ENSEIRB-MATMECA
Télécommunications 2A

TS228 - Intelligence artificielle en traitement de l'image

Classification de données par approches non supervisées

Réalisé par :

Pierre MINIER
Merwan MULLER
Baptiste ROULLIAUX

Encadrant :

Yannick BERTHOUMIEU

Février 2022

Sommaire

1 Introduction

Définie par l'Encyclopédie Larousse comme étant « l'ensemble des théories et des techniques mises en oeuvre en vue de réaliser des machines capables de simuler l'intelligence humaine », l'intelligence artificielle fait l'objet de nombreuses controverses depuis 1956. De l'apprentissage automatique à l'apprentissage profond en passant par les réseaux de neurones, l'IA donne naissance à de nouvelles disciplines et technologies soulevant parfois de nombreuses questions éthiques ou sociétales.

L'apprentissage automatique (ou « *machine learning* » en anglais) est un sous-domaine de l'intelligence artificielle. Il consiste à développer des systèmes capables d'apprendre ou d'améliorer leurs performances à partir des données qu'ils traitent. Il existe deux méthodes d'apprentissage : l'apprentissage supervisé et l'apprentissage non supervisé. Ces deux méthodes se distinguent par leur niveau de connaissance ; l'apprentissage supervisé requiert une forte connaissance a priori tandis que l'apprentissage non supervisé en requiert peu.

La technique la plus utilisée pour résoudre les problèmes d'apprentissage non supervisé est le regroupement (ou « *clustering* » en anglais). Cette technique consiste à diviser un ensemble de données en différents « paquets » homogènes, c'est-à-dire présentant des caractéristiques communes (d'un point de vue informatique).

Le sujet choisi porte sur deux algorithmes de clustering : K-means et DBSCAN (pour « *Density-Based Spatial Clustering of Applications with Noise* »). Dans la suite, nous expliquerons le fonctionnement de chaque algorithme puis leur implémentation, avant de les comparer.

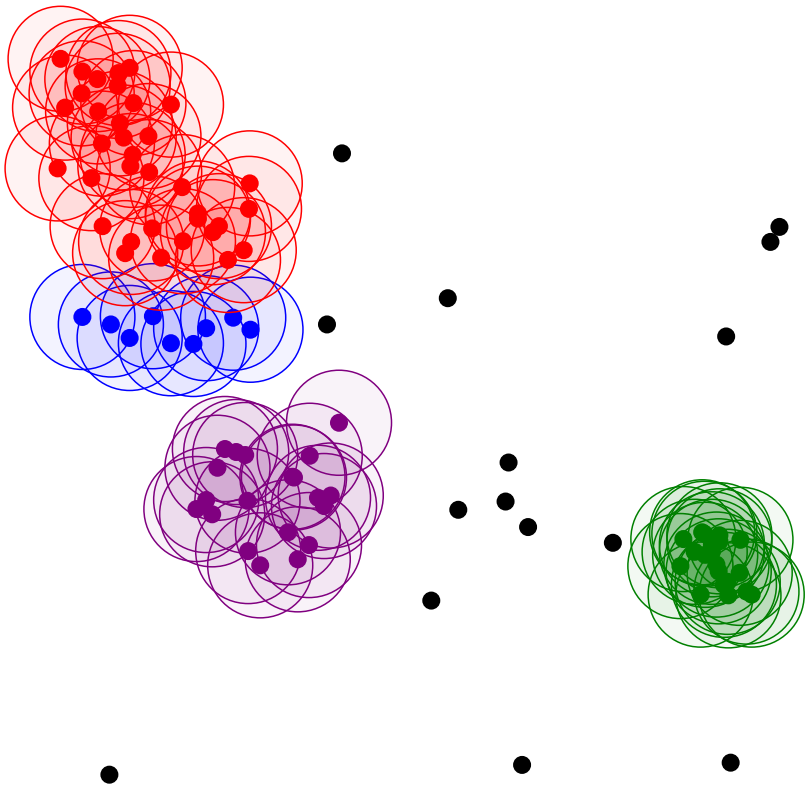

FIGURE 1.1 – Identification de 4 clusters bruités avec DBSCAN

2 Partition des données en K groupes

2.1 Principe de l'algorithme K-means

L'algorithme K-means divise un jeu de données en K clusters. Ces derniers sont identifiés par leur centre de masse, nommés centroïdes. Lors de l'affectation d'un point, K-means cherche le cluster minimisant la distance entre son centroïde et le point étudié.

2.1.1 Initialisation

Les K clusters sont mis en place durant la phase d'initialisation. L'algorithme de base consiste à les définir aléatoirement parmi K points distincts du jeu de données. Cette technique, bien que simple à mettre en oeuvre, ne peut garantir des résultats fiables. Des optimisations sont alors mises en places.

2.2 Améliorations existantes

2.2.1 Répétition de l'algorithme

Une première méthode consiste à répéter l'algorithme jusqu'à la convergence des centroïdes. Concrètement, K-means est appliqué tel quel une première fois. Puis une seconde fois, mais en considérant les points de départ comme étant les centroïdes des clusters de la précédente itération. Et ainsi de suite, jusqu'à ce que les centroïdes des nouveaux clusters correspondent aux point initiaux.

2.2.2 Une initialisation non aléatoire : K-means++

Plus les similarités seront importantes au sein d'un groupe et différentes entre deux groupes, plus notre algorithme sera puissant et fiable. Toutefois, la partie aléatoire de cet algorithme peut amener à des dérives. En effet, la sélection de plusieurs points de référence peut se faire à partir d'un même cluster. Faussant ainsi les résultats obtenus à la sortie de l'algorithme. C'est pourquoi, des algorithmes permettant la sélection de points optimaux ont été mis en place.

La méthode K-means++ consiste à sélectionner un point aléatoirement puis, à définir le point le plus éloigné de lui comme le sommet d'un autre cluster. Ensuite, on cherche à déterminer le point le plus éloigné de ces deux sommets afin de définir le repère d'un autre cluster. Et nous répétons cela jusqu'à obtenir les K clusters demandés.

2.2.3 Une alternative à la distance euclidienne : Mélange Gaussien

Le calcul basé sur la distance euclidienne peut biaiser les résultats puisque les clusters peuvent se recouper. Ainsi, si nous conservons l'algorithme brut, ce fait ne sera pas considéré et les clusters en seront impactés.

Une méthode souvent utilisée, pour pallier ce phénomène, est celle dite de mélange Gaussien. Elle suppose que la répartition des points dans le nuage se fait à l'aide d'une loi normale, et ce, grâce au théorème central limite. Ainsi, en considérant K clusters dans l'univers de points, il nous est possible d'estimer la probabilité qu'ont chaque point d'appartenir à ces dits clusters. Cela se fait en estimant leur poids selon chacune des gaussiennes. Le cluster à l'origine du point sera donc celui qui a le poids le plus élevé.

2.3 Avantages et inconvénients de l'algorithme K-means

Cet algorithme a pour principal atout sa simplicité dans son raisonnement et dans sa complexité temporelle que nous calculerons dans la sous-partie suivante. Ainsi, il peut être perfectionné tout en conservant un temps d'exécution raisonnable. Attention cependant à tout de même sélectionné un algorithme adéquat.
Toutefois, si nous ne prenons pas le temps de coder les versions plus perfectionnées de cette méthode, les résultats obtenus sont inégaux, car reposant en grande partie sur une notion d'aléatoire.
Aussi, la méthode K-means implique de connaître le nombre de clusters auparavant, ce qui n'est pas le cas si nous travaillons dans un cas pratique. Il est donc nécessaire d'exécuter le code avec différentes valeurs de K afin, dans un second temps, de pouvoir sélectionner la valeur la plus probable de clusters.

2.4 Évaluation de la complexité temporelle de l'algorithme K-means

Nous commençons par choisir de manière aléatoire les K points qui seront les sommets des différents clusters. Puis, pour les $(N-K)$ points restants, on mesure leur distance par rapport aux K clusters précédemment établis. Donc, en sommant, nous obtenons : $C(N) = (K(1 + (N - K)))$ où K est le nombre de clusters et N, le nombre de points dans la constellation. L'ordre de cette complexité est donc en $O(N)$.

3 Partition de l'espace avec la densité

3.1 Principe de l'algorithme DBSCAN

L'algorithme DBSCAN partitionne spatialement un jeu de données en mesurant la proximité d'un point vis-à-vis de son voisinage. On étudie ainsi une certaine densité autour du point considéré. Si elle s'avère trop faible, le point étudié est étiquetté comme un bruit. Ce principe de base justifie le sigle de la méthode : *Density-Based Spatial Clustering of Applications with Noise*.

3.1.1 Initialisation

L'algorithme considère trois entrées :

1. Les points à traiter. Initialement, ils sont tous étiquetés comme "non visités".
2. $min_p \in \mathbb{N}^*$: le nombre minimal de points à rassembler pour définir un cluster.
3. $\epsilon \in \mathbb{R}_+^*$: le rayon de voisinage. Il seuille la distance en dessous de laquelle deux points sont considérés comme voisins.

Par distance, nous entendons une application définie positive sur un ensemble $E \times E$ vérifiant l'inégalité triangulaire, ainsi que les propriétés de symétrie et de séparation. Elle peut être euclidienne, mais pas nécessairement.

3.1.2 Détection d'un cluster

Considérons aléatoirement un point P dans le nuage des points marqués comme "non visités". P est alors étiqueté comme "visité" et deux cas de figure se présentent :

1. Si dans une boule $\mathcal{B}$ de rayon ϵ et de centre P se trouvent au moins $(min_p - 1)$ points "non visités" ou "bruits", alors un nouveau cluster est identifié. Les points de $\mathcal{B}$ lui sont alors affectés. En revanche, hormis le centre P, ces points sont toujours considérés comme "non visités".
2. Sinon P est étiqueté comme "bruit". Son statut peut évoluer si il appartient à une autre une boule $\mathcal{B}$ vérifiant le premier point.

3.1.3 Diffusion d'un cluster

Après avoir identifié un nouveau cluster, ses points "non visités" sont étudiés. Chacun d'entre-eux est étiqueté comme "visité" et une boule $\mathcal{B}$ leur est associée. L'ensemble des points à l'intérieur de ces boules sont ajoutés au cluster. Il est impossible de trouver des points appartenant à un autre cluster dans $\mathcal{B}$ car sinon le point au centre de $\mathcal{B}$ en ferait parti, et ne serait pas considéré ici. Les points nouvellement ajoutés ne sont pas encore marqués comme "visités". Leur voisinage sera étudié en suivant la même procédure. Le cluster cesse de se diffuser quand tous ses points ont été "visités". L'algorithme recherche alors un nouveau cluster ou se termine.

3.1.4 Terminaison

L'algorithme se termine lorsque tous les points ont été marqués comme "visités" ou "bruits".

3.1.5 Avantages et inconvénients de la méthode

Les avantages sont multiples. Cet algorithme ne demande pas un nombre de cluster prédéfini, il identifie les bruits et ne présente pas de contraintes sur la morphologie des clusters. En revanche ses faiblesses résident dans ses paramètres d'entrées : ϵ et min_p. Ils doivent être adaptés selon le jeu de données. De plus, la valeur de ϵ se doit être précise pour des données possédant de nombreuses dimensions.

3.2 Implémentations de DBSCAN

3.2.1 Avec des listes chaînées

Principe

La structure élémentaire manipulée est le point. Elle se compose d'un tableau de dimension N stockant les coordonnées et d'un pointeur vers le prochain point. L'ensemble des points sont donc chaînés. L'implémentation de DBSCAN consiste alors à découper cette chaîne et d'en extraire au fur et à mesure certains points pour les regrouper au sein de sous-chaînes : les clusters. Les points restants de la chaîne principale forment le bruit.

Les clusters sont en réalités eux-mêmes des structures. Elles contiennent des pointeurs vers le premier point, le dernier point et le prochain cluster.

Pourquoi utiliser des listes chaînées ?

Lorsqu'un nouveau cluster $\mathcal{C}$ est détecté, DBSCAN considère un à un l'ensemble des points sans cluster pour chaque point identifié à $\mathcal{C}$. L'utilisation d'une liste chaînée permet de réduire dynamiquement le nombre de points à étudier au fur et à mesure, alors qu'un tableau de points statique conduit à des vérifications sur l'ensemble des points déjà classifiés.

Les clusters sont également chaînés car il est impossible de prévoir combien de clusters DBSCAN produira. A noter que lorsque l'algorithme cherche un nouveau cluster, les derniers sont complets : plus aucun point n'est à une distance inférieure ϵ d'un point du cluster. Il n'est donc pas nécessaire de parcourir la liste des clusters si un pointeur vers le dernier cluster est tenu à jour.

Complexité temporelle

On évalue la complexité temporelle en déterminant le nombre de distance calculé.

Soit le $p_c$ème point du cluster numéro c. En notant $|K_i|$ le nombre de points des clusters i précédents, le nombre de calcul de distance relatif au point considéré est :

$$N - p_c - \sum_{i=1}^{c-1} |K_i|$$

Donc au niveau du cluster numéro c :

$$\sum_{p_c=1}^{|K_c|} \left(N - p_c - \sum_{i=1}^{c-1} |K_i| \right)$$

En notant K le nombre de clusters, on obtient finalement :

$$\begin{aligned}
C(N) &= \sum_{c=1}^{K} \sum_{p_c=1}^{|K_c|} \left(N - p_c - \sum_{i=1}^{c-1} |K_i| \right) \\
&= N \sum_{c=1}^{K} \sum_{p_c=1}^{|K_c|} 1 - \sum_{c=1}^{K} \sum_{p_c=1}^{|K_c|} \left(p_c + \sum_{i=1}^{c-1} |K_i| \right) \\
&= N^2 - \sum_{c=1}^{K} \left(\sum_{p_c=1}^{|K_c|} p_c + |K_c| \sum_{i=1}^{c-1} |K_i| \right) \\
&= N^2 - \sum_{c=1}^{K} \left(\frac{|K_c|(|K_c|-1)}{2} + |K_c| \sum_{i=1}^{c-1} |K_i| \right)
\end{aligned}$$

On distingue deux termes : N^2 représente la complexité sans l'optimisation faîtes avec les listes chaînées, et le second terme constitue le gain calculatoire réalisé. Une écriture plus abstraite est :

$$C(N) = N^2 - f(\{|K_c|\}_{c \in [\![1,K]\!]})$$

Le gain réalisé dépend donc de la manière dont DSBCAN groupe les clusters. Plus précisément, si les premiers clusters contiennent une quantité importante de points alors le gain réalisé est important.

4 Conclusion

Même si au premier abord, la méthode de K-means semble rudimentaire, elle devient pertinente si on la répète ou la combine avec des méthodes simples de détection de points comme K-means++ ou la méthode de mélange gaussien. Elle présente également l'inconvénient de nécessiter la connaissance du nombre de clusters, ce qui n'est que rarement le cas. Il est donc indispensable de l'exécuter plusieurs fois avec des valeurs différentes en paramètres afin de choisir le nombre optimal de clusters présents.

La méthode DBSCAN nécessite également d'être réitérée puisqu'elle prend en paramètres la distance maximale entre deux points d'un même cluster et le nombre minimum de points pour définir un cluster. Or, ces paramètres ne peuvent être défini qu'après expérimentation et tâtonnement, montrant une faiblesse commune à ces deux algorithmes. Toutefois, la méthode DBSCAN présente l'avantage de ne nécessiter aucun algorithme complémentaire, tout cela au prix d'une complexité en $O(N^2)$ contre $O(N)$ pour la méthode K-means.

Enfin, pour aller un peu plus loin, nous aurions voulu évaluer la rapidité de l'exécution de ce code en fonction du langage de programmation utilisé. Malheureusement, le temps a manqué pour mener à bien cette idée. Nos codes sont disponibles sur le lien github suivant : https://github.com/Adrial-Knight/k-mean_DBSCAN_clustering.

Bibliographie

[1] Korajac, A. (2018, 29 novembre). *Unsupervised Learning*. LMU München.

[2] Mbengue, M. (2019, 1 décembre). *Clustering avec l'algorithme DBSCAN*. Pensée Artificielle.