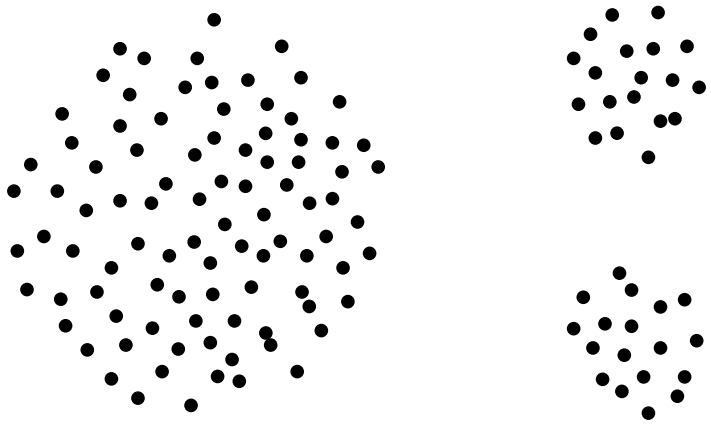
Clustering by Connectivity

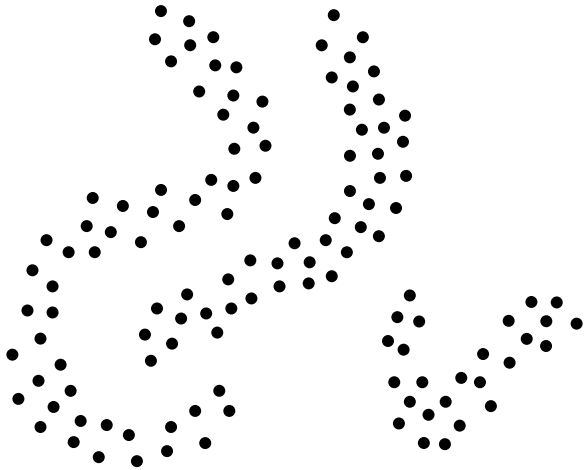
Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

The clusters found by centroid-based clustering (e.g., k-center and k-means) tend to have "ball shapes".



Sometimes clusters may have arbitrary shapes, e.g.:



Clustering by connectivity is a form of clustering that is built on "distance graphs", and deviates significantly from centroid-based clustering. We will discuss two clustering methods under this category:

- Agglomerative clustering — also known as "hierarchical clustering".
- Density-based clustering

Agglomerative Clustering (a.k.a. Hierarchical Clustering)

Given a set P of n objects, the **agglomerative method** works as follows:

1. At the beginning, each object in P forms a cluster by itself.
2. Merge the two clusters that are most similar to each other.
3. Repeat the previous step until only one cluster is left.

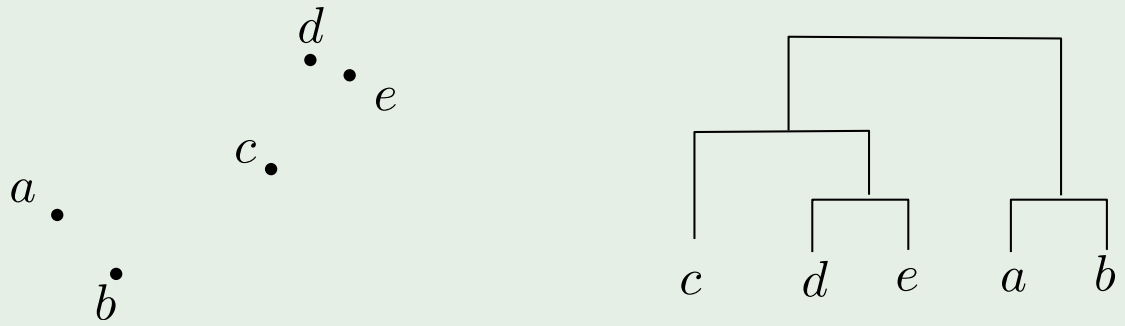
The above framework can be instantiated in many ways depending on how cluster similarity is defined. Specifically, let C_1 and C_2 be two clusters, each being a set of objects. To measure their similarity, we need a function $d(C_1, C_2)$ such that the smaller the function's value, the more similar the two clusters.

Some common definitions for cluster similarity are:

$$
\begin{aligned}
d_{min}(C_1, C_2) &= \min_{o_1 \in C_1, o_2 \in C_2} dist(o_1, o_2) \\
d_{max}(C_1, C_2) &= \max_{o_1 \in C_1, o_2 \in C_2} dist(o_1, o_2) \\
d_{mean}(C_1, C_2) &= \frac{1}{|C_1||C_2|} \sum_{o_1 \in C_1, o_2 \in C_2} dist(o_1, o_2)
\end{aligned}
$$

Among the three, d_{min} is the most popular—when this function is chosen, the agglomerative framework on the previous slide is known as the **single linkage algorithm**. We will focus on d_{min} in the rest of the lecture.

Example



Execution of the agglomerative method using the d_{min} metric:

1. Initially, 5 clusters: $\{a\}, \{b\}, \{c\}, \{d\}, \{e\}$.
2. Merging $\{d\}, \{e\} \Rightarrow \{a\}, \{b\}, \{c\}, \{d, e\}$.
3. Merging $\{a\}, \{b\} \Rightarrow \{a, b\}, \{c\}, \{d, e\}$.
4. Merging $\{c\}, \{d, e\} \Rightarrow \{a, b\}, \{c, d, e\}$.
5. Merging $\{c\}, \{d, e\} \Rightarrow \{a, b, c, d, e\}$.

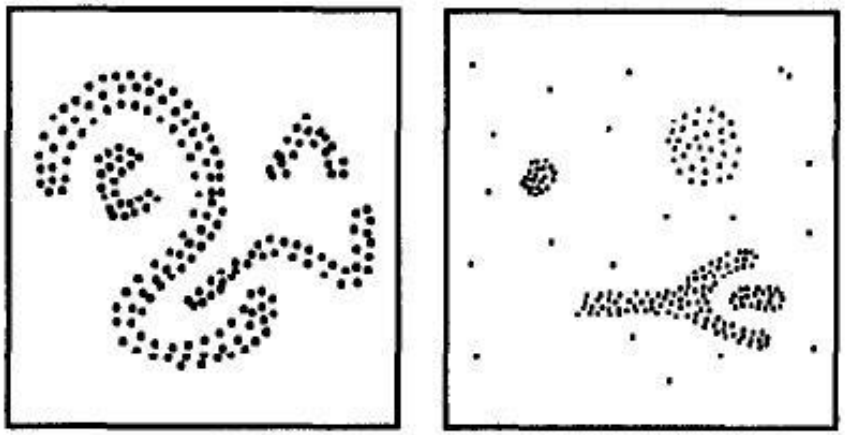
The merging history of the algorithm can be represented as a tree (see above), which is called a **dendrogram**.

Think:

- How many merges are there in total if we have n objects?
- Given a dendrogram, how would you obtain k clusters quickly?

Density-Based Clustering

In some applications, clusters can have arbitrary shapes and may need to be separated from **noise**:



(figures from a KDD96 paper titled "A density-based algorithm for discovering clusters in large spatial databases with noise")

We will learn a method called DBSCAN to find such clusters. It serves as a representative of noise-resistant density-based clustering, which works by enforcing two principles:

- The area around a noise point is "sparse".
- If two points are placed in the same cluster, it should be possible to "walk" from one point to the other by staying only in the "dense" areas.

Parameters and Core Points

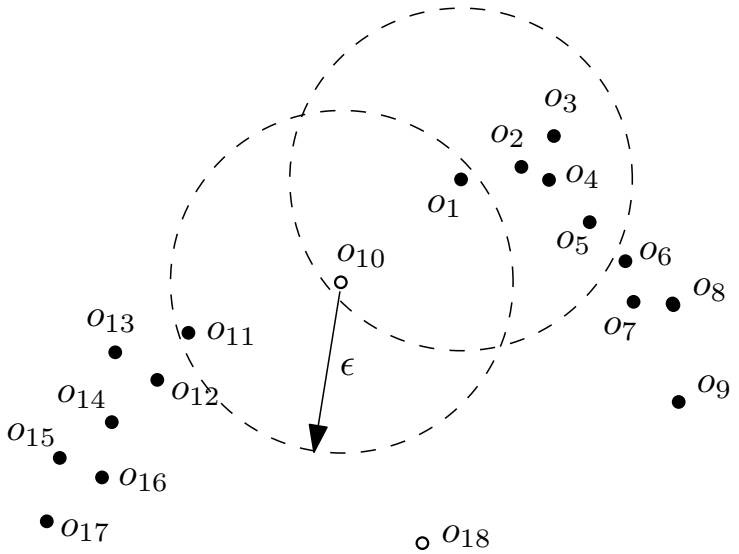
Parameters:

- ϵ: a distance threshold.
- *MinPts*: a constant integer.

$B(p, \epsilon)$: the ball centered at a point with radius ϵ, called the **vicinity area** of p.

P: the set of points to cluster

Core point: a point $p \in P$ such that $B(p, \epsilon)$ covers at least *MinPts* points of P.



MinPts = 4
Core points in black

Forming Clusters

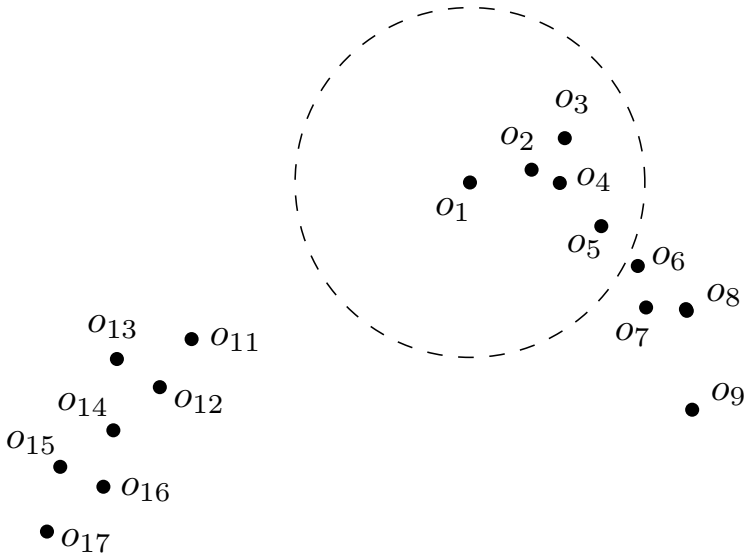
Conceptually, clusters are defined in two steps:

1. Cluster core points.
2. Assign non-core points.

We will explain each step in turn.

Step 1: Cluster core points

This step focuses only on core points.



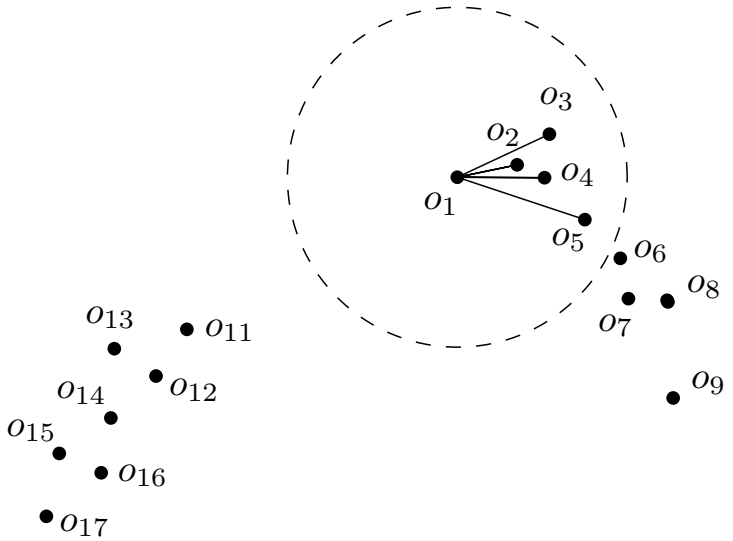
$MinPts = 4$

Core points in black

Step 1: Cluster core points

Connect a core point p to all the points in $B(p, \epsilon)$.

For example, o_1 is connected to 4 points in its vicinity area:

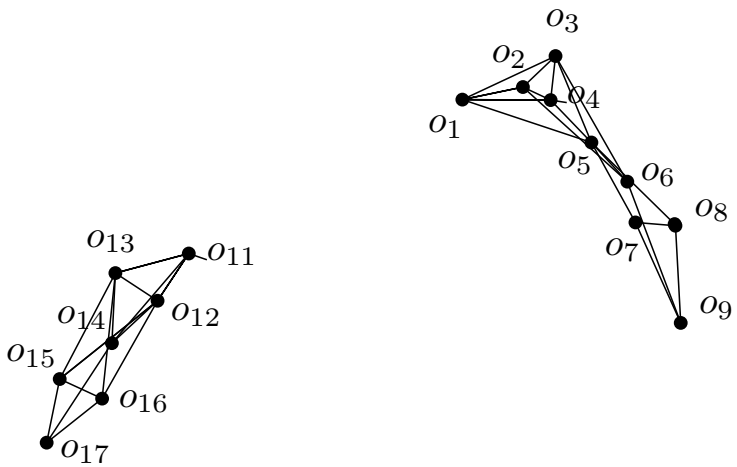


$MinPts = 4$

Core points in black

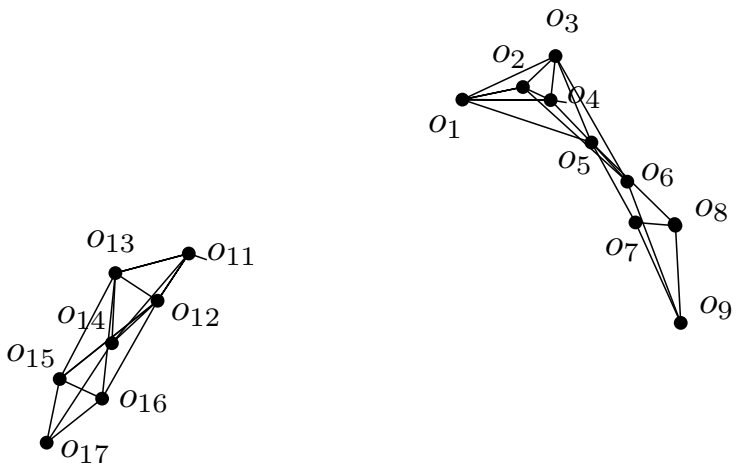
Step 1: Cluster core points

This is the situation after adding all the edges:



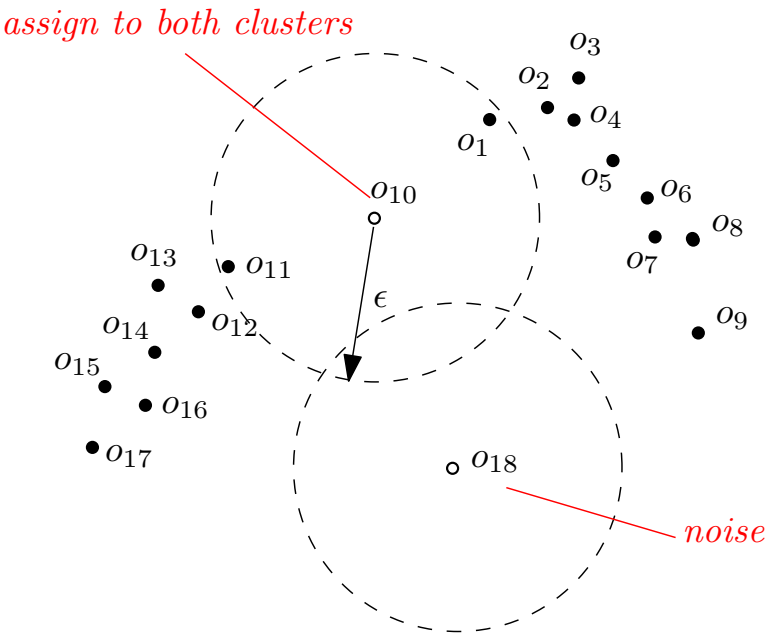
Step 1: Cluster core points

Take each connected component of the resulting a graph as a cluster.



Step 2: Assign non-core points

Every non-core point p is added to the cluster of every core point in $B(p, \epsilon)$. For example, o_{10} is added to two clusters: the cluster of o_1 and the cluster of o_{11}.

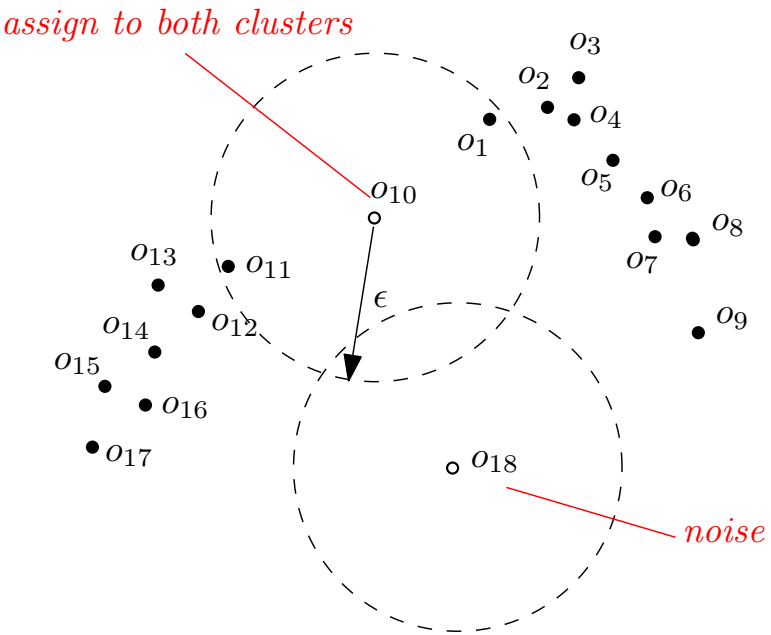


$MinPts = 4$

Each non-core point can be assigned to at most $MinPts - 1 = O(1)$ clusters.

Step 2: Assign non-core points

Final clusters: $\{o_1, o_2, ..., o_9, o_{10}\}$, $\{o_{10}, o_{11}, o_{12}, ..., o_{17}\}$.



$MinPts = 4$

The clustering result is unique.

It is straightforward to obtain the DBSCAN clusters in $O(dn^2)$ time, where n is the number of points.

There is an inherent connection between DBSCAN the single-linkage algorithm we discussed in the previous lecture.

Think: Suppose that you have computed a dendrogram for single-linkage. How would you use the dendrogram to obtain a DBSCAN clustering with parameterized by $\epsilon > 0$ and $minPts = 1$?