Mathématiques Numériques
introduction à la classification non hiérarchique
Telecom Nancy
(version courte)

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

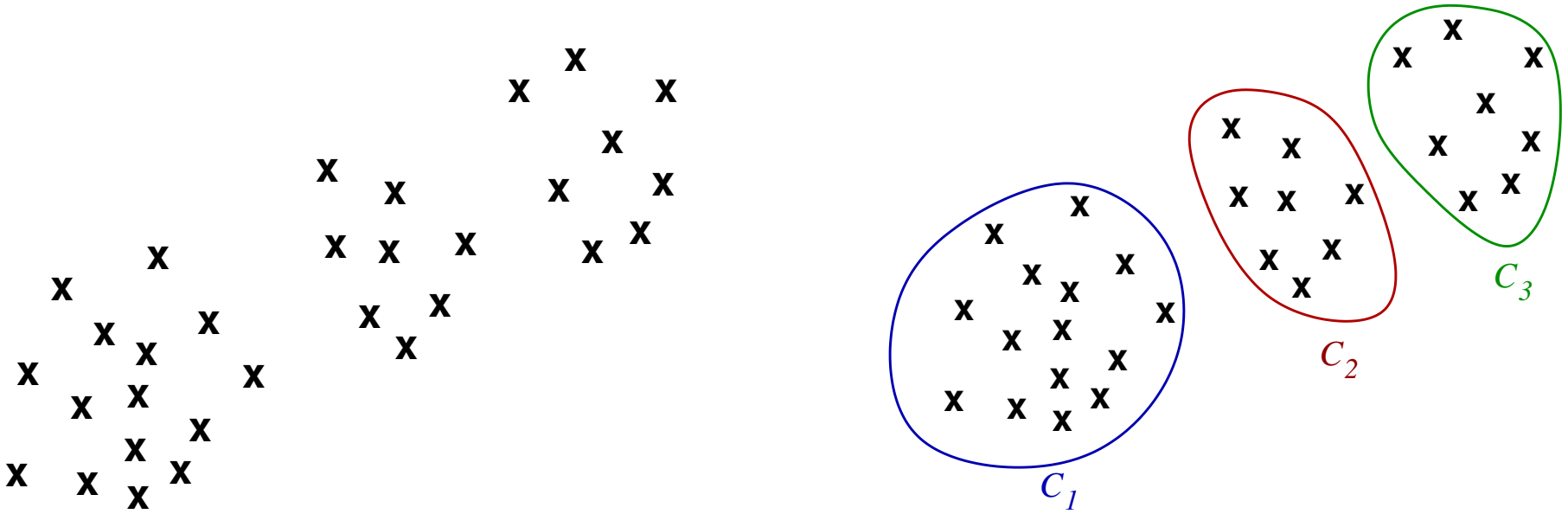
2021-2022

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" :

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

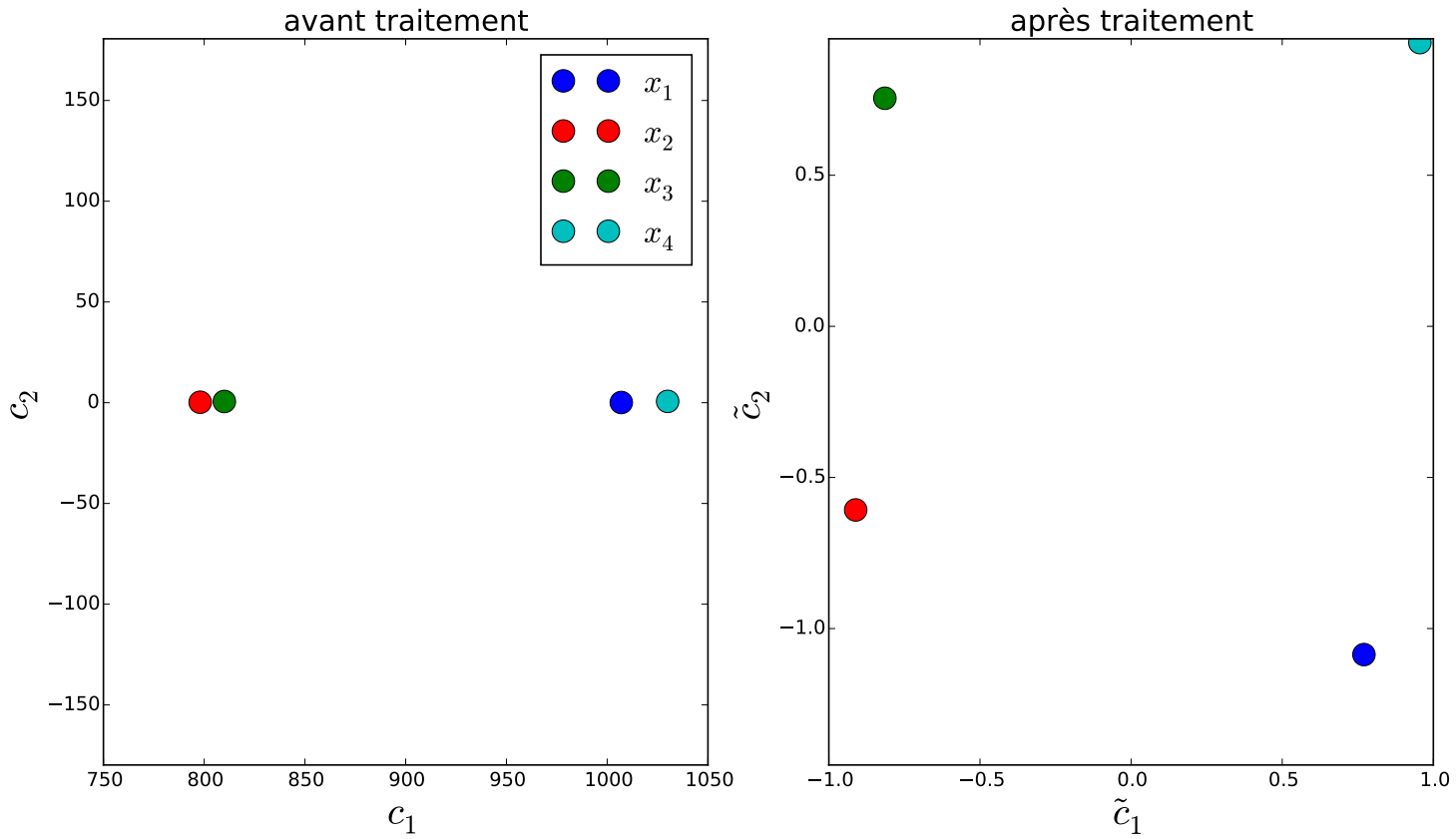
Sur les données précédentes, on obtient :

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 $I_\Pi < I_{opt}$
        $I_{opt} \leftarrow I_\Pi$
        $\Pi_{opt} \leftarrow \Pi$

Considérations générales II

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

$n \setminus k$	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 III

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

Ce point sera évoqué en partie en TD. On montre que si aucune classe ne se vide (ce qui peut arriver mais est rarissime), on obtient la stabilisation de la partition (et donc des barycentres) en un nombre fini d'itérations (voir version longue pour les détails).

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 $I^*(K^*) = 0$!

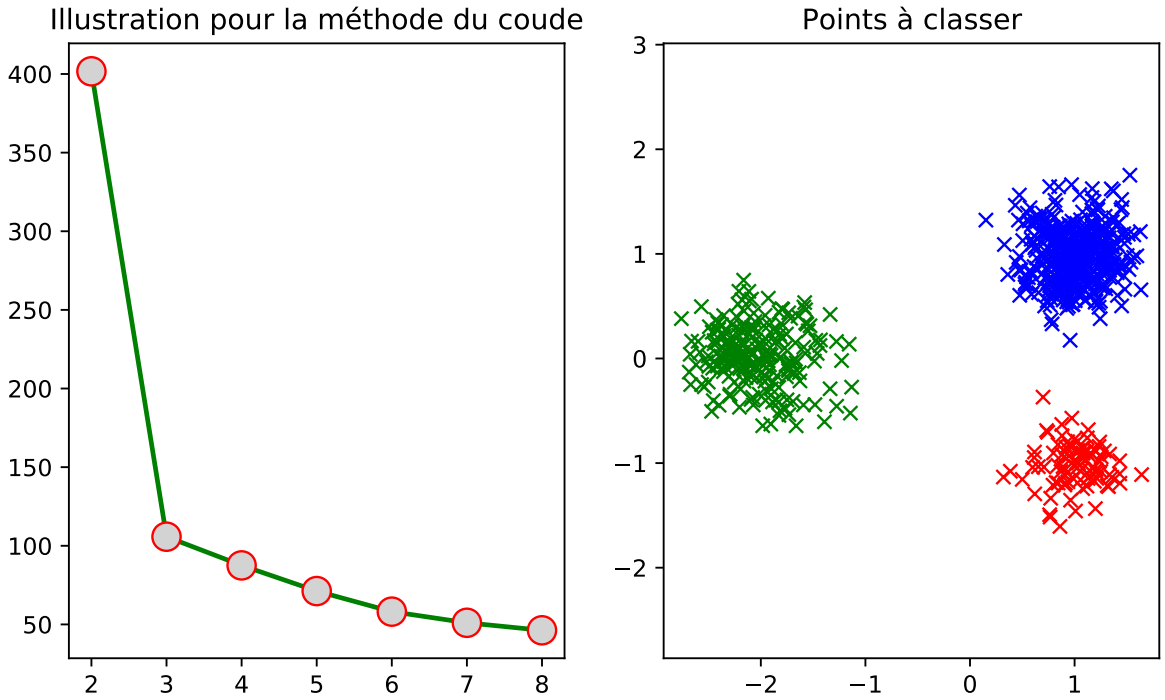
$\Rightarrow$ *Il faut d'autres critères pour décider du bon nombre de classes !*

Nous allons en voir deux :

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

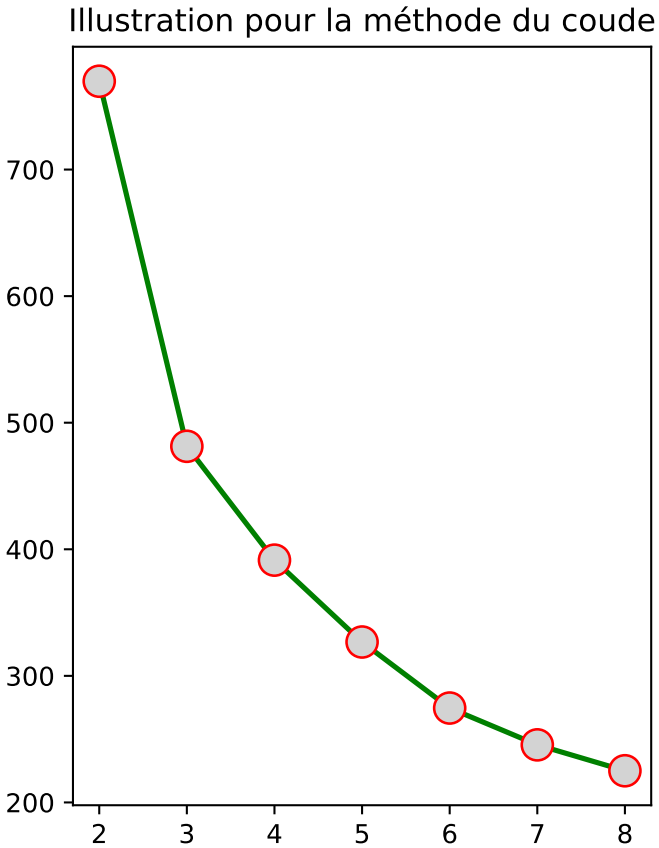
Choix du bon nombre de classes II

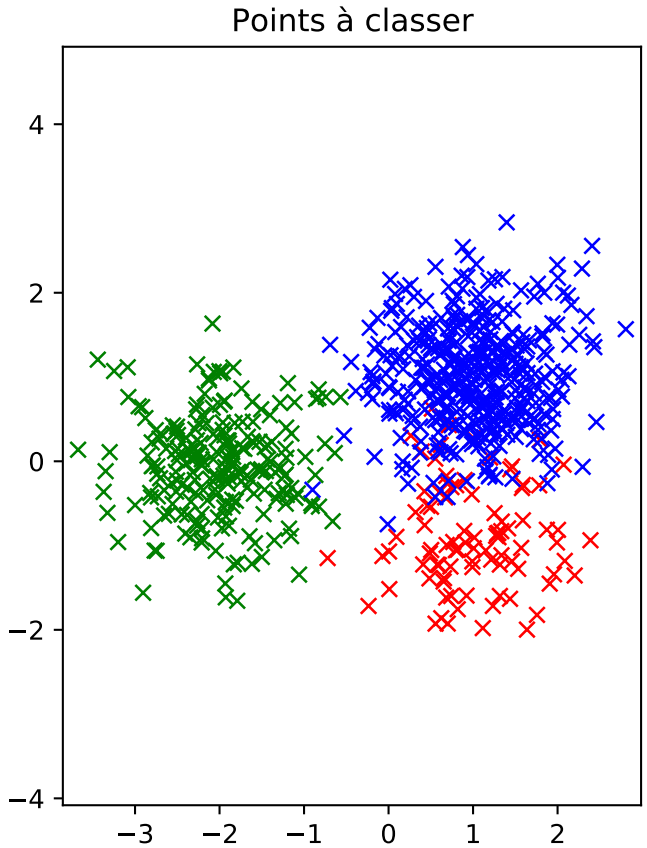
La méthode du coude : on calcule $I^*(2)$, puis $I^*(3)$, $I^*(4)$, etc. Soit K^* le nombre optimal de classes, on s'attend qu'ensuite (i.e. pour $K > K^*$) l'inertie diminue mais moins qu'avant : la courbe doit présenter un coude localisé en K^*. Exemple :



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.

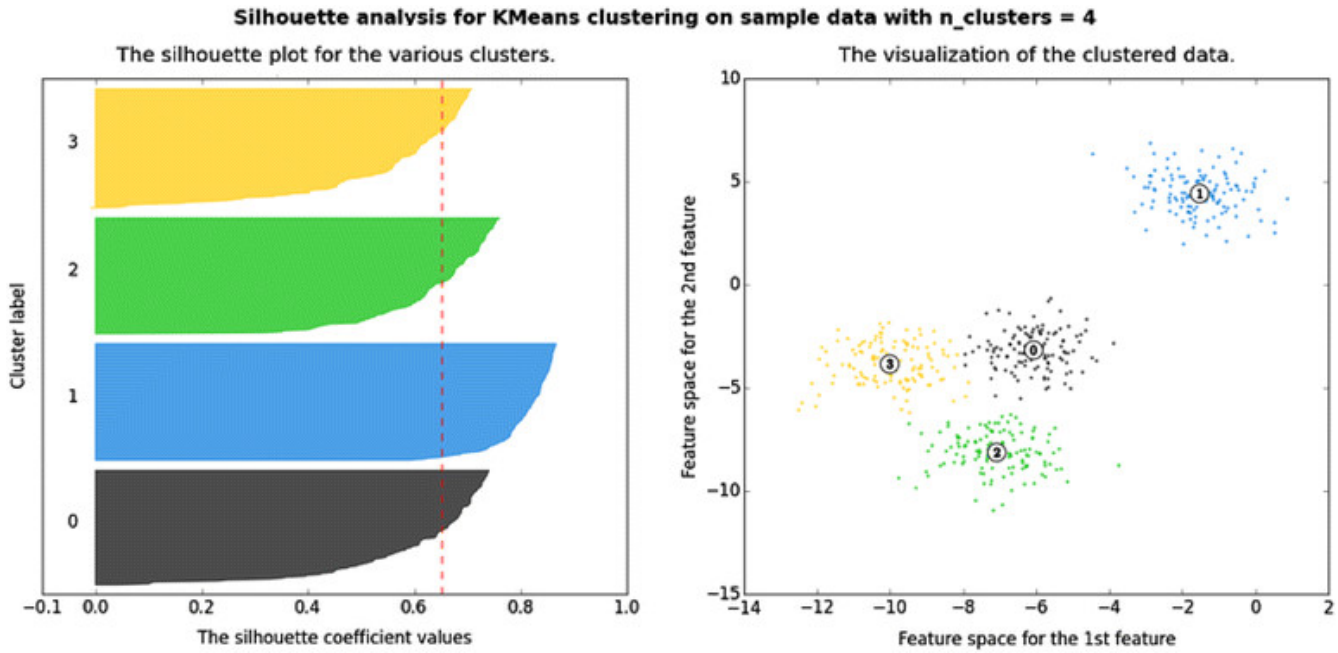
- La moyenne par classe :

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

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.



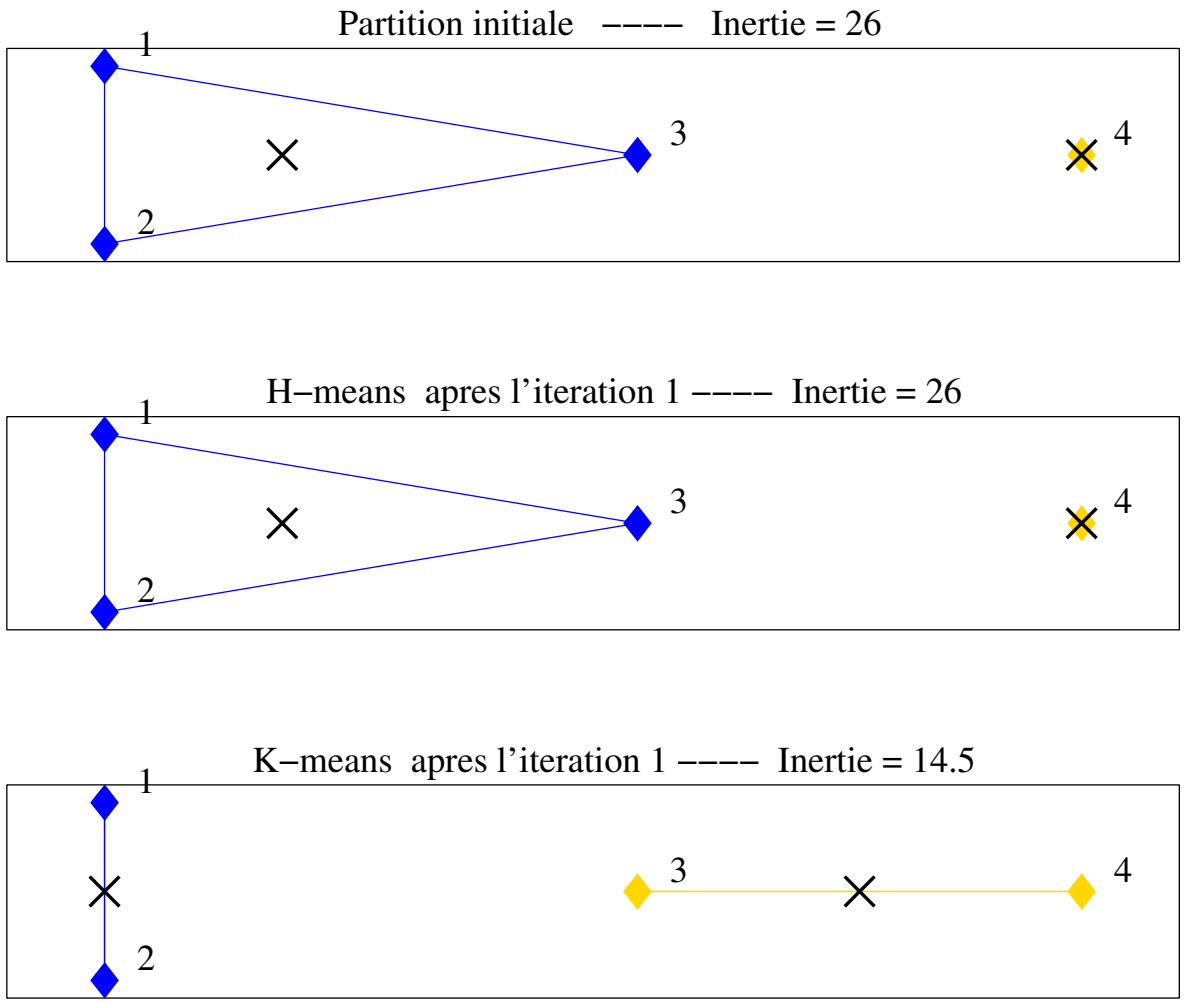
Crédits illustration : Identification of Asthma Subtypes Using Clustering Methodologies, June 2016, Pulmonary Therapy 2(1), DOI : 10.1007/s41030-016-0017-z, LicenseCC BY-NC 4.0, Matea Deliu, Matthew Sperrin, Danielle Belgrave, Adnan Custovic.

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

cf version complète des transparents. Il n'est pas au programme.