COURS : ALGORITHMES POUR L'INTELLIGENCE ARTIFICIELLE == ALGORITHME DES K MOYENNES ==

I) L'APPRENTISSAGE NON SUPERVISÉ ET LE CLUSTERING

Jusqu'à présent, nous avons étudié une méthode d'apprentissage supervisée qui fonctionne en régression et en classification. Dans le cadre de l'apprentissage supervisé, nous disposons généralement d'un ensemble de p variables caractéristiques X_1, X_2, …, X_p, mesurées sur n observations, ainsi que d'une variable réponse Y également mesurée sur ces mêmes n observations. L'objectif est alors de prédire Y à l'aide de X_1, X_2, …, X_p.

Dans ce cours, nous allons étudier une méthode d'apprentissage non supervisé, destinée aux situations dans lesquelles nous ne disposons que d'un ensemble de variables caractéristiques. Nous ne cherchons pas à effectuer une prédiction, car nous ne disposons pas d'une variable réponse associée Y. L'objectif est plutôt de découvrir des informations intéressantes concernant les mesures de X_1, X_2, …, X_p. Existe-t-il une manière informative de visualiser les données ? Pouvons-nous découvrir des sous-groupes parmi les variables ou parmi les observations ?

L'apprentissage non supervisé désigne un ensemble varié de techniques permettant de répondre à ce type de questions. Ici, nous nous concentrerons sur un algorithme appartenant à la classe des méthodes de clustering, visant à découvrir des sous-groupes inconnus au sein des données.

I.1. Le défi de l'apprentissage non supervisé

L'apprentissage supervisé est un domaine bien compris. Par exemple, si on souhaite prédire un résultat binaire à partir d'un jeu de données, on dispose d'un ensemble d'outils très développés (tels que la régression, les réseaux de neurones, la méthode KNN), ainsi qu'une compréhension claire de la manière d'évaluer la qualité des résultats obtenus (en utilisant la validation sur un ensemble de test indépendant par exemple).

L'apprentissage non supervisé est souvent réalisé dans le cadre d'une analyse exploratoire des données, ce qui le rend souvent bien plus difficile que l'apprentissage supervisé :

1 Il n'existe pas d'objectif simple pour l'analyse, tel que la prédiction d'une variable réponse ;
2 Il peut être difficile d'évaluer les résultats obtenus par les méthodes d'apprentissage non supervisé. En effet, si nous ajustons un modèle prédictif à l'aide d'une technique d'apprentissage supervisé, il est alors possible de vérifier notre travail en observant dans quelle mesure notre modèle prédit correctement la variable réponse Y sur des observations qui n'ont pas été utilisées lors de l'ajustement du modèle. En revanche, dans l'apprentissage non supervisé, il n'existe aucun moyen de vérifier notre travail, car nous ne connaissons pas la réponse véritable.

Les techniques d'apprentissage non supervisé prennent une importance croissante dans de nombreux domaines. Voici quelques exemples :

3 Un moteur de recherche peut choisir quels résultats afficher à un individu donné en se basant sur les historiques de clics d'autres individus présentant des schémas de recherche similaires.
4 Un site de commerce en ligne peut chercher à identifier des groupes d'acheteurs présentant des historiques de navigation et d'achats similaires, ainsi que des articles

présentant un intérêt particulier pour les acheteurs au sein de chaque groupe. Un acheteur individuel peut alors se voir proposer de manière préférentielle les articles susceptibles de l'intéresser particulièrement, sur la base des historiques d'achat d'acheteurs similaires.

Ces tâches d'apprentissage statistique, et bien d'autres encore, peuvent être réalisées à l'aide de techniques d'apprentissage non supervisé.

I.2. Les méthodes de clustering

Le clustering désigne un ensemble très large de techniques visant à identifier des sous-groupes, ou classes, au sein d'un jeu de données. Lorsque nous réalisons un clustering des observations d'un jeu de données, nous cherchons à les partitionner en groupes distincts de telle sorte que les observations appartenant à un même groupe soient très similaires entre elles, tandis que les observations appartenant à des groupes différents soient très dissemblables.

Pour rendre cela concret, il est nécessaire de définir ce que signifie le fait que deux observations ou plus soient similaires ou différentes. Il s'agit souvent d'une considération spécifique au domaine, qui doit être établie à partir de la connaissance des données étudiées.

Par exemple, supposons que nous disposions d'un ensemble de n observations, chacune caractérisée par p variables. Les n observations peuvent correspondre à des utilisateurs d'une plateforme de streaming musical, et les p variables peuvent correspondre à des caractéristiques mesurées pour chaque utilisateur : temps moyen d'écoute par jour, genres musicaux les plus écoutés, fréquence de découverte de nouveaux artistes, utilisation de playlists, heure d'écoute habituelle, etc.

Nous pouvons soupçonner qu'il existe une hétérogénéité parmi ces utilisateurs. Par exemple, il pourrait exister plusieurs types d'auditeurs sans que ces catégories soient connues à l'avance : auditeurs occasionnels, passionnés d'un genre précis, explorateurs musicaux, utilisateurs qui écoutent principalement en arrière-plan, etc. Le clustering peut alors être utilisé pour identifier automatiquement ces sous-groupes d'utilisateurs à partir des données disponibles. Il s'agit d'un problème d'apprentissage non supervisé, car l'objectif est de découvrir une structure (ici des groupes d'utilisateurs aux comportements similaires) uniquement à partir des variables observées. À l'inverse, dans un problème d'apprentissage supervisé appliqué au même contexte, l'objectif pourrait être de prédire une variable précise, par exemple la probabilité qu'un utilisateur s'abonne à une offre premium ou qu'il écoute un nouvel album recommandé.

Dans ce cours, nous nous concentrons sur une des approches de clustering les plus connues : *l'algorithme des k moyennes* (*clustering k-means*).

II) LA MÉTHODE DES K MOYENNES

II.1. Notion de clusters et centroïdes

La méthode des k moyennes est une approche simple et élégante pour partitionner un jeu de données en K clusters distincts et non chevauchants. Pour réaliser ce clustering, il faut d'abord spécifier le nombre souhaité de clusters K ; ensuite, l'algorithme des k moyennes attribuera chaque observation à exactement l'un des K clusters.

La Figure 1 montre les résultats obtenus en appliquant l'algorithme des k moyennes à un exemple simulé composé de 150 observations en deux dimensions, en utilisant trois valeurs différentes de K, le nombre de clusters. La couleur et la forme (ronds bleus, croix jaunes, triangles verts et carrés violets) de chaque observation indiquent le cluster auquel elle a été affectée par l'algorithme des k moyennes. Il n'existe pas d'ordre entre les clusters ; l'attribution des couleurs est donc arbitraire. Ces étiquettes de clusters n'ont pas été utilisées pour effectuer le clustering ; elles constituent au contraire les sorties de la procédure de clustering.

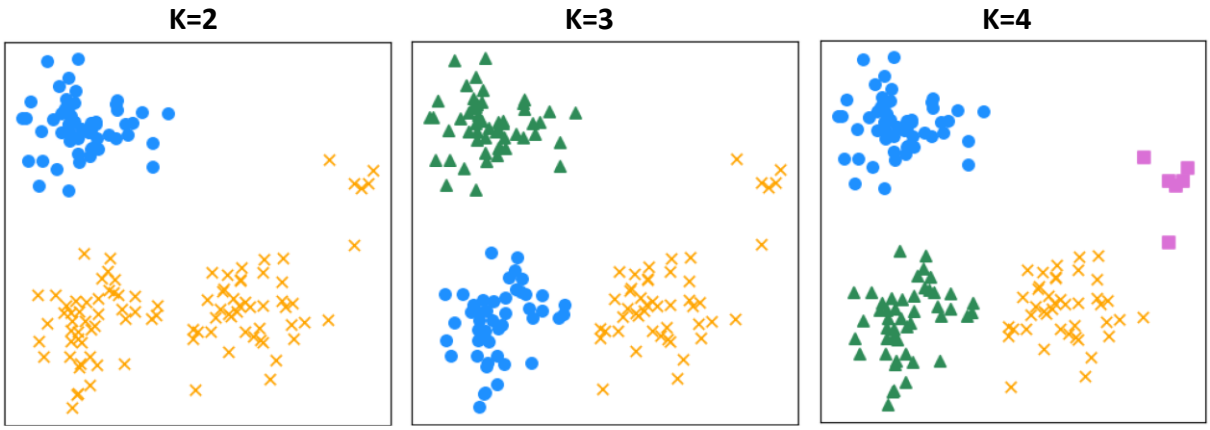

Figure 1 : Simulation d'un jeu de données partitionné avec 150 observations en 2D

Prenons un autre exemple de points de données bidimensionnels, représenté dans le panneau de gauche de la Figure 2. En examinant attentivement les données qui y sont présentées, on peut voir qu'elles se répartissent naturellement en trois groupes ou clusters. Dans le panneau de droite, les points sont associés à 3 clusters. Les centres de chaque cluster sont également représentés. Ces centres de clusters sont souvent désignés, dans le jargon de l'apprentissage automatique, sous le terme de centroïdes de clusters.

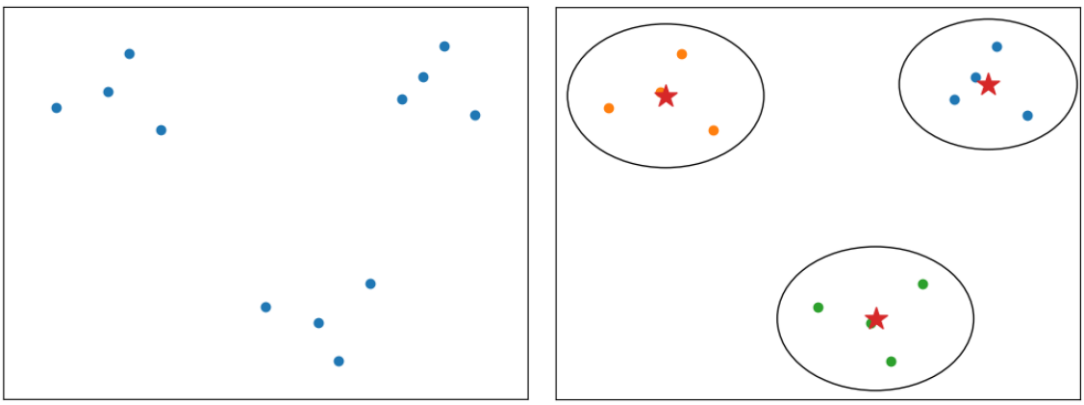

Figure 2 : Jeu de données bidimensionnel (gauche) et regroupement en clusters (K=3)

II.2. Problème d'optimisation pour construire les clusters

Commençons par introduire quelques notations. Soit $\mathrm{D} = \{\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}\}$ les observations et soient C_1, …, C_K des ensembles contenant les observations x_i. Ces ensembles sont appelés des *clusters* et vérifient deux propriétés :

- $C_1 \cup C_2 \cup \ldots \cup C_K = \{\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}\}$: chaque observation appartient au moins à un des K clusters.
- $C_\mathrm{k} \cap C_{\mathrm{k}'} = \emptyset$ pour tous $\mathrm{k} \neq \mathrm{k}'$: les clusters ne se chevauchent pas. Aucune observation n'appartient à plus d'un cluster.

Par exemple, si la i-ème observation appartient au k-ème cluster, alors $\mathrm{x}_\mathrm{i} \in C_k$. L'idée sous-jacente à l'algorithme des k moyennes est qu'un bon partitionnement est celui pour lequel la variation intra-cluster est aussi faible que possible. La variation intra-cluster associée au cluster C_k est une mesure $W(C_\mathrm{k})$ de l'ampleur des différences entre les observations appartenant à ce cluster. Nous cherchons donc à résoudre le problème suivant :

$$\underset{C_1,\ldots,C_K}{\text{minimiser}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

Cette formule signifie que nous souhaitons partitionner les observations en K clusters de manière à ce que la variation intra-cluster totale, obtenue en la sommant sur l'ensemble des K clusters, soit aussi faible que possible.

Pour résoudre ce problème de minimisation, nous devons définir la variation intra-cluster. Il existe de nombreuses manières possibles de formaliser ce concept, mais la plus courante repose sur la distance euclidienne au carré :

$$W(C_\mathrm{k}) = \frac{1}{|C_\mathrm{k}|} \sum_{\substack{\mathrm{x_i} \in C_k \\ \mathrm{x_{i'}} \in C_k}} \|x_i - x_{i'}\|_2^2 \qquad , x_i \in \mathbb{R}^p \; et \; x_i = (x_{i1}, \ldots, x_{ip})$$

$$= \frac{1}{|C_\mathrm{k}|} \sum_{\substack{\mathrm{x_i} \in C_k \\ \mathrm{x_{i'}} \in C_k}} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

… où $|C_k|$ désigne le nombre d'observations dans le k-ième cluster. Autrement dit, la variation intra-cluster du k-ième cluster est égale à la somme de toutes les distances euclidiennes au carré calculées entre chaque paire d'observations appartenant à ce cluster, divisée par le nombre total d'observations du k-ième cluster.

En combinant ces équations, on obtient le problème d'optimisation qui définit l'algorithme des k moyennes :

$$\underset{C_1,\ldots,C_K}{\text{minimiser}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_\mathrm{k}|} \sum_{\substack{\mathrm{x_i} \in C_k \\ \mathrm{x_{i'}} \in C_k}} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

Ce problème fait intervenir les distances entre toutes les paires de points, ce qui le rend extrêmement complexe à résoudre. Il peut cependant être réécrit sous une forme équivalente faisant apparaître explicitement les centroïdes des clusters, mieux adaptée à une résolution algorithmique.

II.3. Problème d'optimisation avec les centroïdes

On peut manipuler l'expression précédente pour faire apparaître le centroïde d'un cluster :

$$\frac{1}{|C_{\mathrm{k}}|}\sum_{\substack{\mathrm{x_i}\in C_k\\ \mathrm{x_{i'}}\in C_k}}\sum_{j=1}^{p}\left(x_{ij}-x_{i'j}\right)^2=\frac{1}{|C_{\mathrm{k}}|}\sum_{\substack{\mathrm{x_i}\in C_k\\ \mathrm{x_{i'}}\in C_k}}\sum_{j=1}^{p}\left({x_{ij}}^2-2x_{ij}x_{i'j}+{x_{i'j}}^2\right)$$

$$=\frac{1}{|C_{\mathrm{k}}|}\left(|C_k|\sum_{\mathrm{x_i}\in C_k}\sum_{j=1}^{p}{x_{ij}}^2+|C_k|\sum_{\mathrm{x_{i'}}\in C_k}\sum_{j=1}^{p}{x_{i'j}}^2-2\sum_{\substack{\mathrm{x_i}\in C_k\\ \mathrm{x_{i'}}\in C_k}}\sum_{j=1}^{p}x_{ij}x_{i'j}\right)$$

$$=\frac{1}{|C_{\mathrm{k}}|}\left(2|C_k|\sum_{\mathrm{x_i}\in C_k}\sum_{j=1}^{p}{x_{ij}}^2-2\sum_{j=1}^{p}\left(\sum_{\mathrm{x_i}\in C_k}x_{ij}\right)\left(\sum_{\mathrm{x_i'}\in C_k}x_{i'j}\right)\right)$$

$$=\frac{1}{|C_{\mathrm{k}}|}\left(2|C_k|\sum_{\mathrm{x_i}\in C_k}\sum_{j=1}^{p}{x_{ij}}^2-2\sum_{j=1}^{p}\left(\sum_{\mathrm{x_i}\in C_k}x_{ij}\right)^2\right)$$

$$=2\sum_{\mathrm{x_i}\in C_k}\|x_i\|_2^2-\frac{2}{|C_{\mathrm{k}}|}\left\|\mu_{\mathrm{C_k}}\right\|_2^2\ \ avec\ \mu_{\mathrm{C_k}}=\frac{1}{|C_k|}\sum_{\mathrm{x_i}\in C_k}x_i$$

$\mu_{\mathrm{C_k}}$ est le vecteur moyen des observations appartenant au cluster k (*centroïde* du cluster k).

$$=2\sum_{\mathrm{x_i}\in C_k}\|x_i\|_2^2-\frac{2}{|C_{\mathrm{k}}|}\left\|\sum_{\mathrm{x_i}\in C_k}x_i\right\|^2=2\sum_{\mathrm{x_i}\in C_k}\|x_i\|_2^2-2\sum_{\mathrm{x_i}\in C_k}x_i\cdot\mu_{\mathrm{C_k}}$$

$$=2\sum_{\mathrm{x_i}\in C_k}\|x_i\|_2^2-4\sum_{\mathrm{x_i}\in C_k}x_i\cdot\mu_{\mathrm{C_k}}+2\sum_{\mathrm{x_i}\in C_k}x_i\cdot\mu_{\mathrm{C_k}}$$

$$=2\sum_{\mathrm{x_i}\in C_k}\|x_i\|_2^2-4\sum_{\mathrm{x_i}\in C_k}x_i\cdot\mu_{\mathrm{C_k}}+2\left\|\mu_{\mathrm{C_k}}\right\|_2^2$$

$$=2\sum_{\mathrm{x_i}\in C_k}\|x_i\|_2^2-4\sum_{\mathrm{x_i}\in C_k}x_i\cdot\mu_{\mathrm{C_k}}+2|C_{\mathrm{k}}|\sum_{\mathrm{x_i}\in C_k}\left\|\mu_{\mathrm{C_k}}\right\|_2^2$$

$$=2\sum_{\mathrm{x_i}\in C_k}\left(\|x_i\|_2^2-2x_i\cdot\mu_{\mathrm{C_k}}+\left\|\mu_{\mathrm{C_k}}\right\|_2^2\right)$$

$$=2\sum_{\mathrm{x_i}\in C_k}\left\|x_i-\mu_{\mathrm{C_k}}\right\|_2^2$$

Le problème d'optimisation peut donc s'écrire sous la forme suivante :

$$\underset{C_1,\ldots,C_K}{\text{minimiser}}\left\{\sum_{k=1}^{K}\sum_{\mathrm{x_i}\in C_k}\left\|x_i-\mu_{\mathrm{C_k}}\right\|_2^2\right\}$$

Le problème consiste maintenant à partitionner les observations en K clusters de sorte à minimiser, sur l'ensemble des clusters, la somme des distances euclidiennes au carré entre chaque observation et le centroïde de son cluster (fonction objectif). Faire apparaître les centroïdes permet d'exprimer le problème initial uniquement en fonction des distances entre chaque observation et un représentant du cluster, ce qui simplifie considérablement l'analyse et sa résolution.

II.4. Algorithme

Le problème d'optimisation reste très difficile à résoudre exactement, puisqu'il existe approximativement K^n façons de répartir n observations en K clusters. Ce nombre devient colossal dès que K ou n ne sont pas extrêmement petits.

Heureusement, un algorithme très simple permet d'obtenir un optimum local (une solution généralement satisfaisante) pour ce problème d'optimisation. Cet algorithme, inventé par Lloyd en 1957, est fondé sur l'heuristique gloutonne. Il consiste à :

1. Choisir au hasard k observations dans D : elles constituent les centres initiaux des k clusters en construction.
2. Ce choix étant fait, on répète les deux opérations suivantes jusqu'à ce que les clusters soient stables :
   a. On place chaque observation dans le cluster dont le centre est le plus proche.
   b. On calcule le centre des clusters

Algorithme des k moyennes

On note $\mathrm{j}(\mathrm{x_i})$ l'indice du cluster de $\mathrm{x_i}$
K : Nombre de clusters ; x : observation

K_MOYENNES (x, K) :

Choisir au hasard les centres μ_{C_1}, μ_{C_2}, μ_{C_K} parmi les observations $\mathrm{x_i}$
Tant que les clusters ne sont pas stables :

Pour tout $\mathrm{i} \in \{1, \ldots, \mathrm{n}\}$:

$$\mathrm{j}(\mathrm{x_i}) \leftarrow \underset{\mathrm{j}\in\{1,..,\mathrm{K}\}}{\arg\min} \left\| x_i - \mu_{\mathrm{C_j}} \right\|_2^2$$

Pour tout $\mathrm{k} \in \{1, \ldots, \mathrm{K}\}$:

$$\mu_{\mathrm{C_k}} \leftarrow \frac{1}{|C_\mathrm{k}|} \sum_{\mathrm{x_i}\in C_\mathrm{k}} x_i = \frac{1}{|C_\mathrm{k}|} \sum_{\mathrm{x_i}\in C_\mathrm{k}} \sum_{j=1}^{p} x_{ij}$$

Renvoyer $C_1, C_2, \ldots, C_\mathrm{K}$

La complexité du calcul de la norme d'un vecteur de $\mathbb{R}^p$ est en O(p), or chaque itération de la boucle « Tant que » de l'algorithme nécessite le calcul de n·k telles distances, d'où une complexité en O(p·n·k) pour chaque itération. Ainsi, la complexité totale de l'algorithme est en O(m·p·n·k) où m est le nombre d'itérations de la boucle « Tant que ».

En pratique, le nombre d'itérations m est souvent faible (quelques dizaines), mais il n'existe pas de garantie simple en O(n) (k, p et m étant négligeables devant n) au pire cas ; on fixe souvent un maximum d'itération.

II.5. Convergence

Lorsque le résultat ne change plus, un optimum local a été atteint. La Figure 3 illustre la progression de l'algorithme appliqué à un jeu de données simulé. L'algorithme des k moyennes tire son nom du fait qu'à l'étape 2(b), les centroïdes des clusters sont calculés comme la moyenne des observations affectées à chaque cluster.

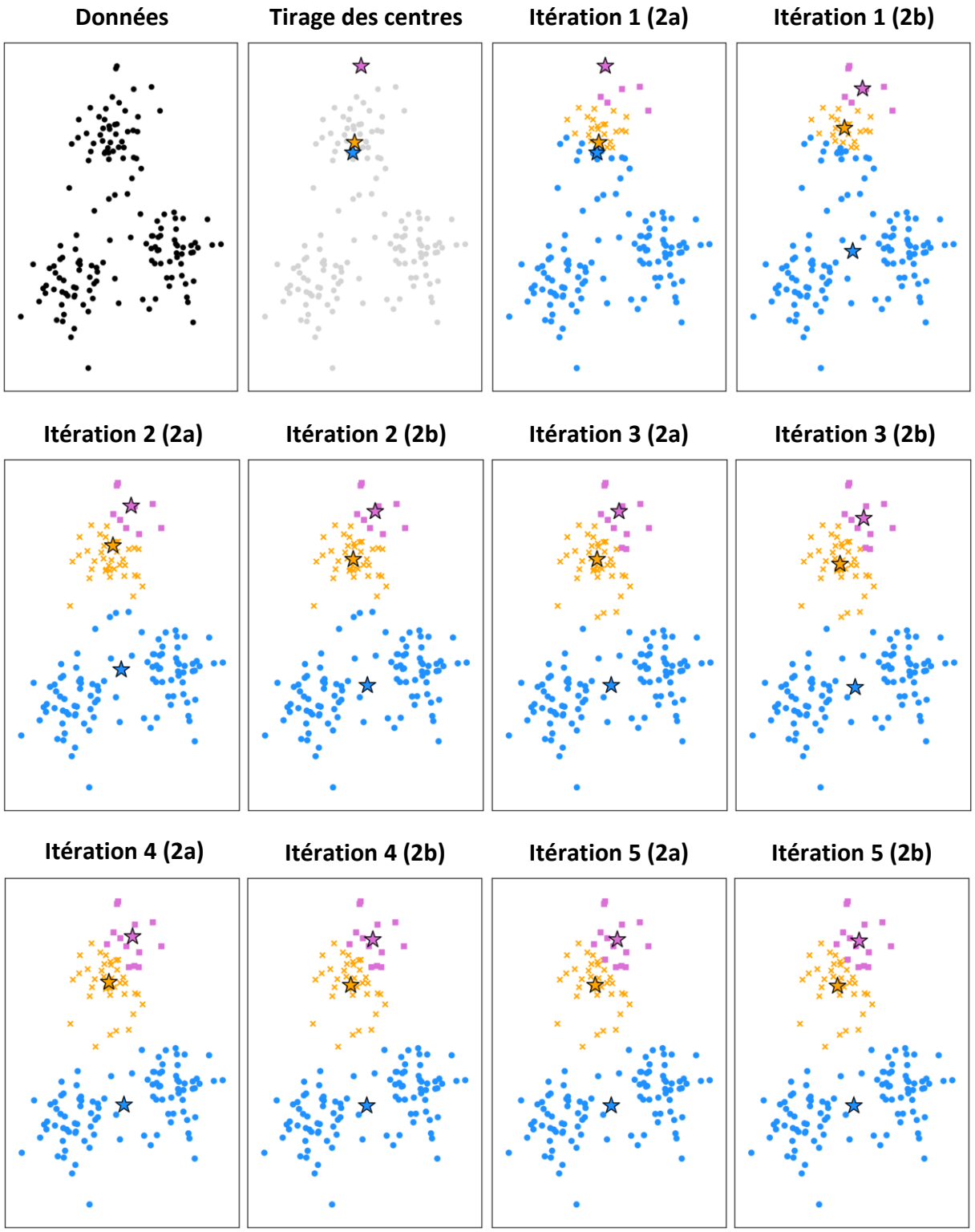

Figure 3 : Progression de l'algorithme des k moyennes avec K = 3

II.6. Impact des choix initiaux

Comme l'algorithme des k moyennes converge vers un optimum local et non vers un optimum global, les résultats obtenus dépendent de l'affectation initiale (aléatoire) des observations aux clusters à l'étape 1 de l'algorithme. Pour cette raison, il est important d'exécuter l'algorithme plusieurs fois à partir de différentes configurations initiales aléatoires. On sélectionne ensuite la meilleure solution, c'est-à-dire celle pour laquelle la fonction objectif est minimale.

La Figure 4 illustre les optima locaux obtenus en lançant l'algorithme six fois avec six affectations initiales différentes, sur un jeu de données simulé. Dans cet exemple, le meilleur partitionnement est celui dont la valeur du critère est égale à 380.0.

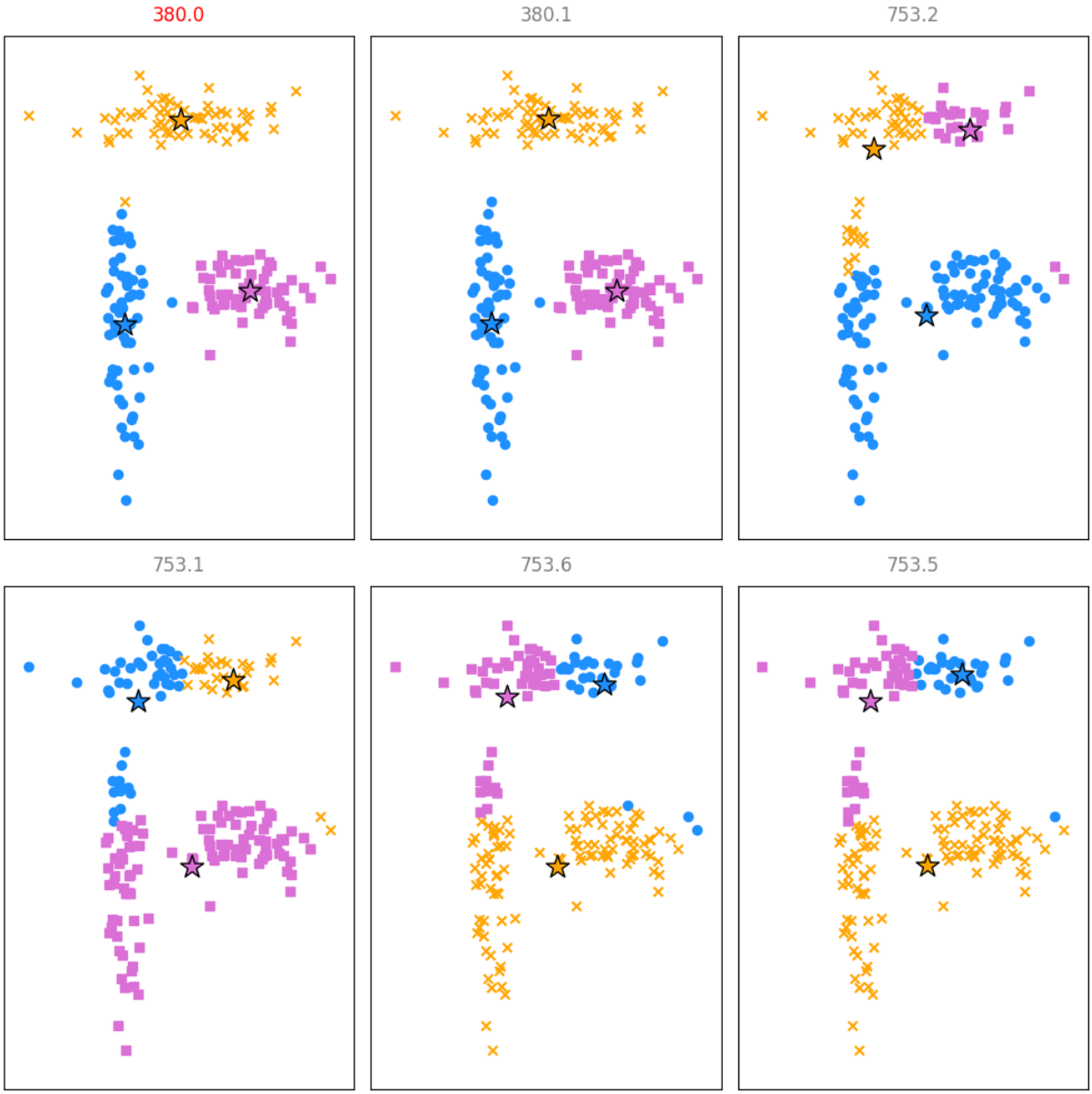

Figure 4 : Algorithme des k moyennes lancé six fois sur un jeu de données simulé, K=3

II.7. Choix du nombre de clusters

Comme nous l'avons vu, pour appliquer l'algorithme des k-moyennes, il est nécessaire de choisir le nombre de clusters que l'on suppose présents dans les données. Le choix de K ne possède pas de solution universelle. Plusieurs critères complémentaires sont utilisés selon le contexte (exploration, modélisation, contraintes métier).

La méthode la plus classique est le critère du coude (*elbow method*). On trace l'inertie intra-classe en fonction de K :

$$J(K) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_{C_k}\|_2^2$$

On choisit la valeur où la décroissance de J(K) ralentit nettement. La limite de cette méthode est que le « coude » peut être peu marqué ou subjectif.

La Figure 5 montre une courbe de coude typique pour le jeu de données utilisé dans les illustrations de la Figure 4:

- Forte décroissance de J(K) pour K entre K=1 et K=3,
- Ralentissement net après K=3

Le coude apparaît ici autour de K=3, ce qui est cohérent avec la structure du jeu de données utilisé (3 amas).

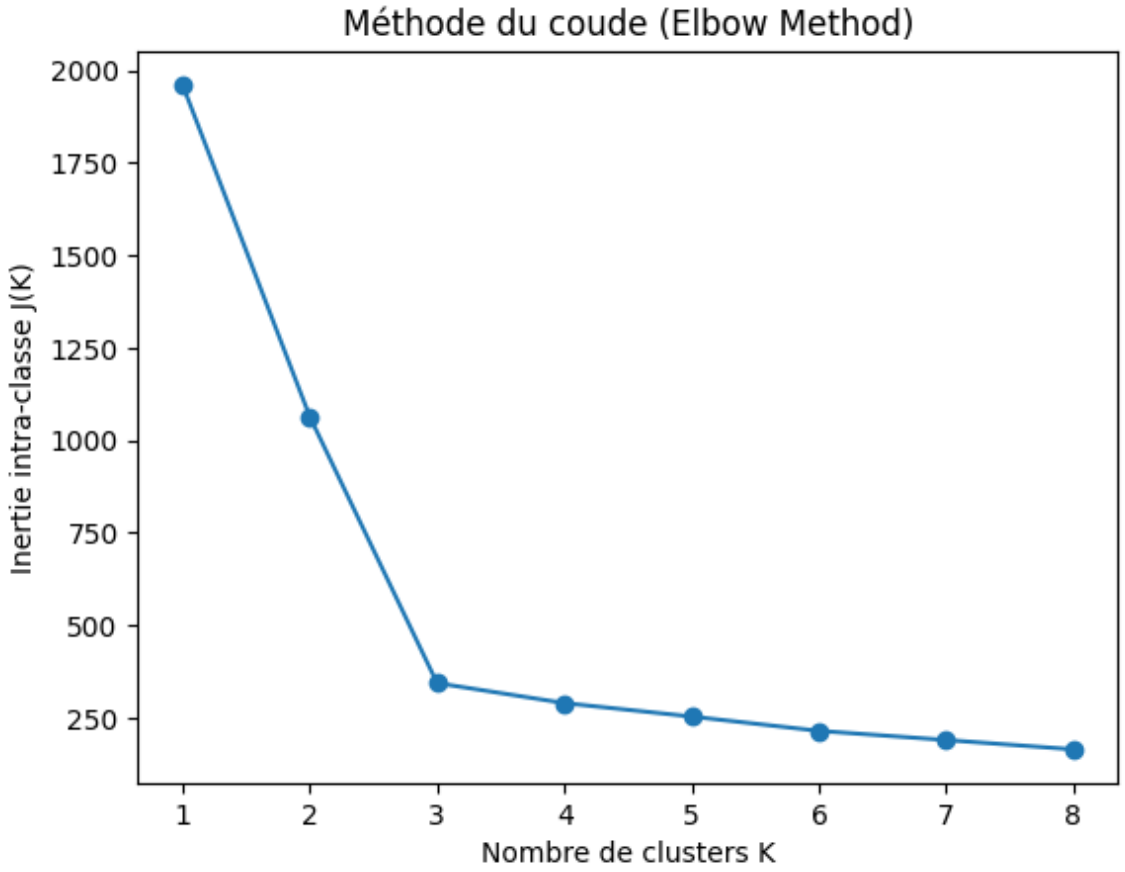

Figure 5 : Courbe de coude typique

III) EXEMPLES D'IMPLÉMENTATION EN PYTHON

Pour implémenter en Python l'algorithme des k moyennes, définissons deux sous-programmes.

III.1. Code de l'étape 2a : Attribution des observations aux clusters

Le premier permet de trouver l'indice du cluster le plus proche d'une liste d'observations x. Il effectue l'étape 2a de l'algorithme :

$$j(x_i) \leftarrow \underset{j\in\{1,..,K\}}{\arg\min} \left\| x_i - \mu_{C_j} \right\|_2^2$$

Remarque : Le code ci-dessous remplace $\left\| x_i - \mu_{C_j} \right\|_2^2$ par $\left\| x_i - \mu_{C_j} \right\|_2$ dans l'argmin. Cela est possible car les deux quantités induisent le même ordre sur les distances (croissance stricte sur $\mathbb{R}^+$.

```
# x : Matrice des observations, de forme (n,p)
# centres : Matrice des centroïdes, de forme (K,p)

def CALCULER_CLUSTERS(x,centres):
    dist=[]
    for z in centres:
        dist.append(np.linalg.norm(x-z,axis=1))
    dist=np.array(dist)
    y=np.argmin(dist,axis=0)
    return y
```

III.2. Code de l'étape 2b : Calcul des centroïdes

Le deuxième sous-programme reçoit les observations avec leurs étiquettes (clusters attribués) provisoires et calcule les centres des clusters. Il implémente l'étape 2b de l'algorithme :

$$\mu_{C_k} \leftarrow \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$

```
# x : Matrice des observations, de forme (n,p)
# y : Vecteur des labels, de forme (n), indiquant le cluster
#     de chaque observation

def CALCULER_CENTRES(x,y):
    centres=[]
    for label in np.unique(y):
        centre=np.mean(x[y==label],axis=0)
        centres.append(centre)
    return np.array(centres)
```

III.3. Algorithme complet

Enfin, l'algorithme complet : après avoir initialisé les centres, on effectue une boucle dans laquelle les algorithmes précédents sont appelés :

```
# x : Observations, de forme (n,p)
# k : Nombre de clusters

def K_MOYENNES(x,K):
    x=np.array(x)

    # Initialisation des centres
    indices=np.random.choice(len(x),size=K,replace=False)
    centres=x[indices]

    # Initialisation des clusters
    y=CALCULER_CLUSTERS(x,centres)
    y_prec=None

    # Boucle
    while not np.all(y==y_prec):
        y_prec=y.copy()
        centres=CALCULER_CENTRES(x,y)
        y=CALCULER_CLUSTERS(x,centres)
    return y,centres
```