DBSCAN

date: 20261105
status: #finished
tags: #machine_learning #clustering #dbscan

---

0. Setup

First of all, DBSCAN stands for density-based spatial clustering of applications with noise. It was introduced by Ester et al..

Let $\mathcal{D}_{\text{data}} = \{x_1, x_2, ..., x_n\}$ where $x_i \in \mathbb{R}^d$ for all i. We want to find the label $c_i \in \{0\} \cup \mathcal{K}$ for each x_i, where $\mathcal{K} = \{1, 2, ..., K\}$. The label $c_i = 0$ means noise once the algorithm ends. Denote $\mathcal{C} = \{c_1, c_2, ..., c_n\}$ as the set of cluster labels of the data points.

---

1. The Algorithm

(i) Construct the ε-neighborhoods

Construct the $epsilon$-neighborhood $N_\varepsilon(x_i)$ for each data point x_i:

$$N_\varepsilon(x_i) = \{j : \|x_i - x_j\| \leq \varepsilon\}.$$

(ii) Identify the core points

Core points are those whose ε-neighborhood size reaches the threshold $m \in \mathbb{N}$. We create an indicator variable z_i for each i:

$$z_i = \begin{cases} 1 & \text{if } |N_\varepsilon(x_i)| \geq m \\ 0 & \text{otherwise} \end{cases}.$$

(iii) Build the clusters

Build the clusters from the core points and their ε-neighborhoods.

- Initialize every label in $\mathcal{C}$ to 0, and set $K = 0$.

- Loop through the data points, i from 1 to n. If $z_i = 1$ and $c_i = 0$: {

1. Open a new cluster:

$$K \leftarrow K + 1, \quad c_i = K, \quad Q \leftarrow N_\varepsilon(x_i).$$

2. Repeat until Q is empty: {

Remove j from Q. If $c_j = 0$, let $c_j = K$, and if also $z_j = 1$, grow Q by $Q \leftarrow Q \cup N_\varepsilon(x_j)$.

}

}

> **Note**
>
> Since the outer loop only opens clusters at core points, some non-core points, especially those before the first core point, might seem to miss the opportunity to be labeled. There is no need to worry, because those points must be in $N_\varepsilon(x_i)$ for some core point x_i if they are not noise, and the inner loop adds every core point's neighborhood to Q.

> **Remark**
>
> Beyond the cluster labels, the points in $\mathcal{D}_{\text{data}}$ are classified into three groups: core points, border points, and noise. A cluster consists of core points and border points, which are in the neighborhoods of those core points.

---

2. Discussion

(i) Comparisons with K-means

Recall that in K-means, the clusters are characterized by centroids, while in DBSCAN, clusters are characterized by core points. Basically, we construct the neighborhood for each data point, and a neighborhood is just the set of points close enough to the underlying data point, including itself. Then, core points are those whose neighborhoods are at least a threshold in size.

What's more, in K-means, a point is in a cluster if the corresponding centroid is the closest one. In DBSCAN, a point is in a cluster if either it's a core point in that cluster or it's in the neighborhood of a core point associated with that cluster. A border point in the neighborhoods of two clusters takes the label of the first cluster that reaches it, so its label depends on the visit order.

There is only one hyperparameter in K-means, which is the number of clusters. In DBSCAN, however, the number of clusters is no longer a hyperparameter but the result of the algorithm. DBSCAN's hyperparameters include ε and m, which are used for the definitions of neighborhoods and core points, respectively.

(ii) Time Complexity

With a direct scan, constructing the $epsilon$-neighborhoods needs quadratic time. Building the clusters costs one check per neighbor of each core point, which is quadratic at worst and linear when neighborhoods stay small. On the other hand, K-means needs roughly linear time in n for fixed K, d, and iteration count.

(iii) How to choose hyperparameters?

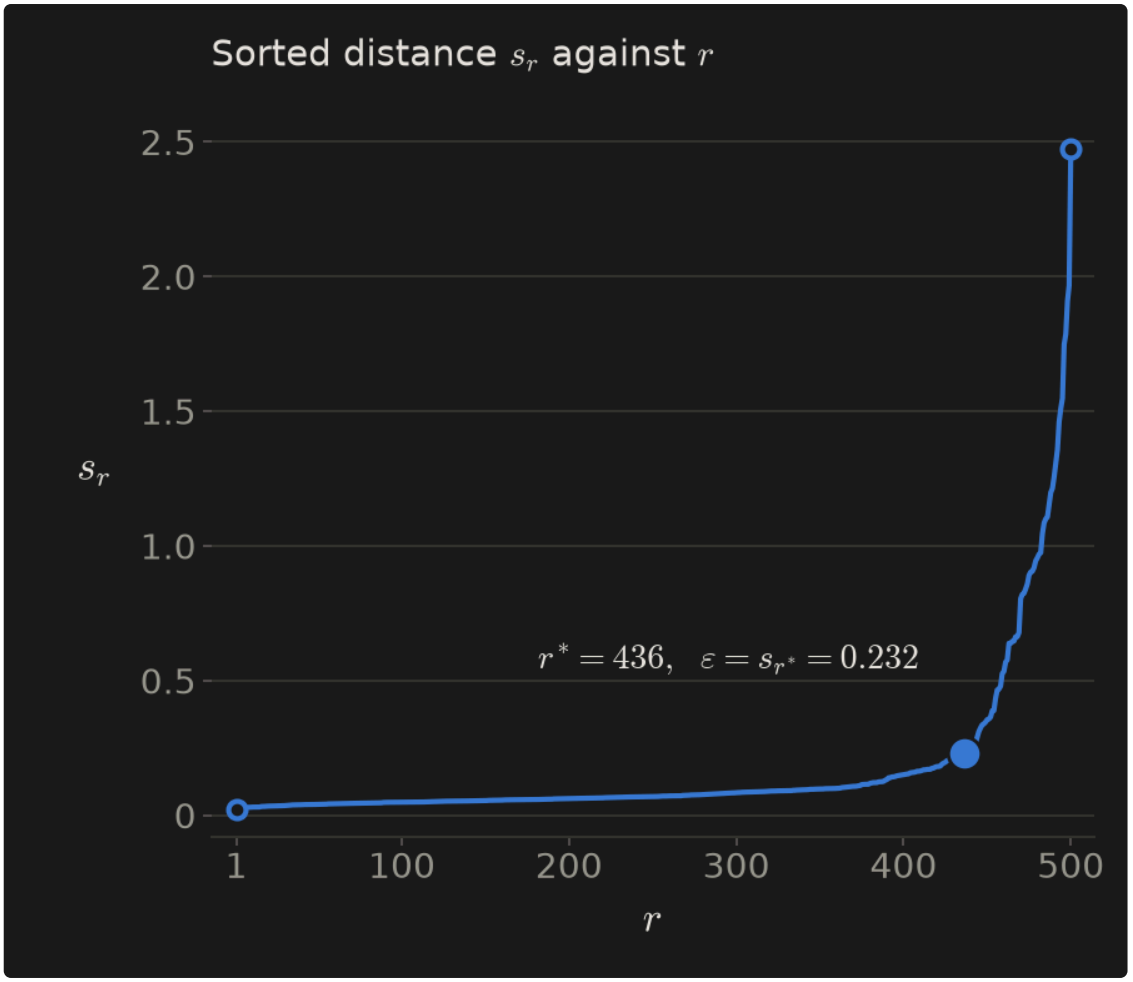
Schubert et al. state that m typically has little effect and can reasonably be set to $2d$, where d is the dimension of the data. For spatial clustering, $d = 2$, so $m = 4$. Given m, ε can be tuned similarly using the elbow method. To do so, we first compute the m-th smallest neighboring distance for each i:

$$\delta_m(x_i) = \text{the m-th smallest of } \|x_i - x_j\| \text{ over } j = 1, \ldots, n.$$

Note that $\delta_m(x_i)$ is the smallest ε that makes x_i a core point, because $|N_\varepsilon(x_i)| \geq m$ holds exactly when at least m distances are at most ε, which means $\varepsilon \geq \delta_m(x_i)$. We collect $\delta_m(x_i)$ of all i and sort them in ascending order, writing s_r for the r-th smallest:

$$s_1 \leq s_2 \leq \ldots \leq s_n,$$

and then plot s_r against r, which looks like the following for a simulated sample of three Gaussian clusters plus uniform noise, where $n = 500$, $d = 2$, and $m = 4$.

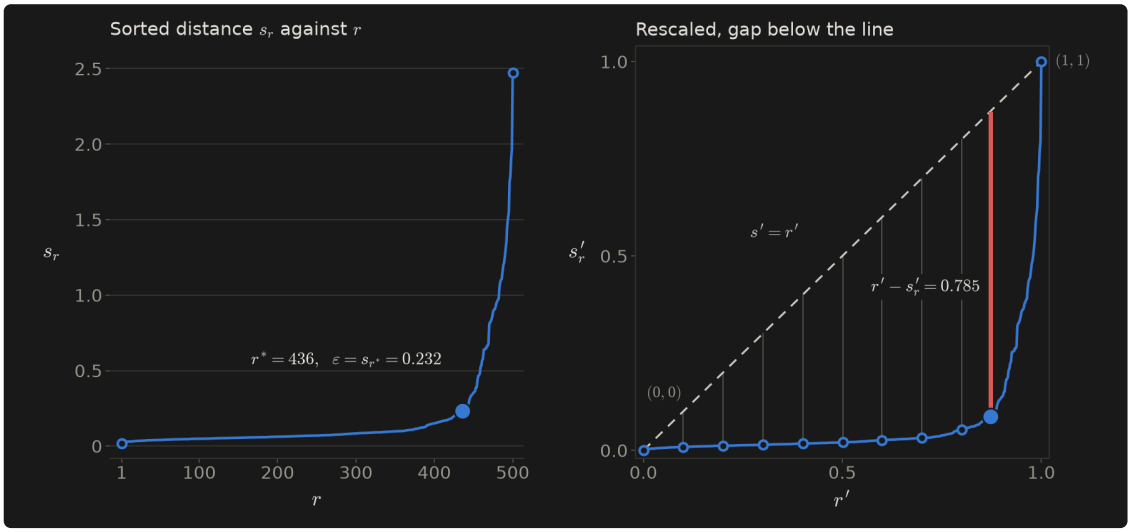


We then normalize both axes:

$$r' = \frac{r-1}{n-1}, \quad s'_r = \frac{s_r - s_1}{s_n - s_1}.$$

Finally,

$$\varepsilon = s_{r^*} \text{ where } r^* = \arg\max_{1 \le r \le n} r' - s'_r.$$



An elbow appears when dense clusters sit among sparse noise. Cluster points become core at a small ε, while noise needs a very large ε, so the sorted curve stays low and then rises sharply. No elbow is guaranteed.

This rule is a heuristic and returns a value even without an elbow. Following Schubert et al., check the result: noise is usually between 1% and 30%, and a largest cluster above 20% to 50% of the clustered points suggests a smaller ε.