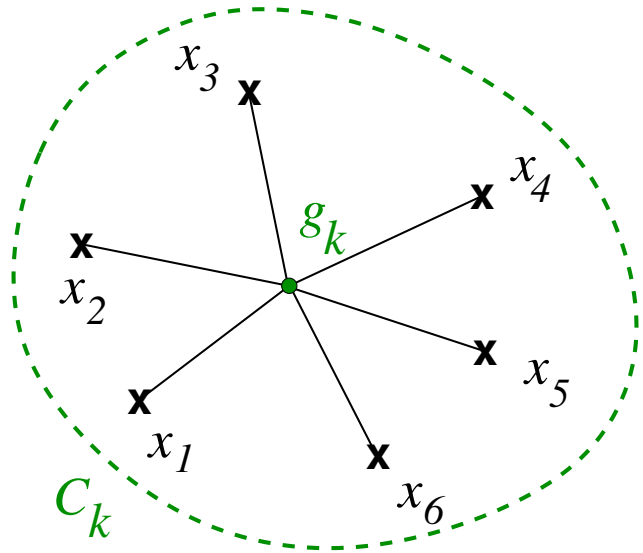
- **Barycentre** (moyenne) de la classe C_k :

$$g_k = \frac{1}{n_k} \sum_{x \in C_k} x \tag{2}$$

où $n_k = \text{card}(C_k)$.

- **Inertie** de la classe C_k par rapport à son barycentre (*variance*):

$$I_k = \sum_{x \in C_k} d(x, g_k)^2 = \sum_{x \in C_k} \|x - g_k\|^2 \tag{3}$$



- On définit l'inertie totale $I(\Pi)$ de la partition $\Pi = \{C_1, \ldots, C_K\}$, appelée aussi **inertie intra-classe** :

$$\boxed{I(\Pi) = \sum_{k=1}^{K} I_k = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - g_k\|^2} \tag{4}$$

- Plus généralement, on définit l'inertie d'un nuage de point $\{x_i\}_{i=1,\cdots,n}$ par rapport à un point quelconque $y \in \mathbb{R}^p$:

$$I(y) = \sum_{i=1}^{n} d(x_i, y)^2 = \sum_{i=1}^{n} \|x_i - y\|^2 \tag{5}$$

Théorème de Huygens

Pour tout $y \in \mathbb{R}^p$, on a

$$I(y) = I(g) + n\,\|y - g\|^2 \tag{6}$$

où g est le barycentre des $\{x_i\}_{i=1,\cdots,n}$.

Preuve.

$$\begin{aligned} I(y) &= \sum_{i=1}^{n} \|x_i - y\|^2 = \sum_{i} (x_i - y \,|\, x_i - y) \\ &= \sum_{i=1}^{n} (x_i - g + g - y \,|\, x_i - g + g - y) \\ &= \sum_{i=1}^{n} \left(\|x_i - g\|^2 + 2(x_i - g \,|\, g - y) + \|g - y\|^2 \right) \end{aligned}$$

$$I(y) = I(g) + n\|g-y\|^2 + 2\sum_{i=1}^{n}(x_i - g \,|\, g-y)$$

Or, $\sum_{i=1}^{n}(x_i - g \,|\, g-y) = (\underbrace{\sum_{i=1}^{n} x_i - ng}_{=0} \,|\, g-y) = 0$ d'où le résultat.

□

Conséquence du Théorème de Huygens.

L'inertie d'un nuage de points par rapport à son barycentre g est **minimale** :

$$I(g) \leq I(y), \quad \forall y \in \mathbb{R}^p.$$

Le Théorème de Huygens permet d'obtenir un résultat de développement de l'inertie totale $I(g)$ de n points par rapport au barycentre g en fonction de l'inertie intra-classe.

Théorème de König–Huygens

Soit les points $\{x_i\}_{i=1,\dots,n}$ et $\Pi = \{C_1, \cdots, C_K\}$ une partition de ces points avec les barycentres $g_1, \cdots, g_K$ des K classes. L'inertie totale I des n points autour du barycentre g vaut

$$I = I(g) = \underbrace{I(\Pi)}_{\text{inertie intra-classe}} + \underbrace{\sum_{k=1}^{K} n_k \|g_k - g\|^2}_{\text{inertie inter-classe}}$$

Remarque. L'inertie inter-classe est l'inertie du nuage des K barycentres.

Preuve du Théorème de König-Huygens.

Théorème de Huygens appliqué dans chaque classe $C_k \Rightarrow$

$$I_k(g) = I_k + n_k \|g_k - g\|^2 \tag{7}$$

L'inertie totale par rapport à g vaut

$$\begin{aligned} I = I(g) &= \sum_{i=1}^{n} \|x_i - g\|^2 = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - g\|^2 = \sum_{k=1}^{K} I_k(g) \\ &= \underbrace{\sum_{k} I_k}_{= I(\Pi)} + \sum_k n_k \|g_k - g\|^2 \quad \text{d'après (7)} \end{aligned}$$

□

Utilisation du Théorème de König-Huygens

Un critère usuel de classification consiste à chercher la partition Π telle que l'inertie **intra-classe** $I(\Pi)$ soit **minimale**.

Remarque. Le nombre K de partitions est **fixé**. Si K n'était pas fixé, alors la solution optimale serait obtenue avec $K = n$ classes (un individu = une classe) et dans ce cas $I(\Pi) = 0$.

IV. Considérations combinatoires

Le nombre de partitions possibles en K classes étant fini, on pourrait calculer l'inertie pour chaque partition et choisir la plus petite. Malheureusement, le nombre de partitions possibles est très grand ...

Soit $P_{n,k}$ le nombre de partitions en k classes d'un ensemble de n éléments.

On a $P_{n,1} = P_{n,n} = 1$ et $P_{n,2} = 2^{n-1} - 1$.

On peut montrer la relation de récurrence suivante :

$$P_{n,k} = P_{n-1,k-1} + kP_{n-1,k} \quad (1 < k < n)$$

$P_{12,3} \simeq 86500$
$P_{12,4} \simeq 611500$
$P_{13,3} \simeq 261600$
$P_{13,4} \simeq 2532500$

V. Algorithmes de type "nuées dynamiques"

Deux algorithmes : *k-means* et *h-means*.
h-means est une amélioration des *k-means* : convergence plus rapide en général et partition de meilleure qualité (inertie plus petite)

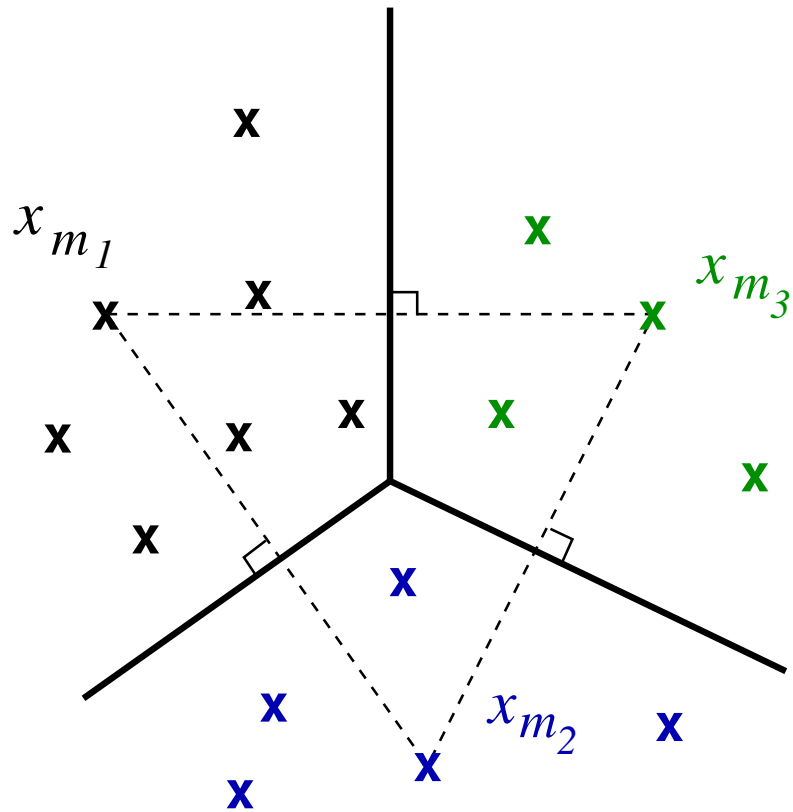
1. **Partition initiale**.

- On tire au hasard K nombres m_i, $i = 1, \cdots, K$, **tous distincts** dans l'ensemble $\{1, \cdots, n\}$.
- Les points x_{m_i} sont choisis comme "centres" des classes initiales.
  Partition initiale $\Pi^{(0)} = \{C_1^{(0)}, \cdots, C_K^{(0)}\}$.
  La classe $C_k^{(0)}$ est formée de tous les points les plus proches du centre x_{m_k} :

$$x_i \in C_k^{(0)} \Leftrightarrow \begin{array}{l} k \text{ est le plus petit entier tq} \\ \|x_i - x_{m_k}\| \leq \|x_i - x_{m_l}\|, \quad \forall l \in \{1, \cdots, K\} \end{array}$$

Remarques :

- En cas d'égalité de distance entre un point et plusieurs centres, le point est attribué à la classe de plus petit indice.
- Chaque classe[1] est non-vide car elle contient au moins x_{m_k}.



[1] pour $p = 2$ ou 3, les classes sont construites à partir des *cellules de Voronoï*.

2. Algorithme des *k-means*.

Idée : on remplace les K centres initiaux par les K barycentres des classes initiales. On détermine les nouvelles classes et on recommence jusqu'à convergence (i.e. les classes ne changent plus).

➔ 2 phases :

1. **Phase de barycentrage** : Pour une partition donnée, on calcule les barycentres de chaque classe.
2. **Phase d'affectation** : On boucle sur chaque point en l'affectant à la classe dont le barycentre est le plus proche

Remarque : Dans la phase d'affectation, on ne recalcule pas les barycentres quand un point change de classe.

Algorithme k-means

entrées : X, classe (la partition initiale)

sorties : classe (partition finale), I, ...

répéter

- phase de calcul des barycentres : $g_k, \quad k = 1, 2, ..., K$
- phase d'affectation des points :

  $nbcht \leftarrow 0; \quad I_k \leftarrow 0$ pour $k = 1, 2, ..., K$

  pour i de 1 à n

  $d2min \leftarrow \min_{1 \leq k \leq K} ||x_i - g_k||^2$

  $kmin \leftarrow \arg\min_{1 \leq k \leq K} ||x_i - g_k||^2$

  si $kmin \neq classe_i$ **alors** (le point a changé de classe)

  $classe_i \leftarrow kmin$; $nbcht \leftarrow nbcht + 1$

  fin si

  $I_{kmin} \leftarrow I_{kmin} + d2min$

  fin pour
- calcul de l'inertie totale : $I_{totale} \leftarrow \sum_{k=1}^{K} I_k$

jusqu'à ce que $nbcht = 0$

(si $nbcht = 0$, aucun point n'a changé de classe $\rightarrow$ stabilisation)

3. Convergence des *k-means*

Partition initiale $\Pi^{(0)}$ donnée.

Pour $m = 1, 2, ...$
$\mathcal{G}^{(m)}$ = barycentres des classes $\Pi^{(m-1)}$
$\Pi^{(m)}$ = partition obtenue après la phase de réaffectation

On considère la suite des inerties $I(\mathcal{G}^{(m)}, \Pi^{(m)})$.

On peut montrer que

$$I(\mathcal{G}^{(m)}, \Pi^{(m)}) \leq I(\mathcal{G}^{(m)}, \Pi^{(m-1)}) \leq I(\mathcal{G}^{(m-1)}, \Pi^{(m-1)})$$

La suite des inerties est décroissante et minorée (≥ 0) donc **convergente**.

L'ensemble des inerties est de cardinal fini (car l'ens. des partitions l'est)
$\Rightarrow$ la suite des inerties devient stationnaire à partir d'un certain rang
$\Rightarrow$ les centres ne bougent plus et les partitions non plus.

Remarques.

- La classification (clustering) par k-means est une classification **non-supervisée** : l'appartenance des objets/individus à l'une des K classes n'est pas connue (mais le nombre de classes K est fixé) et l'objectif est de construire les classes d'appartenance.
- A l'inverse, dans une classification **supervisée**, les classes sont prédéfinies ("chat", "chien", "lapin", "renard",...) et on veut déterminer l'appartenance des objets/individus (images, textes, caractéristiques numériques) à ces classes. Ces méthodes (apprentissage automatique) permettent de trouver une fonction qui associe les individus aux classes et permettent de prédire la classe d'une nouvelle donnée (par ex. régression logistique, SVM, réseaux de neurones...)

4. Algorithme des *h-means*.

Variante des *k-means* basée sur les remarques suivantes :

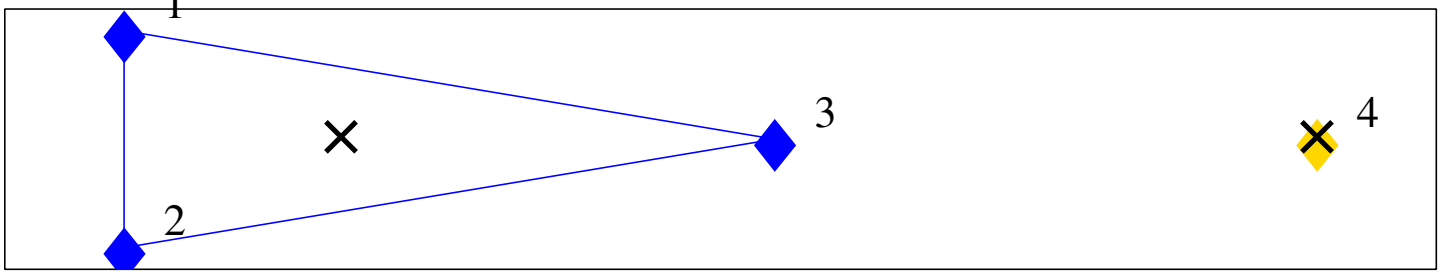
1. Quand on affecte un point à une autre classe, on ne modifie pas les barycentres.
2. Attribuer le point au centre le plus proche n'est pas toujours la meilleure façon de diminuer l'inertie.

Idée des *h-means* : Pour chaque point, on envisage tous les changements de classe possibles et on calcule l'inertie correspondante. On retient alors le changement de classe qui fournit l'inertie totale la plus petite.

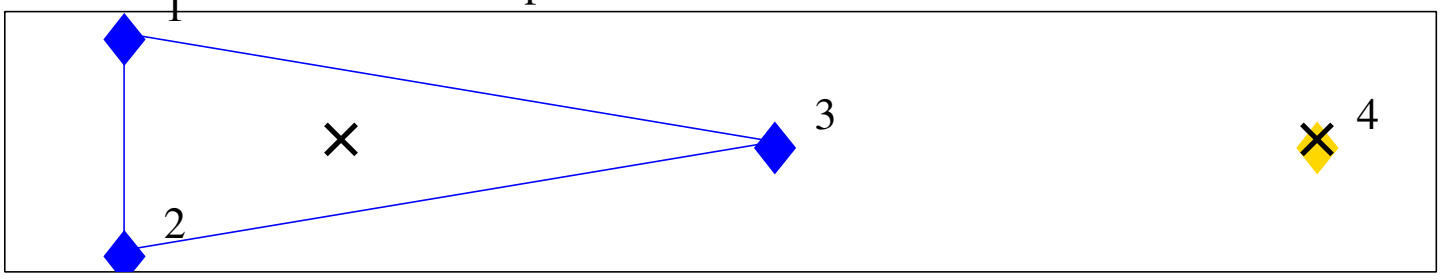
k-means vs h-means

Points 3 et 4 : centres initiaux

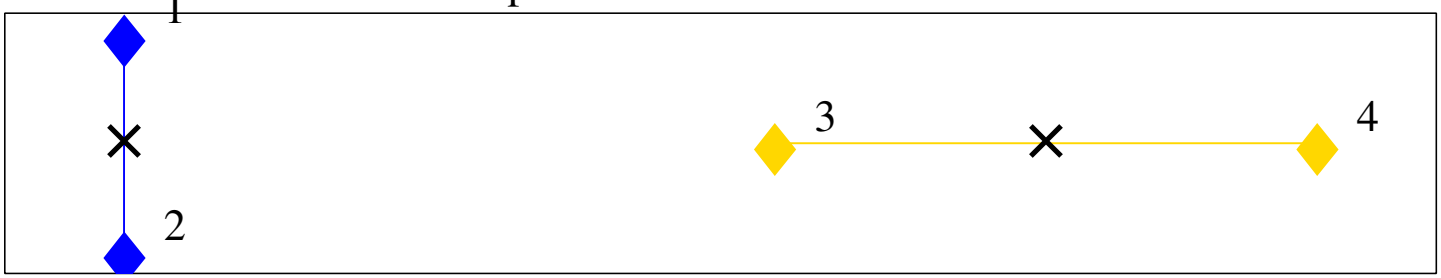
Partition initiale ---- Inertie = 26



k–means apres l'iteration 1 ---- Inertie = 26



h–means apres l'iteration 1 ---- Inertie = 14.5



Algorithme des *h-means*.

1. Partition initiale et calcul des barycentres
2. A chaque itération, on boucle sur tous les points.

   Soit x_i un point et $l = classe(i)$ le numéro de sa classe courante.

   On envisage le passage de x_i de la classe l à la classe $k \neq l$ en calculant la modification $c_{i,k}$ de l'inertie résultant de ce changement :

Inertie avant changement	➔	Inertie après changement
I		$I + c_{i,k}$

   On retient la classe $\bar{k}$ qui fournit l'inertie la plus petite :

   $$\bar{k} = \min\{k \mid c_{i,k} = \bar{c}_i\} \quad \text{où } \bar{c}_i = \min_{k \neq l} c_{i,k}.$$

   Si $\bar{c}_i < 0$ alors le point x_i passe de la classe l à la classe $\bar{k}$. Puis on met à jour les barycentres des 2 classes qui ont changées.
3. Il y a convergence quand il n'y a plus de changement de classe.

Remarques.

- Pour les *h-means*, il existe des formules rapides de mises à jour des $c_{i,k}$ et des inerties.
- L'algo. des *h-means* ne peut pas dégénérer i.e. aucune classe ne peut se vider, contrairement aux *k-means*.
- En général, l'algo. des *h-means* converge plus vite que celui des *k-means* et la partition finale de meilleure qualité (au sens de l'inertie).
- L'algo. des *h-means* peut améliorer une partition *k-means*, mais l'inverse est faux.
- On peut choisir d'autres distances que la distance euclidienne (distance de Manhattan $d(x_i, x_k) = \sum_j |x_i^j - x_k^j|$, d'autres distances pour le traitement d'images,...)

Exemples d'application.

1. Clustering/segmentation/compression d'image : réduire le nombre de couleurs dans une image pour diminuer sa taille sans trop perdre en qualité.
2. Détection d'anomalies (fraude bancaire et cybersécurité) : identifier les transactions suspectes dans un grand volume de données. Clustering des transactions selon des caractéristiques (montant, fréquence, lieu...). Puis détection des transactions qui ne rentrent dans aucun cluster bien défini : ces transactions peuvent être marquées comme potentiellement frauduleuses.
3. Recommandation de produits (E-commerce, Netflix, Spotify) : regrouper les utilisateurs ayant des goûts similaires pour leur proposer du contenu adapté. Clustering des utilisateurs aux préférences similaires. Si un utilisateur A est classé dans le même cluster que l'utilisateur B, alors on peut lui recommander des films préférés par B. Les recommandations sont dynamiques : si un utilisateur change ses préférences, il peut être réassigné à un autre cluster.

Clustering/segmentation d'image

Une image couleur est un ensemble de pixels (individus) qui ont essentiellement 5 caractères en mode RGB (d'autres représentations sont possibles) :

$$p = [x, y, r, g, b]$$

où (x, y) sont les coordonnées du pixel dans l'image, (r, g, b) les composantes de couleurs.

On peut utiliser la distance

$$d(p_1, p_2) = \lambda(x_1 - x_2)^2 + \lambda(y_1 - y_2)^2 + (r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2$$

avec $\lambda \in [0, 1]$ un paramètre à choisir.

☛ Segmentation = clustering des pixels en K classes.

Image originale 329×216

Image modifiée : $K = 10$, $\lambda = 0.0001$

Image modifiée : $K = 10$, $\lambda = 0.7$

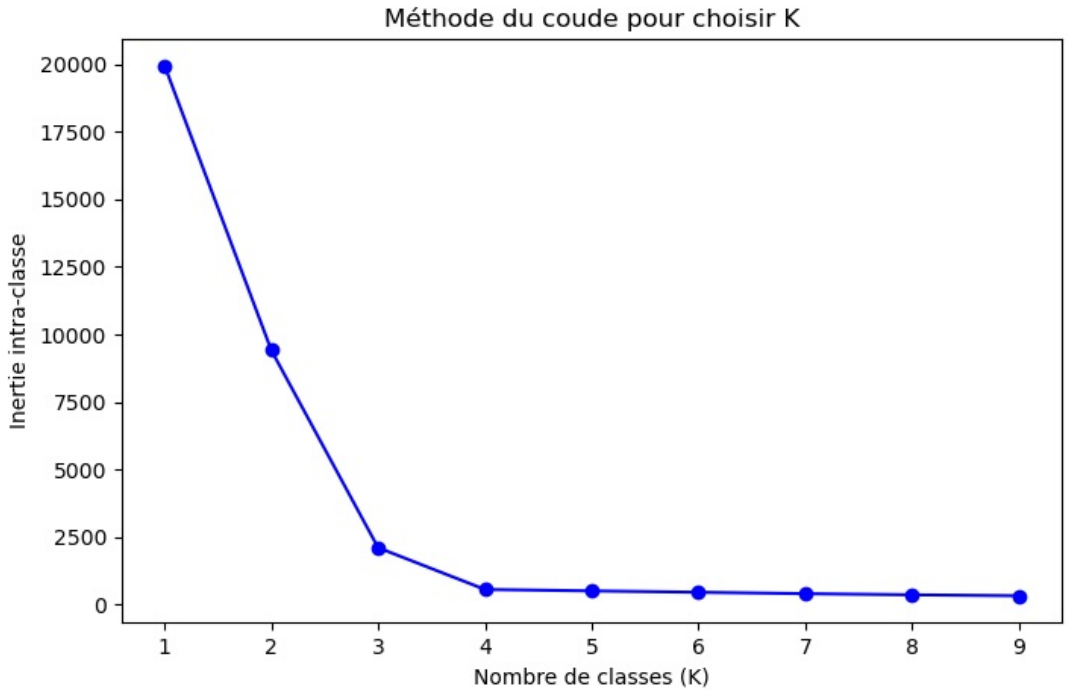
Image modifiée : $K = 16$, $\lambda = 0$

5. Choix du nombre de partitions.

a) La méthode du coude (Elbow method).

On fait varier K et on regarde l'évolution de l'inertie intra-classe $I(\Pi)$ avec $\Pi = \{C_1, \cdots, C_K\}$. *L'inertie intra-classe diminue toujours lorsque K augmente.*

On observe généralement une forte baisse de l'inertie jusqu'à un certain point d'inflexion. Après ce point, la réduction de l'inertie devient moins significative.



Ce point d'inflexion est appelé le "coude" et donne généralement une valeur correcte de K (dans l'exemple ci-dessus, on prendra $K = 4$).

Problèmes possibles : le coude n'est pas toujours évident et pour des données complexes, il n'est pas toujours facile d'identifier le bon K. Le *score silhouette* peut apporter une amélioration.

b) La méthode de la silhouette.

Cette méthode fournit un coefficient entre -1 (très mauvaise partition) et 1 (partition parfaite) qui mesure la qualité d'une partition. Pour chaque point x_i, dont la classe est k ($x_i \in C_k$), on définit son *score silhouette* par :

$$S_i := \frac{b_i - a_i}{\max\{a_i, b_i\}}$$

où

$$a_i := \frac{1}{n_k - 1} \sum_{x \in C_k \setminus \{x_i\}} d(x_i, x), \quad n_k = card(C_k)$$

est la distance moyenne entre x_i et les points de sa classe.

A partir de:

$$b_{i,k'} := \frac{1}{n_{k'}} \sum_{x \in C_{k'}} d(x_i, x), \quad n_{k'} = card(C_{k'})$$

qui est la distance moyenne entre x_i et une autre classe $k' \neq k$ que la sienne, on définit :

$$b_i := \min_{k' \neq k} b_{i,k'}$$

qui est la distance entre x_i et la classe voisine la plus proche.

On remarque que :

- si $a_i \ll b_i$ le point x_i est beaucoup plus proche (en moyenne) des points de sa classe que des points des autres classes, on obtient alors :

$$S_i = \frac{b_i - a_i}{b_i} = 1 - \frac{a_i}{b_i} \simeq 1 \quad (S_i \leq 1)$$

- si $b_i < a_i$ alors le point x_i est plus proche en moyenne des points d'une autre classe, il est mal placé ! Lorsque $b_i \ll a_i$, il vient :

$$S_i = \frac{b_i - a_i}{a_i} \simeq -1 \quad (S_i \geq -1)$$

Le score silhouette d'un point (qui est donc compris entre -1 et 1) est une indication numérique et objective de son bon ou mauvais placement.

Remarque. Lorsque son score est 0 le point est a priori aussi proche de sa classe que de la classe la plus proche.

Que faire avec les scores silhouettes ?

- Leur moyenne :

$$S_\Pi = \frac{1}{n} \sum_{i=1}^{n} S_i$$

qui donne un indicateur scalaire S_Π sur la qualité moyenne de la partition Π.

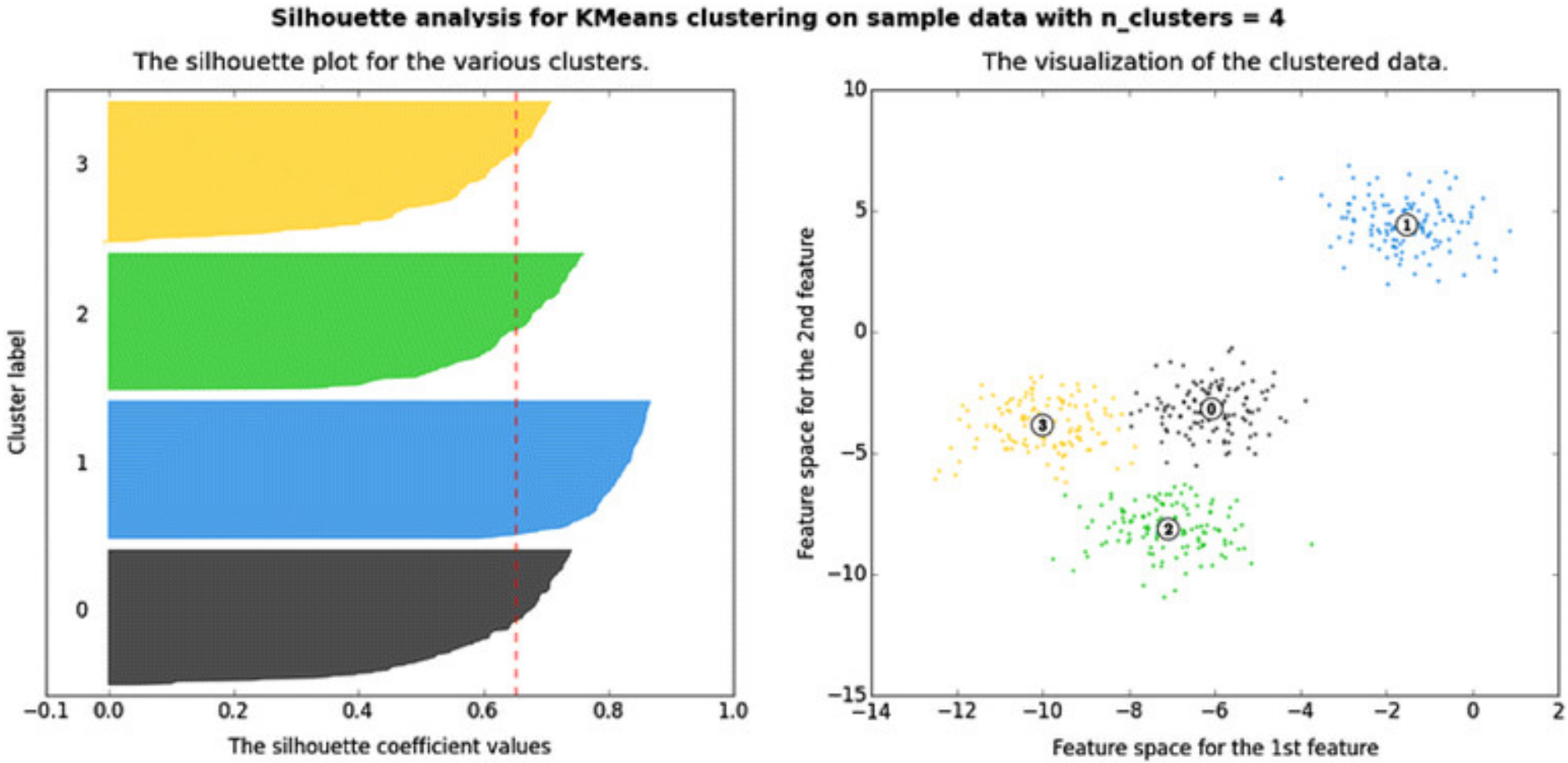
Étant donné K et une partition (quasi) optimale Π pour ce nombre de classes, on retiendra la valeur K^* qui donne le plus gros score silhouette S_Π.

- La moyenne par classe :

$$S_k = \frac{1}{n_k} \sum_{i:x_i \in C_k}^{n} S_i$$

  qui donne une indication plus précise que S_Π.

- Un graphique récapitulant tous les scores agencés d'une certaine manière.



Crédits illustration : Identification of Asthma Subtypes Using Clustering Methodologies, June 2016, Pulmonary Therapy 2(1), DOI: 10.1007/s41030-016-0017-z, LicenseCC BY-NC 4.0, M. Deliu, M. Sperrin, D. Belgrave, A. Custovic.