Mathématiques Numériques
introduction à la classification non hiérarchique
Telecom Nancy

Bruno Pinçon (I.E.C.L.) bruno.pincon@univ-lorraine.fr

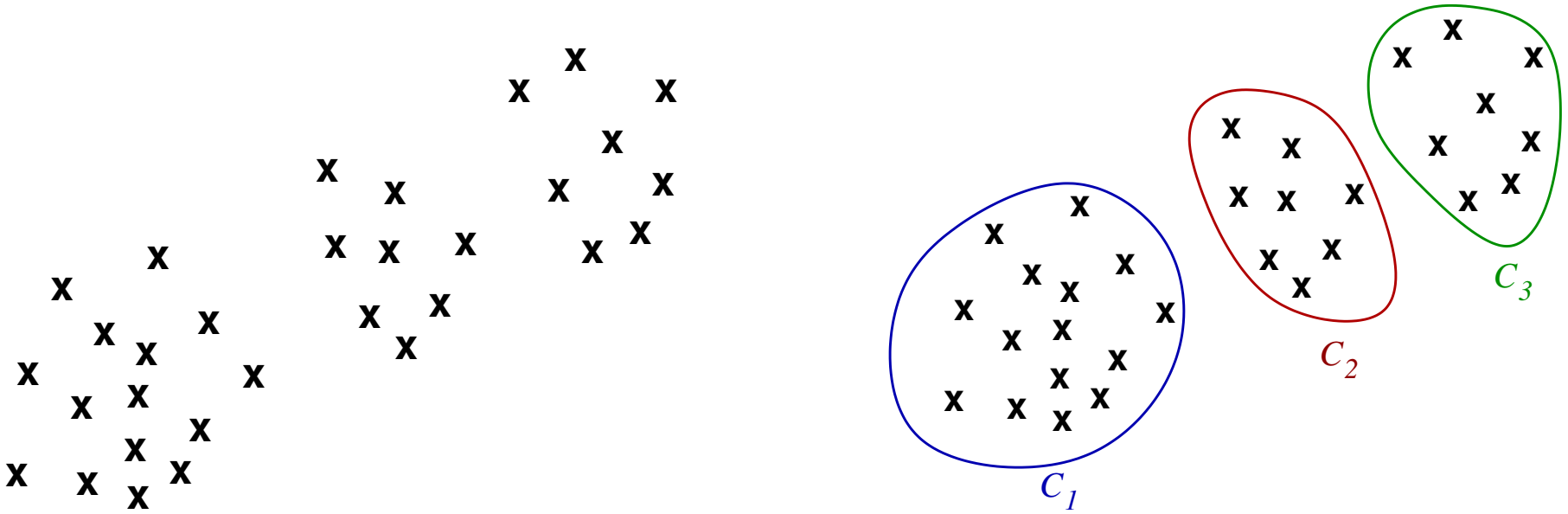
2021-2022

Introduction I

On considère des données (caractères) quantitatives portant sur n individus (p caractères par individu) et on cherche à partitionner ces n individus en K groupes (on dira classes dans la suite) les plus homogènes possibles avec (en général) $K << n$.
Un individu est donc caractérisé par un vecteur de $\mathbb{R}^p$ et la tradition est de stocker les p caractères des n individus dans un tableau X de taille $n \times p$, l'individu numéro i correspondant à la ligne i du tableau X, ce que l'on notera x_i (on utilisera parfois x pour désigner un individu sans préciser son numéro) :

num individu	c_1	.....	c_p
i	x_i^1	.....	x_i^p

Introduction II



Classification d'individus (à 2 caractères) en 3 groupes

Pour classer les individus en plusieurs groupes, on a besoin de définir une distance entre deux individus quelconques (c-a-d entre deux points de $\mathbb{R}^p$) et l'on choisira la distance euclidienne :

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

Introduction III

D'autres choix sont possibles (*), mais la distance euclidienne (en tout cas une distance issue d'un produit scalaire) est bien adaptée pour certaines raisons de simplicité mathématique et algorithmique.

(*) Les codes proposent généralement d'utiliser aussi des distances qui ne dérivent pas d'un produit scalaire comme la distance de Manhattan $d_m(x_i, x_k) = ||x_i - x_k||_1 = \sum_{j=1}^{p} |x_i^j - x_k^j|$.

Après classification (en supposant qu'elle soit satisfaisante) chaque groupe peut être représenté par un individu "moyen" (le barycentre du groupe) et ainsi l'information initiale constituée des n individus est alors résumée, après classification, par les K barycentres (plus le nombre d'individus par classe). Vu sous cet angle la classification peut donc être considérée comme un moyen de compresser de l'information (mais de façon destructive) en passant de $n \times p$ nombres à $K \times p$ nombres (plus K entiers pour le nombre d'individus par classe).

Introduction IV

Le but du jeu est donc de trouver une partition des n individus en K groupes. Soit $\Pi = \{C_1, C_2, ..., C_K\}$ une telle partition (chaque classe (sous-ensemble) C_k étant non vide), on adoptera les notations suivantes :

- g_k désignera le barycentre de la classe C_k :

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

où $n_k = \#C_k$ est le nombre d'éléments de C_k ;

- I_k désignera l'inertie de la classe C_k par rapport à son barycentre :

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

Introduction V

- $I(\Pi)$ désigne l'inertie totale associée à la partition Π :

$$I(\Pi) = \sum_{k=1}^{K} I_k$$

(remarque : les barycentres g_k et les inerties I_k dépendent bien sûr aussi du choix de la partition Π).
Avec ces notations, le problème de classer les n individus en K groupes, consiste à trouver une partition optimale Π^* qui réalise :

$$I(\Pi^*) \leq I(\Pi), \forall \Pi \in P_{ad} \quad (1)$$

où P_{ad} est l'ensemble des partitions admissibles, c'est à dire l'ensemble de toutes les partitions de $\{x_1, x_2, ..., x_n\}$ en K sous-ensembles non vides.

Introduction VI

Rmq : le plus souvent on ne connaît pas le bon nombre de classes K à utiliser ! Il n'y a d'ailleurs pas de réponses mathématiques complètement satisfaisantes à cette question : si on veut simplement minimiser I alors $K = n$ convient, chaque classe étant alors réduite à un point... De plus les algorithmes qui permettent de classifier en changeant (au cours du déroulement de l'algorithme) le nombre de groupes sont assez difficiles à paramétrer. Finalement il semble qu'une solution courante consiste à utiliser les algorithmes fonctionnant avec un nombre de classes fixé et de les essayer successivement avec différentes valeurs de K. Nous verrons deux critères qui permettent de trouver (suite aux résultats obtenus pour différentes valeurs de K) une valeur satisfaisante K^*.

Préparation des données I

En général il est nécessaire d'harmoniser les valeurs numériques de chaque caractère de sorte qu'elles soient comparables. Par exemple, prenons ces données :

id	c_1	c_2
1	1007	0.1
2	798	0.23
3	810	0.6
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 1 *puisque ses valeurs sont beaucoup plus grandes que celle du caractère* 2. *Pourtant il y a des variations importantes dans caractère* 2 *!*

Préparation des données II

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" :

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

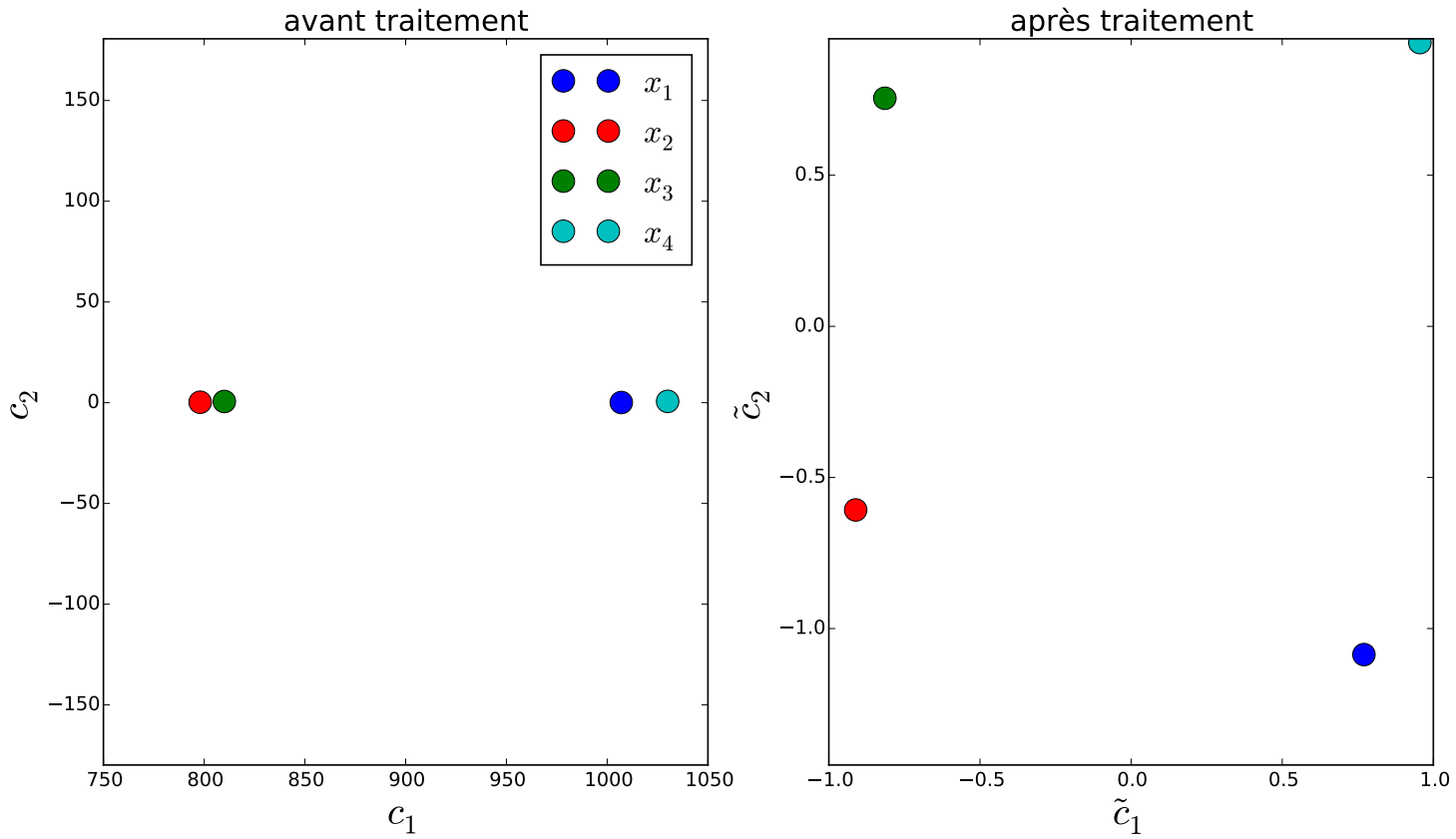
Sur les données précédentes, on obtient :

id	$\tilde{c}_1$	$\tilde{c}_2$
1	0.770	−1.086
2	−0.911	−0.607
3	−0.814	0.755
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

Préparation des données III

Graphiquement :



Préparation des données IV

Les moyennes et écarts types de chaque caractère peuvent être estimés sur une population plus grande : parmi les individus que l'on veut classer, l'un des caractères n'est peut-être pas pertinent ! Par exemple si dans la population totale le caractère c_i varie de 0 *à* 100 *avec une moyenne de* 50 *alors qu'il varie entre* 49 *et* 51 *dans la sous-population à étudier, on ne voudra pas apporter trop d'importance à ce caractère (ce qui sera le cas si on utilise la moyenne et l'écart type de la population totale et pas ceux de l'échantillon).*

Considérations générales I

L'ensemble P_{ad} étant de dimension finie, un algorithme évident pour trouver une partition optimale (il peut y en avoir plusieurs), consiste à calculer l'inertie $I(\Pi)$ pour chaque $\Pi \in P_{ad}$ puis à retenir la partition optimale :

$I_{opt} \leftarrow +\infty$
pour Π parcourant P_{ad}
    $I_\Pi \leftarrow$ calcul_inertie(Π)
    si $I_\Pi < I_{opt}$
        $I_{opt} \leftarrow I_\Pi$
        $\Pi_{opt} \leftarrow \Pi$

Problème : le nombre d'éléments de P_{ad} explose très rapidement !
Remarque : si le principe de l'algorithme est évident, les détails le sont moins... (génération des éléments de P_{ad} ?).

Considérations générales II

exercice : calcul du nombre d'éléments de P_{ad}

On appelle $P_{n,k}$ le nombre de partitions en k sous-ensembles non vides d'un ensemble E à n éléments et $\Pi_k(E)$ l'ensemble de toutes ces partitions ($P_{n,k} = \#\Pi_k(E)$). Montrer les résultats suivants (les deux premiers sont élémentaires) :

1. $P_{n,1} = 1$
2. $P_{n,n} = 1$
3. $P_{n,k} = P_{n-1,k-1} + kP_{n-1,k}$, $1 < k < n$. Aide : soit $e \in E$ un élément quelconque de E, on peut scinder $\Pi_k(E)$ en 2 sous-ensembles distincts :
   - Q_1 rassemblant les partitions dans lesquelles e apparait uniquement comme singleton (l'une des k classes de chaque partition est exactement $\{e\}$)
   - et Q_2 constitué de toutes les autres partitions (dans lesquelles e n'apparait jamais dans un singleton !).

Considérations générales III

Solution :

1. $P_{n,1} = 1$: en effet il est évident que $\Pi_1(E) = \{E\}$;
2. $P_{n,n} = 1$: E est partitionné en singletons et donc $\Pi_n(E) = \{\{\{e_1\}, \ldots, \{e_n\}\}\}$.
3.
   - $\#Q_1 = P_{n-1,k-1}$ puisque ses partitions peuvent être obtenues en prenant les partitions de $\Pi_{k-1}(E \setminus \{e\})$ et en leur rajoutant le singleton $\{e\}$.
   - L'ensemble Q_2 lui peut se construire de la façon suivante : on forme $\Pi_k(E \setminus \{e\})$ et, pour chaque partition de cet ensemble, il suffit de rajouter l'élément e dans l'une quelconque des k classes, soit k choix possible par partition. Ainsi $\#Q_2 = kP_{n,k-1}$.

   Finalement : $P_{n,k} = \#Q_1 + \#Q_2 = P_{n-1,k-1} + kP_{n-1,k}$.

Considérations générales IV

Un autre exercice serait d'écrire une fonction python qui utiliserait un algorithme en "triangle" (comme pour le triangle de Pascal) pour obtenir ces nombres. Voici un tableau donnant quelques valeurs des $P_{n,k}$ (qui sont appelés nombre de *Stirling* de 2 ème espèce) :

$n \setminus k$	2	3	4	5	6
7	63	301	350	140	21
8	127	966	1701	1050	266
9	255	3025	7770	6951	2646
10	511	9330	34105	42525	22827
11	1023	28501	145750	246730	179487
12	2047	86526	611501	1379400	1323652
13	4095	261625	2532530	7508501	9321312
14	8191	788970	10391745	40075035	63436373

Considérations générales V

Malgré cette explosion combinatoire, quelques algorithmes permettant de trouver l'optimum exact ont été mis au point mais ils restent cantonnés à des valeurs de n assez faibles.

Ainsi dans la plupart des cas pratiques, on est obligé de recourir à des algorithmes *heuristiques stochastiques*[1] qui ne permettent pas forcément de trouver la solution optimale.
Dans la pratique, la solution (en général sous optimale) est obtenue en une dizaine d'itérations, ce qui en fait des algorithmes très efficaces que l'on fait tourner plusieurs fois. L'aspect stochastique de ces algorithmes fait qu'une solution différente peut être trouvée. On augmente ainsi les chances d'obtenir une bonne partition du nuage de points.

1. Un algorithme est qualifié de stochastique lorsque le hasard y joue un rôle.

Alg. *H*-means, *K*-means : introduction

- L'algorithme des *K-means* est une amélioration de celui des *H-means*. Cependant il semble peu utilisé (cf rmq suivante).
- L'algorithme *H-means* est facilement vectorisable/parallélisable d'où sans doute sa prédominance sur celui des *K-means*.
- Généralement le nom *K-means* est donné à l'algorithme *H-means*...
- Ces deux algorithmes s'utilisent à partir d'une configuration (partition) initiale choisie aléatoirement.
- Aussi bien l'un que l'autre ne donnent pas, en général, la solution optimale, il faut les faire tourner plusieurs fois (cf transparent précédent) et retenir la meilleure partition obtenue (au sens de l'inertie).

H-means, K-means : choix de la partition initiale I

1. Tirer K nombres $m_{i_{(1 \leq i \leq K)}}$ au hasard tous distincts dans l'ensemble $\{1, 2, \ldots, n\}$. Si on dispose d'une fonction $U(a, b)$ permettant de tirer uniformément un entier entre a et b (compris), on peut utiliser l'algorithme évident :

   pour k de 1 **à** K
   répéter
   $u \leftarrow U(1, n)$
   jusqu'à ce que $u \notin \{m_1, ..., m_{k-1}\}$
   $m_k \leftarrow u$

   On obtient ainsi un tirage uniforme d'un sous-ensemble à K élément de $\{1, 2, \ldots, n\}$.

H-means, K-means : choix de la partition initiale II

2. Les points x_{m_i} sont choisis comme "centres" des classes initiales, la partition initiale $\Pi^{(0)} = \{C_1^{(0)}, ..., C_K^{(0)}\}$ étant obtenu de la façon suivante : la classe k est formée de tous les points qui lui sont les plus proches. En cas d'égalité de distance entre un point et plusieurs centres, le point est attribué à la classe de plus petit indice :

$$x_i \in C_k^{(0)} \iff \begin{cases} k \text{ est le plus petit entier tel que :} \\ ||x_i - x_{m_k}|| \leq ||x_i - x_{m_\ell}|| \ \forall \ell \in [\![1, K]\!] \end{cases}$$

Il est clair que chaque classe C_k est non vide car elle comprend au moins le point x_{m_k}.

Algorithme *H*-means I

Chaque itération est constituée des deux phases suivantes :

- *phase de barycentrage* : étant donné une partition, on calcule les barycentres de chaque classe ;
- *phase d'affectation* : on boucle sur les points (individus) en réaffectant chaque point à la classe dont le barycentre est le plus proche (avec affectation à la classe d'indice le plus petit si ambiguité). Cette phase modifie la partition de l'étape précédente.

Rmq :

- Ce processus est itéré jusqu'à stabilisation de la partition. Parfois, on choisit de ne pas aller jusqu'à la stabilisation complète en arrêtant les itérations lorsqu'on considère que l'inertie ne diminue plus beaucoup.

Algorithme *H*-means II

- Lors de la phase d'affectation, on ne recalcule pas les barycentres lorsqu'un point change de classe (ils deviennent donc faux...). Cette remarque (et une autre) est à l'origine de l'algorithme des *K-means*.

Algorithme *H-means*
entrées : X, classe (la partition initiale)
sorties : classe (partition finale), I, ...
répéter
    phase de calcul des barycentres
    phase d'affectation des points
jusqu'à "stabilisation" (ou "quasi-stabilisation")

Algorithme H-means III

Pour écrire plus précisément les deux étapes de l'algorithme, on va définir :

1. le tableau *classe* de taille n tel que $classe_i$ nous donne le numéro de la classe du point x_i ;
2. un tableau I de taille K donnant l'inertie de chaque classe ;
3. un tableau *ne* de taille K donnant le nombre d'éléments de chaque classe ;
4. un tableau G de taille $K \times p$, g_k servira à stocker le barycentre de la classe k ;
5. une variable *nbcht* donnant le nombre de points qui ont changé de classe lors de la phase de réaffectation.

L'algorithme donné (qui est relativement détaillé) n'est qu'une solution possible parmi d'autres.

Algorithme H-means IV

phase de calcul des barycentres :

pour k de 1 à K
  $g_k \leftarrow [0, ..., 0]$
  $ne_k \leftarrow 0$
pour i de 1 à n
  $k \leftarrow classe_i$
  $g_k \leftarrow g_k + x_i$
  $ne_k \leftarrow ne_k + 1$
pour k de 1 à K
  si $ne_k = 0$ **alors** *traiter l'exception* **fin si**
  $g_k \leftarrow g_k / ne_k$

Algorithme H-means V

phase d'affectation des points :

$nbcht \leftarrow 0$

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

pour i de 1 à n

$d2min \leftarrow +\infty$

pour k de 1 à K

$temp \leftarrow ||x_i - g_k||^2$

si $temp < d2min$ **alors**

$d2min \leftarrow temp$; $kmin \leftarrow k$

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

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

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

calcul de l'inertie totale :

$I_{totale} \leftarrow \sum_{k=1}^{K} I_k$

Algorithme H-means VI

De façon exceptionnelle, une classe ou plusieurs classes peuvent se "vider". Si l'on ne veut pas traiter ce cas comme une exception (arrêt du déroulement du programme, puis gestion de l'erreur ...), il est possible de continuer l'algorithme en réaffectant des points aux classes qui se sont vidées. Soit T le nombre de ces classes vides, alors on prend T points dans les autres classes (en privilégiant les classes avec une forte inertie) qui deviennent les nouveaux centres (et uniques points) de ces T classes.

Convergence de l'algorithme

On va l'établir sous l'hypothèse de non-dégénérescence, c-a-d que l'on supposera qu'aucune classe ne se vide. Par convergence, on sous-entend que la stabilisation de la partition est obtenue en un nombre fini d'itérations (et non que la solution obtenue est la (une) partition réalisant le minimum de l'inertie sur l'ensemble P_{ad}). On

Algorithme H-means VII

notera $I(\mathcal{C}, \Pi)$ l'inertie associée à un couple de centres $\mathcal{C}$ et une partition Π : ici on remarque bien que comme les barycentres ne sont pas mis à jour lors la phase d'affectation, ils sont simplement considérés comme les centres des classes. Avec cette notation, l'algorithme peut s'écrire de façon très abrégée :

$\Pi^{(0)}$ étant donnée
pour $m = 1, 2, ...$
$\mathcal{C}^{(m)} =$ barycentres des classes $\Pi^{(m-1)}$
$\Pi^{(m)} =$ partition obtenue après la phase de réaffectation

Algorithme H-means VIII

On considère alors la suite des inerties suivantes :

$$I_1 = I(\mathcal{C}^{(1)}, \Pi^{(0)}), I_2 = I(\mathcal{C}^{(1)}, \Pi^{(1)}), I_3 = I(\mathcal{C}^{(2)}, \Pi^{(1)}), I_4 = I(\mathcal{C}^{(2)}, \Pi^{(2)}), ..$$

où les inerties successives, $I_{2m-1} = I(\mathcal{C}^{(m)}, \Pi^{(m-1)})$ et $I_{2m} = I(\mathcal{C}^{(m)}, \Pi^{(m)})$ sont obtenues lors de l'itération m, suite, respectivement, au barycentrage puis à la réaffectation. Cette suite est décroissante. En effet, il est clair que :

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

puisque le changement d'un point d'une classe à l'autre se fait sur le critère du centre le plus proche (cf exercice A suivant). D'autre part, le barycentre d'un ensemble de points verifie (ici pour la classe k) :

$$\sum_{x \in C_k} ||x - g_k||^2 < \sum_{x \in C_k} ||x - y||^2 \ , \ \forall y \neq g_k$$

Algorithme H-means IX

(c'est le théorème de *Huygens*, cf exercice B suivant) et donc il est aussi clair que :

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

D'autre part comme cette suite est minorée (les inerties étant nécessairement positives) elle est donc convergente (suite décroissante minorée) : $\exists \bar{I} \in \mathbb{R}$ tel que $\lim_{k \to +\infty} I_k = \bar{I}$. En fait il y a mieux car on peut montrer que la suite devient stationnaire : $\exists \bar{k}$ tel que $I_k = \bar{I}$, $\forall k \geq \bar{k}$. En effet comme l'ensemble des partitions est de cardinal (élevé mais) fini, l'ensemble des centres envisagés par cet algorithme (qui sont à chaque fois les barycentres associés à une certaine partition) est aussi de cardinal fini et par conséquent l'ensemble des inerties $I(\Pi, \mathcal{C})$ est lui aussi de cardinal fini. Ainsi la suite que l'on envisage vit dans ce sous-ensemble fini de $\mathbb{R}$, et étant

Algorithme *H*-means X

monotone, elle devient effectivement stationnaire à partir d'un certain rang (cf exercice C). Si on prend m supérieur à ce rang alors :

$$I(\mathcal{C}^{(m-1)}, \Pi^{(m-1)}) = I(\mathcal{C}^{(m)}, \Pi^{(m-1)})$$

et l'inégalité caractérisant les barycentres montre alors que les centres n'ont pas changé ($\mathcal{C}^{(m)} = \mathcal{C}^{(m-1)}$) et comme les centres ne changent pas, les partitions non plus □.

cvg *H-means*, exercice A

Montrer que, lors de la phase de réaffectation, l'inertie diminue strictement si un point change de classe (on en déduit que dans tous les cas $I(\mathcal{C}^{(m)}, \Pi^{(m)}) \leq I(\mathcal{C}^{(m)}, \Pi^{(m-1)})$).

Algorithme *H*-means XI

cvg *H-means*, exercice B (théorème de Huygens)

Soient n vecteurs de $\mathbb{R}^p$ notés x_i, $i = 1, \cdots, n$. On désigne par $g = \frac{1}{n}\sum_{i=1}^{n} x_i$ le barycentre de ces points et on note $I(y) = \sum_{i=1}^{n} \|x_i - y\|^2$ l'inertie de ces points par rapport à un point $y \in \mathbb{R}^p$ donné. Montrer que :

$$I(y) = I(g) + n\|y - g\|^2, \qquad \text{pour tout } y \in \mathbb{R}^p.$$

Aide : dans $I(y)$ on peut développer les termes $\|x_i - y\|^2$ en utilisant $\|x_i - y\|^2 = \|(x_i - g) - (y - g)\|^2$ puis en développant cette norme au carré comme un produit scalaire.

Algorithme *H*-means XII

cvg *H-means*, exercice C

Montrer qu'une suite (x_k) monotone, telle que $\forall k \in \mathbb{N}, x_k \in E \subset \mathbb{R}$ avec $\#E < +\infty$ (E est un sous-ensemble fini de $\mathbb{R}$) est convergente et stationnaire à partir d'un certain rang :
$\exists n^* \in \mathbb{N} : x_n = x_{n^*}, \forall n \geq n^*$.

Choix du bon nombre de classes I

Soit K le nombre de classes imposé et Π^* la (une) partition optimale pour ce choix, c'est à dire telle que :

$$I(\Pi^*) \leq I(\Pi),\ \forall \Pi \text{ admissible}$$

Posons $I^*(K) = I(\Pi^*)$. Il est assez facile de montrer (lorsque $K < n$) que $I^*(K+1) < I^*(K)$. Au sens de l'inertie, le nombre optimal de classes K^* est égal à n, on a alors $I^*(K^*) = 0$!

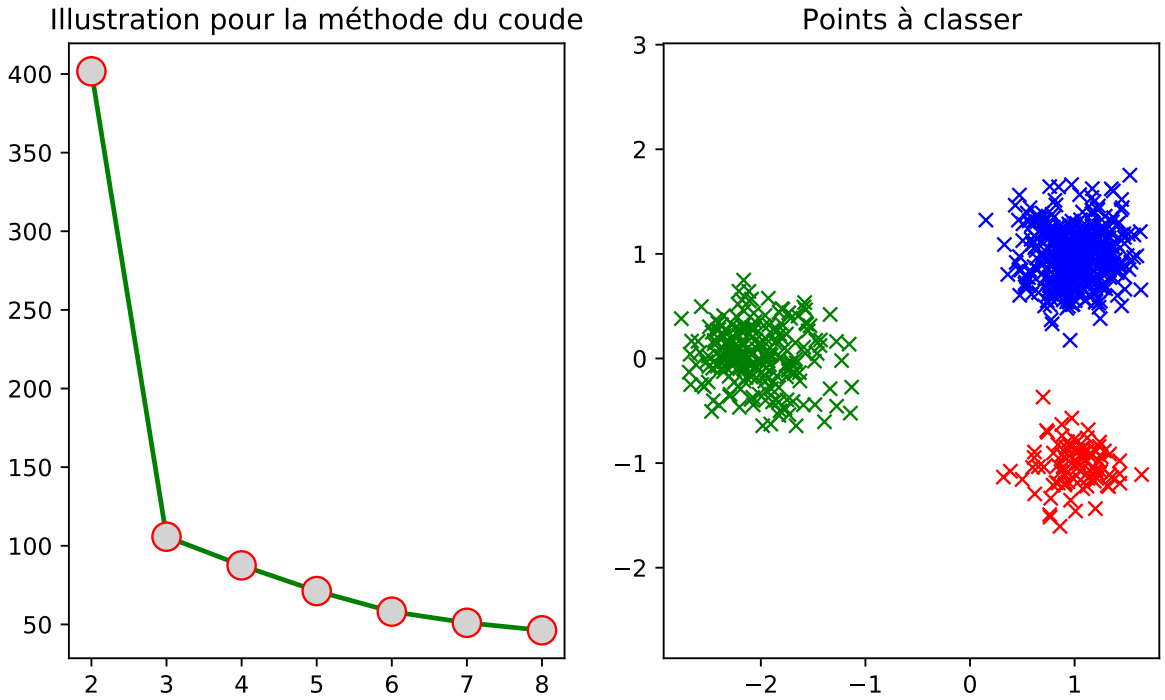
$\Rightarrow$ *Il faut d'autres critères pour décider du bon nombre de classes !*

Nous allons en voir deux :

- la méthode du coude ;
- la méthode des scores silhouettes.

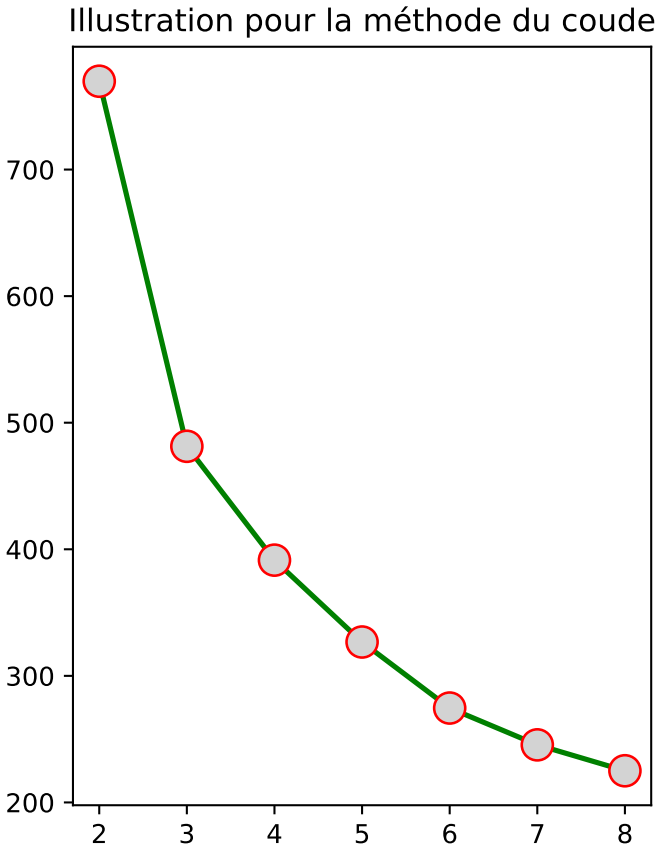
Choix du bon nombre de classes II

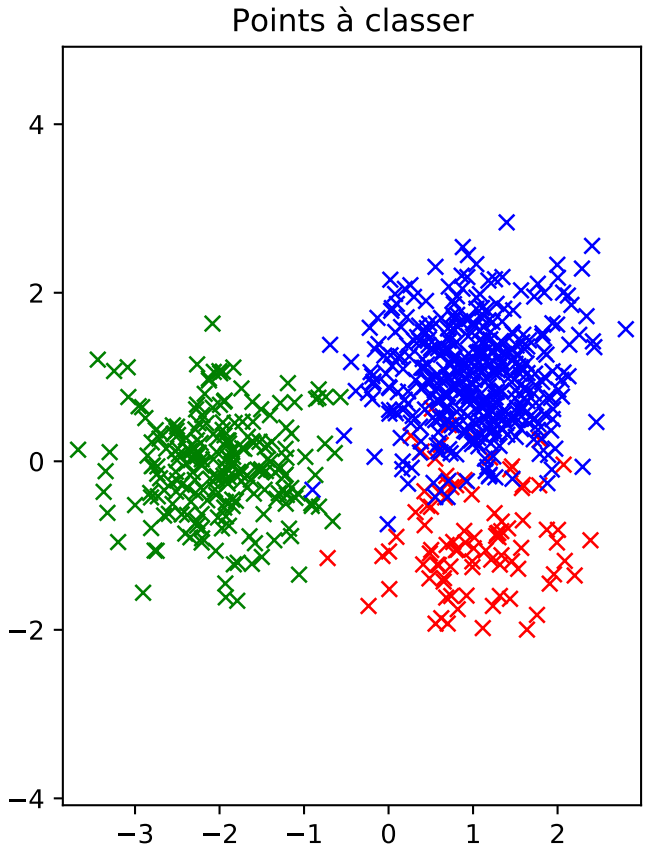
La méthode du coude : on calcule $I^*(2)$, puis $I^*(3)$, $I^*(4)$, etc. Soit K^* le nombre optimal de classes, on s'attend qu'ensuite (i.e. pour $K > K^*$) l'inertie diminue mais moins qu'avant : la courbe doit présenter un coude localisé en K^*. Exemple :



Choix du bon nombre de classes III

Autre exemple (choix plus difficile avec ce critère) :





Choix du bon nombre de classes IV

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 :

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

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

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

Choix du bon nombre de classes V

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

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

La distance entre x_i et la classe voisine la plus proche. On remarque que :

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

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

Choix du bon nombre de classes VI

- 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 $b_i << a_i$, 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. Rmq : lorsque son score est 0 le point est a priori aussi proche de sa classe que de la classe la plus proche.

Choix du bon nombre de classes VII

Que faire avec les scores silhouettes ?

- Leur moyenne :

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

ce qui nous 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.

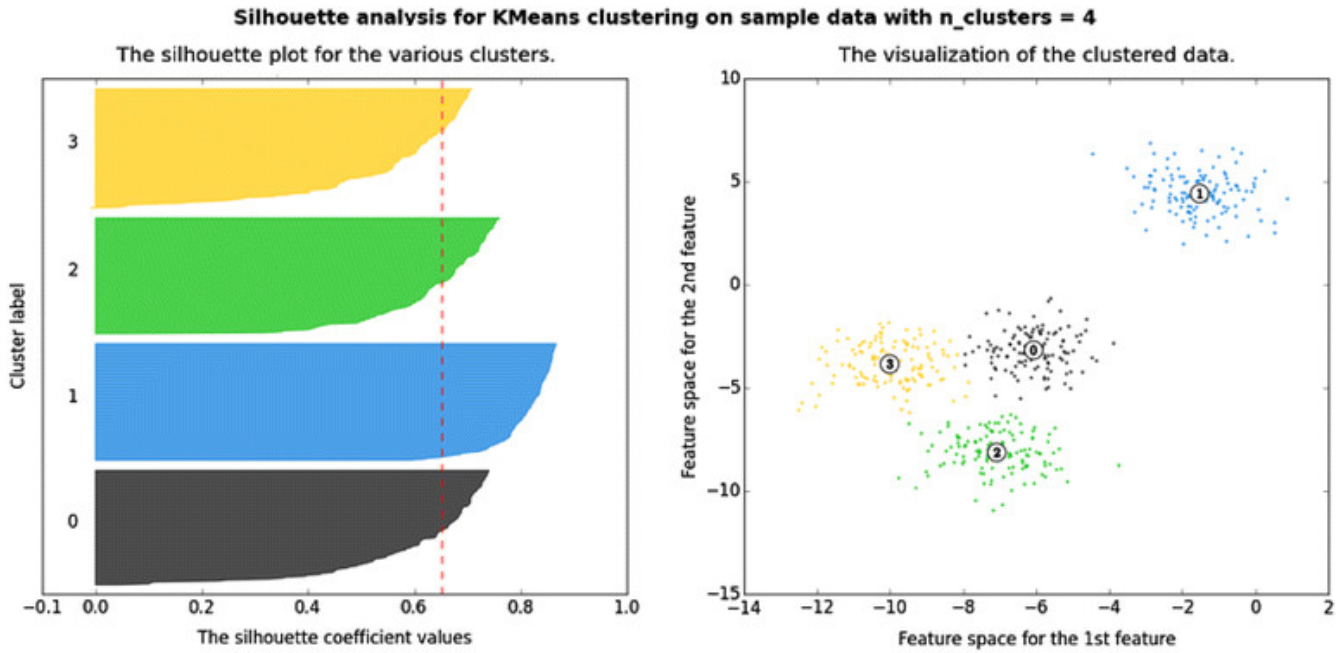
- La moyenne par classe :

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

qui donne une indication plus précise que S_Π.

Choix du bon nombre de classes VIII

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



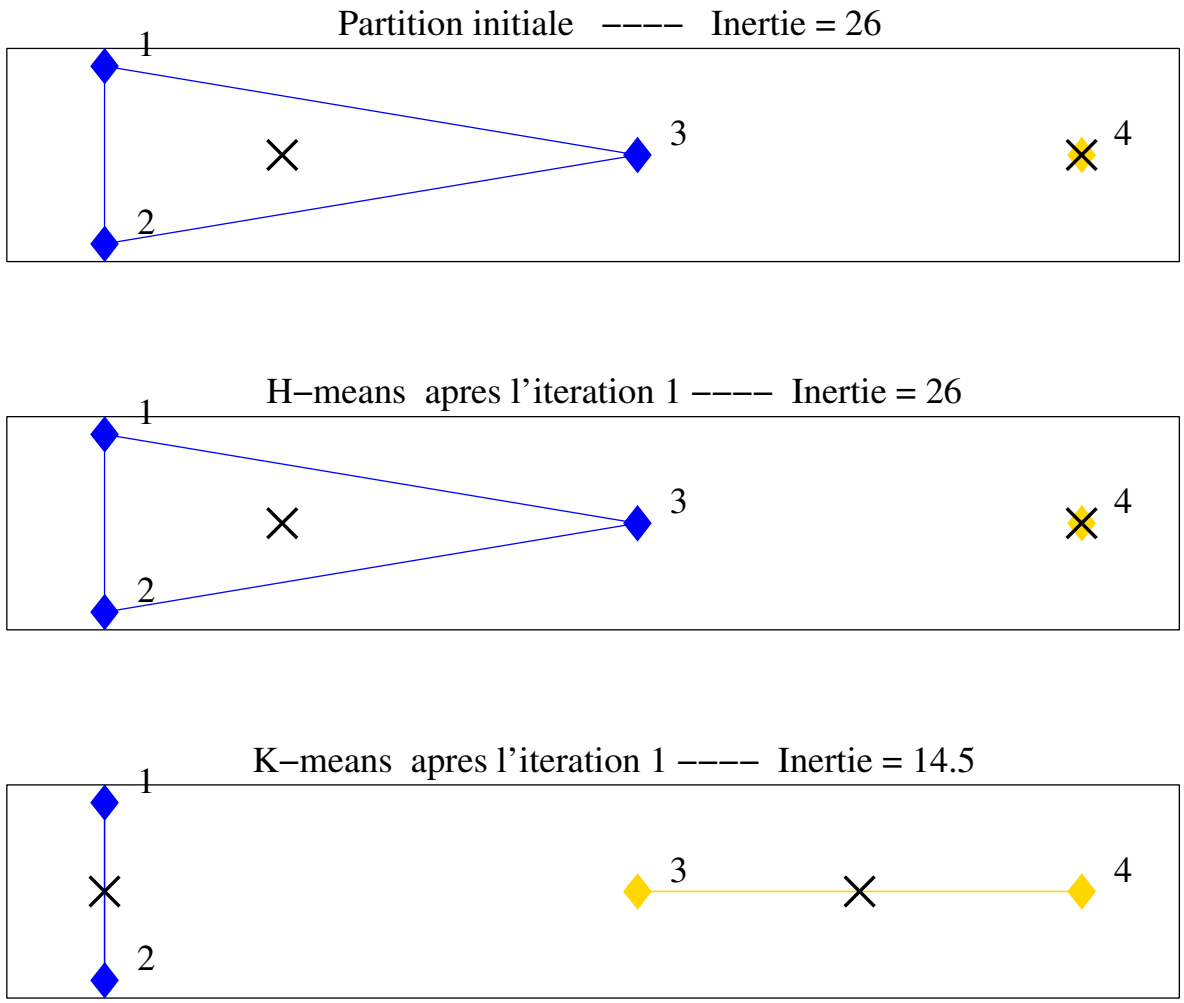
Crédits illustration : Identification of Asthma Subtypes Using Clustering Methodologies, June 2016, Pulmonary Therapy 2(1), DOI : 10.1007/s41030-016-0017-z, LicenseCC BY-NC 4.0, Matea Deliu, Matthew Sperrin, Danielle Belgrave, Adnan Custovic.

Vers les *K*-means I

Deux remarques sur l'algorithme *H-means* conduisent à l'algorithme *K-means* :

1. lorsque l'on affecte un point à une autre classe, les deux barycentres concernés ne sont pas corrigés ; ceci paraît naturel car il peut sembler onéreux de systématiquement recalculer ces barycentres à chaque changement ; nous verrons que cette opération peut se faire à moindre coût et que si les centres des classes correspondent toujours à leur barycentre, la phase (1) de l'algorithme *H-means* devient inutile.
2. en fait on peut remarquer qu'attribuer le point au centre le plus proche n'est pas toujours la meilleure option pour diminuer l'inertie ; la figure suivante illustre ce problème :

Vers les K-means II



K-means versus H-means (les croix montrent les barycentres)

Vers les K-means III

1. lors de la phase d'initialisation, les points numéros 3 et 4 ont été choisi respectivement comme centres inititiaux des classes 1 et 2 ; la classe 1 (en bleu) est donc formée par les points $\{x_1, x_2, x_3\}$ et la classe 2 ne contient que le point x_4 ;
2. l'algorithme *H-means* ne change pas cette partition initiale car le point 3 est plus proche du barycentre de sa classe initiale que de celui de la classe 2 ;
3. dans l'algorithme *K-means*, on envisage successivement pour chaque point, tous les changements de classe possibles en mesurant exactement la contribution de chaque changement à l'inertie finale (ce point sera détaillé ultérieurement), et l'algorithme découvre alors qu'attribuer le point 3 à la classe 2 permet de diminuer l'inertie ; la partition finale (obtenue aussi en 1 seule itération) est donc de meilleure qualité (au sens de l'inertie).

Algorithme *K-means* I

Après le choix d'une partition initiale puis le calcul des barycentres de chaque classe, il consiste à effectuer jusqu'à convergence des itérations constituées par une boucle sur les points :

Soit i le numéro du point courant et $\ell = classe_i$ sa classe actuelle, on envisage le passage de x_i dans chaque classe $k \neq \ell$, en calculant la modification $c_{i,k}$ de l'inertie obtenue par ce changement (si I l'inertie actuelle, alors $I + c_{i,k}$ donnerait la valeur de l'inertie suite à ce changement) ; on retient la valeur minimale des $c_{i,k}$ et la classe $\bar{k}$ correspondante :

$$\bar{c}_i = \min_{k \neq \ell} c_{i,k} \ , \quad \bar{k} = \min\{k | c_{i,k} = \bar{c}_i\}$$

si $\bar{c}_i < 0$ alors le point i passe de la classe ℓ à la classe $\bar{k}$ et l'on met à jour les barycentres des deux classes.

Algorithme *K-means* II

Rmq :

1. la convergence s'obtient lorsqu'il n'y a plus de changement de classe (ainsi pour le test d'arrêt on peut utiliser une variable qui compte le nombre de changements de classe dans la boucle sur les points) ;
2. si x_i est le seul point de la classe ℓ alors il est inutile d'envisager un changement dans une autre classe car l'inertie ne peut alors qu'augmenter (en effet l'inertie associée à la classe ℓ est alors nulle vu que son barycentre est confondu avec le seul point de la classe). Cette remarque montre que si l'initialisation est correcte (pas de classe vide) alors l'algorithme ne peut pas dégénérer, c'est à dire qu'une classe ne peut pas se vider (contrairement à l'algorithme *H-means* où ce problème est néanmoins peu fréquent).

Algorithme *K-means* III

Calcul de $c_{i,k}$: le point x_i est actuellement dans la classe ℓ et on envisage son passage dans la classe $k \neq \ell$; enfin on suppose que la classe ℓ contient au moins deux points (si elle ne contient que le point x_i l'inertie ne pourrait qu'augmenter). On notera :

- $\hat{g}_\ell$ et $\hat{g}_k$ les barycentres après changement de classe,
- C_ℓ et C_k les classes ℓ et k actuelles (avant changement) et $n_\ell = \#C_\ell$, $n_k = \#C_k$ leur nombre d'éléments respectifs :
- $\hat{I}_\ell$ et $\hat{I}_k$ les inerties associées aux classes ℓ et k après changement.

Algorithme *K-means* IV

Tout d'abord remarquons que l'on peut exprimer assez facilement $\hat{g}_\ell$ et $\hat{g}_k$ en fonction de g_ℓ et g_k (et donc avoir une mise à jour rapide des barycentres si le changement envisagé est effectif) :

$$\begin{aligned}
\hat{g}_\ell &= \frac{1}{n_\ell - 1} \sum_{x \in C_\ell \setminus \{x_i\}} x \\
&= \frac{1}{n_\ell - 1} \left(\sum_{x \in C_\ell} x - x_i \right) \\
&= \frac{1}{n_\ell - 1} \left(n_\ell g_\ell - x_i \right) \\
&= \frac{1}{n_\ell - 1} \left((n_\ell - 1) g_\ell + g_\ell - x_i \right) \\
&= g_\ell + \frac{1}{n_\ell - 1} \left(g_\ell - x_i \right)
\end{aligned}$$

Algorithme *K-means* V

Un calcul analogue donne :

$$\begin{aligned}\hat{g}_k &= \frac{1}{n_k+1}\sum_{x\in C_k\cup\{x_i\}} x \\ &= \frac{1}{n_k+1}(n_k g_k + x_i) \\ &= g_k + \frac{1}{n_k+1}(x_i - g_k)\end{aligned}$$

Passons maintenant au changement dans l'inertie : les deux termes modifiés sont les inerties des classes ℓ et k : leurs contributions passe de $I_\ell + I_k$ à $\hat{I}_\ell + \hat{I}_k$ et ainsi $\hat{I} = I - (I_\ell + I_k) + (\hat{I}_\ell + \hat{I}_k)$ soit $c_{i,k} = (\hat{I}_\ell - I_\ell) + (\hat{I}_k - I_k)$. Développons $\hat{I}_\ell$:

$$\hat{I}_\ell = \sum_{x\in C_\ell\setminus\{x_i\}} d^2(\hat{g}_\ell, x) = \sum_{x\in C_\ell\setminus\{x_i\}} ||\hat{g}_\ell - x||^2$$

Algorithme *K-means* VI

en remplaçant $\hat{g}_\ell$ par son expression en fonction de g_ℓ, il vient :

$$\begin{aligned} ||\hat{g}_\ell - x||^2 &= ||g_\ell - x + \frac{g_\ell - x_i}{n_\ell - 1}||^2 \\ &= ||g_\ell - x||^2 + \frac{1}{(n_\ell - 1)^2}||g_\ell - x_i||^2 + \frac{2}{n_\ell - 1}(g_\ell - x|g_\ell - x_i) \end{aligned}$$

ainsi $\hat{I}_\ell = \sum_{x \in C_\ell \setminus \{x_i\}} ||\hat{g}_\ell - x||^2$ s'écrit aussi :

$$\begin{aligned} \hat{I}_\ell &= \sum_{x \in C_\ell \setminus \{x_i\}} \left(||g_\ell - x||^2 + \frac{1}{(n_\ell - 1)^2}||g_\ell - x_i||^2 + \frac{2}{n_\ell - 1}(g_\ell - x|g_\ell - x_i) \right) \\ &= I_\ell - ||g_\ell - x_i||^2 + \frac{1}{n_\ell - 1}||g_\ell - x_i||^2 + \frac{2}{n_\ell - 1} \sum_{x \in C_\ell \setminus \{x_i\}} (g_\ell - x|g_\ell - x_i) \\ &= I_\ell - ||g_\ell - x_i||^2 + \frac{1}{n_\ell - 1}||g_\ell - x_i||^2 + \frac{2}{n_\ell - 1} \left(\sum_{x \in C_\ell \setminus \{x_i\}} (g_\ell - x)|g_\ell - x_i \right) \end{aligned}$$

Algorithme *K-means* VII

On utilise maintenant le fait que g_ℓ est le barycentre de la classe ℓ :

$$g_\ell = \frac{1}{\#C_\ell} \sum_{x \in C_\ell} x \iff \#C_\ell g_\ell - \sum_{x \in C_\ell} x = 0 \iff \sum_{x \in C_\ell} (g_\ell - x) = 0$$

d'où :

$$\sum_{x \in C_\ell \setminus \{x_i\}} (g_\ell - x) = -(g_\ell - x_i)$$

et donc finalement :

$$\begin{aligned}
\hat{I}_\ell &= I_\ell - ||g_\ell - x_i||^2 + \frac{1}{n_\ell - 1}||g_\ell - x_i||^2 - \frac{2}{n_\ell - 1}(g_\ell - x_i | g_\ell - x_i) \\
\hat{I}_\ell &= I_\ell - ||g_\ell - x_i||^2 + \frac{1}{n_\ell - 1}||g_\ell - x_i||^2 - \frac{2}{n_\ell - 1}||g_\ell - x_i||^2 \\
\hat{I}_\ell &= I_\ell + \frac{-(n_\ell - 1) + 1 - 2}{n_\ell - 1}||g_\ell - x_i||^2
\end{aligned}$$

Algorithme *K-means* VIII

Soit :

$$\hat{I}_\ell - I_\ell = -\frac{n_\ell}{n_\ell - 1}||g_\ell - x_i||^2$$

Un calcul semblable nous donne :

$$\hat{I}_k - I_k = \frac{n_k}{n_k + 1}||g_k - x_i||^2$$

et donc la contribution apportée à l'inertie si le point x_i passait de la classe ℓ à la classe k serait de :

$$c_{i,k} = \frac{n_k}{n_k + 1}||g_k - x_i||^2 - \frac{n_\ell}{n_\ell - 1}||g_\ell - x_i||^2$$

Avec ces derniers détails, on peut maintenant écrire un algorithme plus précis. La solution exposée reprend les notations précédemment utilisées pour l'algorithme *H-means*. D'autre part il est possible de

Algorithme *K-means* IX

calculer les inerties tout au début (après le calcul des barycentres) et de les mettre à jour à chaque changement de classe d'un point (il faut alors scinder le terme *temp* en deux pour apporter les modifications de I_ℓ et $I_{\bar{k}}$) et la phase de post-traitement est alors inutile. On peut aussi rajouter une variable (qui additionnerait les termes $\bar{c}$) si l'on veut connaître la diminution de l'inertie sur une itération complète.

Algorithme *K-means*
entrées : X, classe (la partition initiale)
sorties : classe (partition finale), I, ...

initialisation : calcul des barycentres :
$g_k \leftarrow [0,0,0]$ et $ne_k \leftarrow 0$ pour $k = 1, 2, ..., K$
pour i de 1 à n
$k \leftarrow classe_i$
$g_k \leftarrow g_k + x_i$; $ne_k \leftarrow ne_k + 1$
pour k de 1 à K

Algorithme *K-means* X

$g_k \leftarrow g_k / ne_k$

l'algorithme en question :

répéter :

$nbcht \leftarrow 0$

pour i de 1 à n

$\ell \leftarrow classe_i$

si $ne_\ell > 1$ **alors**

$\ell \leftarrow classe_i$; $\bar{c} \leftarrow +\infty$

pour k de 1 à K et $k \neq \ell$

$temp \leftarrow \frac{ne_k}{ne_k+1}||g_k - x_i||^2 - \frac{ne_\ell}{ne_\ell-1}||g_\ell - x_i||^2$

si $temp < \bar{c}$ **alors**

$\bar{c} \leftarrow temp$; $\bar{k} \leftarrow k$

si $\bar{c} < 0$ **alors** le point change de classe

$classe_i \leftarrow \bar{k}$; $nbcht \leftarrow nbcht + 1$

$g_\ell \leftarrow g_\ell + (g_\ell - x_i)/(ne_\ell - 1)$; $ne_\ell \leftarrow ne_\ell - 1$

$g_{\bar{k}} \leftarrow g_{\bar{k}} + (x_i - g_{\bar{k}})/(ne_{\bar{k}} + 1)$; $ne_{\bar{k}} \leftarrow ne_{\bar{k}} + 1$

Algorithme *K-means* XI

jusqu'à ce que $nbcht = 0$ si aucun point n'a changé de classe alors la stabilisation est obtenue

post traitement : calcul des inerties :
$I_k \leftarrow 0$ pour $k = 1, 2, ..., K$
pour i de 1 à n
$k \leftarrow classe_i$
$I_k \leftarrow I_k + ||x_i - g_k||^2$