# *Density-Based Clustering*

*2026-05-06 · cheerful mango Haubentaucher*

It is all about that space, about that space.

## *The Why*

The clustering algorithms we have seen so far share a common mechanism of assigning points to clusters based on their distance to an other point; a centroid or a linkage point. Every point, no matter how isolated, is absorbed into some cluster latest at the final merge. Real data contains noise: GPS measurements with dropouts, sensor readings during interference, survey responses that are genuine outliers.

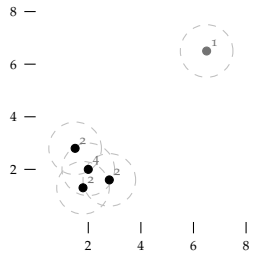



Figure 1: Each point carries its own dashed $\varepsilon$-neighbourhood; the small grey digit next to a point gives the number of points inside that circle, itself included. In the dense cluster at the lower left the four neighbourhoods overlap and chain the points together; the isolated $p_n$ at the upper right has no companions within $\varepsilon$. Density, not distance to a centre, determines membership.

No centroid required. A density-based cluster is a space, not a mean and not a link. Its shape is implicit in the data, seeded by the points themselves; nothing forces it to be convex or spherical; detects noise, outliers and sparse bridges naturally, and avoids committing to $K$ in advance. This geometric-assumption-free view is both the strength of density-based methods and, as we will see, the source of their own limitations.

In order to move from pairwise distances to density-aware neighbourhoods we have good reason to find ways to,

- define what it means for a region of space to be dense or sparse.
- grow clusters of arbitrary shape, including non-convex shapes, from local density, without assuming convex or spherical structure.
- separate signal from noise, so that sparse outliers are labelled as such rather than forced into a cluster.
- let the data itself decide how many clusters exist, without committing to $K$ in advance.

*Hands On Experience*

CONSIDER the same six canonical points, now with a seventh point $x_7$=(4, 5) placed between the two natural groups.

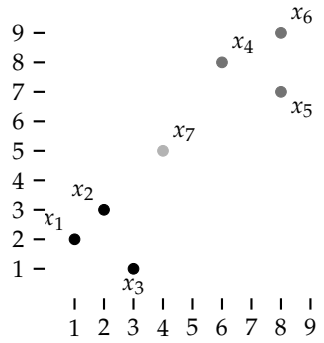



Figure 2: Six canonical points plus $x_7$=(4, 5) sitting between the two natural groups. Which points lie in dense neighbourhoods, and which sits in a sparse region?

DRAW A CIRCLE of radius $\varepsilon$=2.5 around each point and count who sits inside. The three bottom-left points find each other, the three top-right points find each other, and $x_7$ finds empty space; read off the data without ever choosing a number of clusters.

THE NEIGHBOURHOOD RADIUS $\varepsilon$ is the distance within which a point looks for companions. Too small and every point is left alone; too large and everything collapses into one cluster. The right value lives where the data transitions from dense to sparse, and we will read it off the data itself using the $k$-distance plot.

OUR APPROACHES SO FAR WOULD REFUSE to let $x_7$ be lonely. With $K$=2 it hands the point to whichever centroid happens to sit slightly closer, an assignment driven by initialisation rather than by any structure in the data. Hierarchical clustering would fold $x_7$ into some merge and bury it inside a cluster it never really joined.

DENSITY ALLOWS US TO DO SO. A point whose $\varepsilon$-neighbourhood is empty is not handed to the nearest cluster; it is labelled $p_n$. Cluster shape is read off local neighbourhoods with a sense for $p_b$'s, the number of clusters emerges from where density breaks instead of from a chosen $K$, and sparse points are flagged instead of absorbed.

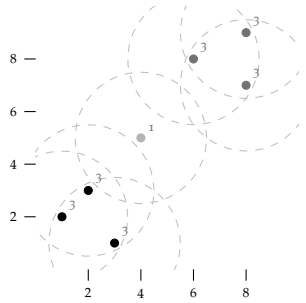



Figure 3: The seven canonical points with their dashed $\varepsilon$=2.5 neighbourhoods. The small grey digit next to each point gives $|N_\varepsilon|$, itself included. Each natural group forms a three-point neighbourhood; $x_7$ sits in a sparse region and finds only itself within $\varepsilon$.

THE LEARNING OBJECTIVES of this lecture:

- Trace a density clustering by hand on a small dataset and learn how to set the neighbourhood radius $\varepsilon$, and the minimum number of points $n_{\min}$ from the data itself.
- Understand the concepts of core ($p_c$), border ($p_b$), and noise ($p_n$) points by formalising neighbourhoods.
- Recognise when density-based clustering succeeds where K-Means and hierarchical fail, where they share characteristics, and where it fails in turn under varying density, and in high dimensionality.

*The Density-Based Algorithm*

DB SCAN[1] or Density-Based Spatial Clustering of Applications with Noise, builds clusters from local density rather than from a chosen number of centroids or a sequence of merges. Two parameters govern the algorithm: $\varepsilon$, the neighbourhood radius, and $n_{\min}$, the minimum number of points required to form a dense region.

[1]

GIVEN DATA $X = \{x_1, \ldots, x_n\}$ and parameters $(\varepsilon, n_{\min})$, DB Scan returns a labelling

$$\tilde{y} : X \longrightarrow \{1, 2, \ldots, K, p_n\}$$

where $K$ is determined by the data rather than fixed in advance. The output is a flat partition plus a $p_n$ set $N$.

EVERY POINT IS CLASSIFIED into one of three roles before any cluster is formed. The $\varepsilon$-neighbourhood of a point $p$ is the set of all points within distance $\varepsilon$:

$$N_\varepsilon(p) = \{q \in X : d(p, q) \leq \varepsilon\}.$$

- A point $p_c$ is a core point if $|N_\varepsilon(p_c)| \geq n_{\min}$.
- A point $p_b$ is a border point if $|N_\varepsilon(p_b)| < n_{\min}$ but $p_b \in N_\varepsilon(p_c)$ for some core point $p_c$.
- A point $p_n$ is noise if it is neither core nor border.

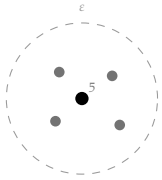



Figure 4:

CORE POINT. The black point has $|N_\varepsilon|$=5 $\geq n_{\min}$=4. Four companions sit inside its dashed $\varepsilon$-neighbourhood. The small grey digit at the upper right gives $|N_\varepsilon|$, the point itself included.

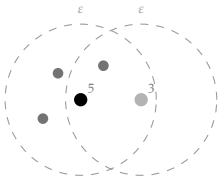



Figure 5:

BORDER POINT $p_b$. The light grey point on the right has $|N_\varepsilon|$=3 < $n_{\min}$=4. It is not a $p_c$ itself, but it falls inside the $\varepsilon$-neighbourhood of the black $p_c$ on the left and is absorbed into that cluster.

DENSITY REACHABILITY is not symmetric. A $p_b$ is reachable from a $p_c$ whenever $p_b \in N_\varepsilon(p_c)$, but $p_c$ is not reachable from $p_b$ because $p_b$ is not a $p_c$. Cluster membership propagates from $p_c$'s outward, never in reverse.

DENSITY CONNECTIVITY extends reachability to full cluster membership. Point $q$ is density-reachable from $p$ if there exists a chain of points $p{=}p_1, p_2, \ldots, p_m{=}q$ such that

$$p_{i+1} \in N_\varepsilon(p_i) \quad \text{and} \quad |N_\varepsilon(p_i)| \geq n_{\min} \quad \text{for } i = 1, \ldots, m-1.$$

Two points $p$, $q$ are density-connected if there exists a third point $o_c$ from which both are density-reachable; a cluster is then defined as a maximal set of pairwise density-connected points.

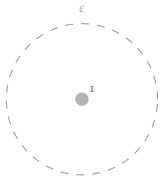



Figure 6:

NOISE POINT $p_n$. With $|N_\varepsilon|$=1 and no $p_c$ within $\varepsilon$, the point belongs to no cluster. The dashed circle is otherwise empty.

TRACE THROUGH OUR SEVEN POINTS. Dashed circles are $\varepsilon$=2.5 neighbourhoods; dark points belong to $C_1$, grey to $C_2$, and an open circle marks $p_n$.

INITIALIZATION is trivial as the set of clusters and their core-memberships is the same regardless of the order in which points are processed. Any unvisited $p_c$ triggers a new cluster; absorbing surrounding $p_b$'s immediately; and growing the cluster by absorbing other $p_c$'s and then their $p_b$'s. Only $p_b$'s that happen to lie in the $\varepsilon$-neighbourhood of several $p_c$'s can be assigned to different clusters depending on visit order, but keeping their point class $p_c$, $p_b$, or $p_n$.

Algorithm 1: Naive Density-Based Clustering (DBSCAN)

**Require:** Data $X$, neighbourhood radius $\varepsilon$, density threshold $n_{\min}$
**Ensure:** Cluster label for each point (or $p_n$)

```
Mark all points as unvisited; C ← 0
for each unvisited point p_i ∈ X do
    Mark p_i as visited
    Compute N ← N_ε(p_i)
    if |N| < n_min then
        Label p_i as p_n
    else
        Label p_i as p_c; C ← C + 1; assign p_i to cluster C
        for each p_j ∈ N \ {p_i} do
            if p_j is unvisited then
                Mark p_j as visited
                Compute N' ← N_ε(p_j)
                if |N'| ≥ n_min then label p_j as p_c; N ← N ∪ N'
                end if
            end if
            if p_j has no cluster then assign p_j to C (as p_c or p_b)
            end if
        end for
    end if
end for
```

TERMINATION AND DETERMINISM. The outer loop visits every point at most once and the inner loop only grows by adding previously unvisited points, so the algorithm halts after $O(n)$ neighbourhood queries. The resulting cluster set is determined uniquely by the data and the pair $(\varepsilon, n_{\min})$: $p_c$'s are fixed by the $|N_\varepsilon|$ threshold.

COMPLEXITY. Each of the $n$ points triggers one $\varepsilon$-neighbourhood query costing $O(n)$, giving $O(n^2)$ overall. A spatial index (k-d or ball tree) reduces each query to $O(\log n)$, yielding $O(n \log n)$ in low dimensions.

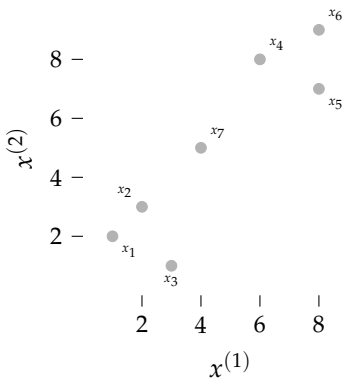



Figure 7:

INITIALIZATION: all $n$=7 points unvisited. Parameters $\varepsilon$=2.5, $n_{\min}$=2 are fixed. No seed necessary. Randomly picking $x_1$ as the first point to process.

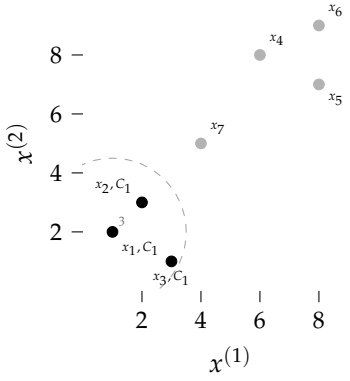



Figure 8:

VISIT $x_1$: $N_{2.5}(x_1)=\{x_1, x_2, x_3\}$, $|N|=3 \geq 2$, so $x_1$ is a $p_c$.

Seed cluster $C_1$; assign $x_1$ to $C_1$ and add $x_2$ and $x_3$ to the expansion queue. Points outside the dashed circle remain unvisited.

Distances: $d(x_1, x_2)=\sqrt{2}\approx 1.41$ ✓, $d(x_1, x_3)=\sqrt{5}\approx 2.24$ ✓.

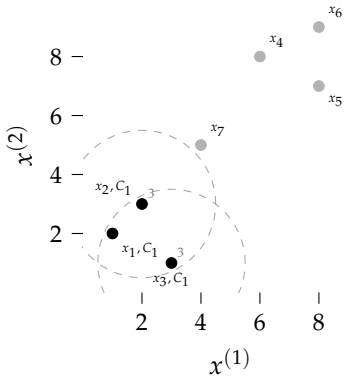



Figure 9:

EXPAND $C_1$: dequeue $x_2$: $N_{2.5}(x_2)=\{x_1, x_2, x_3\}$, all already assigned to $C_1$, no new points added.

Dequeue $x_3$: $N_{2.5}(x_3)=\{x_1, x_2, x_3\}$, same result.

Expansion queue empty: $C_1=\{x_1, x_2, x_3\}$ is sealed.

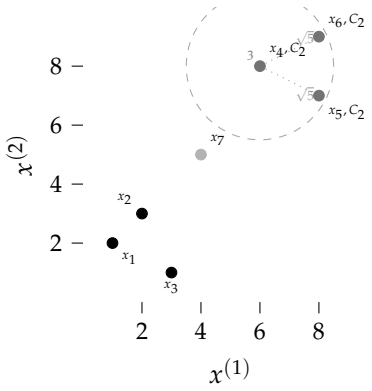



Figure 10:

VISIT $x_4$: $N_{2.5}(x_4)=\{x_4, x_5, x_6\}$, $|N|=3 \geq 2$, so $x_4$ is a $p_c$.

Seed $C_2$; expand from $x_5$ and $x_6$: both are $p_c$'s and their neighbourhoods cover only $\{x_4, x_5, x_6\}$.

Distances: $d(x_4, x_5)=\sqrt{5}\approx 2.24$ ✓, $d(x_4, x_6)=\sqrt{5}\approx 2.24$ ✓, $d(x_4, x_7)=\sqrt{13}\approx 3.61$ ×.

$C_2=\{x_4, x_5, x_6\}$ sealed. Only $x_7$ remains unvisited.

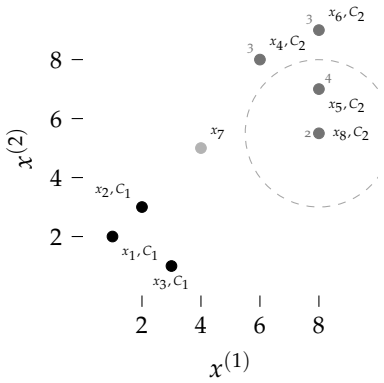



Figure 11:

HYPOTHETICAL POINT $x_8=(8, 5.5)$: $d(x_5, x_8)=1.5 \leq \varepsilon$, so $x_5$ absorbs $x_8$ into its neighbourhood.

$N_{2.5}(x_8)=\{x_8, x_5\}$, so $|N|=2 < n_{\min}=3$; $x_8$ is a border point. Because $x_5$ is already a core point of $C_2$, $x_8$ is classified as $C_2$.

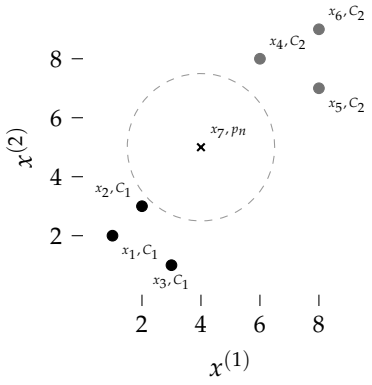



Figure 12:

VISIT $x_7$: its two nearest neighbours are $x_2$ at $d\approx 2.83$ and $x_4$ at $d\approx 3.61$, both beyond $\varepsilon=2.5$. So $N_{2.5}(x_7)=\{x_7\}$, $|N|=1 < 2$, and $x_7$ is labelled $p_n$.

All points visited. Final result: $C_1=\{x_1, x_2, x_3\}$, $C_2=\{x_4, x_5, x_6\}$, and $x_7$ left as noise. No $K$ was chosen; the noise point was never forced into a cluster.

*Hyperparameter Selection*

CHOOSING $n_{\min}$ follows a practical rule of thumb: set $n_{\min} > d+1$ where $d$ is the dimension of the feature space. For two-dimensional data, $n_{\min}=3$ or 4 is typical; higher dimensions require larger values to keep sparse neighbourhoods from being treated as dense.

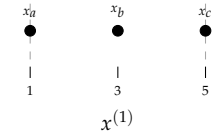



Figure 13: How many points do you need at least to fence in a point in 2D? 3D?

CHOOSING $\varepsilon$ uses the $k$-distance plot: for each point, compute the distance $d_k(x_i)$ to its $k$-th nearest neighbour at $k=n_{\min}-1$. The $(n_{\min}-1)$-th nearest neighbour distance is the tightest radius that still qualifies.

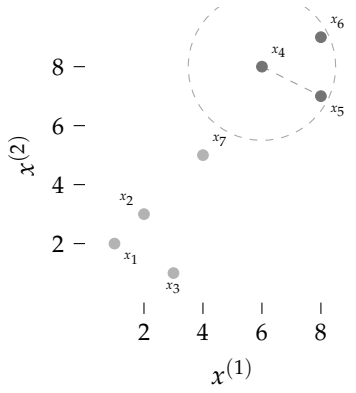



Figure 14:

$k$-TH NEAREST NEIGHBOUR OF $x_4$: $n_{\min}=2$, $k=n_{\min}-1=1$. The dashed circle shows the $\varepsilon=2.5$ neighbourhood; the nearest neighbour of $x_4$ is $x_5$ at distance $d_k(x_4)=\sqrt{5}\approx 2.24$, well inside $\varepsilon$.

and sort these distances in ascending order,

$$d_k^{(1)} \leq d_k^{(2)} \leq \ldots \leq d_k^{(n)}.$$

The curve rises slowly in dense regions and sharply in sparse ones; the elbow marks the transition and is the recommended $\varepsilon$. Notice, that the sorting does not reflect cluster membership: points from different clusters can be interleaved in the $k$-distance plot.

SENSITIVITY SWEEP. Try a few $n_{\min}$ values ($d+1$, $2d$, etc.) and compare the resulting $k$-distance plots. The value that gives the sharpest, most unambiguous elbow is usually the right one.

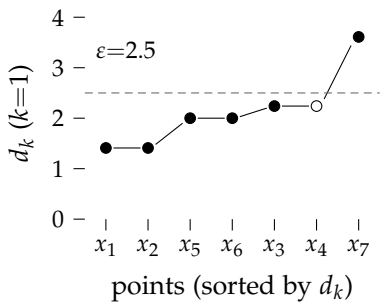
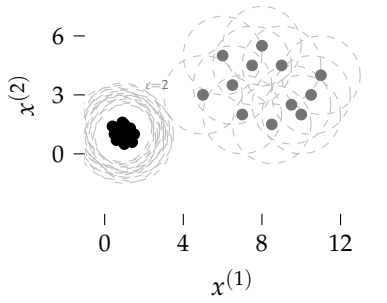


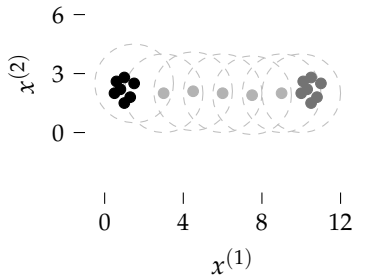

Figure 15: $k$-distance plot for the seven canonical points with $k{=}n_{\min}{-}1{=}1$. Each $d_k^{(i)}$ is the distance from one point to its nearest neighbour, sorted ascending. The first six values sit at $\sqrt{2}{\approx}1.41$, 2.00, or $\sqrt{5}{\approx}2.24$; $d_k^{(7)}$ jumps to $\sqrt{13}{\approx}3.61$ (the lone $x_7$). The unfilled dot marks the elbow, and any $\varepsilon$ in the gap, such as $\varepsilon{=}2.5$ (dashed), separates the dense six from the sparse outlier.

*When DB Scan Fails*

DB SCAN HAS ITS OWN PATHOLOGIES. Every density-based method sets a single threshold. The most common arise from the single global $\varepsilon$, which filters or bridges. Then we have the curse of dimensionality, which makes all points look far apart and destroys the notion of density.




VARYING DENSITY
Figure 16: Varying cluster density. An $\varepsilon$ tight enough to resolve the left cluster leaves the right cluster disconnected; an $\varepsilon$ large enough to connect the right cluster merges everything on the left into one blob, loosing expressiveness of the clustering. No single global threshold works for both.




BRIDGING AND CHAINING
Figure 17: Bridging: a thin chain of $p_c$-eligible points (light) spans the gap between two well-separated clusters. The overlapping $\varepsilon$-circles show how density reachability chains through $p_c$'s, fusing both into one cluster. Increasing $n_{\min}$ or decreasing $\varepsilon$ breaks the bridge at the cost of shrinking the clusters themselves.

OPTICS or Ordering Points To Identify the Clustering Structure [2] introduces two per-point distances. The core distance $\varepsilon_c$ is the per-

[2]

point analogue of the global $\varepsilon$, the smallest radius that captures $n_{\min}$ neighbours:

$$\varepsilon_c(x_i) = \min\{r : |N_r(x_i)| \geq n_{\min}\}.$$

The reachability distance $r(x_i, x_j)$ is defined for every pair where $x_i$ is an already-visited $p_c$ and $x_j$ is an unvisited point. It measures the cost for $x_i$ to absorb $x_j$, and is at least $\varepsilon_c(x_i)$, even if $x_j$ sits closer:

$$r(x_i, x_j) = \max\{\varepsilon_c(x_i),\ d(x_i, x_j)\}.$$

Dense regions have small $\varepsilon_c$, sparse regions large ones. The max ensures that a point is never reached at a distance shorter than its $\varepsilon_c$, so reachability adapts to local density automatically. [3]

HDBSCAN or Hierarchical DBSCAN [3] replaces the Euclidean distance with a mutual reachability distance:

$$d_{\text{mreach}}(x_i, x_j) = \max\{\varepsilon_c(x_i),\ \varepsilon_c(x_j),\ d(x_i, x_j)\}.$$

This spreads apart points in sparse regions; exactly those that should not be merged early. From here HDBSCAN builds a full density hierarchy by applying single-linkage clustering to the mutual reachability distance, and potentially condenses it into a flat clustering by selecting the most persistent clusters across all density levels.

THE ONLY PARAMETER the user must set is $n_{\min}$, and the result is often robust to its choice.

DENSITY HIERARCHIES AND DENDROGRAMS. The density language and the hierarchical-clustering language describe the same structure from different angles.

THE CURSE OF DIMENSIONALITY: All distance-based methods share a deeper vulnerability as the number of features $d$ grows, the volume of the space increases so fast that a fixed number of points $N$ becomes sparse everywhere.

CONSIDER $N$ POINTS drawn uniformly in the unit hypercube $[0,1]^d$. The mean nearest-neighbour distance scales as

$$d_{\text{NN}} \ \propto\ N^{-1/d}.$$

Fix $N$=10 and watch the spacing grow with $d$:

$$\begin{aligned} d = 1: &\quad 10^{-1/1} = 0.10 \\ d = 2: &\quad 10^{-1/2} \approx 0.32 \\ d = 3: &\quad 10^{-1/3} \approx 0.46 \\ d = 10: &\quad 10^{-1/10} \approx 0.79 \\ d = 50: &\quad 10^{-1/50} \approx 0.96 \end{aligned}$$

By $d$=50 every point is nearly as far from its neighbour as from the boundary of the cube. The points have not moved apart; the space has grown underneath them.

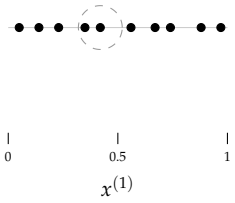



Figure 18: $N$=10 points in $[0,1]^1$: mean NN spacing $\approx 10^{-1}$=0.1. The line is well covered.

$\propto$ means "proportional to": $a \propto b$ says $a = c \cdot b$ for some constant $c$ independent of the variables of interest.

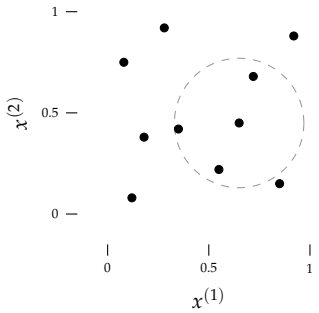



Figure 19: Same $N$=10 points in $[0,1]^2$: the space has grown from a line to a square, and the mean NN spacing rises to $10^{-1/2} \approx 0.32$.

FOR DISTANCE-BASED METHODS THIS IS FATAL. The $\varepsilon$-neighbourhood of a point captures a hypersphere whose volume shrinks relative to the cube: $V_{\text{sphere}} / V_{\text{cube}} \to 0$ as $d$ grows. A radius that includes many neighbours in two dimensions encloses almost nothing in fifty. Either $\varepsilon$ must grow until every point is a neighbour, collapsing all structure into one cluster, or it stays small and every point is labelled $p_n$.

*Examples & Exercises*

COMPUTING BY HAND reveals what the algorithm actually does at each step; run it slowly and feel where density propagates and where it stops.

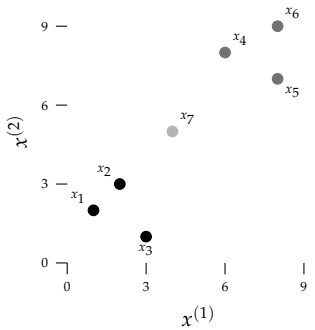



Figure 20: Seven points for the DBSCAN trace. $x_7$=(4,5) sits between the two natural groups.

GIVEN SEVEN POINTS $x_1$=(1,2), $x_2$=(2,3), $x_3$=(3,1), $x_4$=(6,8), $x_5$=(8,7), $x_6$=(8,9), $x_7$=(4,5), run DBSCAN with $\varepsilon$=2.5 and $n_{\min}$=2. Your tasks: compute the neighbourhood size $|N_{2.5}(x_i)|$ for each point, classify each point as $p_c$, $p_b$, or $p_n$, and identify the clusters.

NEIGHBOURHOOD SIZES. For each point $x_i$ count all points within distance 2.5 (including $x_i$ itself). Here is the full calculation for $x_1$:

$$\begin{aligned} d(x_1, x_2) &= \sqrt{(2-1)^2 + (3-2)^2} \\ &= \sqrt{1+1} \\ &= \sqrt{2} \\ &\approx 1.41 \quad (\leq 2.5, \text{ counted}) \\ d(x_1, x_3) &= \sqrt{(3-1)^2 + (1-2)^2} \\ &= \sqrt{4+1} \\ &= \sqrt{5} \\ &\approx 2.24 \quad (\leq 2.5, \text{ counted}) \\ d(x_1, x_7) &= \sqrt{(4-1)^2 + (5-2)^2} \\ &= \sqrt{9+9} \\ &= \sqrt{18} \\ &\approx 4.24 \quad (> 2.5, \text{ not counted}) \end{aligned}$$

So $N_{2.5}(x_1) = \{x_1, x_2, x_3\}$ and $|N_{2.5}(x_1)| = 3 \geq 2$: $x_1$ is a $p_c$. Compute the remaining neighbourhood sizes yourself, then verify against the table below.

VERIFY $x_7$ IS $p_n$. Compute $d(x_7, x_3)$ to confirm that $x_7$ lies outside

| Point | Coords | $\lvert N_{2.5}\rvert$ | Type | Neighbours in $N_{2.5}$ | Cluster |
|---|---|---|---|---|---|
| $x_1$ | $(1,2)$ | 3 | $p_c$ | $\{x_1, x_2, x_3\}$ | $C_1$ |
| $x_2$ | $(2,3)$ | 3 | $p_c$ | $\{x_1, x_2, x_3\}$ | $C_1$ |
| $x_3$ | $(3,1)$ | 3 | $p_c$ | $\{x_1, x_2, x_3\}$ | $C_1$ |
| $x_4$ | $(6,8)$ | 3 | $p_c$ | $\{x_4, x_5, x_6\}$ | $C_2$ |
| $x_5$ | $(8,7)$ | 3 | $p_c$ | $\{x_4, x_5, x_6\}$ | $C_2$ |
| $x_6$ | $(8,9)$ | 3 | $p_c$ | $\{x_4, x_5, x_6\}$ | $C_2$ |
| $x_7$ | $(4,5)$ | 1 | $p_n$ | $\{x_7\}$ only | $-$ |

Table 1: DBSCAN trace for $\varepsilon$=2.5, $n_{\min}$=2. All six canonical points are $p_c$'s; $x_7$ has no neighbour within radius 2.5 and is labelled $p_n$. The two natural clusters are recovered without specifying $K$ and without assigning $p_n$ to either group.

the $\varepsilon$-neighbourhood of its nearest candidate:

$$\begin{aligned} d(x_7, x_3) &= \sqrt{(4-3)^2 + (5-1)^2} \\ &= \sqrt{1+16} \\ &= \sqrt{17} \\ &\approx 4.12 \quad (> 2.5) \end{aligned}$$

Compute $d(x_7, x_4)$ yourself. Why is $x_7$ correctly identified as $p_n$ rather than being assigned to whichever cluster is nearer?

ADDING A BORDER POINT. Place an eighth point $x_8$=(8, 5.5) into the dataset and raise the density threshold to $n_{\min}$=3. Compute $\lvert N_{2.5}(x_8)\rvert$ and classify $x_8$ as $p_c$, $p_b$, or $p_n$. Which cluster, if any, does $x_8$ join, and why? Verify that the six original cluster members remain $p_c$'s under the new threshold.

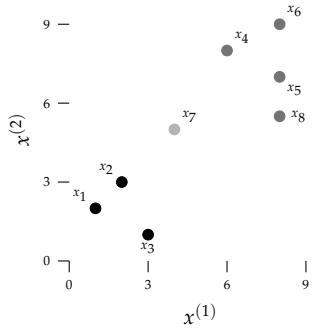



Figure 21: Eight points for the border-point exercise. $x_8$=(8, 5.5) sits below $C_2$.

DENSITY REACHABILITY FROM $x_8$. Still using the eight-point dataset with $\varepsilon$=2.5 and $n_{\min}$=3: is $x_8$ density-reachable from $x_4$? If so, write down the chain of points that connects them. Now reverse the question: is $x_4$ density-reachable from $x_8$? Why or why not?

PARAMETER SENSITIVITY. Re-run the trace with $\varepsilon$=4.5 and $n_{\min}$=2. Does $x_7$ change classification? At what minimum value of $\varepsilon$ does $x_7$ first enter the neighbourhood of some $p_c$?

BUILDING THE $k$-DISTANCE PLOT. Return to the original seven points with $n_{\min}$=2, so $k$=$n_{\min}$−1=1. For each point $x_i$, compute

the distance to its nearest neighbour $d_1(x_i)$. Here is $x_5$:

$$\begin{aligned} d(x_5,\, x_4) &= \sqrt{(8-6)^2 + (7-8)^2} \\ &= \sqrt{5} \\ &\approx 2.24 \\ d(x_5,\, x_6) &= \sqrt{(8-8)^2 + (9-7)^2} \\ &= \sqrt{4} \\ &= 2.00 \\ d_1(x_5) &= 2.00 \quad \text{(nearest neighbour is } x_6\text{)} \end{aligned}$$

Compute $d_1(x_i)$ for the remaining six points. Sort all seven values in ascending order and sketch the curve. Where is the elbow? Does the $\varepsilon$=2.5 from the earlier trace sit in the gap?

Comparison with K-Means. Run K-Means with $K$=2 on the same seven points. Which cluster does K-Means assign $x_7$ to? Is that assignment more or less informative than DBSCAN's $p_n$ label, and why?

K-Means baseline result. After convergence with $K$=2, the centroids settle at $\mu_1 \approx (2.0,\, 2.0)$ and $\mu_2 \approx (7.3,\, 8.0)$. Point $x_7$=(4, 5) is closer to $\mu_1$:

$$\begin{aligned} d(x_7,\, \mu_1) &= \sqrt{(4-2)^2 + (5-2)^2} \\ &= \sqrt{4+9} \\ &= \sqrt{13} \approx 3.61 \\ d(x_7,\, \mu_2) &= \sqrt{(4-7.3)^2 + (5-8.0)^2} \\ &= \sqrt{10.89+9} \\ &= \sqrt{19.89} \approx 4.46 \end{aligned}$$

K-Means assigns $x_7$ to $C_1$, as shown below. The assignment is not wrong in a strict sense; it is simply forced. K-Means has no $p_n$ concept and cannot abstain.

From global to local density. Return to the eight-point dataset ($x_1, \ldots, x_7$ plus $x_8$=(8, 5.5)) with $n_{\min}$=3. For each point $x_i$, compute the core distance $\varepsilon_c(x_i)$: the smallest radius that captures at least $n_{\min}$ points, itself included. Here is $x_5$:

The core distance $\varepsilon_c(x_5)$=2.00 is the radius needed to capture $\{x_8,\, x_6,\, x_5\}$: exactly three points.

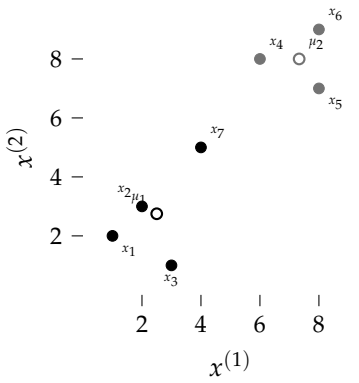



Figure 22: K-Means baseline with $K$=2. o-marks for centroids $\mu_1$ and $\mu_2$. Because K-Means must assign every point, $x_7$ is forced into $C_1$ (dark). Compare with Figure 12: DBSCAN labels $x_7$ as $p_n$ and makes no forced assignment.

$$
\begin{aligned}
d(x_5,\, x_8) &= 1.50 \\
d(x_5,\, x_6) &= 2.00 \\
d(x_5,\, x_4) &= \sqrt{5} \\
&\approx 2.24 \\
\varepsilon_c(x_5) &= 2.00
\end{aligned}
$$

Compute $\varepsilon_c$ for $x_1$ through $x_4$, $x_6$, $x_7$, and $x_8$. Which points have the largest core distance, and what does that tell you about their local density?

Now compute the mutual reachability distance for two pairs:

$$
\begin{aligned}
d_{\text{mreach}}(x_4,\, x_5) &= \max\{\varepsilon_c(x_4),\, \varepsilon_c(x_5),\, d(x_4,\, x_5)\} \\
d_{\text{mreach}}(x_7,\, x_2) &= \max\{\varepsilon_c(x_7),\, \varepsilon_c(x_2),\, d(x_7,\, x_2)\}
\end{aligned}
$$

Compare each to the raw Euclidean distance. For which pair does $d_{\text{mreach}}$ differ from $d$, and why? Why does HDBSCAN not need a global $\varepsilon$?

THE CURSE OF DIMENSIONALITY BY THE NUMBERS. The mean nearest-neighbour spacing scales as $d_{\text{NN}} \propto N^{-1/d}$. Compute $N^{-1/d}$ for $N$=100 at $d = 2,\ 10,\ 50,\ 100$. At what dimension $d$ does the spacing first exceed 0.9? If you tried to run DBSCAN at that dimension, what would happen to the $\varepsilon$-neighbourhoods?

BACK AT THE COFFEETABLE CODE. Same 1 797 aroma fingerprints, two coordinates. Implement epsilon-neighborhoods and the core-point classifier, use the $k$-distance plot to pick $\varepsilon$, then compare DBSCAN with $K$-Means and Ward at the same cluster count.

CODE will be provided as a Python notebook. Use it as a starting point, break things, and observe what changes.

WHAT TO OBSERVE. Core points fill the dense hearts of each variety;

noise points land in the gaps no method should claim. DBSCAN follows density contours rather than drawing spherical cuts, so it agrees with $K$-Means in the cores but diverges at irregular boundaries. A single $\varepsilon$ cannot capture clusters of very different densities. In next steps, learn about sensitivity sweeps and HDBSCAN to address this.

## *Self-Reflection and Recap*

SELF-REFLECTION questions to guide your thinking:

- What does density reachability gain over distance-to-centroid? Which K-Means failures disappear, which new ones appear?
- Why is density reachability asymmetric, and what does that imply for the determinism of $p_b$ labels?
- A $p_b$ sits inside two $p_c$'s from different clusters. Is its label unique? What does that say about what a DB Scan cluster really is?
- Why is bridging the density-based dual of single-linkage chaining, and why does every remedy cost something?
- The $k$-distance plot shows a straight line with no elbow. What does this tell you, and which method would you try next?
- What is the essential difference between an OPTICS reachability plot and an HDBSCAN density hierarchy?
- When would you prefer DB Scan over Ward's method even if both produce a sensible partition?

RECAP of Key Concepts:

- DB Scan clusters are maximal sets of density-connected points; no centroid, no pre-specified $K$
- Every point is $p_c$ when $|N_\varepsilon| \geq n_{\min}$, $p_b$ when inside a $p_c$'s neighbourhood without being $p_c$ itself, or $p_n$
- Density reachability is transitive along $p_c$'s but not symmetric; $p_b$ labels depend on visit order
- The partition is determined by $(X, \varepsilon, n_{\min})$ up to shared $p_b$'s; $O(n)$ queries, no restarts
- The $k$-distance plot with $k{=}n_{\min}{-}1$, sorted ascending, chooses $\varepsilon$ at the elbow

- Failure modes: varying density, where no global $\varepsilon$ fits two scales, and high dimensionality, where distances concentrate
- Bridging fuses clusters through chains of $p_c$'s, the density dual of single-linkage chaining
- OPTICS exposes all density scales via a reachability plot; HDBSCAN extracts the most persistent clusters from a full density hierarchy

Two ways out of the curse of dimensionality. Either we collect exponentially more data [4], which is rarely possible, or we reduce the dimension so that the samples we already have sit densely in the space we actually care about. The observation that makes the second route tractable is that real datasets almost never use their ambient dimensions fully: the meaningful variation lives on a much lower-dimensional subset, and the rest is redundancy, correlation, or noise. Finding that signal-carrying subset is what the next part of these notes is about.

[4]

Up next. Part II, Representing Pattern, leaves the original feature space behind and learns to construct new coordinates. Chapter 5 opens with principal component analysis as the linear baseline. Only once a dataset is described in the dimensions that carry signal do the clustering methods of Part I regain their footing.

Teaser. The curse of dimensionality breaks every method in this chapter, yet real data rarely uses all its ambient dimensions. Can we find the directions that carry the most signal and build a lower-dimensional representation around them, so that the clustering methods of Part I regain their footing?

Feedback.