Chapitre 12

Clustering

Comment étudier des données non étiquetées ? La dimension de réduction nous permet de les visualiser ; mais les méthodes dites de *partitionnement de données*, ou *clustering*, nous permettent d'aller beaucoup plus loin. Il s'agit de séparer les données en sous-groupes homogènes, appelés *clusters*, qui partagent des caractéristiques communes. Dans ce chapitre, nous détaillerons l'intérêt de ces techniques et verrons trois types d'approches de clustering : le clustering hiérarchique, le clustering par centroïdes, et le clustering par densité.

Objectifs

- Expliquer l'intérêt d'un algorithme de clustering
- Évaluer le résultat d'un algorithme de clustering
- Implémenter un clustering hiérarchique, un clustering par la méthode des k moyennes, et un clustering par densité.

12.1 Pourquoi partitionner ses données

Les algorithmes de partitionnement de données permettent d'effectuer une analyse exploratoire sur des données non étiquetées. Ils permettent par exemple d'identifier des utilisateurs qui ont des comportements similaires (ce que l'on appelle la *segmentation de marché*), des communautés sur un réseau social, des motifs récurrents dans des transactions financières, des pixels d'un même objet dans une image (*segmentation d'image*) ou des patients dont la maladie s'explique par un même profil génétique.

Ils permettent aussi de visualiser les données, en se contentant de regarder un exemple représentatif par cluster.

Enfin, les algorithmes de clustering permettent de transférer à toutes les observations du même cluster les propriétés que l'on sait vraies de l'un des éléments de ce cluster. Cela est particulièrement utile dans le cas où l'étiquetage des données est difficile ou coûteux.

Exemple

Prenons l'exemple de l'annotation d'un corpus de documents texte. Annoter manuellement chacun de ces documents par le ou les sujets qu'il couvre est un travail fastidieux. Les personnes qui l'effectuent sont d'ailleurs susceptibles de commettre involontairement des erreurs d'inattention. Il est ainsi moins coûteux, voire plus efficace, d'utiliser un algorithme de clustering pour regrouper automatiquement ces documents par sujet. Il suffira alors d'avoir recours à un intervenant humain pour assigner un sujet à chaque cluster en lisant uniquement un ou deux des documents qu'il contient.

12.2 Évaluer la qualité d'un algorithme de clustering

Le clustering n'étant pas supervisé, il nous faut mettre au point des critères d'évaluation qui ne dépendent pas d'une vérité terrain (c'est-à-dire d'étiquettes connues). La tâche est plus délicate que dans le cadre de l'apprentissage supervisé, dans lequel le but à atteindre est beaucoup plus clair. Cependant, elle n'est pas impossible, et il existe un large éventail de mesures de la performance d'un algorithme de partitionnement des données.

Par la suite, nous supposons avoir partitionné un jeu de données non étiqueté $\mathcal{D} = \{\vec{x}^{\,1}, \vec{x}^{\,2}, \ldots, \vec{x}^{\,n}\}$ de n points d'un espace $\mathcal{X}$ en K clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$. d est une distance sur $\mathcal{X}$ et $k(\vec{x})$ est l'indice du cluster auquel $\vec{x}$ a été assigné.

12.2.1 La forme des clusters

Pour évaluer la qualité des clusters obtenus par un algorithme de partitionnement, il est possible de s'appuyer sur le souhait formulé de regrouper entre elles les observations similaires. Ainsi, les observations appartenant à un même cluster doivent être proches, tandis que les observations dissimilaires doivent appartenir à des clusters différents.

Pour quantifier ces notions, nous allons avoir besoin de celle de *centroïde*, qui est le barycentre d'un cluster.

Définition 12.1 (Centroïde et médoïde) On appelle *centroïde* du cluster $\mathcal{C}$ le point défini par

$$\vec{\mu}_{\mathcal{C}} = \frac{1}{|\mathcal{C}|} \sum_{\vec{x} \in \mathcal{C}} \vec{x}.$$

Le *médoïde* est le point du cluster le plus proche du *centroïde* (il peut ne pas être unique, auquel cas il sera choisi arbitrairement). Il sert de représentant du cluster :

$$\vec{m}_{\mathcal{C}} = \arg\min_{\vec{x} \in \mathcal{C}} d(\vec{x}, \vec{\mu}_{\mathcal{C}}).$$

■

Que des observations proches appartiennent au même cluster peut se traduire par la notion d'homogénéité, illustrée sur la figure 12.1 :

Définition 12.2 (Homogénéité) On appelle *homogénéité* du cluster $\mathcal{C}_k$, ou *tightness* en anglais, la moyenne des distances des observations de ce cluster à son centroïde :

$$T_k = \frac{1}{|\mathcal{C}_k|} \sum_{\vec{x} \in \mathcal{C}_k} d(\vec{x}, \vec{\mu}_k).$$

Ici $\vec{\mu}_k$ est le centroïde de $\mathcal{C}_k$.

L'*homogénéité globale* d'un clustering de $\mathcal{D}$ se calcule comme la moyenne des homogénéités des clusters :

$$T = \frac{1}{K} \sum_{k=1}^{K} T_k.$$

■

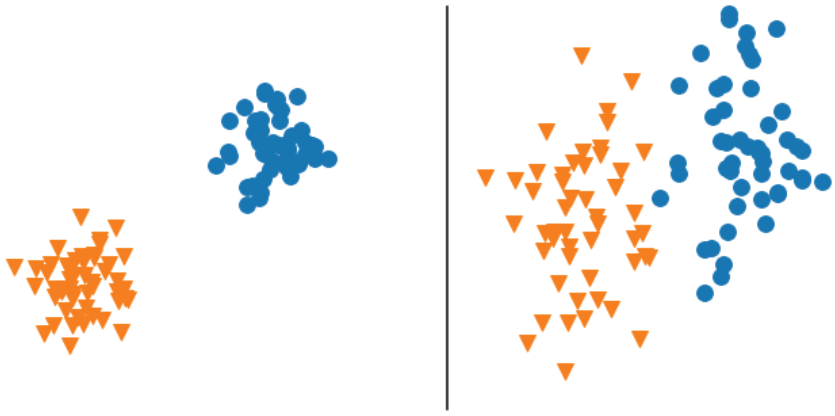


FIGURE 12.1 – Les deux clusters représentés sur le panneau de gauche sont homogènes, resserrés sur eux-mêmes : ils sont composés de points proches les uns des autres. À l'inverse, les deux clusters représentés sur le panneau de droite sont moins homogènes.

Pour quantifier à quel point les clusters sont distants les uns des autres, nous pouvons utiliser le critère de *séparabilité*, illustré sur la figure 12.2.

Définition 12.3 (Séparabilité) On appelle *séparabilité* des clusters $\mathcal{C}_k$ et $\mathcal{C}_l$ la distance entre leurs centroïdes :

$$S_{kl} = d(\vec{\mu}_k, \vec{\mu}_l).$$

La *séparabilité globale* d'un clustering de $\mathcal{D}$ se calcule comme la moyenne des séparabilités des clusters deux à deux :

$$S = \frac{2}{K(K-1)} \sum_{k=1}^{K} \sum_{k=k+1}^{K} S_{kl}.$$

■

Plutôt que de considérer les deux critères de séparabilité (que l'on souhaite élevée) et d'homogénéité (que l'on souhaite faible) séparément, il est possible de les comparer l'un à l'autre grâce à l'indice de Davies-Bouldin.

Définition 12.4 (Indice de Davies-Bouldin) On appelle *indice de Davies-Bouldin* du cluster $\mathcal{C}_k$ la valeur

$$D_k = \max_{l \neq k} \frac{T_k + T_l}{S_{kl}}.$$

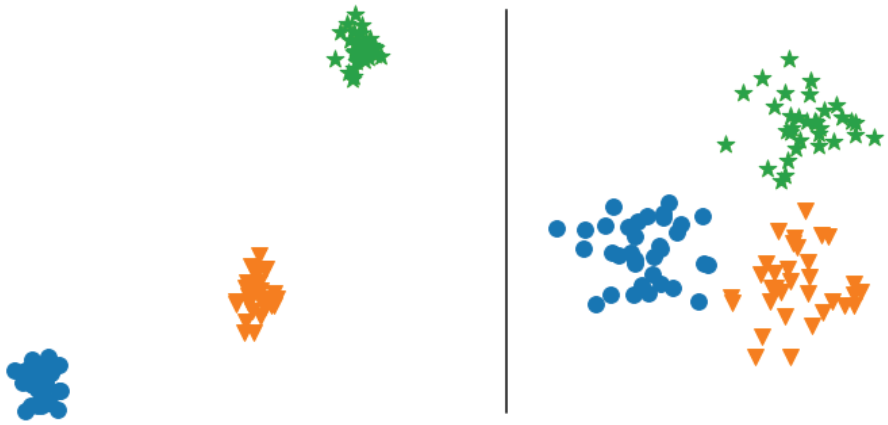


FIGURE 12.2 – Les trois clusters représentés sur le panneau de gauche sont bien séparés, contrairement à ceux représentés sur le panneau de droite qui sont proches les uns des autres.

L'*indice de Davies-Bouldin* global d'un clustering de $\mathcal{D}$ se calcule comme la moyenne des indices de Davies-Bouldin des clusters :

$$D = \frac{1}{C}\sum_{k=1}^{K} D_k.$$

■

Une autre façon de prendre en compte séparabilité et homogénéité est de calculer le *coefficient de silhouette*, qui permet de quantifier pour chacune des observations si elle appartient ou non au bon cluster.

Définition 12.5 (Coefficient de silhouette) On appelle *coefficient de silhouette* de l'observation $\vec{x} \in \mathcal{D}$ la valeur

$$s(\vec{x}) = \frac{b(\vec{x}) - a(\vec{x})}{\max(a(\vec{x}), b(\vec{x}))}$$

où $a(\vec{x})$ est la distance moyenne de $\vec{x}$ à tous les autres éléments du cluster auquel il appartient et $b(\vec{x})$ est la plus petite valeur que pourrait prendre $a(\vec{x})$, si a appartenait à un autre cluster :

$$a(\vec{x}) = \frac{1}{|\mathcal{C}_{k(\vec{x})}| - 1}\sum_{\vec{u}\in\mathcal{C}_{k(\vec{x})}, \vec{u}\neq\vec{x}} d(\vec{u}, \vec{x}); \qquad b(\vec{x}) = \min_{l\neq k(\vec{x})} \frac{1}{|\mathcal{C}_l|}\sum_{\vec{u}\in\mathcal{C}_l} d(\vec{u}, \vec{x}).$$

Le *coefficient de silhouette global* du clustering est son coefficient de silhouette moyen :

$$s = \frac{1}{n}\sum_{i=1}^{n} s(\vec{x}^{\,i}).$$

■

Le coefficient de silhouette de $\vec{x}$ est d'autant plus proche de 1 que son assignation au cluster $\mathcal{C}_{k(\vec{x})}$ est satisfaisante.

12.2.2 La stabilité des clusters

Un autre critère important est celui de la *stabilité* des clusters. On s'attend en effet à obtenir les mêmes clusters si on supprime ou perturbe quelques observations, ou en initialisant différemment l'algorithme de partitionnement.

Ce critère peut être utilisé pour choisir les hyperparamètres de l'algorithme : si on obtient des clusters très différents pour différentes initialisations de l'algorithme de partitionnement, cela peut indiquer que les hyperparamètres sont mal choisis.

12.2.3 Les connaissances expert

Enfin, il arrive parfois que l'on dispose d'un jeu de données partiellement étiqueté par des classes que nous aimerions retrouver par clustering. Cela peut être le cas d'un corpus de documents dont un sous-ensemble est étiqueté par sujet, ou d'une base de données d'images dont un sous-ensemble est étiqueté en fonction de ce qu'elles représentent.

On peut alors évaluer le résultat d'un algorithme de partitionnement comme on évaluerait celui d'un algorithme de classification multi-classe. Attention, il y a cependant une différence : dans le cas du clustering, peu importe que les objets de la classe k se retrouvent dans le premier, le deuxième ou le k-ème cluster. Il faut donc évaluer la concordance de la partition des données par l'algorithme de clustering avec celle induite par les étiquettes. C'est ce que permet de faire l'*indice de Rand*.

Nous supposons maintenant les observations $\vec{x}^{\,i}$ étiquetées par $y^i \in \{1, 2, \ldots, C\}$.

Définition 12.6 (Indice de Rand) On appelle *indice de Rand* la proportion de paires d'observations qui sont soit de même classe et dans le même cluster, soit de classe différente et dans deux clusters différents :

$$\mathrm{RI} = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{l=i+1}^{n} \delta\left(k(\vec{x}^{\,i}) = k(\vec{x}^{\,l})\right) \delta(y^i = y^l) + \delta\left(k(\vec{x}^{\,i}) \neq k(\vec{x}^{\,l})\right) \delta(y^i \neq y^l).$$

■

En bioinformatique, où il est souvent délicat d'étiqueter les objets d'étude par manque de connaissances, il est fréquent d'évaluer un clustering en le comparant à une *ontologie*, c'est-à-dire une classification d'objets (par exemple, des gènes) en catégories décrites par un vocabulaire commun et organisées de manière hiérarchique. On peut évaluer la cohérence d'un clustering avec une ontologie par une *analyse d'enrichissement*, qui consiste à évaluer s'il y a plus d'objets d'une catégorie de l'ontologie au sein d'un cluster que ce à quoi on pourrait s'attendre par hasard. Si l'on suppose les données issues d'une distribution hypergéométrique, il s'agit alors de calculer, pour un cluster $\mathcal{C}_k$, une catégorie $\mathcal{G}$, et un seuil $t \in \mathbb{N}$,

$$\mathbb{P}[|\mathcal{G} \cap \mathcal{C}_k| \geq t] = 1 - \sum_{s=1}^{t} \frac{\binom{|\mathcal{G}|}{s}\binom{n-|\mathcal{G}|}{|\mathcal{C}_k|-s}}{\binom{n}{|\mathcal{C}_k|}}. \tag{12.1}$$

En effet, le terme sous la somme est la probabilité que, lorsqu'on tire $|\mathcal{C}_k|$ éléments parmi n, s d'entre eux appartiennent à $\mathcal{G}$.

Muni de ces différentes façons d'évaluer un algorithme de partitionnement de données, nous pouvons maintenant découvrir ces algorithmes eux-mêmes. Ces algorithmes cherchent à optimiser les critères d'homogénéité et de séparabilité que nous venons de définir. Comme il n'est pas possible de le faire de manière exacte, il s'agit de le faire de manière approchée. Nous nous concentrerons dans ce chapitre sur les trois principales familles d'algorithmes de clustering : clustering hiérarchique, clustering par centroïdes, et clustering par densité.

12.3 Clustering hiérarchique

Le *clustering hiérarchique* forme des clusters séparés par récurrence : il s'agit de partitionner les données pour toutes les échelles possibles de taille de partition, dans une hiérarchie à plusieurs niveaux.

12.3.1 Dendrogramme

Le résultat d'un clustering hiérarchique peut se visualiser sous la forme d'un *dendrogramme*. Il s'agit d'un arbre dont les n feuilles correspondent chacune à une observation. Chaque nœud de l'arbre correspond à un cluster :

— la racine est un cluster contenant toutes les observations
— chaque feuille est un cluster contenant une observation
— les clusters ayant le même parent sont agglomérés en un seul cluster au niveau au-dessus
— un cluster est subdivisé en ses enfants au niveau au-dessous.

Ce sont donc les nœuds intermédiaires qui nous intéresseront le plus.

Enfin, la longueur d'une branche de l'arbre est proportionnelle à la distance entre les deux clusters qu'elle connecte.

La figure 12.3 représente un exemple de dendrogramme. Dans le cas où n est trop grand pour représenter l'intégralité de l'arbre, il est classique de le couper et de n'en représenter que la partie de la racine à un niveau que l'on choisit.

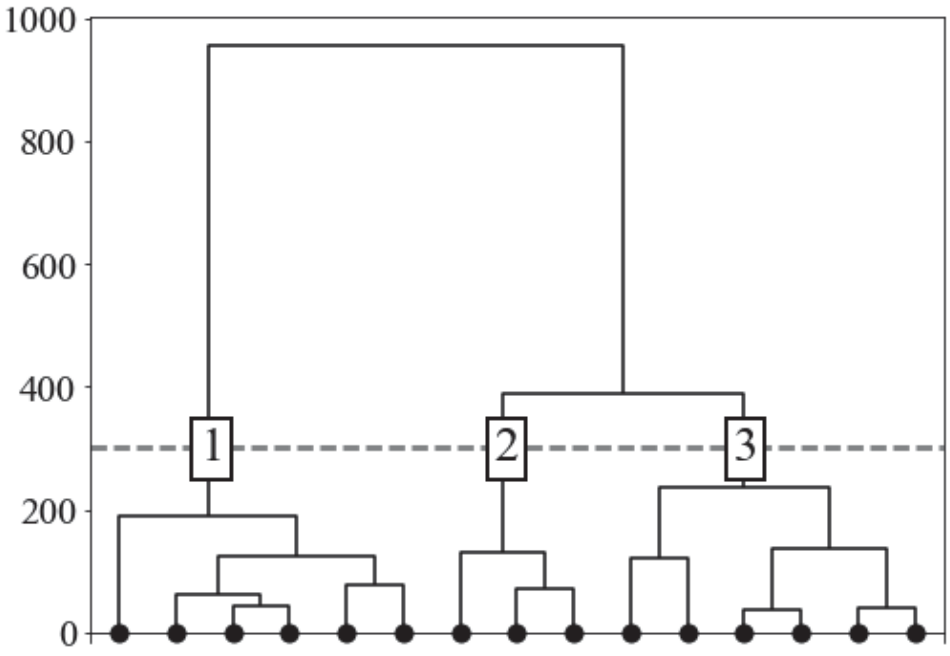


FIGURE 12.3 – Un exemple de dendrogramme. En coupant au niveau de la ligne en pointillés, on obtient 3 clusters. Chaque feuille de l'arbre (sur l'axe des abscisses) correspond à une observation.

Cette facilité à représenter visuellement le résultat d'un algorithme de clustering hiérarchique fait qu'il est très utilisé dans des domaines comme la bioinformatique.

12.3.2 Construction agglomérative ou divisive

Un clustering hiérarchique peut se construire de manière *agglomérative* ou *divisive*.

Le *clustering agglomératif*, ou *bottom-up clustering*, commence par considérer les feuilles du dendrogramme : initialement, chaque observation forme un cluster de taille 1. À chaque itération de l'algorithme, on trouve les deux clusters les plus proches, et on les agglomère en un seul cluster, et ce jusqu'à ne plus avoir qu'un unique cluster contenant les n observations.

Le *clustering divisif*, ou *top-down clustering*, est l'approche inverse. On l'initialise en considérant un seul cluster, la racine du dendrogramme, contenant toutes les observations. À chaque itération, on sépare un cluster en deux, jusqu'à ce que chaque cluster ne contienne plus qu'une seule observation.

Par la suite, nous nous concentrerons sur l'approche agglomérative.

12.3.3 Fonctions de lien

Déterminer les deux clusters les plus proches afin de les agglomérer requiert de définir une distance entre clusters ; c'est ce qu'on appelle dans le cadre du clustering une *fonction de lien*, ou *linkage* en anglais. Plusieurs approches sont possibles, qui reposent toutes sur une distance d sur $\mathcal{X}$.

Tout d'abord, on peut choisir d'agglomérer deux clusters si deux de leurs éléments sont proches. On utilise alors le *lien simple*.

Définition 12.7 (Lien simple) On appelle *lien simple*, ou *single linkage*, la distance entre deux clusters définie par

$$d_{\text{simple}}(\mathcal{C}_k, \mathcal{C}_l) = \min_{(\vec{u},\vec{v})\in\mathcal{C}_k\times\mathcal{C}_l} d(\vec{u}, \vec{v}).$$

■

On peut aussi choisir d'agglomérer deux clusters si *tous* leurs éléments sont proches. On utilise alors le *lien complet*, qui est la distance maximale entre un élément du premier cluster et un élément du deuxième.

Définition 12.8 (Lien complet) On appelle *lien complet*, ou *complete linkage*, la distance entre deux clusters définie par

$$d_{\text{complet}}(\mathcal{C}_k, \mathcal{C}_l) = \max_{(\vec{u},\vec{v})\in\mathcal{C}_k\times\mathcal{C}_l} d(\vec{u}, \vec{v}).$$

■

Une approche intermédiaire consiste à considérer la distance moyenne entre un élément du premier cluster et un élément du deuxième. C'est le *lien moyen*.

Définition 12.9 (Lien moyen) On appelle *lien moyen*, ou *average linkage*, la distance entre deux clusters définie par

$$d_{\text{moyen}}(\mathcal{C}_k, \mathcal{C}_l) = \frac{1}{|\mathcal{C}_k|}\frac{1}{|\mathcal{C}_l|}\sum_{\vec{u}\in\mathcal{C}_k}\sum_{\vec{v}\in\mathcal{C}_l} d(\vec{u}, \vec{v}).$$

Cette distance est aussi parfois appelée *UPGMA* pour *Unweighted Paired Group Method with Arithmetic mean.* ■

Une alternative au lien moyen est le *lien centroïdal*, qui considère la distance entre les centroïdes des clusters.

Définition 12.10 (Lien centroïdal) On appelle *lien centroïdal*, ou *centroid linkage*, la distance entre deux clusters définie par

$$d_{\text{centroidal}}(\mathcal{C}_k, \mathcal{C}_l) = d\left(\frac{1}{|\mathcal{C}_k|}\sum_{\vec{u}\in\mathcal{C}_k}\vec{u}, \frac{1}{|\mathcal{C}_l|}\sum_{\vec{v}\in\mathcal{C}_l}\vec{v}\right) = d(\vec{\mu}_k, \vec{\mu}_l).$$

Cette distance est aussi parfois appelée *UPGMC* pour *Unweighted Paired Group Method with Centroid.* ■

Les fonctions de lien ci-dessus s'attachent à garantir la *séparabilité* des clusters. À l'inverse, il est possible de chercher à maximiser leur *homogénéité* : il s'agit de minimiser $T_k = \frac{1}{|\mathcal{C}_k|}\sum_{\vec{x}\in\mathcal{C}_k} d(\vec{x}, \vec{\mu}_k)$. Dans le cas où d est la distance euclidienne, alors $T_k = \frac{1}{|\mathcal{C}_k|}\sum_{\vec{x}\in\mathcal{C}_k} ||\vec{x} - \vec{\mu}_k||_2$. Le *clustering de Ward* utilise une formulation similaire : il s'agit d'agglomérer deux clusters de sorte à minimiser la variance intra-cluster du résultat.

Définition 12.11 (Inertie) On appelle *variance intra-cluster*, ou *inertie* du cluster $\mathcal{C}$ la valeur

$$\text{Var}_{\text{in}}(\mathcal{C}) = \frac{1}{|\mathcal{C}|}\sum_{\vec{x}\in\mathcal{C}} ||\vec{x} - \vec{\mu}||_2^2 .$$

L'*inertie globale* d'un clustering de $\mathcal{D}$ est alors donnée par la somme des inerties des clusters :

$$V = \sum_{k=1}^{K}\frac{1}{|\mathcal{C}_k|}\sum_{\vec{x}\in\mathcal{C}_k} ||\vec{x} - \vec{\mu}_k||_2^2 .$$

■

12.3.4 Choix du nombre de clusters

Le clustering hiérarchique a l'avantage de ne pas requérir de définir à l'avance le nombre de clusters, ce qui permet d'explorer toutes les possibilités le long du dendrogramme. Cependant, il faut généralement prendre cette décision à un moment.

Il est possible pour cela d'utiliser un dendrogramme, pour y déceler un « niveau » auquel les clusters sont clairement distants les uns des autres – on se rappellera que la longueur d'une branche est proportionnelle à la distance entre les deux clusters qu'elle sépare.

Une solution alternative est d'évaluer les différentes partitions trouvées, c'est-à-dire les différents nœuds du dendrogramme, à l'aide d'une mesure de performance telle que le coefficient de silhouette.

12.3.5 Complexité algorithmique

La complexité algorithmique du clustering hiérarchique est élevée : à chaque itération, pour décider quels clusters regrouper, il nous faudra calculer les distances deux à deux entre toutes les paires d'observations du jeu de données, pour une complexité en $\mathcal{O}(pn^2)$. Une alternative est de stocker ces distances en mémoire pour pouvoir les réutiliser, ce qui a une complexité quadratique en le nombre d'observations.

Le clustering hiérarchique est donc plus adapté aux jeux de données contenant peu d'échantillons.

12.4 Méthode des k-moyennes

Plutôt que d'explorer toutes les partitions possibles à différentes échelles, il peut être préférable de se fixer un nombre K de clusters, ce qui permet de réduire les temps de calculs.

Nous avons vu ci-dessus que le clustering de Ward cherche à minimiser la variance intra-cluster afin de créer des clusters homogènes. La méthode dite des *k-moyennes*, ou du *k-means*, proposée par Hugo Steinhaus (1957), cherche à résoudre ce problème pour un nombre de clusters K fixé. Il s'agit alors de trouver l'affectation des observations à K clusters qui minimise la variance intra-cluster globale :

$$\underset{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_K}{\arg\min} \sum_{k=1}^{K} \sum_{\vec{x}\in\mathcal{C}_k} ||\vec{x} - \vec{\mu}_k||_2^2 \,. \tag{12.2}$$

Cependant, résoudre ce problème de manière exacte n'est pas possible. On utilise donc une heuristique, l'*algorithme de Lloyd*, proposé par Stuart Lloyd (1982).

12.4.1 Algorithme de Lloyd

Définition 12.12 (Algorithme de Lloyd) Étant données n observations dans $\mathbb{R}^p$ et un nombre K de clusters, l'*algorithme de Lloyd* procède de la manière suivante :

1. Choisir K observations $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_K$ parmi les n observations, pour servir de centroïdes initiaux ;
2. Affecter chaque observation $\vec{x}^{\,i} \in \mathcal{D}$ au centroïde dont elle est le plus proche :
$$k(\vec{x}^{\,i}) = \underset{k=1,\ldots,K}{\arg\min} \left|\left|\vec{x}^{\,i} - \vec{\mu}_k\right|\right|_2 ;$$
3. Recalculer les centroïdes de chaque cluster :
$$\vec{\mu}_k = \frac{1}{|\mathcal{C}_k|} \sum_{\vec{x}^{\,i}\in\mathcal{C}_k} \vec{x}^{\,i};$$
4. Répéter les opérations 2–3 jusqu'à convergence, c'est-à-dire jusqu'à ce que les affectations ne changent plus.

■

L'algorithme de Lloyd implémente une stratégie gloutonne, et s'il converge en général très rapidement, il peut tomber dans un minimum local. Il peut donc être pertinent de le faire tourner plusieurs fois, et de garder la solution qui a la plus faible variance intra-cluster.

Si l'on doit itérer t fois l'algorithme de Lloyd, sachant que le coût de calculer Kn distances en p dimensions est de l'ordre de $\mathcal{O}(npK)$, la complexité algorithmique de l'algorithme de Lloyd est en $\mathcal{O}(npKt)$. K et t sont typiquement négligeables devant n, et cet algorithme est donc *linéaire* en le nombre d'observations, par opposition au clustering hiérarchique dont le coût est *quadratique* en n : nous avons remplacé le calcul des distances d'une observation $\vec{x}^{\,i}$ aux $n-1$ autres points du jeu de données par un calcul de sa distance à K centroïdes.

12.4.2 Forme des clusters

D'après la formulation de l'algorithme des k-moyennes (équation 12.2), chaque cluster $\mathcal{C}_k$ est constitué des observations de $\mathcal{D}$ qui sont les plus proches du centroïde $\vec{\mu}_k$. Ils forment donc un diagramme de Voronoï (cf. section 8.1.2). En particulier, cela signifie que les clusters trouvés par l'algorithme des k-moyennes sont nécessairement convexes.

Données aberrantes

L'algorithme du k-means est sensible aux données aberrantes : elles vont en effet tirer un cluster à elle. Si une observation $\vec{x}^{\,i}$ est très éloignée des autres observations, alors elle se retrouvera dans son propre cluster, tandis que le reste des données sera partitionné en $K-1$ clusters.

Cette propriété est néanmoins intéressante, car elle permet d'utiliser l'algorithme des k-moyennes justement pour détecter les observations aberrantes : ce sont celles qui sont seules dans leur cluster.

12.4.3 Variantes

k-means++

L'algorithme du k-means est stochastique : on peut obtenir des résultats différents selon l'initialisation, et certains de ces résultats peuvent avoir une inertie bien plus grande que la solution optimale.

Pour éviter ce problème, l'algorithme *k-means++* commence par initialiser les centroïdes de manière à les disperser au maximum parmi les données. Plus précisément, la procédure consiste à

1. Choisir un premier centroïde $\vec{u}^{\,1}$ aléatoirement parmi les observations $\mathcal{D}$
2. Pour $k = 2, \dots, K$:
   Choisir le k-ème centroïde $\vec{u}^{\,k}$ parmi $\mathcal{D} \setminus \vec{u}^{\,k-1}$, en suivant une loi proportionnelle au carré de la distance à $\vec{u}^{\,k-1}$, c'est-à-dire que $\vec{u}^{\,k}$ aura de fortes chances d'être éloigné de $\vec{u}^{\,k-1}$.

Cette approche ne rend pas le k-means déterministe, mais permet d'éviter les « pires » solutions.

Méthode des k-moyennes avec noyau

La méthode des k-moyennes requiert de décrire les données dans un espace euclidien, et ne peut former que des clusters convexes, ce qui peut être limitant. Cependant, l'*astuce du noyau* (cf. section 10.3.3) s'applique à l'algorithme de Lloyd.

Démonstration. Soit $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ un noyau. Il existe un espace de Hilbert $\mathcal{H}$ et une application $\phi : \mathcal{X} \to \mathcal{H}$ telle que, pour tout $\vec{x}, \vec{x}' \in \mathcal{X} \times \mathcal{X}$, $\kappa(\vec{x}, \vec{x}') = \langle \phi(\vec{x}), \phi(\vec{x}') \rangle_{\mathcal{H}}$.

Pour appliquer l'algorithme de Lloyd aux images $\{\phi(\vec{x}^{\,1}), \phi(\vec{x}^{\,2}), \dots, \phi(\vec{x}^{\,n})\}$ des éléments de $\mathcal{D}$ dans $\mathcal{H}$, nous avons besoin de calculer, à chaque itération, la distance de $\phi(\vec{x}^{\,i})$ à chacun des K centroïdes $\vec{h}_1, \vec{h}_2, \dots, \vec{h}_K$.

La position d'un centroïde $\vec{h}_k$ se calcule comme la moyenne des images des observations appartenant à $\mathcal{C}_k$:

$$\vec{h}_k = \frac{1}{|\mathcal{C}_k|} \sum_{\vec{x}^{\,i} \in \mathcal{C}_k} \phi(\vec{x}^{\,i}).$$

La distance de l'image d'une observation $\vec{x}^{\,i}$ à un centroïde se calcule alors comme

$$\begin{aligned}
\left|\left| \phi(\vec{x}^{\,i}) - \vec{h}_k \right|\right|_2^2 &= \left|\left| \phi(\vec{x}^{\,i}) - \frac{1}{|\mathcal{C}_k|} \sum_{\vec{u} \in \mathcal{C}_k} \phi(\vec{u}) \right|\right|_2^2 \\
&= \left\langle \phi(\vec{x}^{\,i}) - \frac{1}{|\mathcal{C}_k|} \sum_{\vec{u} \in \mathcal{C}_k} \phi(\vec{u}),\ \phi(\vec{x}^{\,i}) - \frac{1}{|\mathcal{C}_k|} \sum_{\vec{u} \in \mathcal{C}_k} \phi(\vec{u}) \right\rangle_{\mathcal{H}} \\
&= \kappa(\vec{x}^{\,i}, \vec{x}^{\,i}) - \frac{2}{|\mathcal{C}_k|} \sum_{\vec{u} \in \mathcal{C}_k} \kappa(\vec{u}, \vec{x}^{\,i}) + \frac{1}{|\mathcal{C}_k|^2} \sum_{\vec{u} \in \mathcal{C}_k} \sum_{\vec{v} \in \mathcal{C}_k} \kappa(\vec{u}, \vec{v}).
\end{aligned}$$

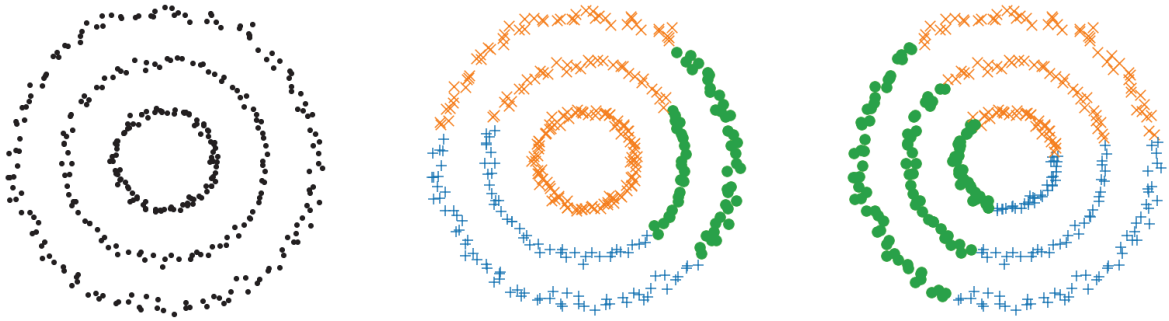
On peut ainsi réécrire l'étape (2) de l'algorithme de Lloyd sans faire appel à l'application ϕ, et sans avoir besoin de recalculer explicitement les centroïdes à l'étape (3). □

Remarque

La version à noyau de la méthode des k-moyennes ne permet généralement pas de connaître les centroïdes des clusters, puisqu'ils vivent dans l'espace de redescription $\mathcal{H}$ qui n'est pas accessible sans connaître ϕ.

12.5 Clustering par densité

Une autre façon d'obtenir des clusters non convexes est le clustering par densité. Prenons l'exemple présenté sur la figure 12.4 : il semblerait naturel de partitionner les données en 3 cercles concentriques, ce que ni le clustering agglomératif ni la méthode des k-moyennes n'arrivent à faire, car ces clusters ne sont pas convexes.



(A) Il nous semble naturel de partitionner ces données en 3 cercles concentriques.

(B) Partitionnement en 3 clusters par clustering agglomératif (lien moyen).

(C) Partitionnement en 3 clusters par k-moyennes.

FIGURE 12.4 – Motivation du clustering par densité.

C'est le problème que le *clustering par densité* cherche à résoudre, en observant que les points les plus proches d'un point donné sont sur le même cercle et non sur un autre cercle. Il s'agit de former des clusters d'observations proches les unes des autres, au sens où si deux éléments $\vec{x}^{\,i}$ et $\vec{x}^{\,l}$ d'un même cluster $\mathcal{C}_k$ peuvent être éloignés l'un de l'autre, il existe une séquence d'éléments $\vec{u}^{\,1}, \vec{u}^{\,2}, \ldots, \vec{u}^{\,m} \in \mathcal{C}_k$ tels que $\vec{u}^{\,1}$ est proche de $\vec{x}^{\,i}$, $\vec{u}^{\,m}$ est proche de $\vec{x}^{\,l}$, et pour tout $t \in \{1, \ldots, m\}$, $\vec{u}^{\,t}$ est proche de $\vec{u}^{\,t-1}$. Cette idée est illustrée sur la figure 12.5

Pour formaliser cette idée, nous allons avoir besoin de quelques définitions.

Définition 12.13 (ϵ-voisinage) Soient $\mathcal{D} = \{\vec{x}^{\,1}, \vec{x}^{\,2}, \ldots, \vec{x}^{\,n}\}$ le jeu d'éléments de $\mathcal{X}$ à partitionner, d une distance sur $\mathcal{X}$, et $\epsilon > 0$. On appelle *ϵ-voisinage* d'un élément $\vec{x} \in \mathcal{X}$ l'ensemble des observations de $\mathcal{D}$ dont la distance à $\vec{x}$ est inférieure à ϵ :

$$\mathcal{N}_\epsilon(\vec{x}) = \{\vec{u} \in \mathcal{D} : d(\vec{x}, \vec{u}) < \epsilon\}.$$

■

Définition 12.14 (Point intérieur) Étant donné $n_{\min} \in \mathbb{N}$, on dit de $\vec{x} \in \mathcal{X}$ qu'il est un *point intérieur*, ou *core point* en anglais, si son ϵ-voisinage contient au moins $n_{\min}$ éléments : $|\mathcal{N}_\epsilon(\vec{x})| \geq n_{\min}$. ■

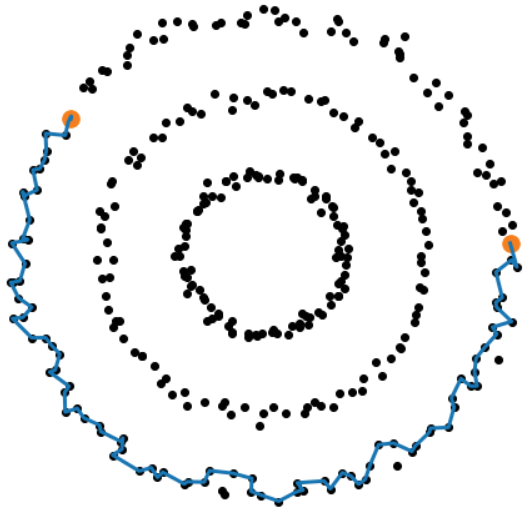


FIGURE 12.5 – Il existe un chemin entre voisins proches permettant de passer d'un point à un autre du même cluster.

Définition 12.15 (Connexion par densité) On dit que $\vec{x}$ et $\vec{v} \in \mathcal{X}$ sont *connectés par densité* si l'on peut les « relier » par une suite d'ϵ-voisinages contenant chacun au moins $n_{\min}$ éléments. Formellement, cela signifie qu'il existe $m \in \mathbb{N}$, et une séquence $\vec{u}^1, \vec{u}^2, \ldots, \vec{u}^m$ d'éléments de $\mathcal{D}$ tels que $\vec{u}^1$ est un point intérieur de $\mathcal{N}_\epsilon(\vec{x})$, $\vec{u}^2$ est un point intérieur de $\mathcal{N}_\epsilon(\vec{u}^1), \ldots,$ $\vec{u}^m$ est un point intérieur de $\mathcal{N}_\epsilon(\vec{u}^{m-1})$, $\vec{v}$ est un point intérieur de $\mathcal{N}_\epsilon(\vec{u}^m)$. ■

L'algorithme *DBSCAN*, proposé en 1996 par Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiaowei Xu (Ester et al., 1996), partitionne les données en créant des clusters de points atteignables par densité les uns depuis les autres. C'est un algorithme de partitionnement populaire, qui a obtenu en 2014 une distinction de contribution scientifique ayant résisté à l'épreuve du temps, le *test of time award* de la prestigieuse conférence KDD.

Définition 12.16 (DBSCAN) On appelle *DBSCAN*, pour *Density-Based Spatial Clustering of Applications with Noise*, ou partitionnement dans l'espace par densité pour des applications bruitées, la procédure de partitionnement suivante :

- Initialisation : Un ensemble d'éléments visités $\mathcal{V} = \emptyset$, une liste de clusters $\mathcal{K} = \emptyset$, une liste d'observations aberrantes $\mathcal{A} = \emptyset$.
- Pour chaque élément $\vec{x} \in \mathcal{D} \setminus \mathcal{V}$:
  1. Construire $\mathcal{N}_\epsilon(\vec{x})$
  2. Si $|\mathcal{N}_\epsilon(\vec{x})| < n_{\min}$: considérer (temporairement) $\vec{x}$ comme une observation aberrante :

     $$\mathcal{A} \leftarrow \mathcal{A} \cup \{\vec{x}\}$$

     Sinon :
     - créer un cluster $\mathcal{C} \leftarrow \{\vec{x}\}$
     - augmenter ce cluster par la procédure `grow_cluster`$(\mathcal{C}, \mathcal{N}_\epsilon(\vec{x}), \epsilon, n_{\min})$
  3. Ajouter $\mathcal{C}$ à la liste des clusters : $\mathcal{K} \leftarrow \mathcal{K} \cup \mathcal{C}$

4. Marquer tous les éléments de $\mathcal{C}$ comme visités : $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{C}$.

La procédure `grow_cluster`$(\mathcal{C}, \mathcal{N}, \epsilon, n_{\min})$ est définie comme suit :
Pour tout $\vec{u} \in \mathcal{N} \setminus \mathcal{V}$:
— créer $\mathcal{N}' \leftarrow \mathcal{N}_\epsilon(\vec{u})$
— si $|\mathcal{N}'| \geq n_{\min}$: actualiser $\mathcal{N}$ de sorte à prendre les éléments de $\mathcal{N}'$ en considération : $\mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{N}'$
— si $\vec{u}$ n'appartient à aucun autre cluster, l'ajouter à $\mathcal{C}$: $\mathcal{C} \leftarrow \mathcal{C} \cup \{\vec{u}\}$
— si $\vec{u}$ était précédemment cataloguée comme aberrante, l'enlever de la liste des observations aberrantes : $\mathcal{A} = \mathcal{A} \setminus \{\vec{u}\}$.

■

Un des avantages de DBSCAN est sa robustesse aux données aberrantes, qui sont identifiées lors de la formation des clusters.

Le fléau de la dimension (voir section 11.1.3) rend DBSCAN difficile à appliquer en très grande dimension : les ϵ-voisinages auront tendance à ne contenir que leur centre. De plus, la densité étant définie par les paramètres ϵ et $n_{\min}$, DBSCAN ne pourra pas trouver de clusters de densité différente. Cependant, DBSCAN ne requiert pas de prédéfinir le nombre de clusters, et est efficace en temps de calcul. Son application aux données de la figure 12.4a est illustrée sur la figure 12.6.

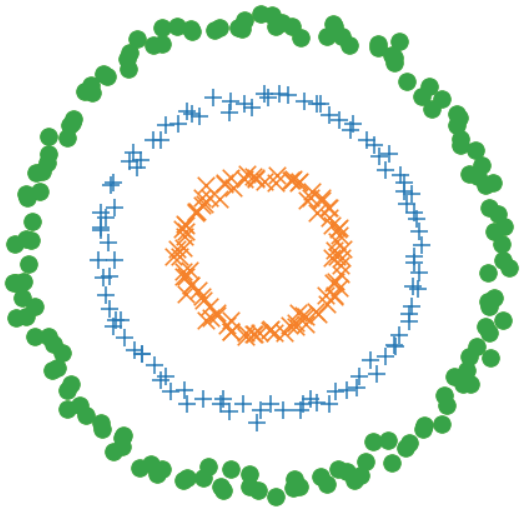


FIGURE 12.6 – Partitionnement des données de la figure 12.4a par DBSCAN.

Points clefs

— Le clustering, ou partitionnement de données, cherche à identifier des classes sans utiliser d'étiquettes.
— En l'absence d'étiquette, la qualité d'une partition peut s'évaluer sur des critères de séparabilité et d'homogénéité