7

DBSCAN

7. DBSCAN

DBSCAN est un apprentissage non supervisé non paramétrique basé sur la densité proposé en 1996 par Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiaowei Xu.

Moins d'hypothèses, plus de flexibilité dans le modèle construit, une application plus large.

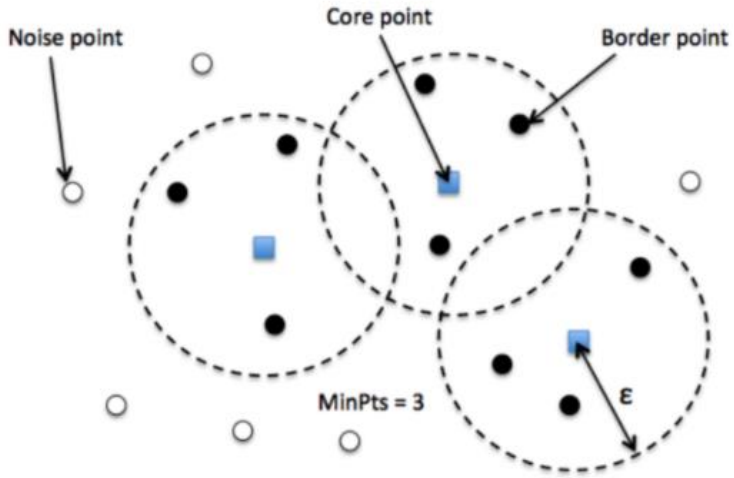
Semblable à K-Means, DBSCAN regroupe les données en fonction de leurs similarités sur la base de fonctions de distance et de densité.
La densité signifie ici le nombre de points dans la région définie par les paramètres du modèle.

L'algorithme DBSCAN utilise 2 paramètres : la distance et le nombre minimum de points MinPts devant se trouver dans un rayon epsilon, pour que ces points soient considérés comme un cluster.
Les paramètres d'entrées sont donc une estimation de la densité de points des clusters.
L'idée de base de l'algorithme est ensuite, pour un point donné, de récupérer son epsilon-voisinage et de vérifier qu'il contient bien MinPts points ou plus. Ce point est alors considéré comme faisant partie d'un cluster.
On parcourt ensuite l'epsilon -voisinage de proche en proche afin de trouver l'ensemble des points du cluster.

7. DBSCAN

L'algorithme fonctionne en calculant la distance entre chaque point et tous les autres points. Nous plaçons ensuite les points dans l'une des trois catégories suivantes.

- Point central : un point avec au moins **min_samples** de points dont la distance par rapport au point est inférieure au seuil défini par epsilon.
- Point frontière : un point qui n'est pas à proximité d'au moins **min_samples** de points mais qui est suffisamment proche d'un ou plusieurs points centraux. Les points limites sont inclus dans le cluster du point central le plus proche.
- Point de bruit (aberrant) : points qui ne sont pas assez proches des points centraux pour être considérés comme des points limites. Les points de bruit sont ignorés. C'est-à-dire qu'ils ne font partie d'aucun cluster.



7. DBSCAN

Le DBSCAN est un algorithme simple qui définit des clusters en utilisant l'**estimation de la densité locale**. On peut le diviser en 4 étapes :

- Pour chaque observation on regarde le nombre de points à au plus une distance **ε** de celle-ci. On appelle cette zone le **ε-voisinage de l'observation**.
- Si une observation compte au moins un certain nombre de voisins y compris elle-même, elle est considérée comme une **observation cœur**. On a alors décelé une **observation à haute densité**.
- Toutes les observations au voisinage d'une observation cœur appartiennent au même cluster. Il peut y avoir des observations cœur proche les unes des autres. **Par conséquent de proche en proche on obtient une longue séquence d'observations cœur qui constitue un unique cluster.**
- Toute observation qui n'est pas une observation cœur et qui ne comporte pas d'observation cœur dans son voisinage est considérée comme une **anomalie**.

Vous avez donc besoin de définir deux informations avant d'utiliser le DBSCAN :

> *Quelle distance **ε** pour déterminer pour chaque observation le **ε**-voisinage ? Quel est le **nombre minimal de voisins nécessaire** pour considérer qu'une observation est une **observation cœur** ?*

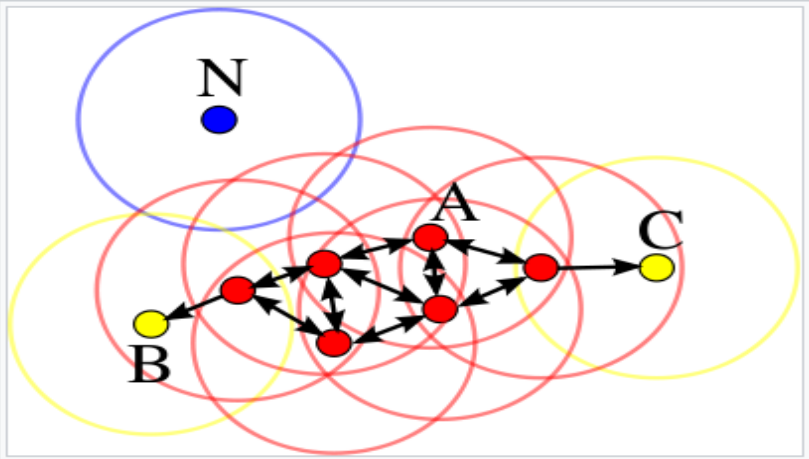
Ces deux informations sont renseignées librement par l'utilisateur. Contrairement à l'algorithme des k-moyennes ou la classification ascendante hiérarchique, il n'y a **pas besoin de définir en amont le nombre de clusters** ce qui rend l'algorithme moins rigide.

Un autre avantage de DBSCAN est qu'il permet aussi de **gérer les valeurs aberrantes ou anomalies**. Vous remarquerez dans la figure ci-dessus que l'algorithme a déterminé 3 clusters principaux : le bleu, le vert et le jaune. Les points colorés en violet constituent des anomalies détectées par le DBSCAN. Évidemment suivant la valeur de **ε** et le nombre de voisins minimal le partitionnement peut varier.

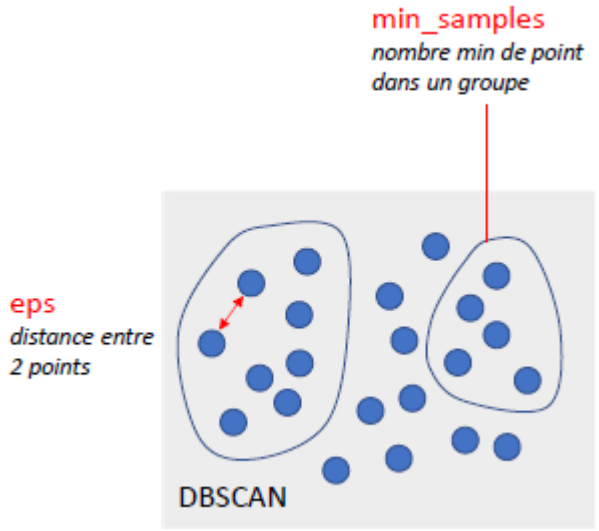
7. DBSCAN

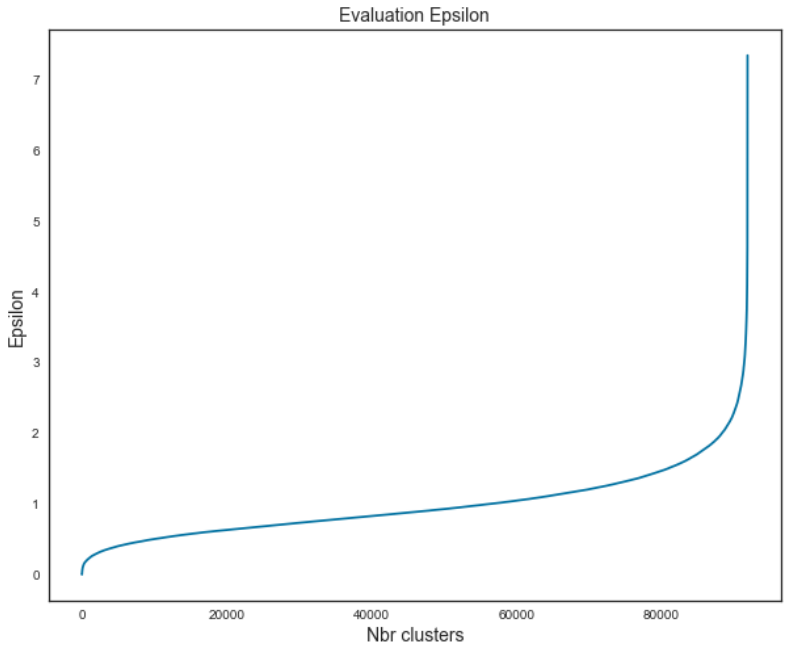
Hyperparamètres :

• **eps** : définit le voisinage d'un point, x, ou le rayon d'un cercle (ou d'une hypersphère dans l'espace N-D) avec x comme centre. Deux points sont considérés comme voisins si la distance entre les deux points est inférieure au seuil epsilon.
• **minPts** : nombre minimum de points dans le voisinage pour former une région dense.
• **min_samples** : le nombre minimum de voisins qu'un point donné doit avoir pour être classé comme un point central. Il est important de noter que le point lui-même est inclus dans le nombre minimum d'échantillons.
• **metric** : la métrique à utiliser lors du calcul de la distance entre les instances d'un tableau de caractéristiques (c'est-à-dire la distance euclidienne).



Les points A sont les points déjà dans le cluster. Les points B et C sont atteignables depuis A et appartiennent donc au même cluster. Le point N est une donnée aberrante puisque son epsilon voisinage ne contient pas MinPts points ou plus.





7. DBSCAN

Estimation des paramètres [modifier | modifier le code]

L'estimation des paramètres ϵ et $MinPts$ n'est pas un problème facile, car ces deux valeurs sont intrinsèquement liées à la topologie de l'espace à partitionner. Une trop faible valeur de ϵ et/ou une trop grande valeur de $MinPts$ peuvent empêcher l'algorithme de propager les clusters. A l'inverse, une trop grande valeur pour ϵ et/ou une trop faible valeur pour $MinPts$ peuvent conduire l'algorithme à ne renvoyer que du bruit. Une heuristique[3] permettant de déterminer conjointement ϵ et $MinPts$ pour un certain espace pourrait être donnée par :

- ϵ : calculer pour chaque point de l'espace la distance à son plus proche voisin. Prendre ϵ tel qu'une part « suffisamment grande » des points aient une distance à son plus proche voisin inférieure à ϵ;
- $MinPts$: calculer pour chaque point le nombre de ses voisins dans un rayon de taille ϵ (la taille de son ϵ-voisinage). Prendre $MinPts$ tel qu'une part « suffisamment grande » des points aient plus de $MinPts$ points dans leur ϵ-voisinage.

Par « suffisamment grand » on entend, par exemple, 95% ou 90% des points.

Une autre heuristique pour les cas en 2D (définie dans l'article originale de DBSCAN[1]) consiste à fixer la valeur de $MinPts$ à 4, et à tracer la courbe (triée dans l'ordre décroissant) des distances de chaque point à leur $4^{\text{ème}}$ plus proche voisin. On fixe alors ϵ à la valeur du "point seuil" repéré sur le graphe. Si ce seuil n'est pas clairement identifiable, l'utilisateur peut le fixer en estimant le pourcentage de bruit dans le jeu de données : ϵ est donc tel que seuls les termes de bruit ont une distance à leur $4^{\text{ème}}$ plus proche voisin plus grande que ϵ.

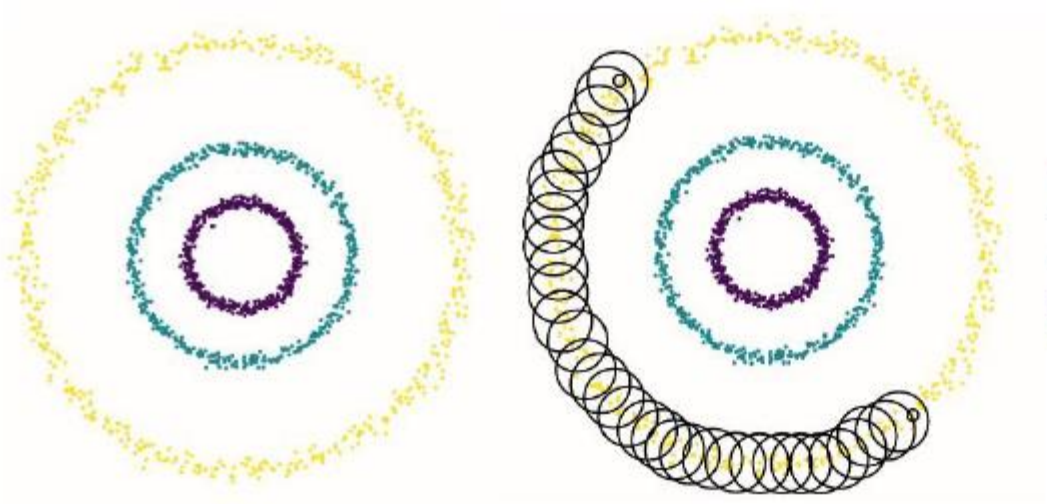
7. DBSCAN

Estimation des paramètres : pour DBSCAN, les paramètres ε et minPts sont nécessaires.

- **minPts** : En règle générale, un minimum minPts peut être dérivé du nombre de dimensions D dans l'ensemble de données, comme minPts ≥ D + 1. La faible valeur minPts = 1 n'a pas de sens, car alors chaque point à lui seul sera déjà un cluster. Avec minPts ≤ 2, le résultat sera le même que celui du clustering hiérarchique avec la métrique du lien unique, avec le dendrogramme coupé à la hauteur ε. Par conséquent, minPts doit être choisi au moins 3. Cependant, des valeurs plus grandes sont généralement meilleures pour les ensembles de données avec du bruit et donneront des clusters plus significatifs. En règle générale, minPts = 2-dim peut être utilisé, mais il peut être nécessaire de choisir des valeurs plus grandes pour les données très grandes, pour les données bruyantes ou pour les données qui contiennent de nombreux doublons.
- **ε :** La valeur de ε peut alors être choisie en utilisant un graphique de k-distance, en traçant la distance au voisin le plus proche k = minPts-1 ordonnée de la plus grande à la plus petite valeur. Les bonnes valeurs de ε sont celles où ce graphique montre un "coude" : si ε est choisi beaucoup trop petit, une grande partie des données ne seront pas regroupées ; alors que pour une valeur trop élevée de ε, les clusters fusionneront et la majorité des objets seront dans le même cluster. En général, les petites valeurs de ε sont préférables, et en règle générale, seule une petite fraction des points doit se trouver à cette distance les uns des autres.
- **Fonction de distance** : Le choix de la fonction de distance est étroitement lié au choix de ε, et a un impact majeur sur les résultats. En général, il faudra d'abord identifier une mesure raisonnable de la similarité pour l'ensemble des données, avant de pouvoir choisir le paramètre ε. Il n'y a pas d'estimation pour ce paramètre, mais les fonctions de distance doivent être choisies de manière appropriée pour l'ensemble des données.

7. DBSCAN

DBSCAN – Principe :



Epsilon Voisinage :
=> Pour chaque point d'un cluster, on doit pouvoir trouver n-voisins à une distance epsilon du point observé

Algorithme :

```
DBSCAN(D, eps, MinPts)
   C = 0
   pour chaque point P non visité
des données D
      marquer P comme visité
      PtsVoisins =
epsilonVoisinage(D, P, eps)
      si tailleDe(PtsVoisins) <
MinPts
         marquer P comme BRUIT
      sinon
         C++
         etendreCluster(D, P,
PtsVoisins, C, eps, MinPts)

etendreCluster(D, P, PtsVoisins, C,
eps, MinPts)
   ajouter P au cluster C
   pour chaque point P' de
PtsVoisins
      si P' n'a pas été visité
         marquer P' comme visité
         PtsVoisins' =
epsilonVoisinage(D, P', eps)
         si tailleDe(PtsVoisins') >=
MinPts
            PtsVoisins = PtsVoisins
U PtsVoisins'
      si P' n'est membre d'aucun
cluster
         ajouter P' au cluster C

epsilonVoisinage(D, P, eps)
   retourner tous les points de D
qui sont à une distance inférieure à
epsilon de P
```

7. DBSCAN

Avantages :

- L'algorithme est très simple et ne nécessite pas qu'on lui précise le nombre de clusters à trouver.
- Il est capable de gérer les données aberrantes en les éliminant du processus de partitionnement.
- Les clusters n'ont pas pour obligation d'être linéairement séparables (tout comme pour l'algorithme des k-moyennes par exemple).

Inconvénients :

- Cependant, il n'est pas capable de gérer des clusters de densités différentes.
- Difficile à paramétrer (eps et minPts fortement liés et mènent souvent à 1 seul cluster).
- difficile à utiliser en très grande dimension : souvenez-vous du fléau de la dimensionalité.
- Les boules de rayon epsilon et de grande dimension ont tendance à ne contenir aucun autre point.
- efficace en temps de calcul sans requérir de prédéfinir le nombre de clusters
- permet de trouver des clusters de forme arbitraire.
- Il ne fonctionne pas aussi bien que d'autres lorsque les clusters ont une densité variable (le réglage du seuil de distance ε et de minPoints pour l'identification des points de voisinage varie d'un cluster à l'autre lorsque la densité varie)
- Il n'est pas performant avec des données de très haute dimension car le seuil de distance ε devient difficile à estimer.

7. DBSCAN

7. DBSCAN

7. DBSCAN