Projet Calcul Scientifique et Statistique

Master 1 Mathématiques

Université Paris Diderot

Clustering : méthode des k-means

Rahma Bourredjem
Raphaël Kanapitsas

20-04-2020

Table des matières

1. Introduction

Le machine learning est sans conteste l'un des domaines phares de ces derniers temps. Ses contributions sont multiples, notamment dans le partitionnement de données (data clustering) où il intervient aussi bien dans la segmentation de marché en marketing que dans la classification des plantes en biologie.

Il existe deux principaux types d'apprentissage ; le supervisé et le non-supervisé. Nous comptons pour chacun d'eux plusieurs algorithmes déjà élaborés plus ou moins adaptés à un type de données et ou de domaines.

L'une des méthodes les plus simples pour aborder la classification non-supervisée est celle des K-means, proposée dès la fin des années 50 par Stuart Lloyd. C'est sur elle que portera notre projet. Son algorithme tient ses principaux atouts dans sa rapidité et sa facilité à implémenter et à calculer.

Nous commencerons par une présentation générale de l'algorithme, que illustrerons à travers une première application. Celle-ci sera l'occasion d'aborder quelque faiblesses. Nous tenterons d'en lever certaines par des procédés bien connus. Enfin, nous terminerons par une application concrète et plus classique en mettant en œuvre notre algorithme sur le classique jeu de données Iris.

2. Méthode des K-means

2.1 Présentation générale et formalisation

L'algorithme K-means est un algorithme d'apprentissage non supervisé. Cette procédure de classification consiste à trouver des tendances communes au sein d'un échantillon de données et de les rassembler en groupes (clusters) **différents** et **homogènes**, sans qu'il y ait de classe donnée à l'avance.

L'idée est la suivante : A partir d'un ensemble de N points $x_1,\ldots,\ x_N$, nous cherchons à déterminer pour un nombre de classes fixées à K, une répartition des points qui minimise le critère dit d'*inertie* ou *variance intra-classe* au sein d'un même groupe, tout en maximisant la *variance inter-classe*. Nous obtenons ainsi des clusters homogènes, bien séparés et distincts.

Cette condition provient de l'étude de la dispersion de notre nuage de points $x_1,\ldots,x_N$ autour de son barycentre $x_G{=}\frac{1}{n}\sum_{i=1}^{n} x_i$, à travers la notion d'inertie qui se décompose d'après la formule d'**Huygens** de la manière suivante :

$$I_T = I_W + I_B.$$

Où : $\begin{cases} I_T & \text{représente l'inertie totale de l'ensemble,} \\ I_W & \text{l'inertie intra-classe (W pour within) et} \\ I_B & \text{l'inertie inter-classe (B pour Between)} \end{cases}$

Nous allons dans ce qui va suivre retrouver cette formule, et par conséquent, le critère de variance sur lequel se base le procédé des K-means.

Formellement, en choisissant de travailler avec la distance euclidienne, nous définissons l'inertie totale par :

$$I_T(G) = \sum_{i=1}^{n} \|x_i - x_G\|^2$$

Dans le cadre de la méthode des K-means, nous supposons qu'il existe K classes distinctes, notées C_1, C_2, ..., C_K, dans lesquelles nous regroupons nos points. Nous associons à chacune de ces classes, son barycentre respectif x_{C_k}, l'inertie totale se décompose alors :

$$
\begin{aligned}
I_T(G) &= \sum_{i=1}^{n} \|x_i - x_G\|^2, \text{ en sommant sur les K classes, nous avons :} \\
&= \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - x_G\|^2 \\
&= \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - x_{C_k} + x_{C_k} - x_G\|^2 \\
&= \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - x_{C_k}\|^2 + \|x_{C_k} - x_G\|^2, \text{ (d'après les propriétés du produit scalaire et du barycentre).} \\
&= \underbrace{\sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - x_{C_k}\|^2}_{\textbf{inertie intra-classe}} + \underbrace{\sum_{k=1}^{K} n_k \|x_{C_k} - x_G\|^2}_{\textbf{inertie inter-classe}}, \text{ avec } n_k \text{ le nombre d'élements de la classe } C_k.
\end{aligned}
$$

Le premier terme exprime la dispersion des points d'une même classe autour de leur barycentre, il s'agit donc de l'inertie intra-classe. Celle-ci est faible lorsque les classes sont homogènes, c'est-à-dire lorsque les points d'une même classe sont proches les unes des autres. Le second terme quantifie, quant à lui, la distance entre les barycentres des classes et le barycentre central de notre distribution. Il indique ainsi à quel point les classes sont distantes les une des autres, et mesure par conséquent l'inertie inter-classe. Nous retrouvons donc la formule de Huygens.

Il s'en suit que pour obtenir un bon regroupement de nos éléments en K clusters homogènes et séparés, nous devons choisir la partition qui minimise l'inertie intra-classe I_W (ou qui maximise I_B, cela revient au même d'après la formule que nous avons retrouvée). Nous définissons cette partition optimal $\mathcal{C}_K^*$ ci-dessous, avec $\mathcal{C}_\mathcal{K}$ l'ensemble des partitions possibles.

$$
\mathcal{C}_K^* = \underset{\mathcal{C} \in \mathcal{C}_\mathcal{K}}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{i \in C_k} \|x_i - x_{C_k}\|^2
$$

Ainsi, la variance intra-classe peut être vue comme une fonction de coût paramétrée sur l'ensemble des partitionnements possibles. Cependant, en pratique, il nous est impossible de visiter toutes les configurations de classes possibles au vu de la complexité combinatoire du problème. C'est pour cette raison que nous devons faire des choix stratégiques sur les partitions à visiter lors de l'initiation, nous étudierons cette sensibilité dans la partie 4.

Le but de l'algorithme va donc être de trouver le partitionnement minimisant cette variance globalement dans l'idéal, ou localement dans une moindre mesure.

2.2 Algorithme

Comme nous l'avons expliqué précédemment, la méthode des centres mobiles consiste à attribuer à chaque élement de notre ensemble de données, une classe parmi les K disponibles.

Nous nous proposons dans cette partie d'expliquer les différentes étapes de l'algorithme qui, rappelons-le, a pour objectif de trouver la répartition minimisant le critère de la variance intra-classe.

2.2.1 Principe

Notons avant de commencer que le nombre de classes K doit être fixé à l'avance, nous reviendrons sur la délicatesse de ce choix par la suite.

Considérons à présent notre ensemble d'apprentissage $\{X_1,..., X_n\}$, avec $X_i \in \mathbb{R}^d$, l'algorithme des K-means se déroule globalement en deux temps :

1. **Initialisation** : En supposant qu'il contienne K classes distinctes, nous commençons par choisir aléatoirement les centres de ces classes, μ_1, μ_2, ..., μ_K parmi les données.
2. **Procédure itérative** :

   Dans un deuxième temps, nous alternons les deux étapes suivantes :

   — Nous attribuons à chacun des n-K élements restant, la classe C_i dont le centre μ_i est le plus proche de l'élément X_i considéré. Ainsi nous construisons une *partition* de notre ensemble. En pratique, nous cherchons pour chaque X_i, l'indice du centre μ_j qui minimise la distance :

   $$\|X_i - \mu_j\|^2$$

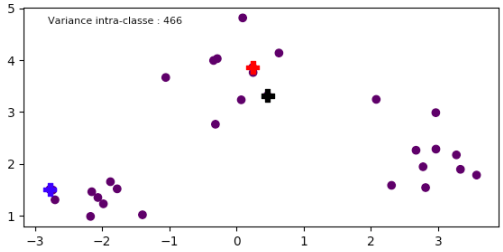
   — Puis, chaque cluster C_k se voit attribuer un *nouveau centre* μ_k [1], qui n'est autre que le barycentre de ses points, calculé via la formule ci-dessous où C_k représente l'ensemble des éléments affectés au centre k.

   $$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} X_i$$

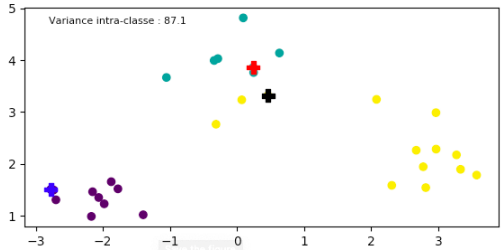
   Ce procédé est réitéré jusqu'à la stabilisation de l'algorithme, c'est à dire jusqu'à ce que les clusters formés restent inchangés entre deux itérations.

Voici une illustration de chaque étape de l'algorithme sur un jeu de données aléatoire (généré en partant de trois points auxquels on ajoute des perturbations suivant une loi normale) :

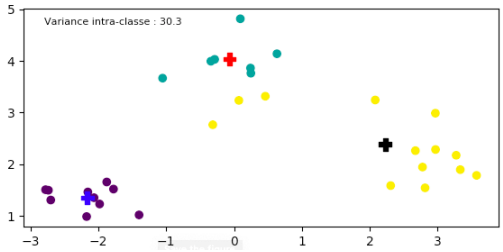
1. μ_k représente ici le barycentre de la classe C_k que nous avons noté x_{C_k} dans la section précédente.



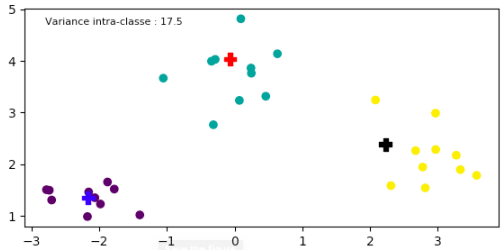
(a) 3 points sont choisis aléatoirement, ils font offices de centres initiaux.



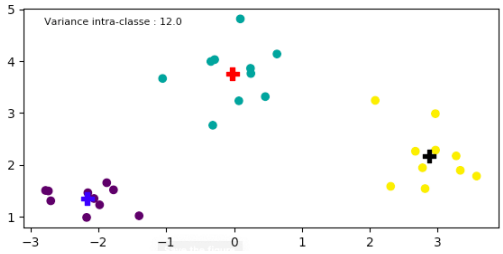
(b) Une première partition est formée ; les points sont regroupés autour des centres les plus proches.



(c) De nouveaux centres de classes sont définis comme étant les barycentres des clusters nouvellement créés.



(d) Construction d'une nouvelle partition ; les points sont regroupés dans de nouvelles classes autour des centres calculés précédemment.



(e) Les classes obtenues restent les même après la définition des nouveaux centres ; stabilité et fin de l'algorithme.

Nous remarquons que la variance intra-classe décroît à chaque itération de l'algorithme, montrons ce résultat.

2.2.2 Convergence de K-means

Théorème (Convergence des k-means). *Quelque soit l'initialisation choisie, la suite* $(I_w^{(t)})_{t\geq 0}$ *des variances intra-classe construite par l'algorithme des K-means converge. Avec :*

$$I_w^{(t)} = \sum_{k=1}^{K} \sum_{i \in C_k} \|X_i - \mu_k^{(t)}\|^2$$

La démonstration du théorème nécessite le lemme suivant :

Lemme (Inertie minimum). *Soit* $\{X_1, ..., X_n\}$, *avec* $X_i \in \mathbb{R}^d$, *n points de* $\mathbb{R}^d$.

Le minimum de la quantité $Q(Y) = \sum_{i=1}^{n} \|X_i - Y\|^2$, *avec* $Y \in \mathbb{R}^d$ *est atteint en* $Y = G = \frac{1}{n}\sum_{i=1}^{n} X_i$, *où* G *est le barycentre des points* $(X_1, ..., X_n)$.

Preuve (Lemme)**.** *Soit* $Y \in \mathbb{R}^d$, *nous avons :*

$$Q(Y) = \sum_{i=1}^{n} \|X_i - Y\|^2 = \sum_{i=1}^{n} \|X_i - G + G - Y\|^2$$

En utilisant les propriétés du produit scalaire et celle de l'identité remarquable, nous obtenons :

$$\|X_i - G + G - Y\|^2 = \|X_i - G\|^2 + \|G - Y\|^2 + 2\langle X_i - G, G - Y\rangle$$

Ainsi, en sommant sur les i, nous avons :

$$\sum_{i=1}^{n} \|X_i - G\|^2 + n\|G - Y\|^2 + 2\sum_{i=1}^{n} \langle \overrightarrow{GX_i}, \overrightarrow{YG}\rangle$$

Or G est le barycentre des points $(X_1, ..., X_n)$, *nous avons alors par définition* $\sum_{i=1}^{n} \overrightarrow{GX_i} = 0$.
Ainsi :

$$\sum_{i=1}^{n} \overrightarrow{GX_i} = 0 \Rightarrow \sum_{i=1}^{n} \langle \overrightarrow{GX_i}, \overrightarrow{YG}\rangle = 0.$$

D'où :

$$Q(Y) = \sum_{i=1}^{n} \|X_i - G\|^2 + n\|G - Y\|^2.$$

Le minimum de Q(Y) est donc atteint pour Y vérfiant $G - Y = 0$, *c'est-à-dire* $Y = G$ *(le barycentre).*
D'où le résultat.

Preuve (Théorème)**.** *Considérons une itération de l'algorithme.*
Soient $C_1^{(t-1)}, ..., C_K^{(t-1)}$ *les clusters formés à* $t-1$, *notons* $\mu_i^{(t-1)}$ *le centre du cluster* $C_i^{(t-1)}$.
Soient $C_1^{(t)}, ..., C_K^{(t)}$ *la nouvelle partition assignée à l'instant t.*
Montrons que la suite des variances intra-classe $(I_w^{(t)})_{t\geq 0}$ *décroît à cette itération, i.e :*

$$\sum_{k=1}^{K} \sum_{i \in C_k^{(t)}} \|X_i - \mu_k^{(t)}\|^2 \leq \sum_{k=1}^{K} \sum_{i \in C_k^{(t-1)}} \|X_i - \mu_k^{(t-1)}\|^2$$

D'apès le ***lemme d'inertie minimum****, à k fixé, le minimum pour* $i \neq j$ *de* $\sum_{i \in C_k} \|X_i - X_j\|^2$ *est atteint en* μ_k.
Ainsi :

$$\sum_{i \in C_k^{(t)}} \|X_i - \mu_k^{(t)}\|^2 \leq \sum_{i \in C_k^{(t)}} \|X_i - \mu_k^{(t-1)}\|^2$$

En sommant sur toutes les classes, nous avons :

$$\sum_{k=1}^{K} \sum_{i \in C_k^{(t)}} \|X_i - \mu_k^{(t)}\|^2 \leq \sum_{k=1}^{K} \sum_{i \in C_k^{(t)}} \|X_i - \mu_k^{(t-1)}\|^2$$

Or, à l'itération t, nous attribuons à chaque X_i le centre μ_k qui lui est le plus proche. Par définition, ce centre est issu de la collection de centres calculés à l'étape t-1, i.e $(\mu_1^{(t-1)}, ..., \mu_K^{(t-1)})$.
Ainsi, la partition $C_1^{(t)}, ..., C_K^{(t)}$ minimise le membre de droite de la précedente inéquation sur toutes les partitions $\mathcal{C}_k$, ce qui donne :

$$\sum_{k=1}^{K} \sum_{i \in C_k^{(t)}} \|X_i - \mu_k^{(t-1)}\|^2 \leq \sum_{k=1}^{K} \sum_{i \in C_k^{(t-1)}} \|X_i - \mu_k^{(t-1)}\|^2$$

Donc, nous avons bien le résultat :

$$\sum_{k=1}^{K} \sum_{i \in C_k^{(t)}} \|X_i - \mu_k^{(t)}\|^2 \leq \sum_{k=1}^{K} \sum_{i \in C_k^{(t-1)}} \|X_i - \mu_k^{(t-1)}\|^2$$

3. Application : reconnaissance de chiffres manuscrits

3.1 Présentation du problème

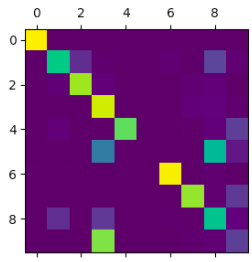
Dans cette section, nous allons tenter d'appliquer la méthode des K-Moyennes à un jeu de données réelles. Celui-ci comporte près de 6000 images de chiffres (entre 0 et 9) écrits à la main. Les images ont une taille de 8 par 8 pixels, la dimension est donc de 64. Chaque pixel est codé par un entier compris entre 0 et 16 (0 étant le blanc, 16 le noir). Ces données sont disponibles sur le site de l'UCI. Dans un premier temps, nous prendrons $K = 10$, qui semble être le choix le plus naturel. Le fichier `optdigits.tra` sera utilisé pour l'apprentissage, et `optdigits.tes` pour tester la catégorisation obtenue. Le fichier `kmeans_digits.py` contient le code spécifique à cet exemple. On désignera par *label* la classification correcte de l'image.

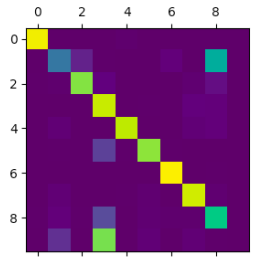
Voici l'idée de la démarche :

1. On charge le fichier `optdigits.tra` en supprimant la dernière colonne, qui contient le *label*.
2. On lance l'algorithme des K-Means dessus, en prenant $K = 10$. L'attente est que chaque chiffre corresponde à une des classes ainsi obtenues. L'initialisation est aléatoire, et plusieurs itérations de l'algorithme sont faites. Les résultats (c'est à dire la position des 10 centres), est enregistrée dans le fichier `digits.result`.
3. Vient ensuite la phase de test. On charge le fichier test, et pour chaque point, on regarde quel centre est le plus proche pour savoir à quel classe il appartient. On compare cela au *label*, pour connaître le taux d'erreur. Le chiffre qu'une classe prédit est donnée par la majorité des *labels* de ses points.
4. Le résultat est donnée sous forme d'une matrice carré A de dimension 10, où A_{ij} donne le nombre de d'image où i est le chiffre correct, et j la chiffre prédit (ces matrices seront affichées graphiquement, pour mieux les visualiser).

3.2 Sensibilité à l'initialisation

Utilisons tout d'abord cet exemple pour illustrer la sensibilité à l'initialisation de la méthode des K-Means. En effet, l'initialisation étant aléatoire, elle peut être très mauvaise, et faire en sorte que l'algorithme s'arrête dans un minimum local de l'inertie intra-classe. En pratique il est difficile de garantir une solution optimale, mais on peut s'en approcher.
L'image de droite montre les résultats d'une seule itération de l'algorithme, pour une initialisation particulièrement mauvaise : le taux de bonne réponse est d'environ 67%.

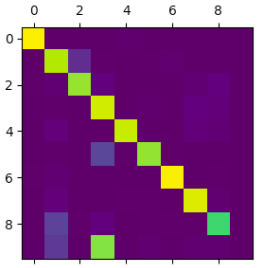




L'image de gauche montre le résultat d'une initialisation un peu meilleure : cette fois-ci, 74% des images sont correctement identifiées. En particulier, le 5 est bien mieux reconnu, mais on a toujours le problème de distinction entre le 3 et le 9. Dans d'autres cas, ce problème n'existe pas. Mais en général, il y a toujours un chiffre qui pose problème.
C'est pour palier à ce problème que l'algorithme est lancé plusieurs fois, en gardant le meilleur résultat. Cela ne garanti pas un résultat optimal, mais permet de s'en approcher. Nous évoquerons dans la partie suivante, une méthode permettant de faire une initialisation de meilleure qualité.

3.3 Premiers résultats

En faisant une centaine d'itérations de l'algorithme, pour s'assurer d'obtenir de bonnes performances (au sens de l'inertie intra-classe), voilà le résultat obtenu. Le taux de bonnes réponses est d'environ 80%. Mais on constate un gros problème : les 3 et les 9 sont confondus.



Deux origines viennent à l'esprit. L'algorithme des K-means est connu pour ses performances qui se dégradent lorsque le nombre de dimensions augmente. Ici nous sommes en dimensions 64, ce qui n'est pas si élevé. Plus probablement, il est possible que les nuages de points réels correspondants à chaque chiffre, ne soient pas tous convexes, ou même connexes. De telles situations mettent cette méthode en défault, car les partitions de R^d formées en prenant le centre le plus proche, sont forcément convexes [1].

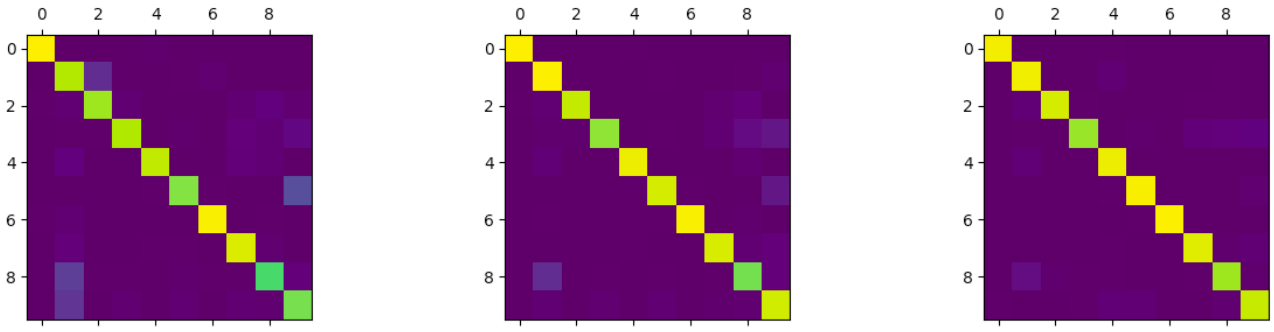
Une solution possible serait de traiter les données avant de les fournir à l'algorithme K-means. Ici, nous allons explorer une autre solution, qui se prête bien à ces données particulières.

3.4 Meilleures prédictions et apprentissage supervisé

L'idée de cette section est simple : augmenter K. En effet, cela va nous permettre de décrire nos données avec plus de détail. En contrepartie, la correspondance idéalisée entre classe et chiffre sera perdue. Il est alors nécéssaire d'associer un **ensemble** de centres à chaque chiffre. En faisant cela, nous sortons quelque peu du cadre, pour rentrer dans l'apprentissage supervisé. En effet, il sera nécéssaire de connaître les *labels* pour associer les centres aux chiffres. [2]

1. En effet, considérons E_{ij}^- les demi-espace formés par les points qui sont plus proches du centre i que du centre j. Alors la partition associé à ce centre est l'intersections des E_{ij}^- pour $i \neq j$. Elle est donc convexe, comme intersection de convexes.

2. On pourrait penser que cette situation est la même que dans les sections précédentes. Mais là où on utilisait les *labels* uniquement pour donner un nom aux classes, on les utilise ici pour créer des "classes de classes".



(a) $K = 11$
$p = 0,87$

(b) $K = 20$
$p = 0,92$

(c) $K = 100$
$p = 0,95$

FIGURE 3.1

La démarche reste la même qu'en 3.1, avec une étape en plus entre (3) et (4), qu'on appellera "classification" :

1. En utilisant les données d'apprentissage, on regarde, pour chaque classe associée à un centre, quel *label* est présent en majorité. Celui-ci dictera quel chiffre prédit ce centre.
2. Grâce à cela, on dresse une liste qui contient en position i le chiffre associé au centre i.
3. Lors du test, on regarde quel centre est le plus proche du point, puis on regarde à quel chiffre ce centre est associé.

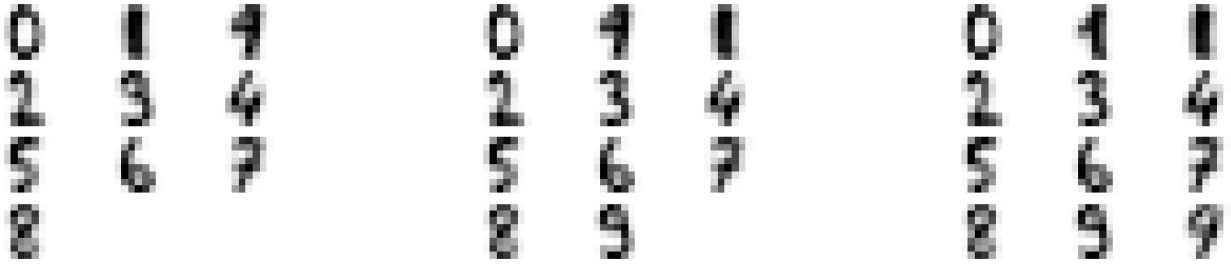
Cette fois-ci, nos partition de l'espace sont des **unions** de régions associées à des centres. Elles peuvent donc être non-convexe et non-connexes. Grâce à cette méthode, on peut obtenir de bien meilleures prédictions. Dans la figure 3.2, on a lancé 10 itérations de l'algorithme pour différentes valeurs de K (p représente le taux de prédictions correctes). De manière plutôt surprenante, simplement passer de 10 à 11 fait disparaitre le problème de distinction entre 3 et 9, et fait passer le taux d'erreur de 20% à 13%. Globalement, plus K est grand, plus le taux d'erreur est faible.

Cependant, cela reste à relativiser :

- Plus K est grand, plus on perd "l'esprit" de K-means. En effet, cela vient à réduire la taille des classes, et l'importance de la convergence de l'algorithme. Prenons le cas extrême où $K = n$ le nombre de points. Alors chaque point des données d'apprentissage est un centre, et l'algorithme revient simplement à trouver le plus proche point parmis les données d'apprentissage.
- D'autre part, augmenter K a un effet linéaire sur la taille nécessaire pour représenter les données, et sur le temps de calcul lors du test (puisqu'il faut comparer la distance d'un point à chaque centre). Dans le cadre de ces données, cela a peu d'importance, mais pour des applications réelles avec une quantité de données biens supérieure, c'est primordial.
- Finalement, on risque la sur-interprétation. En effet, en gagnant en précision, la prédiction est plus sujette à des petites variations dans les données.

3.5 Analyse des centres

Pour terminer cette étude de cas, et tenter en particulier de comprendre l'origine de la différence significative entre $K = 10$ et $K = 11$, nous allons regarder à quoi ressemblent les centres trouvés par K-means. Les données étant des images, on peut visualiser les centres comme telles.



(a) $K = 10$ (b) $K = 11$ (c) $K = 12$

FIGURE 3.2 – Exemples des centres obtenus pour trois valeurs de K différentes (15 itérations)

Ces images apportent immédiatement des réponses. Premièrement, on constate que dans le cas $K = 10$ (et les autres aussi), deux centres sont en fait des 1. L'un écrit avec un seul trait, l'autre avec deux. D'autre part le centre correspondant au 3 ressemble beaucoup à un 9, comme si on avait fusionné les deux. Ce qui se passe est alors clair : il y a une différence plus grande entre les deux types de 1, plutôt qu'entre 3 et 9. L'algorithme atteint alors le minimum de la l'inertie intra-classe lorsqu'il associe deux centres aux deux types de 1, plutôt que de différencier les 3 et 9.

Cela explique le bond de performance lorsqu'on prend $K = 11$. Le nouveau centre disponible fait disparaître le "dilemme" : on peut distinguer les deux types de 1, et avoir deux centres dédiés aux 3 et aux 9. De manière similaire, pour $K = 12$, on retrouve en plus deux types de 9 : les "ronds" ou les "droits". De manière générale, plus K sera grand, plus on va distinguer des versions différentes de chaque chiffre, et plus ces différences seront minimes.

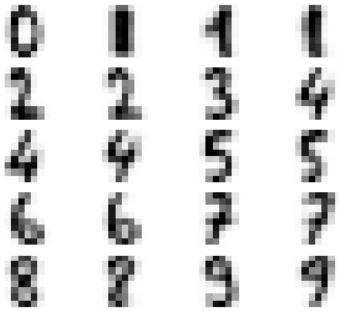


FIGURE 3.3 – Un dernier exemple avec $K = 20$

4. Améliorations possibles

4.1 Amélioration de de l'initialisation : Global K-means

En plus de la méthode d'initialisation aléatoire présentée précédemment, nous pouvons citer celle du Global K-means, qui vise également à palier à la sensibilité liée à l'initialisation.

Au lieu de sélectionner aléatoirement les centres initiaux des clusters comme c'est le cas avec avec le k-means, l'algorithme que nous allons présenter ici procède de manière incrémentielle en ajoutant un nouveau centre de cluster à chaque étape. L'objectif est donc de trouver ces centres puis d'appliquer la procédure itérative du k-means pour atteindre une solution globalement optimale.

La démarche est la suivante :

1. **Début** : Nous commençons par désigner le barycentre de notre essemble de données comme étant le premier centre μ_1 de la classe C_1.
2. **Procédure dynamique** : Puis, nous cherchons à déterminer le centre μ_k suivant (avec k variant de 2 à K), sachant que nous avons les k-1 précédents. Recurcivement, nous parcourons notre ensemble en considérant chaque élément X_k comme un prétendu candidat à notre collection de centre. Ainsi :
   — Nous fixons X_k comme un "pseudo-centre" et nous affectant les données restantes au centre le plus proche afin de former une partition.
   — Nous calculons la variance intra-classe,

$$\sum_{i=1}^{K} \sum_{i \in C_i} \|X_i - X_k\|^2$$

   — Enfin, nous sélectionnons l'élements X_k minimisant cette variance, qui devient notre k ieme centre μ_k.
   Nous réactualisons les centres et répétons jusqu'à obtenir nos K centres d'initialisation.
3. **Retour à k-means** : Nous appliquons la méthode des K-means avec cette collection de centres intiaux jusqu'à convergence.

Remarquons que le Global K-means est un algorithme déterministe, le résultat sera toujours le même pour un jeu de données fixé.

4.2 Choix du nombre K

4.2.1 Méthode naïve

Une première méthode, assez naturelle, pour choisir le nombre de classe, consiste à faire un graphe de l'inertie intra-classe en fonction de K. Naturellement, celle-ci décroit avec K. On choisit le moment où le graphe fait un "coude", c'est à dire là où on a une forte diminution de l'inertie intra-classe pour une augmentation de K.

Cette méthode a l'avantage d'être simple, et facile à implémenter. En revanche, elle manque de finesse, comme nous allons le voir dans la section suivante. Elle est de plus relativement subjective, dans le sens où elle nécéssite une interpretation humaine.

4.2.2 Méthode de Davies-Bouldin

4.2.2.1 La théorie

Nous avons vu dans la section précédente que l'algorithme K-means nécéssite de connaître K, le nombre de classes. En pratique, cela n'est pas toujours évident, voire complètement inconnu. Nous présentons donc un critère de qualité pour déterminer le nombre de classe optimal. L'indice de Davies-Bouldin sera minimal pour K optimal. Nous avons d'abord besoin de quelques définitions.

Définition 1 (Fonction de partition)**.** *La fonction de partition associe à chaque point sa classe :* $p : [\![1; N]\!] \longrightarrow [\![1; K]\!]$
On définit alors $C_k = \{k \in [\![1; N]\!] : p(k) = j\}$

Définition 2 (Écart-type intra-classe)**.** *Pour mesurer la dispersion des point à l'interieur de chaque classe, on pose :*

$$\sigma_k = \frac{1}{|C_k|} \left(\sum_{i \in C_k} \|X_i - \mu_k\|^2 \right)^{1/2} \qquad \forall k \in [\![1; K]\!]$$

Définition 3 (Indice de Davies-Bouldin)**.**

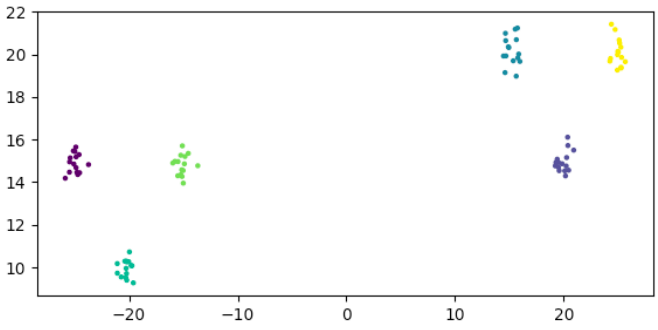
$$I_{DB} = \frac{1}{K} \sum_{i=1}^{K} \max_{i \neq j} \frac{\sigma_i + \sigma_j}{\|\mu_i - \mu_j\|}$$

Nous remarquons que I_{DB} est plus grand lorsque les centres des classes μ_k sont rapprochés et que la dispersion des points au sein des classes est grande. Au contraire, quand I_{DB} est petit, les classes sont éloignées et "compactes". Nous chercherons donc le K qui minimise I_{DB}.

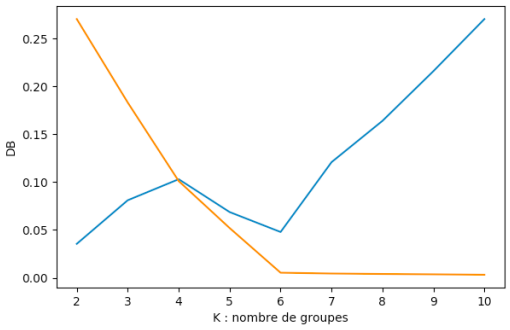
4.2.2.2 La pratique

Nous allons étudier le comportement de cet indice sur des exemples de données (en deux dimension pour pouvoir facilement les visualiser). Plus précisement, des données aléatoires sont générées (à partir d'une liste de centre, en rajoutant des perturbations suivant une loi normale), et l'algorithme est lancé avec différentes valeurs de K. Nous calculons I_{DB} pour chaque partition obtenue.

Dans le premier exemple (Figure 4.1), nous distinguons clairement deux minimums pour $I_{DB} : K = 2$ et $K = 6$. La première possiblité sépare les deux trios de groupes, et la deuxième sépare les six groupes qu'on distingue

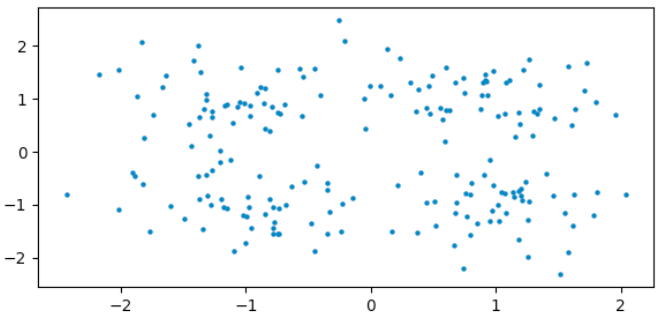


(a) Le jeu de donné (aléatoire)

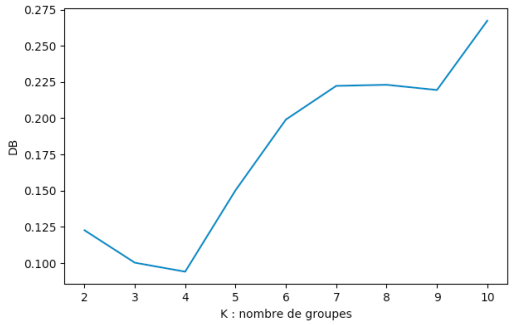


(b) en bleu I_{DB} en fonction de K

FIGURE 4.1



(a) Un jeu de donné moins "facile"



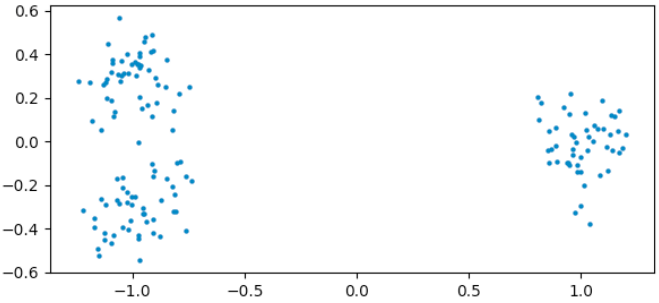
(b) I_{DB} en fonction de K

FIGURE 4.2

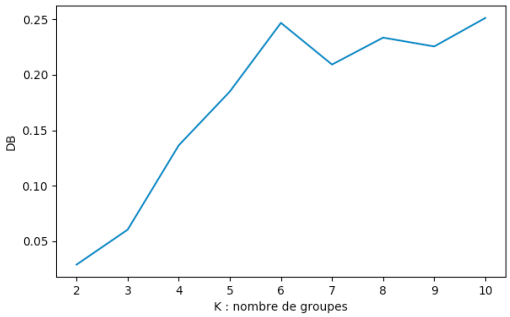
visuellement. Dans ce cas, I_{DB} semble très pertinent. Pour comparer, l'inertie intra-classe est tracée en orange. Nous voyons qu'avec la méthode du "coude" abordée plus haut, nous choisirions $K = 6$. Mais nous passons à côté du choix $K = 2$ qui pourrait être pertinent.

Dans le deuxième exemple (Figure 4.2), les données sont générée à partir de 4 centres : $(-1, -1)$, $(-1, 1)$, $(1, -1)$, $(1, 1)$. Cette fois-ci, la dispersion des points est plus grandes, les frontières moins évidentes. Mais même ici, I_{DB} atteint sont minimum en $K = 4$.

En revanche, dans certains cas, comme le montre le troisième exemple, le minimum de I_{DB} n'est pas forcément atteint pour le K qui serait considéré "optimal" pour un humain. Comme le montre la figure 4.3, il semble plus raisonnable de partitionner les données en 3 groupes plutôt que 2, mais I_{DB} atteint son minimum pour $K = 2$.



(a) Un jeu de donné qui parait moins ambigu, mais qui est problematique



(b) I_{DB} en fonction de K

FIGURE 4.3

5. Application : Iris

Enfin, nous terminons ce projet par l'étude d'un jeu de données bien connu dans le domaine de la classification, celui d'Iris présenté en 1936 par Ronald Fisher dans son papier *The use of multiple measurements in taxonomic problems*.

5.1 Présentation du problème

Le jeu est composé de 150 données, réparties de manière égale en trois espèces de fleurs d'Iris (Setosa, Versicolor et Virginica). Chaque catégorie compte 50 fleurs ayant chacune quatre attributs exprimés en cm : longueur des sépales (sepal length), largeur des sépales (sepal width), longueur des pétales (petal length), et largeur des pétales (petal width). On a en plus, le label correct pour chaque fleur. Ainsi, nos éléments sont representés par un vecteur à 4 dimension. Ces données sont disponibles sur le site de l'UCI. On utilisera le fichier `iris.data`, légerement modifié : les labels sous forme de texte sont remplacés par des entiers entre 1 et 3.

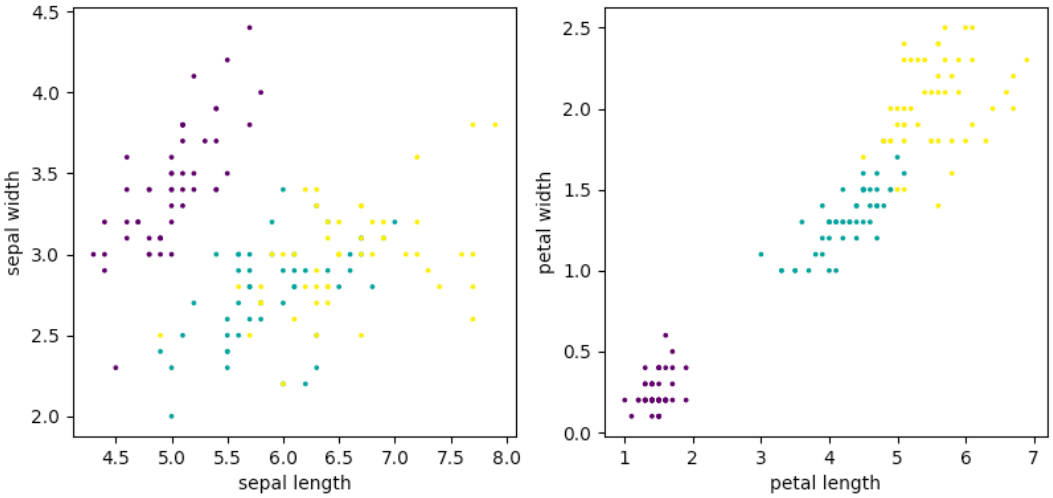


FIGURE 5.1 – Représentation des données, avec une couleur par type de fleur

5.2 Choix de K

Pour appliquer la méthode du k-means à cet exemple, nous devons d'abord fixer le nombre de clusters K. Il est naturel de prendre $K = 3$ ici puisque nous savons que les fleurs sont issues de trois différentes espèces. Voyons si les deux méthodes du choix de K que nous avons évoqués nous permette de retrouver cela.

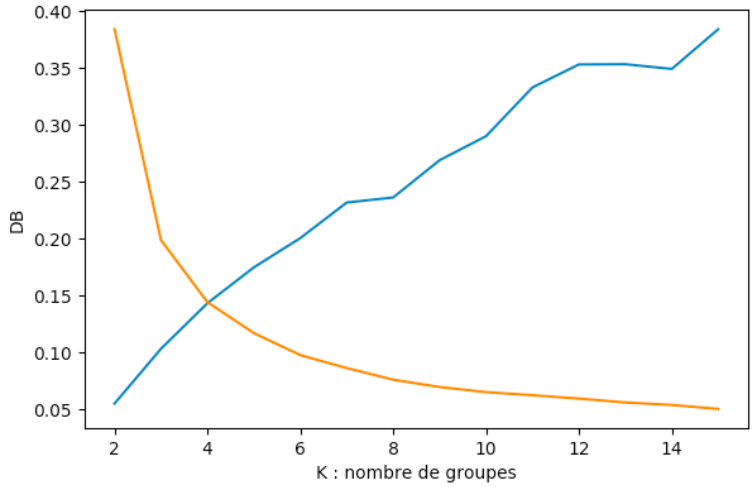


FIGURE 5.2 – En bleu I_{DB}, en orange l'inertie intra-classe (à un facteur multiplicatif près)

Ces deux indicateurs ne donne pas une réponse très claire. On voit tout de même qu'il n'est pas utile de prendre un grande valeur de K. On a une très forte diminution de l'inertie intra-classe au début, alors que I_{DB} augmente assez progressivement. Une lecture de ces graphes hors-contexte amènerai probablement à un choix de K entre 2 et 4.

5.3 Comparaison des deux méthodes d'initialisation

En réalisant plusieurs itérations de l'algorithme avec des initialisations aléatoire, on se rend compte qu'il y a deux type des résultats : soit le (quasi) optimal, soit les types Versicolor et Virginica sont confondus.

Dans le cas de Global K-Means, on retrouve la première situation directement (c'est un algorithme déterministe, on a jamais d'autre résultat). Ces deux situations sont représentées dans les figures 5.3 et 5.4. (Ici encore, p représente le taux de bonne réponse).

5.4 Résultats

On constate que dans le cas optimal, la classification est très bonne. Presque 90% des fleurs sont identifiées correctement. En regardant les graphes, un humain peut facilement distinguer le type Setosa des deux autres. Mais les types Versicolor et Virginica semble faire partie d'un même groupe. Cela montre que même un algorithme aussi simple que K-Means permet de mieux exploiter des données, dès lors qu'elle sont difficile à représenter pour un humain (ici en dimensions 4).

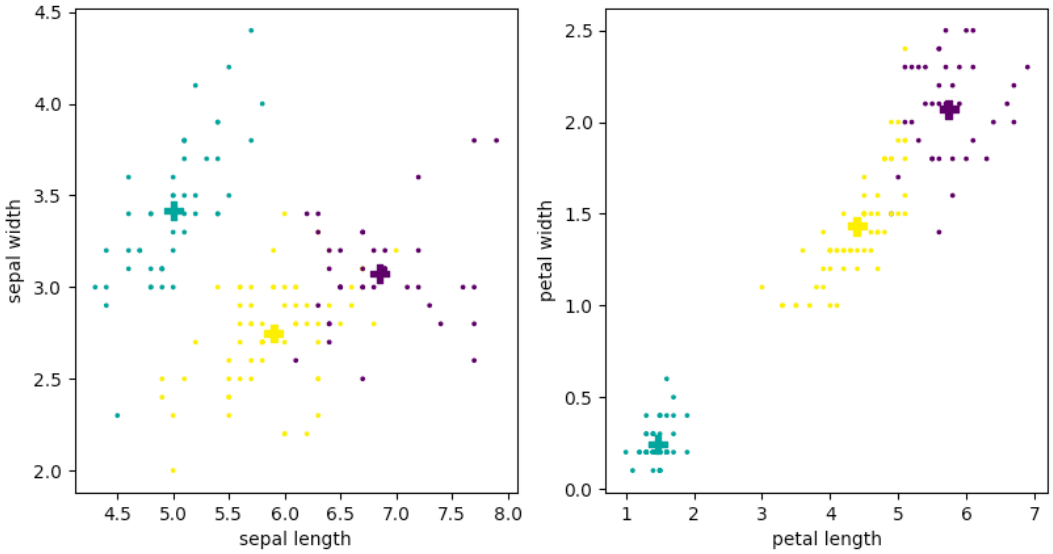


FIGURE 5.3 – Premier cas : le résultat optimal (au sens de K-Means)
$p = 0,89$

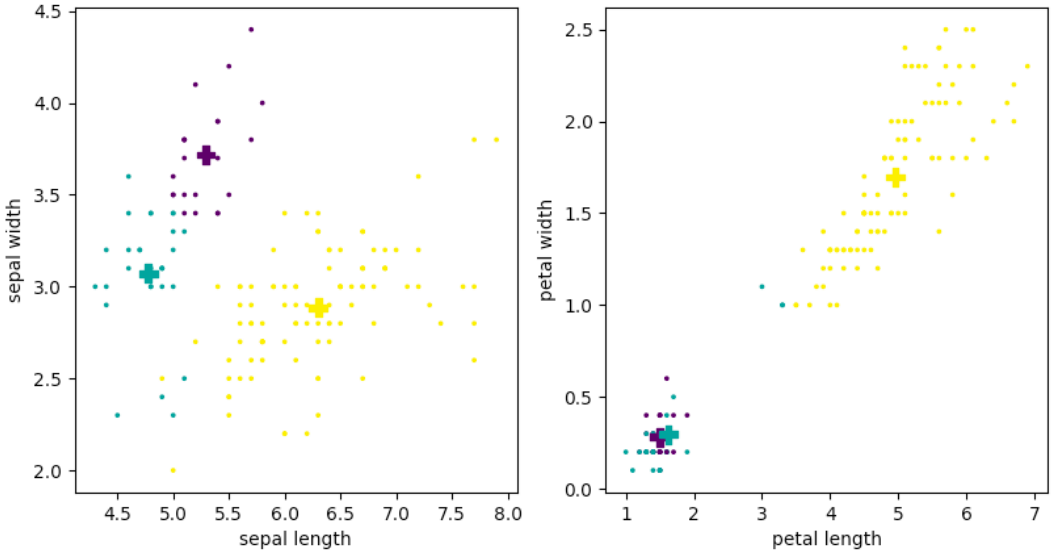


FIGURE 5.4 – Deuxième cas : l'algorithme s'arrête dans un minimum local
$p = 0,55$

6. Conclusion

Nous avons détaillé dans ce projet la méthode des K-means à travers nos deux études de cas. Un algorithme simple à implémenter et à comprendre, qui donne des résultats globalement satisfaisants. Il présente toutefois quelques inconvénients comme ses problèmes liés au choix du nombre de classes, et sa sensibilité à l'initialisation. Nous avons proposé des variantes pour les atténuer.

Il est à noter que la forme des données joue un role important dans cette méthode. En effet, le K-means ne peut être probant sur des clusters non sphériques ou des classes ayant un nombre de données très disparates, d'autres algorithmes sont alors nécessaires pour palier à cela...

Bibliographie

* E. Lebarbier, T. Mary-Huard. Classification non supervisée, AgroParisTech.

* Cours de Xavier Dupre, Clustering, disponible sur www.xavierdupre.fr.

* Z.Guellil, L.Zaoui. Proposition d'une solution au problème d'initialisation cas du K-means, Université des sciences et de la technologie d'Oran MB, Université Mohamed Boudiaf USTO, Algérie.

* David Bouldin, Papier original.