Apprentissage non-supervisé

Augustin Bresset | Guillaume Macquart De Terline

Février 2024

1 Introduction

L'apprentissage non-supervisé est une branche de l'apprentissage automatique qui consiste à apprendre à partir de données non étiquetées. L'objectif est de trouver des structures dans les données, comme des groupes ou des clusters. L'apprentissage non-supervisé est souvent utilisé pour explorer des données et pour trouver des modèles cachés. Il est également utilisé pour la réduction de dimensionnalité, la visualisation de données et la génération de données. Dans cet article nous allons étudier et comparer les algorithmes suivants : K-means, K-medoids DBSCAN et Modèle à Mélange de Gaussiennes.

Pour faire ceci, nous allons utiliser le jeu de données MNIST qui est un jeu de données de chiffres manuscrits. Cela va nous permettre de facilement visualiser les résultats des algorithmes à travers un exemple concret. On s'attend à ce que les algorithmes trouvent des clusters correspondant à des chiffres similaires. Cependant, il est important de noter que les algorithmes ne connaissent pas les étiquettes des chiffres et ne savent pas quels chiffres sont similaires. C'est pourquoi on s'attend aussi à une confusion entre les chiffres similaires, comme le 3 et le 8 par exemple.

Enfin on va pouvoir comparer les performances des algorithmes en utilisant des métriques comme l'indice de silhouette.

1.1 Base de données MNIST

La base de données MNIST est un jeu de données de chiffres manuscrits. Chaque chiffre manuscrit est représenté par une image de 28x28 pixels Chaque pixel est représenté par un nombre entier entre 0 et 255, qui correspond à la luminosité du pixel. Il y a 60 000 images dans l'ensemble d'entraînement et 10 000 images dans l'ensemble de test.

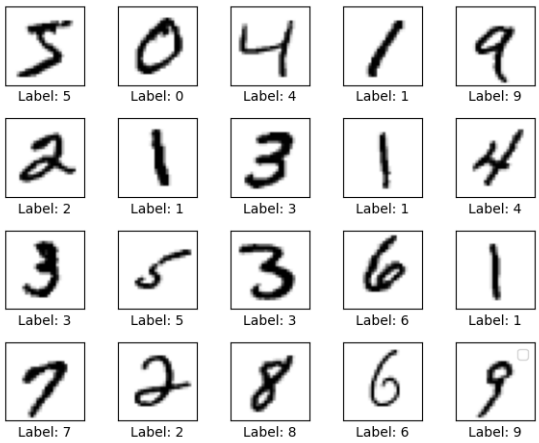

FIGURE 1 – Exemple de chiffres de la base de données MNIST

Ce jeu de données est souvent utilisé pour tester des algorithmes de classification et de clustering. On peut déjà remarqué ici la grande diversité des écritures des chiffres, ce qui rendra la tâche de clustering difficile.

1.2 Présentation des modèles

Pour montrer leur fonctionnement, on génère un jeu de données synthétique avec 300 points regroupés dans 5 clusters avec une variance de 0.8.

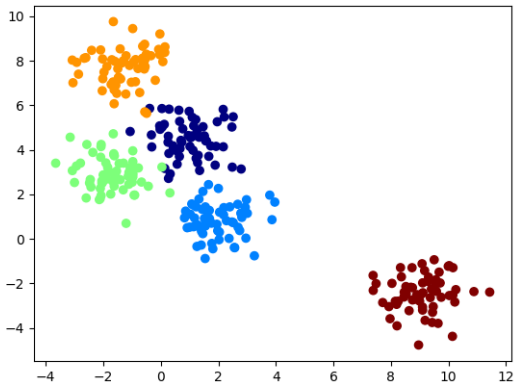

FIGURE 2 – Data généré de manière synthétique

On viendra dans la suite chercher à trouver 4 clusters dans ces données.

1.2.1 K-means

L'algorithme K-means est un algorithme de clustering qui cherche à partitionner les données en K clusters. L'algorithme fonctionne de la manière suivante :

1. Choisir K points aléatoires dans les données, qui serviront de centres de clusters.
2. Assigner chaque point de données au centre de cluster le plus proche.
3. Mettre à jour les centres de clusters en prenant la moyenne des points assignés à chaque cluster.
4. Répéter les étapes 2 et 3 jusqu'à ce que les centres de clusters ne changent plus.

On obtient les résultats suivants :

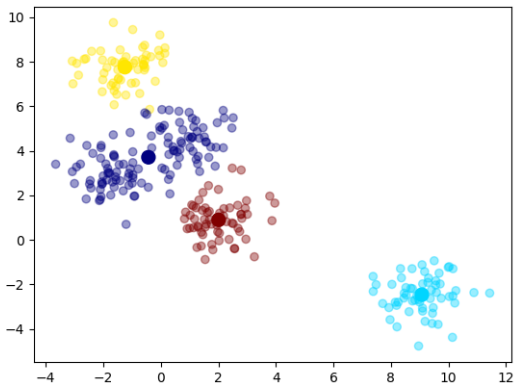

FIGURE 3 – Résultat de l'algorithme K-means

On remarque la fusion des deux clusters les plus proches ce qui est cohérent étant données que l'on cherche 4 clusters dans un ensemble de données regroupés en 5 clusters. On verra que cette remarque s'appliquera à tous les algorithmes.

Fonction de coût La fonction de coût utilisée par l'algorithme prend en compte la distance entre les points et les

centres des clusters. Elle est aussi appelée "inertie" et est définie comme suit :

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} 1\{c_i = k\} ||x_i - \mu_k||^2 \quad (1)$$

Avec :
— n le nombre de points de données
— K le nombre de clusters
— c_i le cluster auquel le point de données x_i est assigné

1.2.2 K-medoids

L'algorithme K-medoids est une variante de l'algorithme K-means. La différence principale est que K-medoids utilise des points de données comme centres de clusters, alors que K-means utilise des moyennes. Cela rend K-medoids plus robuste aux valeurs aberrantes que K-means. L'algorithme fonctionne de la manière suivante :

1. Choisir K points aléatoires dans les données, qui serviront de centres de clusters.
2. Assigner chaque point de données au centre de cluster le plus proche.
3. Mettre à jour les centres de clusters en prenant le point de données qui minimise la somme des distances aux autres points assignés à chaque cluster.
4. Répéter les étapes 2 et 3 jusqu'à ce que les centres de clusters ne changent plus.

On a aussi appliquer cet algorithme sur le jeu de données précédent.

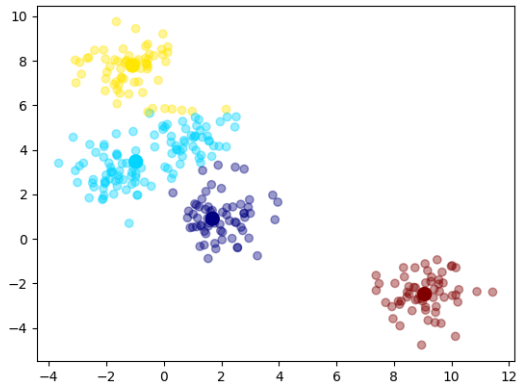

FIGURE 4 – Résultat de l'algorithme K-medoids

1.2.3 Modèle à Mélange de Gaussiennes

Un modèle de mélange gaussien (GMM) est un modèle probabiliste qui cherche à modéliser les données comme un mélange de distributions gaussiennes. L'algorithme fonctionne de la manière suivante :

1. Choisir K points aléatoires dans les données, qui serviront de centres de clusters.
2. Assigner chaque point de données à une distribution gaussienne en utilisant la loi de Bayes.
3. Mettre à jour les centres de clusters et les paramètres des distributions gaussiennes en maximisant la vraisemblance des données.
4. Répéter les étapes 2 et 3 jusqu'à ce que les centres de clusters ne changent plus.

Résultat de l'algorithme sur le jeu de données précédent.

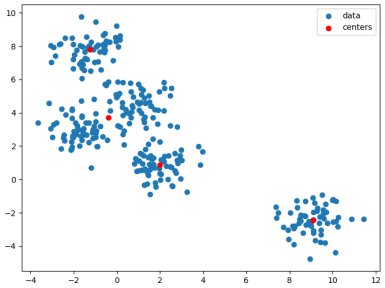

FIGURE 5 – Résultat de l'algorithme Modèle à Mélange de Gaussiennes

1.2.4 DBSCAN

DBSCAN est un algorithme de clustering qui est basé sur la densité des points. L'algorithme fonctionne de la manière suivante :

1. Choisir un point de données aléatoire.
2. Trouver tous les points de données qui sont à une distance inférieure à un seuil ϵ du point de données.
3. Si le nombre de points de données trouvés est supérieur à un seuil $minPts$, alors on crée un cluster avec ces points de données et on recommence à l'étape 1 avec un autre point de données.
4. Si le nombre de points de données trouvés est inférieur à $minPts$, alors on marque le point de données comme un point de bruit et on recommence à l'étape 1 avec un autre point de données.

On a appliqué cet algorithme sur le jeu de données précédent avec une valeur de ϵ de 0.2 et une valeur de $minPts$ de 5. Résultat de l'algorithme sur le jeu de données précédent.

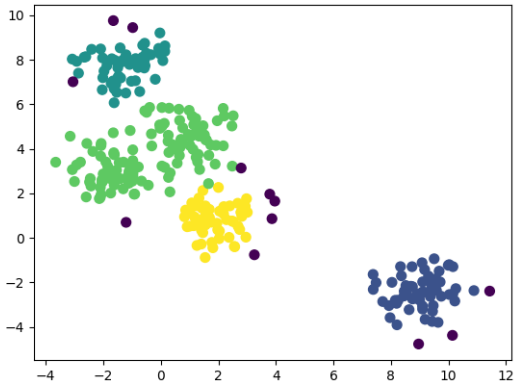

FIGURE 6 – Résultat de l'algorithme DBSCAN

On remarque ici que certains points ne sont pas assignés à un cluster, ce sont les points de bruit.

2 Application sur la base de données MNIST

2.1 K-means

Afin de baisser le temps de calcul, on ne va étudier que les 10 000 premières images de la base de données MNIST. On aura donc 10 clusters contenant chacun :

- label 0 : 1001 images
- label 1 : 1127 images
- label 2 : 991 images
- label 3 : 1032 images
- label 4 : 980 images
- label 5 : 863 images
- label 6 : 1014 images
- label 7 : 1070 images
- label 8 : 944 images
- label 9 : 978 images

2.1.1 Étude pour 10 clusters

On clusterise les images en utilisant les pixels comme features et on recherche 10 clusters. Cela nous permet d'avoir une première visualisation des résultats de l'algorithme pour un nombre de cluster cohérent avec le problème à résoudre.

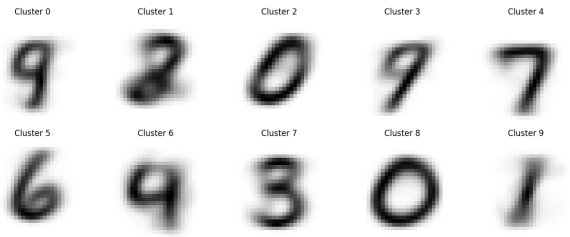

FIGURE 7 – Résultat de l'algorithme K-means sur la base de données MNIST

On remarque ici que malgré la présence de 10 clusters, on ne retrouve pas les 10 chiffres. En effet, deux clusters semblent décrirent le chiffre 0 et trois autres sont flous quant à leur description des chiffres 4 et 9. Enfin les clusters sont très hétérogènes, ce qui est cohérent avec la diversité des écritures des chiffres.

2.1.2 Étude des performances en fonction du nombre clusters : méthode usuelle

Fonction de coût On peut étudier la fonction de coût[1] par rapport au nombre de clusters. Ici on a tracé la fonction de coût pour un nombre de clusters allant de 2 à 20.

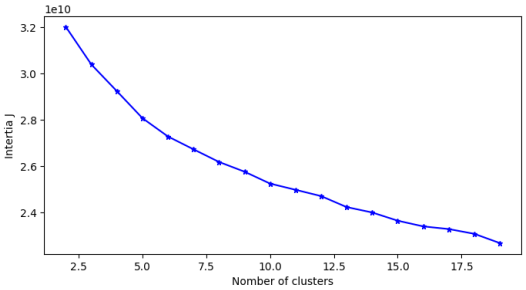

FIGURE 8 – Fonction de coût en fonction du nombre de clusters

Il est totalement cohérent que la fonction soit décroissante en fonction du nombre de clusters. En effet, plus on a de clusters, plus on peut réduire la distance entre les points et les centres des clusters.

La méthode du coude consiste à choisir le nombre de clusters qui correspond à l'endroit où la fonction de coût commence à décroître plus lentement. Ce choix permet de déterminer le nombre optimal de clusters.

Dans ce cas, la courbe ne présente pas de coude net. Elle ne permet donc pas de déterminer clairement le nombre de clusters optimal.

Indice de silhouette L'indice de silhouette est une mesure de la qualité des clusters. Il est défini comme suit :

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{2}$$

Avec :

- $a(i)$ la distance moyenne entre le point i et les autres points du même cluster
- $b(i)$ la distance moyenne entre le point i et les points du cluster le plus proche
- $s(i)$ l'indice de silhouette du point i

On peut caluler la moyenne groupée par cluster de l'indice de silhouette pour chaque nombre de clusters. Cela nous permet d'estimer l'homogénéité et la séparation des clusters.

On a tracé la moyenne de l'indice de silhouette pour un nombre de clusters allant de 2 à 12.

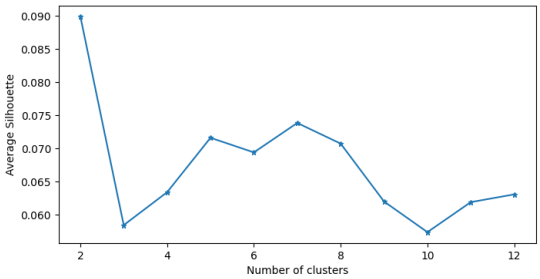

FIGURE 9 – Indice de silhouette en fonction du nombre de clusters

On remarque que l'indice de silhouette est maximal pour 2 clusters. Cependant on ne va pas prendre cette mesure en compte car en séparant simplement l'espace en deux, elle n'est pas représentative de la qualité des clusters. On va donc plutôt regarder le nombre suivant de clusters pour lequel l'indice de silhouette est maximal. Dans ce cas, le **nombre de cluster optimal est 7**.

2.1.3 Étude des performances en fonction du nombre clusters : méthode alternative

Matrice de confusion On peut aussi étudier la matrice de confusion pour chaque nombre de clusters. Celle ci représente pour la ligne i et la colonne j, le nombre de point de label i

qui sont dans le cluster j.

$$\begin{bmatrix} 812 & 14 & 1 & 4 & 61 & 71 & 38 & 0 & 0 & 0 \\ 0 & 2 & 3 & 1109 & 4 & 9 & 0 & 0 & 0 & 0 \\ 9 & 27 & 6 & 213 & 60 & 627 & 49 & 0 & 0 & 0 \\ 6 & 29 & 19 & 89 & 736 & 142 & 11 & 0 & 0 & 0 \\ 1 & 524 & 366 & 63 & 0 & 5 & 21 & 0 & 0 & 0 \\ 13 & 63 & 124 & 236 & 368 & 33 & 26 & 0 & 0 & 0 \\ 16 & 24 & 0 & 114 & 6 & 15 & 839 & 0 & 0 & 0 \\ 1 & 325 & 646 & 91 & 1 & 5 & 1 & 0 & 0 & 0 \\ 3 & 25 & 68 & 172 & 333 & 326 & 17 & 0 & 0 & 0 \\ 5 & 437 & 477 & 38 & 17 & 2 & 2 & 0 & 0 & 0 \end{bmatrix}$$

TABLE 1 – Matrice de confusion pour 7 clusters

2.2 K-medoids