I. Hierarchical Clustering

A clustering method that builds a hierarchy (tree structure) of clusters instead of just producing a fixed number of clusters like K-Means does. You can think of it as gradually merging or splitting clusters while recording all the steps — this record forms a dendrogram, a tree-like diagram showing how clusters combine.

The Core Idea: Hierarchical clustering looks for structure in the data by repeatedly grouping similar points (or clusters) together (or apart), based on how close they are to each other. There are two main types:

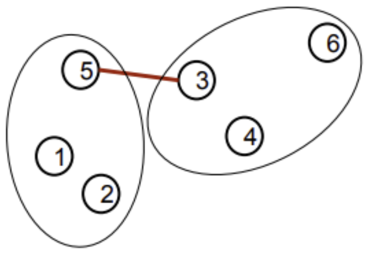
1. Agglomerative (bottom-up)
    a. Start: every point is its own cluster
    b. Compute the distance between all pairs of clusters
    c. Merge the two closest clusters
    d. Keep repeating b and c, stop when everything becomes one big cluster
2. Divisive (top-down)
    a. Start: all points in one cluster
    b. Repeatedly split clusters into smaller ones
    c. Stop: when every point is it's own cluster

II. Distance Functions for Hierarchical Clustering

Distance between points: d(p1, p2)
Distance between clusters: D(C1, C2)

1. Single-Link Distance

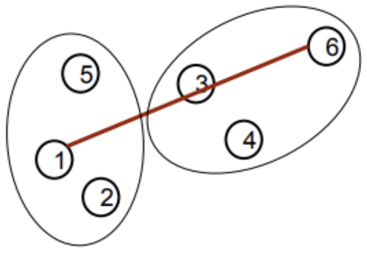
The distance between two clusters = the minimum distance between any pair of points (one from each cluster).



$D_{single}(C1,C2) = \min_{i \in C1, j \in C2} d(i,j)$

Good at handling clusters of different shapes and sizes. But it's sensitive to noise and can form "chain-like" clusters (elongated).

2. Complete-Link Distance

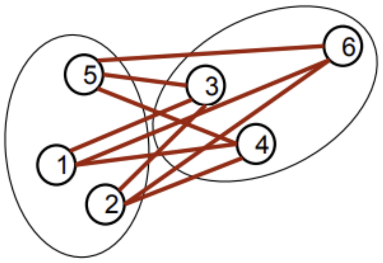


The distance between two clusters = the minimum distance between any pair of points (one from each cluster).

$D_{complete}(C1,C2) = \max_{i \in C1, j \in C2} d(i,j)$
Produces compact, spherical clusters.
Less sensitive to noise, but may split large clusters unnaturally.

3. Average-Link Distance

The distance between clusters = the average of all pairwise distances between points in the two clusters.

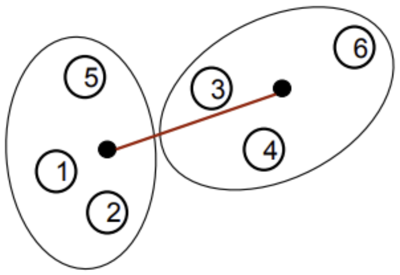


$D_{average}(C1,C2) = \frac{1}{|c1||c2|} \sum_{i \in C1, j \in C2} d(i,\ j)$

A balance between single and complete link.
Less affected by noise, but tends to favor round (globular) clusters.

4. Centroid Distance

The distance between the centroids (means) of the clusters.



$D_{centroid}(C1,C2) = d(mean(C1),mean(C2))$

Simple and intuitive.
But can cause issues (called inversions) where merged clusters appear closer than their subclusters.

5. Ward's Distance

Measures how much the variance (spread) increases when you merge two clusters.
You merge clusters that cause the smallest increase in total within-cluster variance.

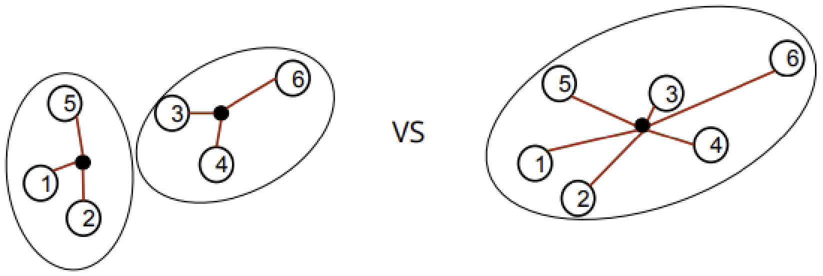
$D_{WD}(C1,C2)$=(spread after merging)−(spread before merging)

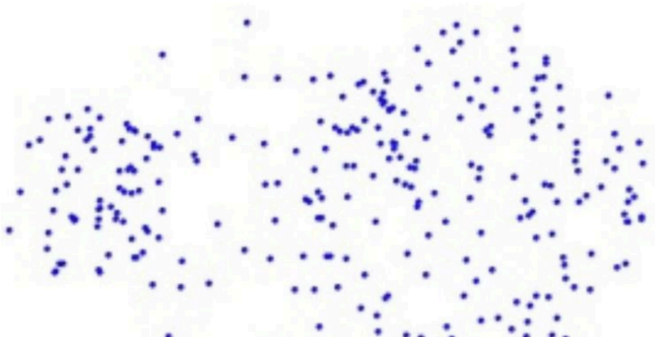
$$D_{WD}(C_1, C_2) = \underbrace{\sum_{p \in C_{12}} d(p, \mu_{12})}_{\text{spread of merged cluster}} - \underbrace{\sum_{p_1 \in C_1} d(p_1, \mu_1)}_{\text{spread of cluster 1}} - \underbrace{\sum_{p_2 \in C_2} d(p_2, \mu_2)}_{\text{spread of cluster 2}}$$

Each cluster has some "tightness" — the sum of squared distances of its points to its centroid.

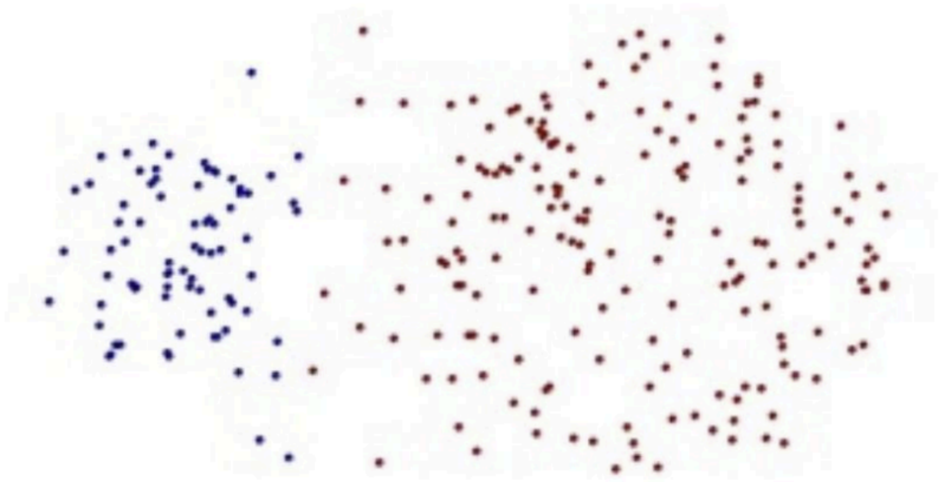
When you merge two clusters, that tightness usually increases (because points are now farther from the new, common centroid).

Ward's method picks the merge that causes the smallest increase in this spread.

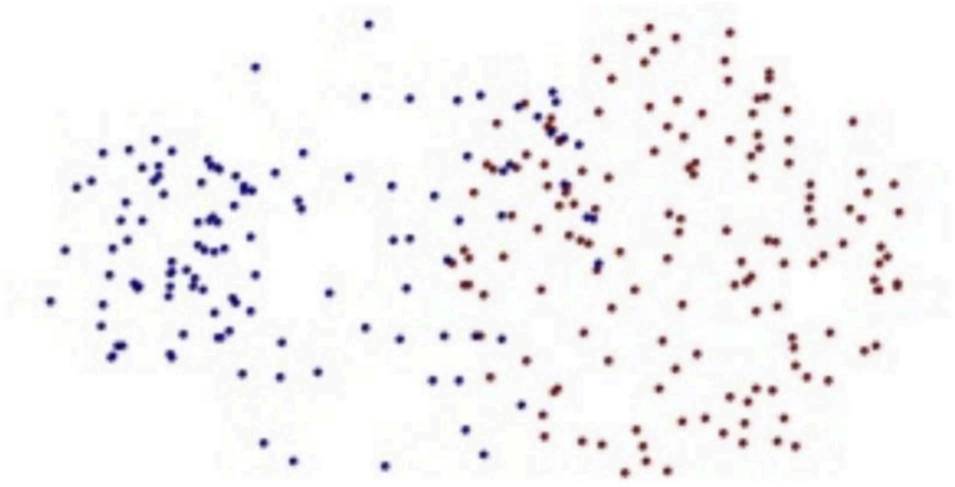




Dataset



Clustering with Complete-Link Distance



Clustering with Single-Link Distance

III. Agglomerative Clustering Algorithm
Start with each point as its own cluster → step by step, merge the closest clusters until everything becomes one cluster. At each step, we record the merge so we can later draw a dendrogram (the tree structure). Steps:

1. Start: every datapoint as its own cluster

2. Compute pairwise cluster distances ….. use your chosen distance function (single-link, complete-link…..) and create a distance matrix.

3. Merge: find the pair of clusters with the smallest distance, merge them. (smallest value on matrix, merge the 2 clusters for which that distance represents)

4. Update the distance matrix: the clusters now look different, recalculate distances based on what it looks like now ….. update the matrix.

5. Repeat steps 3 and 4: continue merging the two nearest clusters at each step → until only one big cluster remains that contains all points.

IV. Distance Matrix
A distance matrix is simply a table that shows the pairwise distances between all points (or clusters) in your dataset. It's how the algorithm keeps track of how close or far apart each pair of objects (points or clusters) is.

	A	**B**	**C**	**D**
A	0	d(A,B)	d(A,C)	d(A,D)
B	d(B,A)	0	d(B,C)	d(B,D)
C	d(C,A)	d(C,B)	0	d(C,D)
D	d(D,A)	d(D,B)	d(D,C)	0

Properties:
- Symmetric: because $d(A,B) = d(B,A)$
- Diagonal = 0: each point's distance to itself is zero.
- The values depend on the distance metric used (e.g., Euclidean, Manhattan).
- As clustering progresses, entries change -- we recompute the distances between clusters, not individual points.

Example:

Let's say we have 4 points in 2D space:

- A(0,0)
- B(1,1)
- C(3,0)
- D(0,-2)

Say we choose to use Euclidean distance, let's compute the pairwise distances:

- d(A,B) = √2
- d(A,C) = 3
- d(A,D) = 2
- d(B,C) = √5
- d(B,D) = √10
- d(C,D) = √13

So the matrix looks like:

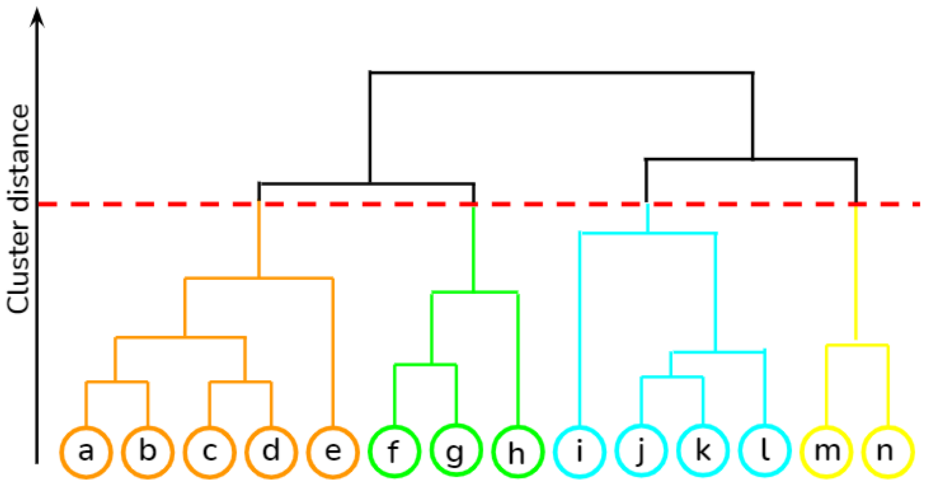
	A	**B**	**C**	**D**
A	0	√2	3	2
B	√2	0	√5	√10
C	3	√5	0	√13
D	2	√10	√13	0

In hierarchical clustering, this matrix is the engine of the agglomerative process -- it tells the algorithm who should merge next.

V. Cutting the Dendrogram

After the process, we have 1 large cluster…. to actually get useful smaller clusters, you have to decide where to "cut" the resulting clustering tree -- determining the amount of resulting clusters.

The height of the cut line (the red dashed line in your example) represents a similarity threshold -- or equivalently, a maximum cluster distance you're willing to accept within a cluster.



In this example, we end up with 4 resulting clusters (based on where the cut is)

The lower you cut → the more strict you are about similarity.

- Points must be very close (highly similar) to be in the same cluster.
- You'll get many small, tight clusters.

The higher you cut → the more relaxed you are about similarity.

- You allow points that are farther apart to merge into the same cluster.
- You'll get fewer, larger, more general clusters.

In general, hierarchical clustering is used to explore and reveal the structure or hierarchy within data -- showing how data points or groups naturally cluster together at different levels of similarity.

It doesn't just give you one final clustering (like K-Means); it gives you an entire hierarchy of relationships -- from small, tight clusters to large, broad groupings.
