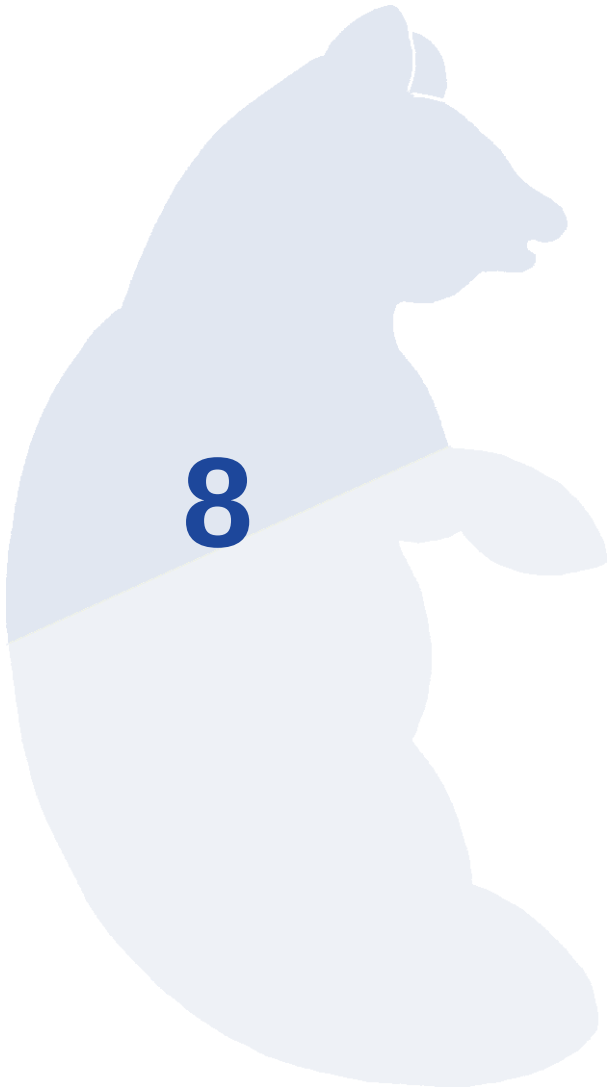


HDBSCAN

8. HDBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) est un algorithme de clustering développé par Campello, Moulavi et Sander.

Il étend DBSCAN en le convertissant en un algorithme de clustering hiérarchique, puis en utilisant une technique pour extraire un clustering plat basé sur la stabilité des clusters.

HDBSCAN et comment il excelle même lorsque les données ont :

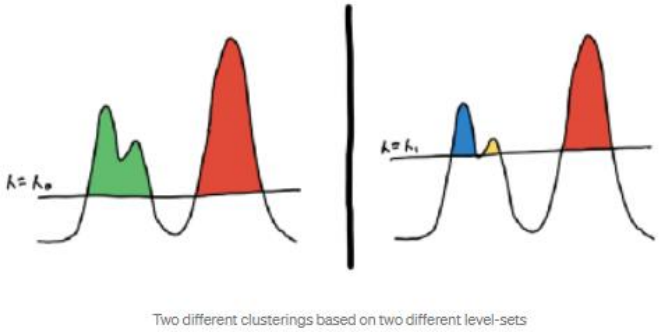
- des clusters de forme arbitraire
- des clusters de tailles et de densités différentes
- Du bruit

HDBSCAN utilise une approche basée sur la densité qui fait peu d'hypothèses implicites sur les clusters. Il s'agit d'une méthode non paramétrique qui recherche une hiérarchie de clusters formée par les modes multivariés de la distribution sous-jacente. Plutôt que de rechercher des clusters ayant une forme particulière, elle recherche des régions de données qui sont plus denses que l'espace environnant. L'image mentale que vous pouvez utiliser est d'essayer de séparer les îles de la mer ou les montagnes de ses vallées.

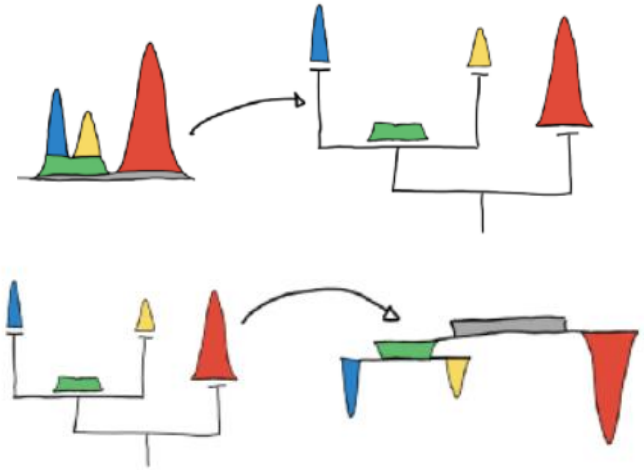
Source
Source

8. HDBSCAN

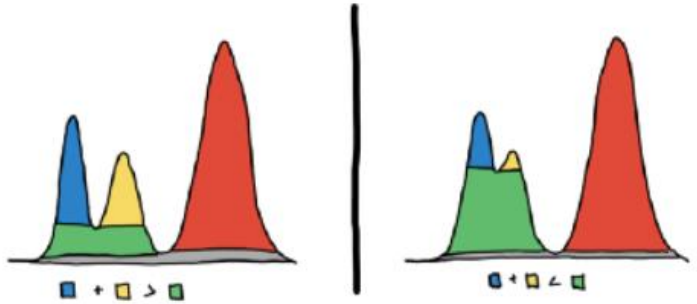
Une façon de définir cela est de fixer un certain seuil global pour le PDF de la distribution sous-jacente. Les composantes connectées des ensembles de niveaux résultants sont vos clusters. C'est ce que fait l'algorithme DBSCAN, et si on le fait à plusieurs niveaux, on obtient DeBaCl.

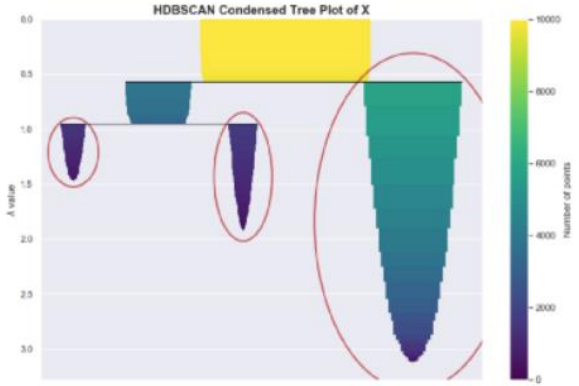


Two different clusterings based on two different level-sets



Visualizing the tree top-down





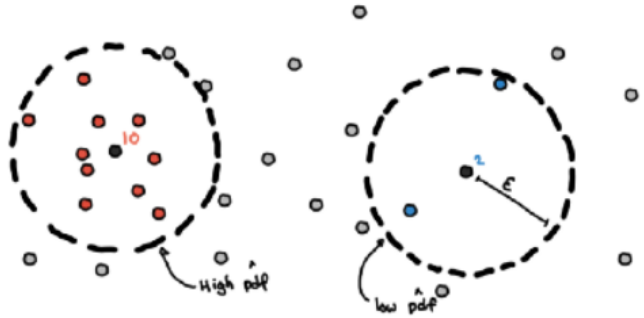
Condensed tree plot from HDBSCAN

Sur la gauche, nous voyons que la somme des aires des régions bleue et jaune est plus grande que l'aire de la région verte. Cela signifie que les 2 pics sont plus saillants, nous décidons donc qu'il s'agit de deux clusters distincts.
À droite, nous voyons que la zone du vert est beaucoup plus grande. Cela signifie que ce ne sont que des "bosses" plutôt que des pics. Nous disons donc qu'il s'agit d'un seul groupe.
Dans la littérature, la surface des régions est la mesure de la persistance, et la méthode est appelée eom ou excès de masse. Un peu plus formellement, nous maximisons la somme totale de la persistance des clusters sous la contrainte que les clusters choisis ne se chevauchent pas.

8. HDBSCAN

Par conséquent, nous devons estimer la PDF en utilisant la densité empirique. Nous avons déjà discuté d'une façon de le faire, en utilisant un histogramme. Cependant, cette méthode n'est utile que pour les données unidimensionnelles et devient difficile à calculer lorsque le nombre de dimensions augmente.
Nous avons besoin d'autres moyens pour obtenir le PDF empirique.
En voici deux :

- Compter le nombre de voisins d'un point particulier dans son ε-radius.
- Trouver la distance au K-ième plus proche voisin (c'est ce qu'utilise HDBSCAN)



Estimation of pdf using neighbor counts

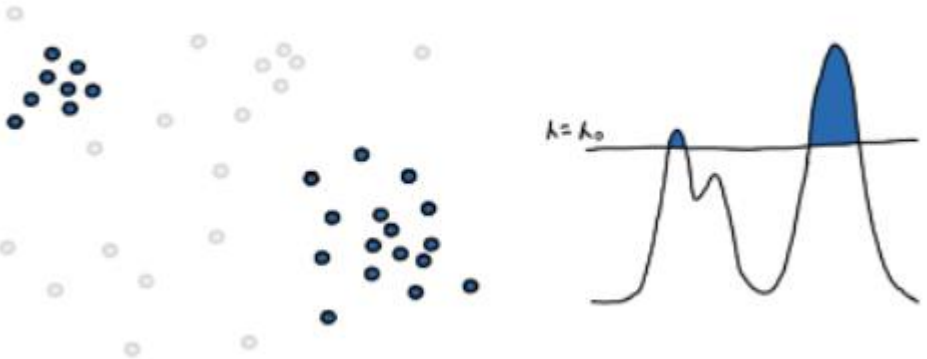
Compter les voisins dans un rayon de ε.
Pour chaque point, nous dessinons une hypersphère de ε-radius autour du point et comptons le nombre de points qui s'y trouvent. Il s'agit de notre approximation locale de la densité en ce point de l'espace.

Les résultats sont ce que nous appelons les distances centrales dans HDBSCAN. Les points avec des distances centrales plus petites sont dans des régions plus denses et auraient une estimation élevée pour le PDF. Les points avec des distances centrales plus grandes sont dans des régions plus clairsemées parce que nous devons parcourir de plus grandes distances pour inclure suffisamment de voisins.

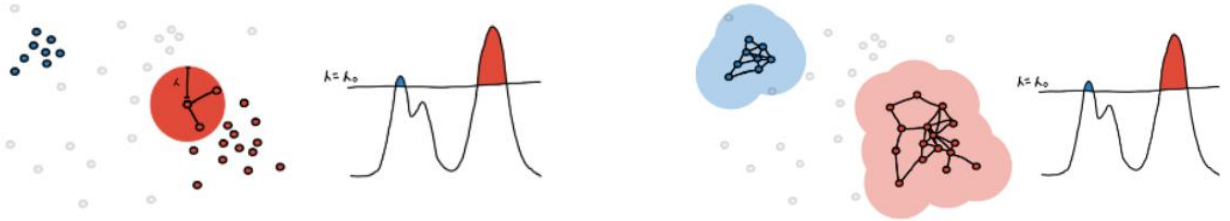
8. HDBSCAN

Trouver le niveau-ensemble et colorer les régions
Rappelez-vous que dans les exemples précédents, nous obtenons un ensemble de niveaux à partir du PDF et les régions résultantes sont nos clusters. C'était facile car une région était représentée par une forme quelconque. Mais lorsque nous avons affaire à des points, comment savoir ce que sont les différentes régions ?

La première étape consiste à trouver le level-set à un certainλ. Nous filtrons pour les régions pdf(x) $\geq \lambda$ ou nous filtrons pour les points dont la core_distance $\leq \lambda$.



Maintenant, nous devons trouver les différentes régions. Cela se fait en connectant les points "proches" les uns aux autres. "Proche" est déterminé par le niveau de densité de courant défini par λ et nous disons que deux points sont suffisamment proches si leur distance euclidienne est inférieure à λ.
Nous dessinons une sphère de rayon λ autour de chaque point.

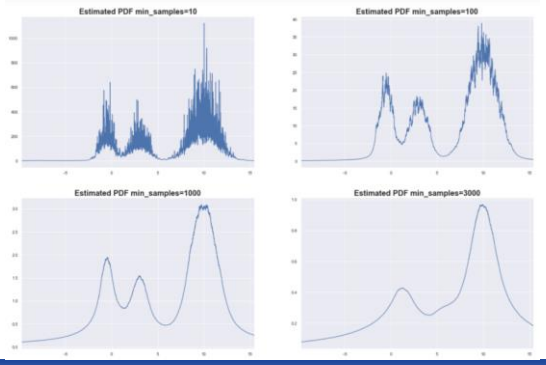


8. HDBSCAN

Hyperparamètres :

Alors que la classe HDBSCAN possède un grand nombre de paramètres qui peuvent être réglés lors de l'initialisation, en pratique, il y a un très petit nombre de paramètres qui ont un effet pratique significatif sur le clustering.

• **min_cluster_size** : la principale différence entre dbscan et hdbscan est que au lieu de compter les points à l'intérieur d'un rayon fixe eps pour définir les points de noyau, de frontière et de bruit, hdbscan le fait effectivement en utilisant un rayon extensible, de sorte que le seul hyperparamètre important est le d'importance est le min_cluster_size (la taille minimale d'un cluster). L'augmentation de la taille min_cluster_size réduit le nombre de clusters, en fusionnant certains ensemble. Ceci est le résultat de la réoptimisation de HDBSCAN*, dont le clustering plat offre une plus grande stabilité sous une notion légèrement différente de ce qui constitue un cluster. Valeurs : int(gamma * np.sqrt(n)) avec gamma in range (1, int(np.log(n))) et n= nb lignes dataframe.
• **min_samples** : Plus la valeur de min_samples est grande, plus le clustering est conservateur - plus de points seront considérés comme du bruit, et les clusters seront limités à des zones progressivement plus denses. valeurs : range(1, int(2 * np.log(n)) n= nb lignes dataframe.



8. HDBSCAN

• **cluster_selection_epsilon** : Dans certains cas, nous voulons choisir une petite taille de min_cluster_size car même les groupes de quelques points peuvent être intéressants pour nous. Cependant, si notre ensemble de données contient également des partitions avec de fortes concentrations d'objets, ce paramétrage peut donner lieu à un grand nombre de micro-clusters. La sélection d'une valeur pour cluster_selection_epsilon nous aide à fusionner les clusters dans ces régions. En d'autres termes, cela garantit que les clusters inférieurs au seuil donné ne sont pas divisés davantage. Le choix de cluster_selection_epsilon dépend des distances données entre vos points de données. Par exemple, définissez la valeur à 0,5 si vous ne voulez pas séparer les clusters qui sont séparés de moins de 0,5 unités.

8. HDBSCAN

L'ébauche de l'implémentation du HDBSCAN est la suivante :

1. Calculer les distances centrales par points : C'est essentiellement la façon dont nous "estimons la pdf sous-jacente".
2. Utiliser `mutual_reachability` **(a, b)** comme mesure de distance pour chaque a, b. La distance d'accessibilité mutuelle est un résumé du niveau de λ auquel deux points seront connectés. C'est ce que nous utilisons comme nouvelle métrique.
3. Construire un arbre d'envergure minimale. La construction de l'arbre de portée minimale équivaut à un regroupement à lien unique dans l'espace λ, ce qui revient à itérer à travers chaque ensemble de niveaux possibles et à garder la trace des regroupements.
4. Élaguer l'arbre. En bref, puisque nous n'avons qu'une estimation du PDF, nous nous attendons à avoir une certaine variance. Ainsi, même si la distribution sous-jacente est très lisse, la PDF estimée peut être très bosselée, et donc aboutir à un arbre hiérarchique très compliqué.
Nous utilisons le paramètre min_cluster_size pour lisser les courbes de la distribution estimée et, par conséquent, simplifier l'arbre dans le condensed_tree_.
5. Choisir les clusters en utilisant "l'excès de masse". En utilisant l'arbre condensé, nous pouvons estimer la persistance de chaque cluster et ensuite calculer pour le clustering optimal comme discuté dans la section précédente.

8. HDBSCAN

8. HDBSCAN

8. HDBSCAN