DBSCAN Clustering Tutorial

September 17, 2025

1 Introduction

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) discovers clusters of arbitrary shapes while separating noisy points. By relying on two parameters—the neighborhood radius ε and the minimum number of points `minPts`—it groups dense regions without predefining the number of clusters. DBSCAN is robust to outliers and especially useful when clusters exhibit varying shapes, though it can struggle when densities differ dramatically between groups.

2 Theory and Formulas

2.1 Neighborhoods and Core Points

For any point p, the ε-neighborhood is defined as

$$\mathcal{N}_\varepsilon(p) = \{q \mid \|p - q\|_2 \leq \varepsilon\}. \tag{1}$$

A point is a *core point* if $|\mathcal{N}_\varepsilon(p)| \geq$
$textttminPts$. A point within a core point's neighborhood but not itself core is termed a *border point*. Points that are neither core nor border are labelled as noise.

2.2 Density Reachability and Connectivity

Point q is directly density reachable from p if p is core and $q \in \mathcal{N}_\varepsilon(p)$. Density reachability is transitive across chains of core points, enabling DBSCAN to form clusters as maximal sets of mutually density-connected points:

$$C = \{p \mid \exists p_0 \in C_0, p \text{ density reachable from } p_0\}, \tag{2}$$

where C_0 contains at least one core point. The algorithm iteratively expands clusters by exploring neighborhoods of core points.

2.3 Algorithm Outline

1. Standardize features where necessary so that distances are meaningful.
2. For each unvisited point, query $\mathcal{N}_\varepsilon(p)$. If p is not core, mark it as noise temporarily.

3. If p is core, create a new cluster and add all points in its neighborhood. Recursively expand by checking neighbors that qualify as core points.

4. Continue until every point has been assigned to a cluster or marked as noise.

The distance metric influences the notion of density; Euclidean distance is common for numeric features, while cosine or Manhattan distances may be better for sparse or high-dimensional data.

3 Applications and Tips

- **Spatial analytics**: DBSCAN detects hot spots such as GPS trajectory stops or crime regions without having to guess cluster counts.
- **Anomaly detection**: Points labelled noise provide a principled way to flag rare events or device faults.
- **Parameter tuning**: Use a k-distance plot (typically with $k = \texttt{minPts} - 1$) to select ε; the elbow indicates a stable choice. Start with $\texttt{minPts} = 2d$ for d features, and adjust based on noise levels.
- **Preprocessing**: Scale features, remove duplicates, and consider dimensionality reduction when dealing with high-dimensional data so that distance queries remain reliable and efficient.

4 Python Practice

The accompanying script `gen_clustering_dbscan_figures.py` generates synthetic two-dimensional data with three dense regions and peripheral noise. It fits DBSCAN for a range of ε values, stores an illustrative clustering, and produces an ordered k-distance curve for diagnosing parameter choices.

Listing 1: Excerpt from $\text{gen}_c lustering_d bscan_f igures.py$

```python
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors

# Fit DBSCAN with Euclidean distance
dbscan = DBSCAN(eps=0.35, min_samples=5, metric="euclidean")
dbscan.fit(points)
labels = dbscan.labels_

# Compute ordered distances to the k-th nearest neighbor
neighbors = NearestNeighbors(n_neighbors=5)
neighbors.fit(points)
dists, _ = neighbors.kneighbors(points)
k_distance = np.sort(dists[:, -1])
```

5 Result

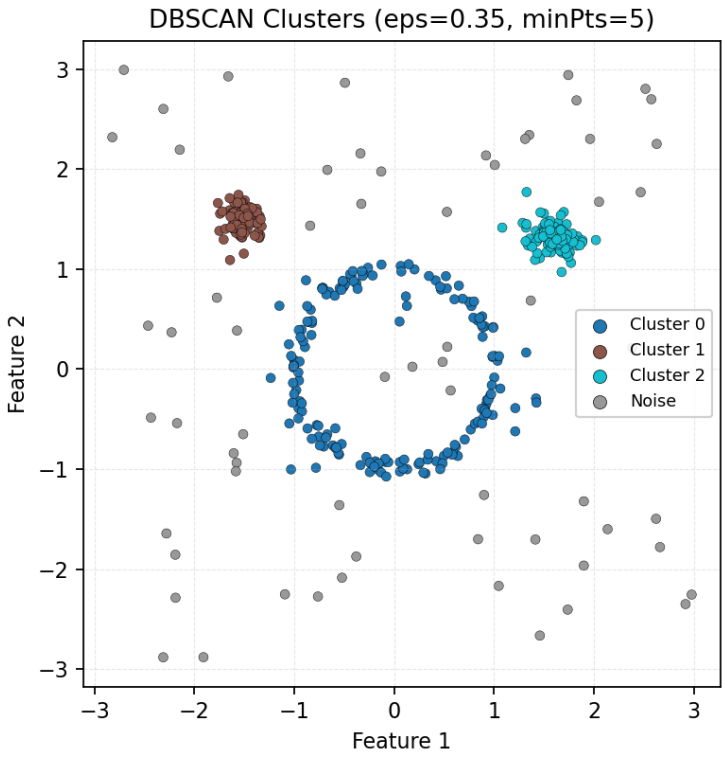

Figure 1: DBSCAN clusters on synthetic data with noise ($\varepsilon = 0.35, \texttt{minPts} = 5$)

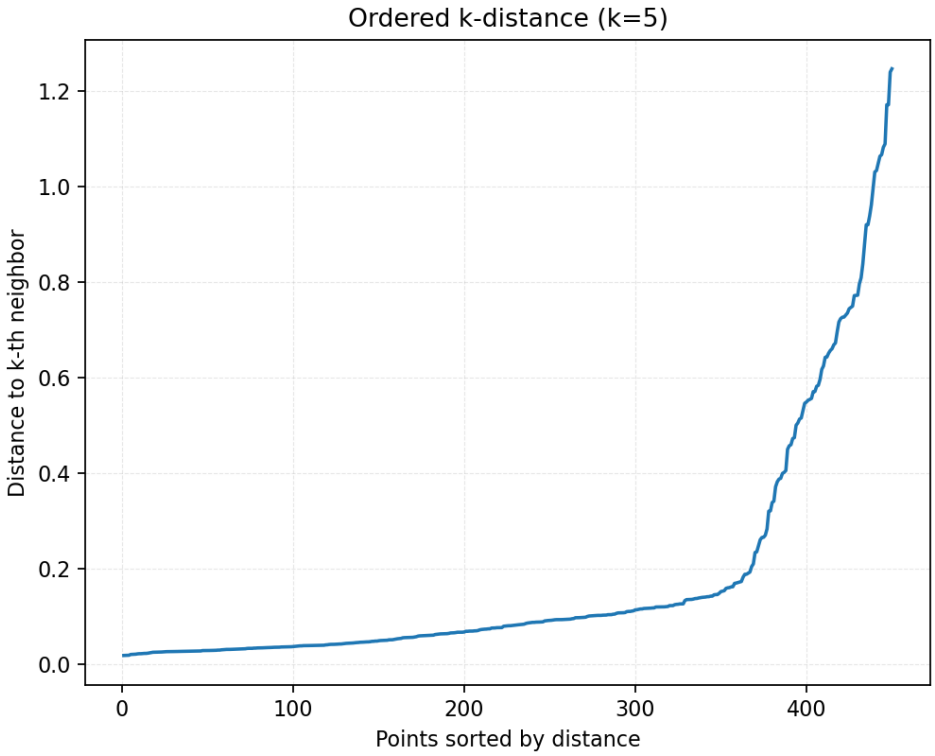

Figure 2: Ordered k-distance plot highlighting the elbow near $\varepsilon = 0.35$

6 Summary

DBSCAN excels at uncovering arbitrarily shaped clusters without specifying their count while labelling sparse regions as noise. Effective use hinges on meaningful distance metrics, thoughtful scaling, and data-driven selection of ε and `minPts`. The synthetic example demonstrates how clustering output and k-distance diagnostics work together to justify parameter choices.