

# Apprentissage non supervisé

Baudoint Emma et Baures Vincent

October 2021

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Démarches entreprises</b>	<b>3</b>
<b>3</b>	<b>Analyse et compréhension des différentes méthodes de clustering</b>	<b>4</b>
3.1	Clustering k-means . . . . .	4
3.2	Clustering agglomératif . . . . .	7
3.3	Clustering DBSCAN . . . . .	10
3.4	Clustering HDBSCAN . . . . .	12
<b>4</b>	<b>Analyse d'échantillons inconnus</b>	<b>13</b>
4.1	Dataset a . . . . .	14
4.2	Dataset t . . . . .	16
4.3	Dataset h . . . . .	17
4.4	Dataset zgo . . . . .	18
4.5	Dataset zgn . . . . .	20
4.6	Dataset tr . . . . .	22
4.7	Avantage et Inconvénient des différentes méthodes . . . . .	23
<b>5</b>	<b>Conclusion</b>	<b>24</b>

# Chapitre 1

## Introduction

Le clustering est une méthode utilisée pour organiser des données brutes en cluster homogènes. Les données sont regroupées selon des caractéristiques communes telles que leur proximité qui est mesuré à partir de critères définis.

## Chapitre 2

# Démarches entreprises

Afin d'évaluer la qualité de nos clusters, nous les avons évalués grâce à la métrique de la silhouette. Pour chaque point, c'est la différence entre la distance moyenne avec les points de son cluster et la distance moyenne avec les points des autres clusters. On fait la moyenne de ces valeurs : la valeur est comprise entre -1 et 1. Cela représente notre silhouette.

Afin de trouver le nombre de cluster idéal pour chaque algorithme, nous avons itérer en augmentant le nombre de clusters et en calculant la silhouette à chaque fois. Nous avons gardé à chaque fois le meilleur résultat.

Au début, nous avons choisi de nous arrêter dès que la silhouette cessait de s'améliorer. Cela nous coînçait dans un maximum local. Nous avons donc choisi ensuite d'itérer jusqu'à une limite que nous déterminions "à la main". Notre limite était bornée par le nombre de points présents dans notre cluster divisé par 2 et si on n'observait plus d'amélioration ou si le temps était trop long, on cessait de l'augmenter.

## Chapitre 3

# Analyse et compréhension des différentes méthodes de clustering

### 3.1 Clustering k-means

La méthode k-means se base sur la division de point en k clusters. Pour cela, elle est basée sur des centres mobiles. Le nombre de cluster K est fixé en avance et le but est de minimiser la distance entre les points à l'intérieur d'un cluster.

Dans cette approche, les centres des clusters sont d'abord choisis aléatoirement. La moyenne de poids du cluster est donc initialement aléatoire. Chaque nouveau centre d'un cluster est ensuite choisi avec une probabilité proportionnelle au carré de la distance entre le point et le cluster le plus proche. En iterant, le but est ensuite de diminuer cette moyenne de poids afin d'atteindre un minimum global.

Durant le TP, nous avons d'abord testé cette méthode sur un jeu de données simple en fixant le nombre de cluster K à celui qu'on pouvait observer sur le résultat attendu. Nous avons obtenu la figure suivante en appliquant l'algorithme sur le jeu de donnée 2d-4c :

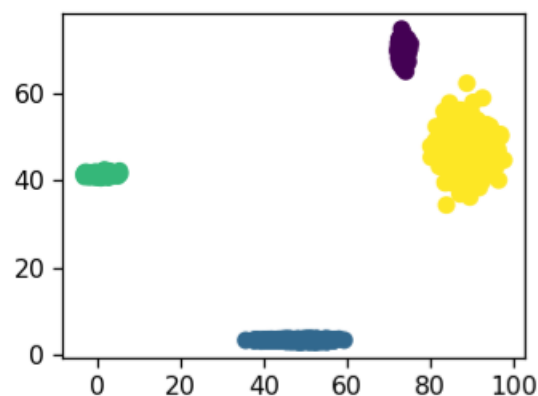


Figure n°1 : Graphique obtenu

Nous retrouvons bien ici les clusters correctement délimités.

Afin de connaître la performance de notre réponse, nous avons utilisé la métrique de la silhouette pour évaluer les algorithmes implémentés. Nous avons ensuite repris le même jeu de données mais en itérant cette fois-ci sur le nombre de clusters avec la démarche vue dans le chapitre 2. Nous avons ainsi retrouvé le même nombre de clusters que sur la figure précédente donnée comme solution.

K-means fonctionne très bien pour des clusters convexes et avec peu de points. Si la densité est trop importante, le temps de l'algorithme devient exponentiel. Les points ne seront plus associés au bon cluster car ils seront soumis à l'influence d'un autre cluster proche et plus dense. Sur le set long3 on obtient les résultats suivants :

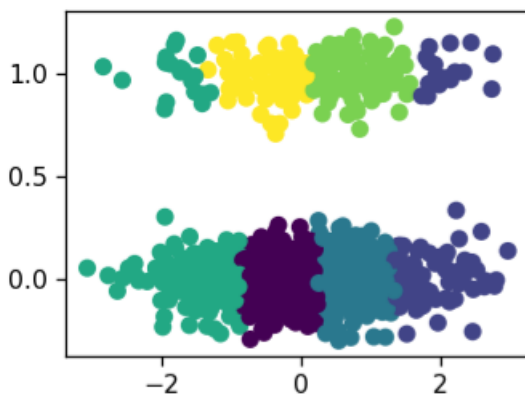


Figure n°2 : Graphique obtenu

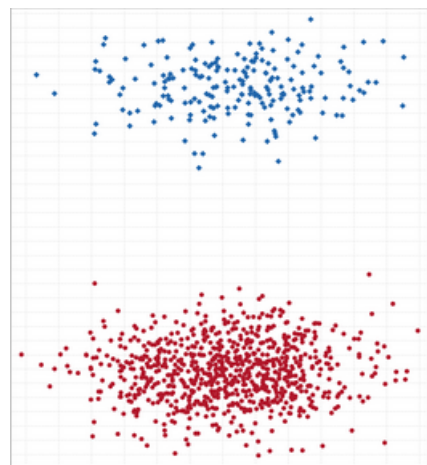


Figure n°3 : Graphique attendu

En itérant de manière croissante sur les clusters, et en fixant la limite à 50, on obtient le résultat ci-dessus avec les données suivantes :

- métrique : 0.4818106162564601
- nombre de cluster : 6
- temps : 9.300729990005493

Ici, k-means a donc détecté beaucoup plus de clusters que prévu dû aux problèmes de densité.

Si les clusters sont non convexes, k-means sera incapable de les déterminer puisque la distance entre le centre du cluster et certains points n'aura plus aucun sens. Voici un exemple de résultat du k-means sur des clusters non convexes qui illustre l'incapacité de k-means à repérer les clusters non convexes :

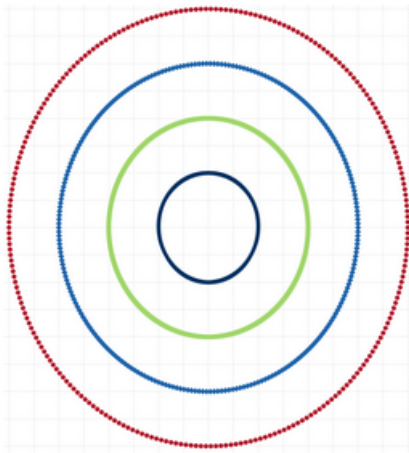


Figure n°4 : Graphique obtenu

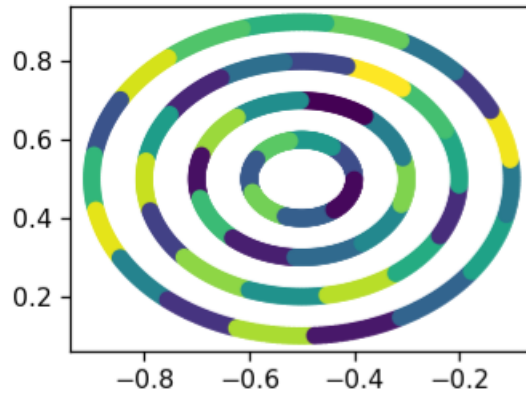


Figure n°5 : Graphique attendu

On obtient les résultats suivants :

- métrique : 0.48019278656249587
- nombre de cluster : 48
- temps : 11.437515497207642

Ici, dès que les clusters suivent des formes complexes, on peut observer l'incapacité de k-means à les repérer car il n'y a pas de notion de voisinage dans cet algorithme.

## 3.2 Clustering agglomératif

Le clustering agglomératif est basé lui sur des centres mobiles. En effet, on va considérer que chaque point est un cluster à lui tout seul. On va ensuite chercher à savoir quel est le cluster le plus proche de chaque cluster est les "agglomérer". On répète ensuite cette étape jusqu'à ce que tous les clusters appartiennent à un même cluster. Afin de visualiser ces clusters sur chaque niveau d'agglomération, on peut utiliser un dendrogramme.

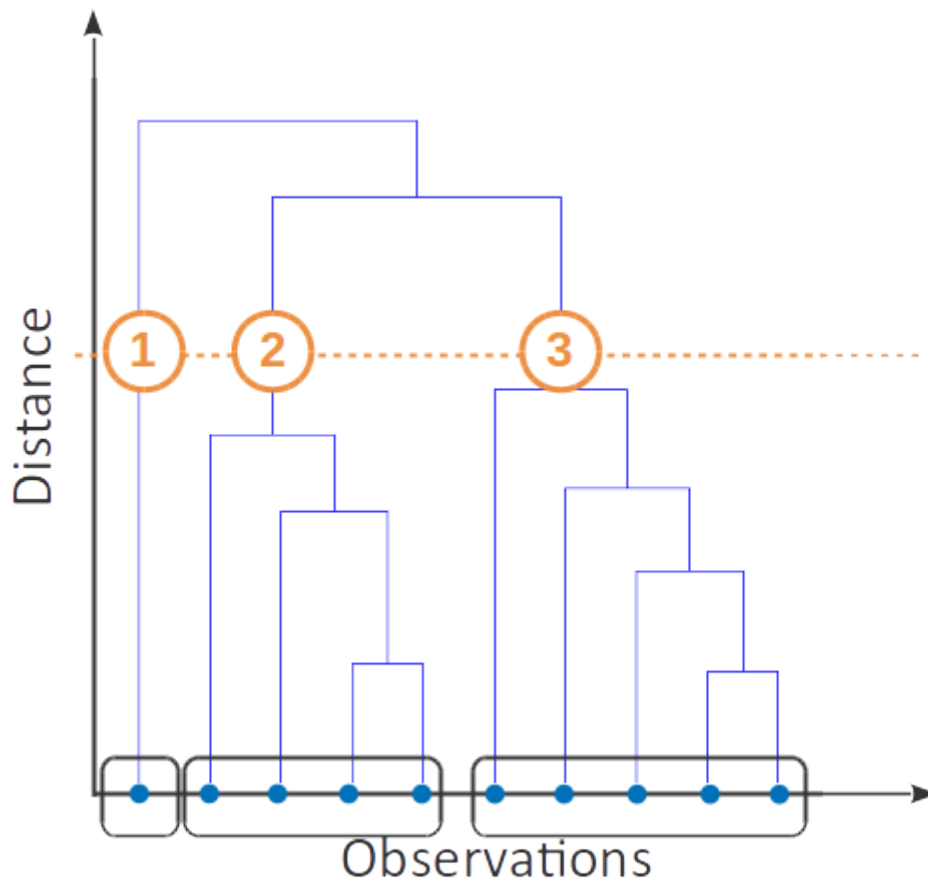


Figure 6 : Exemple de dendrogramme

En lançant notre algorithme, on va définir le nombre de cluster que l'on veut obtenir à la fin. Sur le dendrogramme ci-dessus, on veut trois cluster on va donc "couper" notre diagramme à l'endroit où on observe encore trois branches parallèles. C'est ainsi qu'on va déterminer nos clusters.

La distance entre deux clusters est déterminée grâce au linkage. Il existe 4 méthodes de linkage sur lesquels on va itérer avec le même exemple. Observons l'influence de ce linkage sur un même exemple de cluster, le set de donnée long3 vu plus tôt dans la partie k-means :

1. Simple : La distance entre deux cluster est définie par la distance entre deux points de se cluster



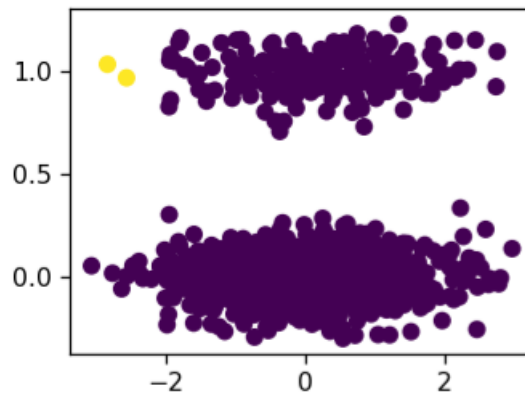


Figure n°7 : Single linkage

2. Complet : La distance entre deux cluster s'appuie ici sur la distance entre tous les points des deux clusters. On va donc ici calculer la distance maximale entre deux points de deux clusters différents.

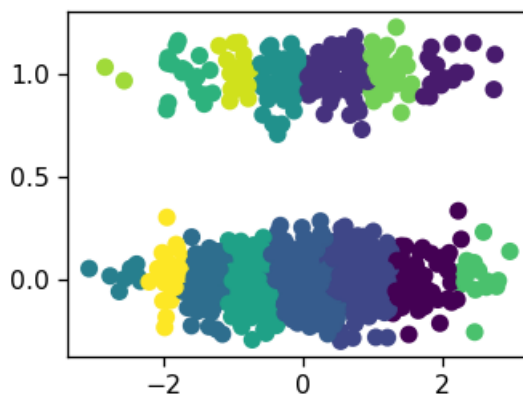


Figure n°8 : Complete linkage

3. Moyenne : La distance entre deux clusters s'appuie sur la distance moyenne entre toutes les paires de points

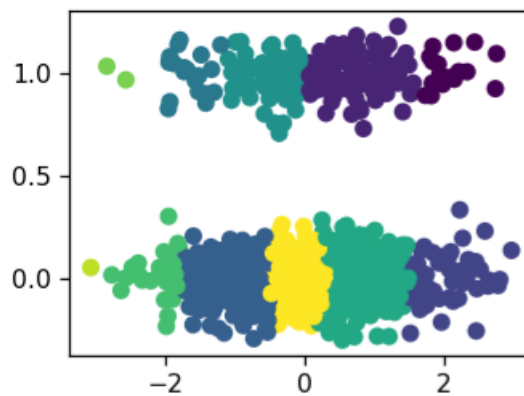


Figure n°9 : Average linkage

4. Ward : Le clustering de Ward permet à chaque étape d'agréger deux clusters afin de minimiser l'augmentation de la variance inter-cluster. C'est une méthode de clustering hiérarchique.

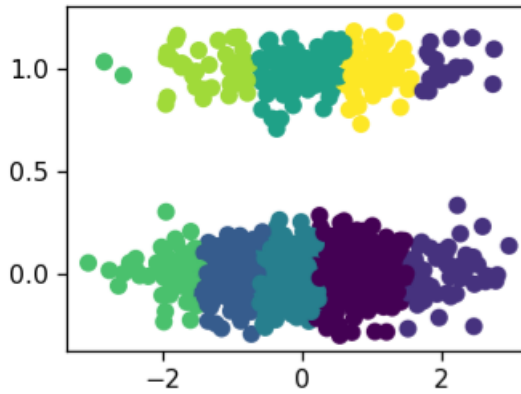


Figure n°10 : Ward linkage

Les trois premières méthodes permettent de garantir la séparation mais pas l'homogénéité des clusters. Le linkage ward sert à prendre en compte cette homogénéité. On observe cependant ici que quelque soit la méthode de linkage utilisée, on n'arrive toujours pas à obtenir de bons résultats.

On peut observer que cette méthode de clustering est moins sensible à la densité de donnée que k-means. En effet, on peut obtenir ce genre de résultat sur des données dense.

Cette méthode permet aussi d'identifier de nombreuses formes si l'on paramètre correctement le linkage :

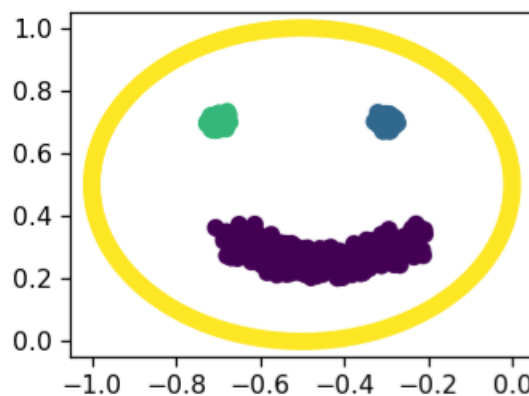


Figure n°11 : Smile1

### 3.3 Clustering DBSCAN

Le clustering DBSCAN introduit une notion de voisinage qui n'était pas présente dans les précédentes méthodes de clustering. Il permet de déterminer des formes de clusters non convexes et ne nécessite pas de renseigner le nombre de cluster au lancement de l'algorithme.

DBSCAN s'appuie sur deux paramètres : la distance epsilon et le nombre minimum de points devant se trouver dans ce rayon epsilon. Les points se trouvant dans ce rayon sont considérés comme un cluster. Si la valeur d'epsilon est trop grande, tous les points vont se retrouver dans le même cluster. Si au contraire elle est trop petite, on va constater beaucoup d'anomalie.

Il est possible d'arriver sur de bons résultats en "tatonnant" sur les paramètres d'epsilon et de min sample. On peut obtenir les figures suivantes qui correspondent aux exemples donnés et vus ci-dessus ( min sample = 2, epsilon = 0,01 pour dartboard1 et min sample = 2, epsilon = 0,1 pour smile1 ) :

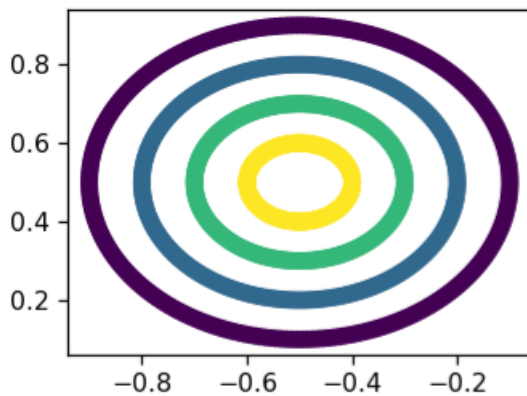


Figure n°12 : dartboard1

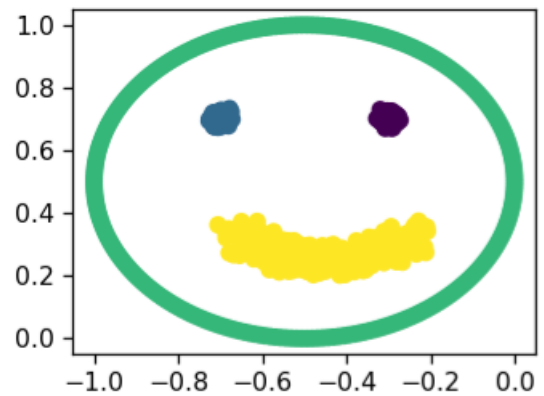


Figure n°13 : smile1

On a choisit de faire choisir une plage de points minimum pour former un cluster et de faire varier epsilon sur cette plage de point. On va appliquer cette méthode sur le dartboard1. En calculant la silhouette sur le modèle suivant on obtient la variation de la silhouette en fonction de epsilon pour un nombre minimum de point donné suivant.

Cette méthode de clustering permet de déterminer des formes non convexes et est sensible face au bruit et aux anomalies. Elle reste cependant compliquée à mettre en place à cause de la complexité du paramétrage et reste sensible aux clusters de différentes densités. Par exemple, nous n'avons pas réussi à trouver de paramètres satisfaisants pour long3. Le meilleur résultat obtenu avec les paramètres : min sample et epsilon est le suivant :

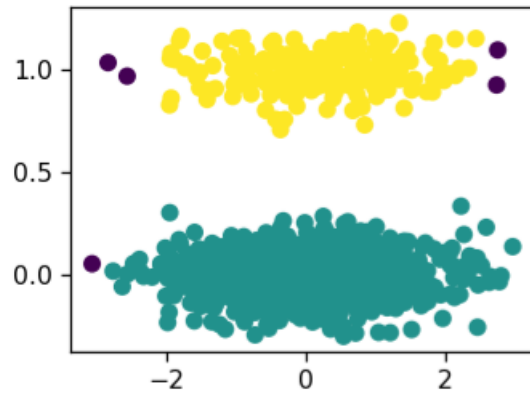


Figure n°12 : obtenu

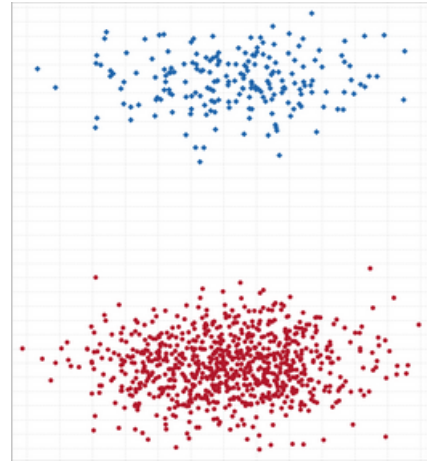


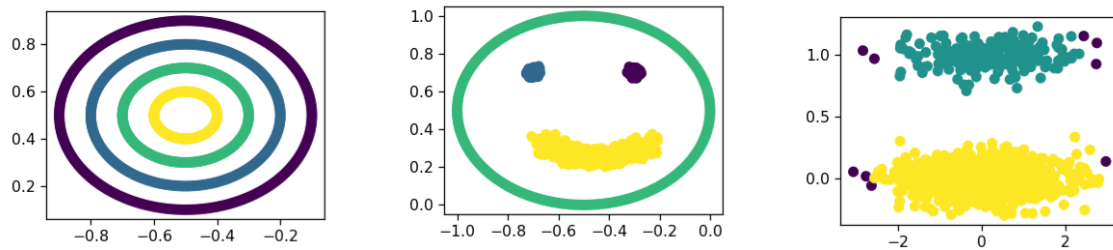
Figure n°13 : attendu

On obtient bien deux clusters et on peut remarquer que cet algorithme est sensible au bruit. On observe effectivement les points violets qui peuvent représenter le bruit. Il reste cependant très difficile à paramétrer. En effet on doit faire de la recherche à "tâton" peu précise.

### 3.4 Clustering HDBSCAN

HDBSCAN fonctionne au départ comme DBSCAN avec la transformation de l'espace en fonction de la densité. Cependant, au lieu de prendre une valeur epsilon, le dendrogramme est divisé en petit nombre de points transformés en un seul point global. Cela se traduit pas un arbre plus petit. Cet arbre est ensuite utilisé pour sélectionner les clusters les plus stables.

En appliquant cette méthode sur les datasets vu précédemment, on observe les résultats suivants. On retrouve bien les clusters attendus avec en plus la détection du bruit.



Cette méthode permet d'améliorer la résistance aux variations de densité et d'éliminer les problèmes de paramétrage rencontrés avec epsilon dans DBSCAN. HDBSCAN semble être l'algorithme le plus performant sur des formes complexes.

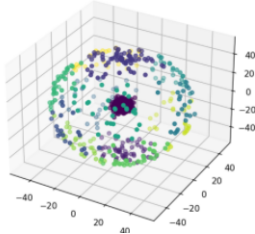
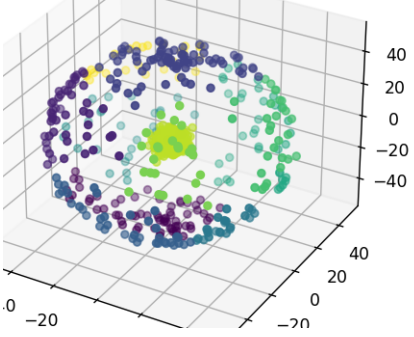
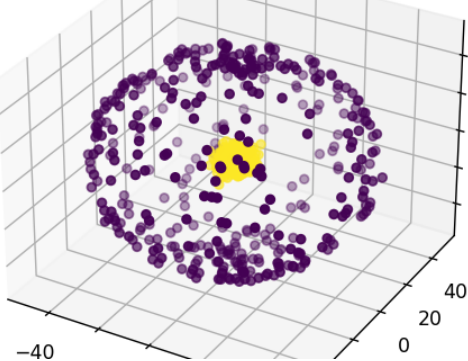
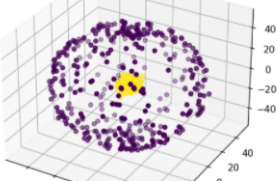
## Chapitre 4

# Analyse d'échantillons inconnus

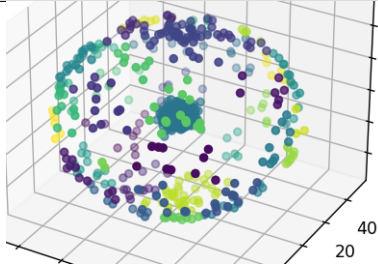
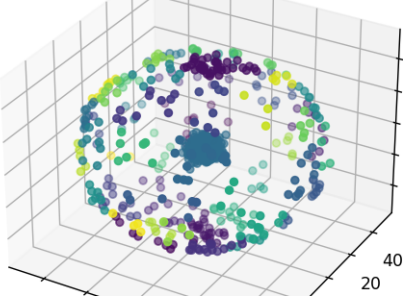
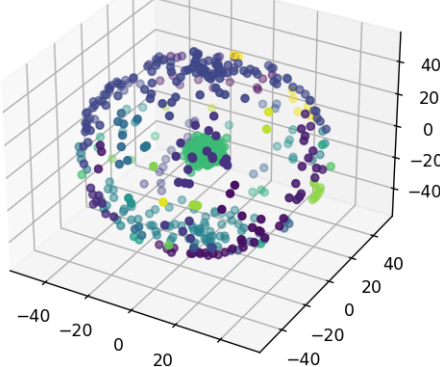
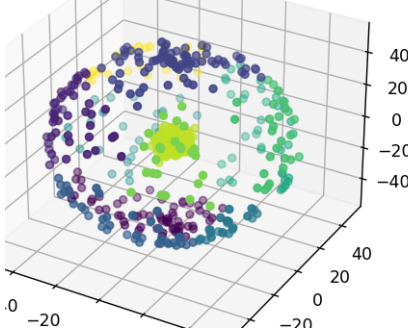
Dans cette partie, on va analyser les différentes méthodes de clustering sur différents datasets inconnus. Pour Kmeans et clustering agglomératif, on met en place une méthode de recherche liée à un nombre d'itération sur des clusters : on recherche automatiquement la meilleure silhouette en augmentant le nombre de cluster et en bloquant à un max d'itération fixé à 50. Pour DBSCAN et HDBSCAN, on va chercher notre résultat à "taton" car les paramètres sont très sensibles et difficiles à trouver.

## 4.1 Dataset a

Le dataset a est une forme non convexe. On s'attend donc à ce que kmeans ne nous donne pas de résultats très intéressants. On obtiendra des résultats plus pertinents avec les autres algorithmes. Au vue de la forme dessinée on s'attend à trouver deux clusters, un au centre et un autour. Pour la méthode de clustering agglomératif, on affiche ici le meilleur linkage qui est celui ward ici (informations détaillées plus bas).

Kmean		métrique: 0.6224202013307731 nombre de cluster: 13 temps: 27.797864198684692
Clustering agglomératif		métrique: 0.6012875373212041 nombre de cluster: 11 temps: 3.065884590148926
DBSCAN		score: 0.31149297508558416 eps: 20 min sample: 1
HDBSCAN		métrique: 0.4611686741443048 Taille minimale du cluster: 33 temps: 3.0757153034210205

On observe bien ici effectivement que le k-means est incapable de repérer la forme non convexe autour qu'il repère la forme convexe au centre. HDBSCAN donne de bons résultats et trouve bien les deux clusters que nous souhaitons identifier. La méthode DBSCAN ne fonctionne que si on choisit un min sample=1, pour les autres, cela nous trouve un seul et unique cluster. Cela peut venir des différences de densité au sein de notre dataset. Finalement on observe les résultats suivants pour la méthode de clustering agglomératif :

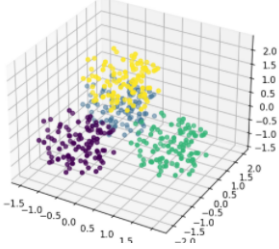
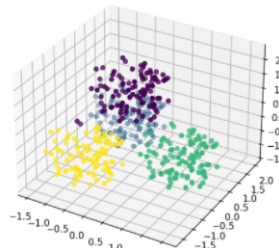
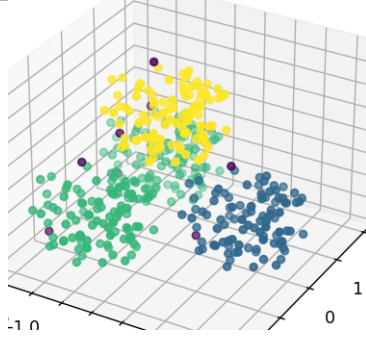
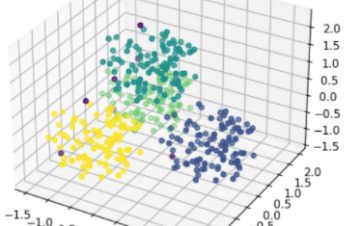
Average		métrique: 0.6167270963497942 nombre de cluster: 40 temps: 1.5067260265350342
Complete		métrique: 0.6071547148117817 nombre de cluster: 49 temps: 2.9610507488250732
Single		métrique: 0.4068197148493168 nombre de cluster: 48 temps: 1.239497184753418
Ward		métrique: 0.6012875373212041 nombre de cluster: 11 temps: 3.065884590148926

On peut en déduire ici, que le meilleur linkage est le linkage ward pour ce type de dataset. C'est celui pour lequel le nombre de clusters se rapproche le plus de ce qui est recherché. Cela paraît logique puisque notre dataset est de forme non convexe et que le linkage le plus efficace sur ce type de dataset est le linkage ward (comme vu dans la première partie).



## 4.2 Dataset t

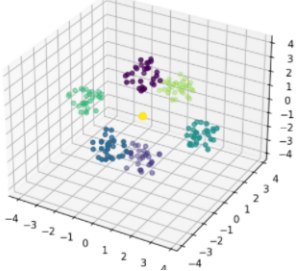
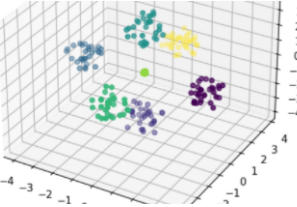
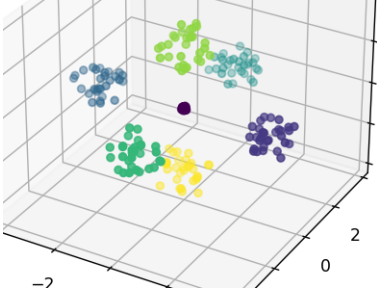
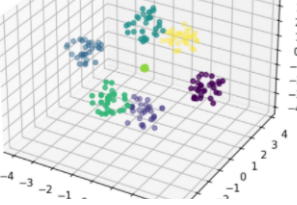
Le dataset t a des formes convexes. EN regardant la figure on peut identifier 4 clusters, de densité assez égale. On s'attend donc à ce que tous les algorithmes arrivent à interpréter et reconnaître les clusters. Pour la méthode de clustering agglomératif, on ne présentera ici que le meilleur linkage qui est le linkage average et qui est performant sur les formes convexes que l'on a ici..

Kmean		<p>métrique: 0.5057889289788572  nombre de cluster: 4  temps: 23.060959577560425</p>
Clustering agglomératif		<p>métrique: 0.5010021884877641  nombre de cluster: 4  temps: 1.0321087837219238</p>
DBSCAN		<p>score: 0.31475918776109435  eps: 0.44  min sample: 3</p>
HDBSCAN		<p>métrique: 0.7019231989948802  Taille minimale du cluster: 19  temps: 0.18745207786560059</p>

On peut observer ici que tous les algorithmes arrivent à reconnaître les 4 clusters et à les identifier. HDBSCAN et DBSCAN arrivent en plus à reconnaître le bruit de ce dataset. Au niveau des temps d'exécutions, on peut remarquer que k-means a un temps d'exécution assez important comparé au clustering agglomératif.

### 4.3 Dataset h

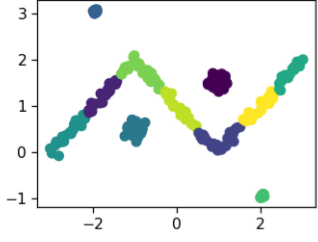

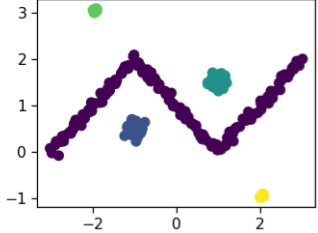
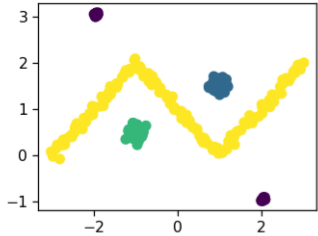
Le dataset a est une forme convexe. Les clusters sont bien espacés et de densités égales. Tous les algorithmes devraient être en mesure de reconnaître ces clusters.

Kmean		<pre>métrique: 0.7019231989948802 nombre de cluster: 7 temps: 18.548879861831665</pre>
Clustering agglomératif		<pre>métrique: 0.7019231989948802 nombre de cluster: 7 temps: 0.5350384712219238</pre>
DBSCAN		<pre>score: 0.7019231989948802 eps: 0.9 min sample: 5</pre>
HDBSCAN		<pre>métrique: 0.7019231989948802 Taille minimale du cluster: 19 temps: 0.18745207786560059</pre>

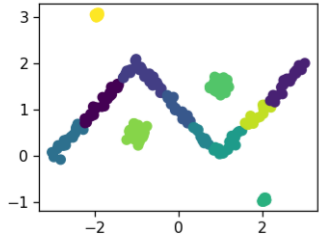
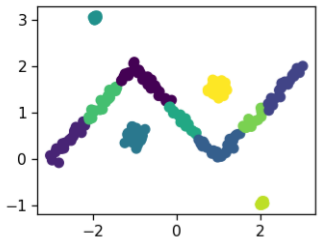
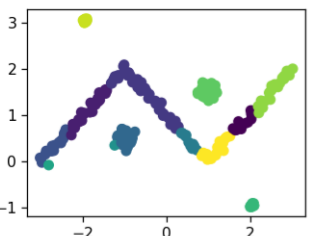
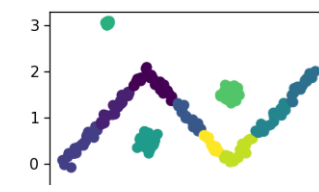
On observe bien ici que tous nos algorithmes sont capables d'identifier les clusters. On peut noter le temps assez important que met Kmean à reconnaître le bon nombre de cluster. On peut aussi noter que DBSCAN reste très difficile à configurer afin de retrouver le bon nombre de cluster et le bruit.

## 4.4 Dataset zgo

Le dataset zgo a des formes non convexes. On est censé pouvoir identifier une forme centrale et de petits clusters autour. L'algorithme kmeans risque de ne pas être efficace sur ce dataset.

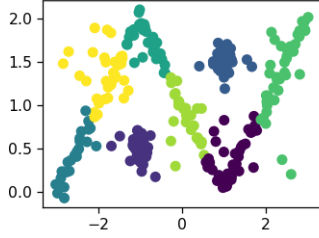
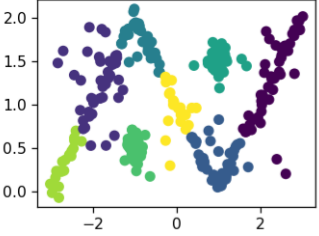
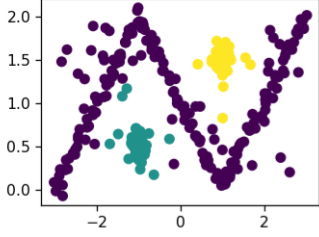
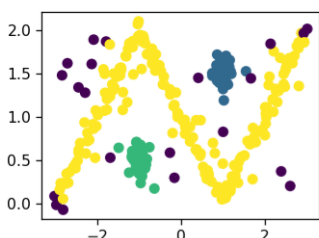
Kmean		métrique: 0.6876525786780527 nombre de cluster: 11 temps: 5.469648599624634
Clustering agglomératif		métrique: 0.6827730873266927 nombre de cluster: 12 temps: 0.296083927154541
DBSCAN		score: 0.21486984278572413 eps: 0.5 min sample: 10
HDBSCAN		métrique: 0.21486984278572413 Taille minimale du cluster: 15 temps: 0.19018840789794922

On peut observer que kmeans n'arrive pas du tout à identifier cette forme non convexe comme on s'y attendait. HDBSCAN et DBSCAN arrivent à bien identifier les différents clusters avec les paramètres renseignés. Suivant la taille minimal du cluster renseignée pour HDBSCAN et DBSCAN, on peut obtenir que les deux petits clusters aux extrémités soient considérés comme du bruit ou non. Concernant la méthode de clustering agglomératifs, le meilleur résultat obtenu est celui obtenu avec le linkage ward :

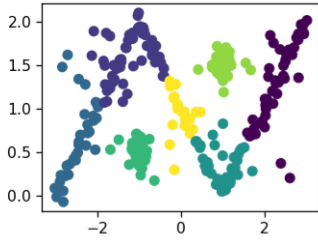
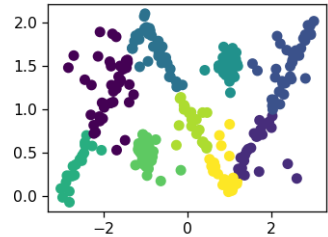
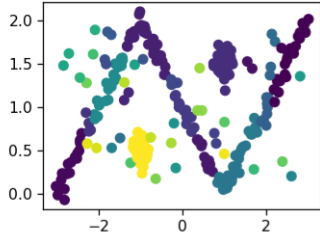
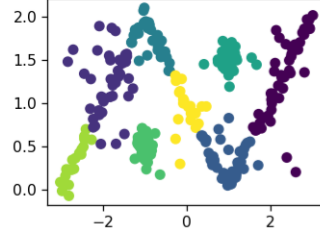
Average		<p> métrique: 0.6785059540435935  nombre de cluster: 12  temps: 0.23154377937316895 </p>
Complete		<p> métrique: 0.6718499349227364  nombre de cluster: 11  temps: 0.2693004608154297 </p>
Single		<p> métrique: 0.5274872161458021  nombre de cluster: 13  temps: 0.25682902336120605 </p>
Ward		<p> métrique: 0.6827730873266927  nombre de cluster: 12  temps: 0.296083927154541 </p>

## 4.5 Dataset zgn

Le dataset zgn a des formes non convexes. Nous sommes censé pouvoir identifier une forme centrale, un cluster et du bruit. L'algorithme kmean risques de ne pas être efficace sur cet algorithme. Il présente aussi du bruit que HDBSCAN devrait être en mesure d'identifier.

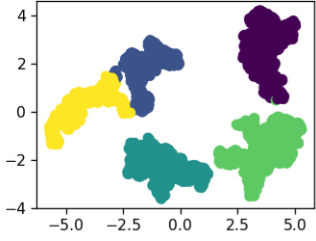
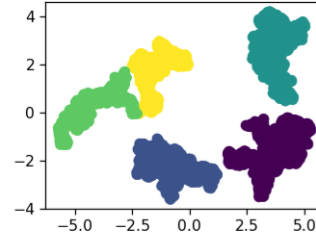
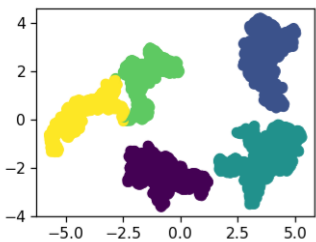
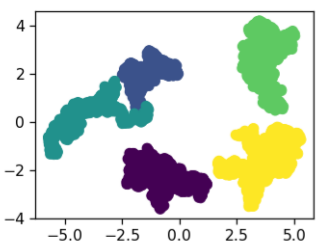
Kmean		métrique: 0.5925272025254529 nombre de cluster: 8 temps: 7.542797565460205
Clustering agglomératif		métrique: 0.5743819808245126 nombre de cluster: 8 temps: 0.48215222358703613
DBSCAN		score: 0.06789543854848062 eps: 0.5 min sample: 10
HDBSCAN		métrique: 0.052412133031889185 Taille minimale du cluster: 19 temps: 0.3345212936401367

On peut observer ici que kmeans n'arrive pas du tout à identifier cette forme non convexes comme on s'y attendait. HDBSCAN et DBSCAN arrivent à bien identifier les différents cluster avec les paramètres renseignés. HDBSCAN arrive même à identifier du bruit. Avec une meilleure paramétration de HDBSCAN, on pourrait réussir à identifier encore plus précisément les clusters et le bruit. Ici on a fixé min samples à 13 et min cluster size à 19. De même sur DBSCAN avec une évaluation plus fine des paramètres il doit être possible d'identifier le bruit mais nous n'avons pas réussi. Pour le clustering agglomératif, nous avons gardé la linkage ward car c'est le plus adapté aux formes non convexes. cependant ici, le linkage average semble donner un meilleur résultat :

Average		<p>métrique: 0.5737179826659661  nombre de cluster: 7  temps: 0.37294888496398926</p>
Complete		<p>métrique: 0.5658392809840586  nombre de cluster: 9  temps: 0.47658824920654297</p>
Single		<p>métrique: 0.2135091346277793  nombre de cluster: 47  temps: 0.40441465377807617</p>
Ward		<p>métrique: 0.5743819808245126  nombre de cluster: 8  temps: 0.48215222358703613</p>

## 4.6 Dataset tr

Le dataset tr a des formes convexes mais une densité extrêmement importante. On va donc pouvoir analyser l'efficacité en temps de nos algorithmes sur ce dataset. Pour HDBSCAN, nous allons itérer le même nombre de fois que pour clustering agglomératif et Kmean sur l'algorithme.

Kmean		<code>métrique: 0.6537600400912208</code> <code>nombre de cluster: 5</code> <code>temps: 186.5591971874237</code>
Clustering agglomératif		<code>métrique: 0.6538846549209778</code> <code>nombre de cluster: 5</code> <code>temps: 349.4967608451843</code>
DBSCAN		<code>score: 0.6532803251538253</code> <code>eps: 0.42</code> <code>min sample: 40</code>
HDBSCAN		<code>métrique: 0.6492858972614843</code> <code>Taille minimale du cluster: 6</code> <code>temps: 62.825504302978516</code>

On peut observer ici que tous les algorithmes parviennent à identifier les clusters correctement. La différence se situe ici sur le temps mis pour identifier les clusters. Kmeans et clustering agglomératif sont des méthodes qui, sur des datasets de cette densité, sont très lentes. HDBSCAN et DBSCAN reste plus rapide. Pour le clustering agglomératif, nous avons décidé de garder le linkage ward.

## 4.7 Avantage et Inconvénient des différentes méthodes

1. Kmeans : Pratique sur des clusters convexes et de petites densités, trop long sur des clusters dense, et incapable de repérer des clusters non convexes. Il est facile a paramétrer
2. Clustering agglomératif : Pratique sur des formes non convexes et parfois convexes, il atteind ses limites sur des formes non convexes complexes ou de densité inégale. Le temps d'exécution sur des clusters très dense est de même beaucoup trop long.
3. DBSCAN : Difficile à paramétrer mais donne de bon résultat sur toutes les formes de cluster.
4. HDBSCAN est plus facile à paramétrer sur DBSCAN et donne de bon résultats sur toutes les formes de cluster, y compris ceux de densité inégale et est sensible au bruit.



## Chapitre 5

# Conclusion

Pour conclure, évaluer les performances d'un algorithme de clustering en classification non supervisée peut être compliqué. On peut comparer les méthodes données par les différents algorithmes pour se faire une idée et avoir des éléments de références. Dans la première partie nous pouvions nous appuyer sur les résultats que nous étions censé obtenir donné sur le git.

Concernant la seconde partie, les observations faites nous ammenent à la conclusion que, avec un bon paramétrage, HDBSCAN est l'algorithme de clustering le plus performant. L'algorithme Kmeans est celui qui nécessite le plus long temps d'exécution mais qui est le plus simple à paramétrer et qui s'exécute convenablement uniquement sur des clusters convexes. Utiliser plusieurs méthodes sur un même dataset permet d'affiner la recherche.