Chapitre 5

Introduction à l'analyse de données : Classification non hiérarchique.

5.1 Introduction

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).

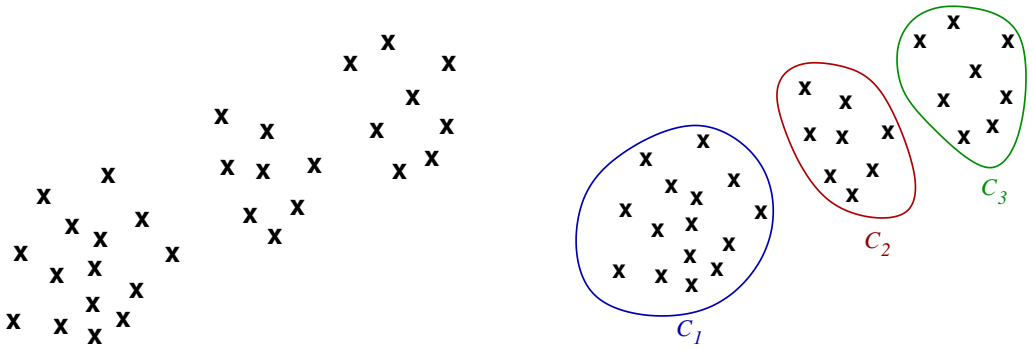

FIGURE 5.1 – Classification d'individus (à 2 caractères) en 3 groupes

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'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 (comme pour les moindres carrés) [1] :

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

1. Cependant les codes proposent généralement d'utiliser aussi des distances qui ne dérivent pas d'un produit scalaire.

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).

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 :

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

— $I(\Pi)$ 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 :

$$I(\Pi^*) \leq I(\Pi), \forall \Pi \in P_{ad} \tag{5.1}$$

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.

Remarque : 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 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 classe fixé et de les essayer successivement avec différentes valeurs de K.

5.2 Considérations générales sur le problème (5.1)

On peut remarquer que 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 (ou les) partition(s) optimale(s). Le problème est que le nombre d'éléments de P_{ad} à tendance à être gigantesque ! Appelons $P_{n,k}$ 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 ($P_{n,k} = \#\Pi_k(E)$). On a les résultats élémentaires suivants :

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 $\Pi_n(E) = \{\{\{e_1\}, \ldots, \{e_n\}\}\}$.

Avec ces cas simples, on peut maintenant calculer $P_{n,k}$ dans le cas général grâce à la formule de récurrence suivante :

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

Preuve : soit donc $\Pi_k(E)$ l'ensemble des partitions qui nous intéressent et $e \in E$ 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 !).

Il est clair que : $\#Q_1 = P_{n-1,k-1}$ 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. D'où finalement $\#Q_2 = kP_{n,k-1}$ et le résultat cherché $\square$.

Comme exercice vous pouvez coder une fonction récursive en python [2] ou encore, un peu plus difficile mais beaucoup plus efficace, coder une fonction qui utilisera un algorithme en "triangle" (comme pour le triangle de Pascal).

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

Malgré cette explosion combinatoire, quelques algorithmes permettant de trouver l'optimum exact ont été mis au point [3], 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* et *stochastiques* [4] 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. En effet 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.

2. Ou dans un autre langage en utilisant une classe d'entiers arbitrairement grands.
3. Ces algorithmes sont basés sur la force brute mais en incluant des astuces permettant de ne pas tester toutes les partitions.
4. Un algorithme est qualifié de stochastique lorsque le hasard y joue un rôle.

5.3 Quelques algorithmes

5.3.1 Introduction

Les deux algorithmes que nous allons détailler sont connus sous les noms de *K-means* et *H-means*, ce dernier étant appelé méthode des *centres mobiles* en français. Enfin ils apparaissent comme des cas particuliers d'une méthode plus générale nommée *nuées dynamiques*. L'algorithme des *K-means* est une amélioration de celui des *H-means* : en général, il converge plus rapidement et on obtient une partition de meilleure qualité (au sens où l'inertie totale est plus faible). De plus, il peut améliorer une partition obtenue par les *H-means*, l'inverse étant faux. Cependant dans certains cas, partant de la même configuration initiale, l'algorithme des *H-means* peut donner une meilleure partition (qui est ensuite améliorable par *K-means*!) il est donc bon d'avoir ces deux méthodes dans sa musette ! Notons que généralement le nom de *K-means* est donné à l'algorithme *H-means* et que l'algorithme *H-means* est facilement vectorisable/parallélisable d'où sans doute sa prédominance sur celui des *K-means*.

Le résultat obtenu par ces algorithmes sur un cas particulier dépend (généralement) de la configuration (partition) initiale choisie. Nous allons tout d'abord détailler la méthode usuellement utilisée pour obtenir cette partition initiale.

5.3.2 Choix de la partition initiale

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 $\{1, 2, \ldots, n\}$.

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 :

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

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

5.3.3 Algorithme des *H-means*

Chaque itération étant 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 chaque point (individu) en le réaffectant 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.

Ce processus est itéré jusqu'à stabilisation de la partition. Il faut noter que lors de la phase d'affectation, on ne recalcule pas les barycentres lorsqu'un point change de classe. Cette remarque (et une autre) est à l'origine de l'algorithme des *K-means*. Dans certains cas, on peut ne pas aller jusqu'à la stabilisation complète mais arrêter les itérations lorsqu'on considère que l'inertie ne diminue plus beaucoup [5]. Pour une écriture plus algorithmique, 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, on écrira une version un peu différente en python numpy en TP.

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

répéter

phase de calcul des barycentres :

$g_k \leftarrow [0, ..., 0]$ 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
si $ne_k = 0$ **alors** *traiter l'exception* **fin si**
$g_k \leftarrow g_k / ne_k$

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 :

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

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

5. Cette stratégie s'explique du fait que l'inertie diminue assez rapidement lors des premières itérations puis plus lentement par la suite.

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 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

On considère alors la suite des inerties suivantes :

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

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 (à développer un peu ...). D'autre part, le barycentre d'un ensemble de points verifie (ici pour la classe k) :

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

(c'est le théorème de *Huygens*) 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 monotone, elle devient effectivement stationnaire à partir d'un certain rang (exercice). 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 $\square$.

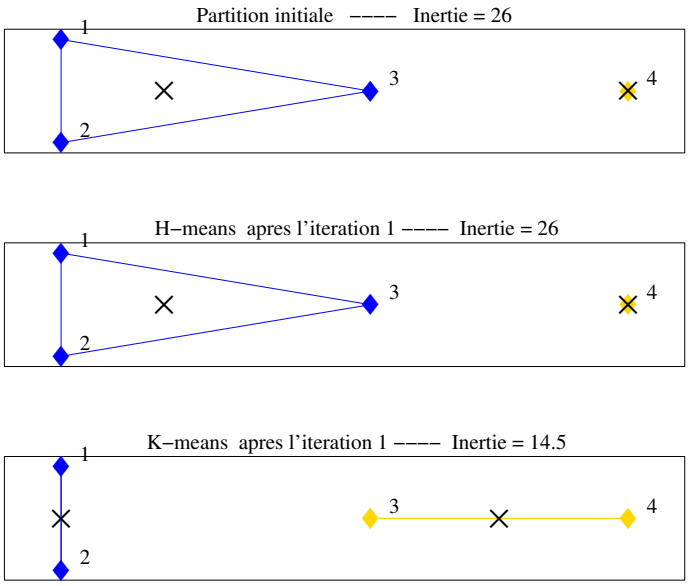

FIGURE 5.2 – K-means versus H-means

5.3.4 Algorithme des *K-means*

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 5.2 illustre ce problème :
   (a) 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 ; sur le dessin on a dessiné avec des croix les barycentres des deux classes ;
   (b) 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 ;
   (c) 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).

Ainsi l'algorithme *K-means*, après choix d'une partition initiale puis calcul des barycentres de chaque classe, consiste à effectuer jusqu'à convergence [6] des itérations constituées par une boucle sur les points :

6. comme pour l'algorithme *H-means* on peut aussi arrêter les itérations avant stabilisation complète.

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 [7] :

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

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.

Remarque :

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* [8]).

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. 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.

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) :

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

7. celle de plus petit indice si le minimum est atteind pour plusieurs choix
8. où ce problème est néanmoins peu fréquent.

Un calcul analogue donne :

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

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 $\hat{I}_\ell$:

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

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

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

ainsi :

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

On utilise maintenant le fait que g_ℓ est le barycentre de la classe ℓ : $\sum_{x\in C_\ell} g_\ell - x = 0$ et donc :

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

et donc finalement :

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

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 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
    $g_k \leftarrow g_k / ne_k$

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
                $classe_i \leftarrow \bar{k}$; $nbcht \leftarrow nbcht + 1$
                $g_\ell \leftarrow g_\ell + (g_\ell - x_i)/(ne_\ell - 1)$; $ne_\ell \leftarrow ne_\ell - 1$
                $g_{\bar{k}} \leftarrow g_{\bar{k}} + (x_i - g_{\bar{k}})/(ne_{\bar{k}} + 1)$; $ne_{\bar{k}} \leftarrow ne_{\bar{k}} + 1$
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
    $k \leftarrow classe_i$
    $I_k \leftarrow I_k + ||x_i - g_k||^2$