WIKIPÉDIA

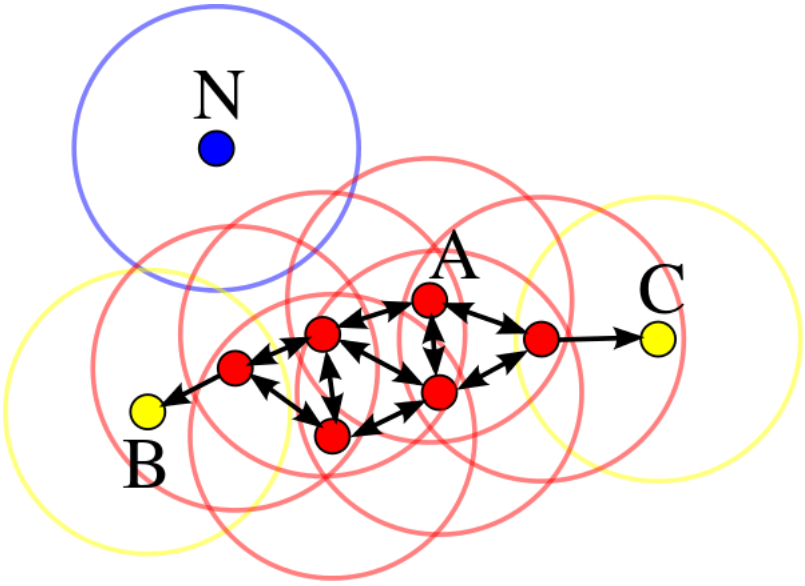
DBSCAN

DBSCAN (*density-based spatial clustering of applications with noise*) est un algorithme de partitionnement de données proposé en 1996 par Martin Ester, Hans-Peter Kriegel, Jörg Sander et Xiaowei Xu[1]. Il s'agit d'un algorithme fondé sur la densité dans la mesure qui s'appuie sur la densité estimée des clusters pour effectuer le partitionnement.

Sommaire

Principe général

L'algorithme DBSCAN utilise 2 paramètres : la distance ϵ et le nombre minimum de points $MinPts$ devant se trouver dans un rayon ϵ 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 ϵ-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'ϵ-voisinage de proche en proche afin de trouver l'ensemble des points du cluster.



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.

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
```

Estimation des paramètres

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[2] 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.

Avantages et inconvénients

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). Cependant, il n'est pas capable de gérer des clusters de densités différentes.

Articles connexes

- OPTICS

Références

1. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining, 1996, pp. 226–231.
2. alitouka, *spark_dbscan: DBSCAN clustering algorithm on top of Apache Spark*, 18 août 2017 (lire en ligne (https://github.com/alitouka/spark_dbscan))

La dernière modification de cette page a été faite le 18 avril 2018 à 20:54.