# *Hierarchical Clustering*

*2026-05-06 · cheerful mango Haubentaucher*

Ich springe von Level zu Level zu Level.

## *The Why*

K-Means left us with three sharp problems.

The centroid assumption forces every cluster to be convex and compact. A single mean position must stand in for all members, so the algorithm breaks whenever clusters are elongated, ring-shaped, or split across multiple densities. All cluster characteristics and structure information are lost and filtered when the mean is computed. The centroid then lands somewhere no data actually lives, and every assignment made from that position is wrong by construction.

Sensitivity to initialization and outliers means results are non-deterministic. A single extreme point can pull a centroid far from the dense mass of its cluster, silently distorting every assignment downstream. Different random starts can converge to entirely different partitions of the same data.

A single scale of structure is all K-Means can see at once. With $K{=}2$ you recover two broad groups; with $K{=}10$ you recover ten fine subgroups. Both are valid descriptions of the same data at different levels of zoom, yet K-Means forces you to choose one and discard the other. This is not a flaw in the algorithm but a flaw in the question: asking "how many clusters?" presupposes that data has one natural scale.

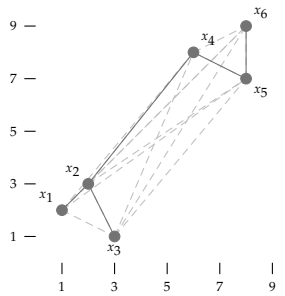


Figure 1: The canonical six points with all pairwise distances (dashed) and the full single-linkage hierarchy (solid). The last merge connects $x_2$ and $x_4$ across the two groups. No centroids needed; only pairwise distances drive the clustering.

Hierarchy is the natural answer. Structure in real data rarely lives at a single scale. Biological taxonomy nests species inside genera inside families inside orders. A document corpus organizes words into topics and topics into domains. File systems nest folders inside

folders. In each case, the right question is not "how many clusters?" but "which level of abstraction do I need right now?".

HIERARCHICAL CLUSTERING produces a dendrogram: a binary tree in which leaves are individual data points and each internal node records the distance at which two sub-trees were merged. The full range of partitions, from $K{=}1$ to $K{=}n$, lives in this one structure.

THE SAME DISTANCE MATRIX that K-Means uses to assign points to centroids can be used to build an entire tree of clusterings. The tree is built in one pass; every value of $K$ is then available for free.

IN ORDER TO MOVE FROM A FLAT PARTITION TO A FULL HIERARCHY of clusterings, we have good reason to find ways to,

- extend pairwise point distances to cluster distances, making the notion of closest pair well-defined at every scale.
- record each merge in a tree structure that makes every partition from $K{=}1$ to $K{=}n$ readable in a single pass.
- understand how the choice of cluster distance and linkage design shapes the resulting hierarchy and where it fails.

*Hands-On Experience*

How would you build a hierarchy of clusterings over these six points? Which is the cluster on the highest level of abstraction, $K$=1? How easy is it to build hierarchy top down? Which are the two clusters on the second highest level, $K$=2? Still obvious, then it gets trickier when $K$=3 the first non-trivial design-choices are to be made.

Divisive clustering (top-down) starts with all points in one cluster and recursively splits the cluster with the largest diameter. It is rarely used in practice: splitting is more expensive than merging, and early splits made with global information can be noisier than local merges.

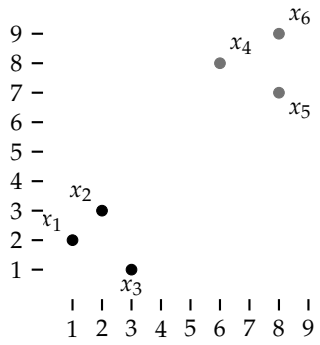



Figure 2: The canonical six points. Which two are closest? Try building a hierarchy top-down, then bottom-up, which one is easier?

Building Hierarchy Bottom Up is more intuitive. Which points are closest to each other? See how this again draws on our understanding of distance as a proxy for similarity? Connect them with a line. Now find the next closest pair and connect them. Keep going until all six points are linked into a tree. At what level of distance would you cut that tree to recover two groups? Three groups?

A horizontal cut at any chosen height yields a flat partition, as we are used to as a result of K-Means.

The fundamental question this chapter answers is how do we formalize the bottom-up merging process? We will see how agglomerative clustering builds a dendrogram from pairwise distances; and how different linkage criteria encode different assumptions about what makes two clusters close.

Linkage, is non-trivial. Two points intuitively merge into a cluster, but beyond that we need a rule for measuring the distance from that cluster to everything else. Do we use the closest pair of points across the two clusters? The furthest? The average? Each answer is a different linkage criterion, and each one tells a different story about what "nearby clusters" means.

The Learning Objectives of this lecture:

- Understand how agglomerative clustering builds a dendrogram through successive minimum-distance merges, and read merge-height gaps to identify the natural number of clusters.
- Apply single, complete, and Ward's linkage and explain how each criterion shapes the resulting hierarchy.
- Identify when hierarchical clustering succeeds where K-Means fails, recognise its failure modes, and explain why scalability requires approximations such as BIRCH.

*The Hierarchical Clustering Algorithm*

THE AGGLOMERATIVE STRATEGY[1] is bottom-up: given data $X = \{x_1, \ldots, x_n\}$, initialize $n$ singleton clusters $C_i = \{x_i\}$ in an active set and precompute a distance matrix $D[i, j] = d(x_i, x_j)$. At each step the closest pair $(i^*, j^*) = \arg\min_{i \neq j} D[i, j]$ is merged into $C_m = C_{i^*} \cup C_{j^*}$; the distances to remaining active clusters are updated via linkage criterion $L$. Every merge is recorded as an internal node of the dendrogram $T$, labeled with the distance $D[i^*, j^*]$ at which it occurred.

[1]

THE RESULT $T$ is a binary tree. Every valid flat partition from $K$=1 to $K$=$n$ can be read off by making a horizontal cut at the desired height. The $n - 1$ merge heights span the full range from the smallest pairwise distance to the diameter of the data.

**Require:** Data $X = \{x_1, \ldots, x_n\}$, linkage criterion $L$
**Ensure:** Dendrogram $T$
1: Initialize: $C_i \leftarrow \{x_i\}$ for $i = 1, \ldots, n$; active $\leftarrow \{C_1, \ldots, C_n\}$
2: Compute $D[i, j] \leftarrow d(x_i, x_j)$ for all $i \neq j$
3: **while** |active| $> 1$ **do**
4: $(i^*, j^*) \leftarrow \arg\min_{i \neq j} D[i, j]$
5: **Record merge:** $(C_{i^*}, C_{j^*})$ at height $D[i^*, j^*]$ in $T$
6: $C_m \leftarrow C_{i^*} \cup C_{j^*}$
7: Update $D[m, k] \leftarrow L(C_m, C_k)$ for each active $C_k \neq C_{i^*}, C_{j^*}$
8: Remove $C_{i^*}$ and $C_{j^*}$ from active; add $C_m$
9: **end while**

Algorithm 1: Agglomerative Hierarchical Clustering.

NOTICE THAT $L$ IS THE ONLY HYPERPARAMETER TO DETERMINE. The algorithm is otherwise fully deterministic.

INITIALIZATION IS DETERMINISTIC. Each point starts as its own singleton cluster, there is nothing to choose. This stands in sharp contrast to $k$-means, where a random centroid draw forces multiple restarts for optimization. Here, the starting state is uniquely determined by the data, so two runs on the same dataset always produce the same dendrogram.

Strictly, the dendrogram is unique only when all pairwise distances are distinct. When ties occur, the merge order depends on the tie-breaking rule, typically lowest index first.

TRACE THROUGH OUR SIX POINTS. Dotted edges represent pairwise distance calculations; the dashed edge marks the current merge; gray edges record prior ones.

REMOVE TWO. ADD ONE. In every iteration reduces |active| by exactly one (two clusters are removed and one merged cluster is added)

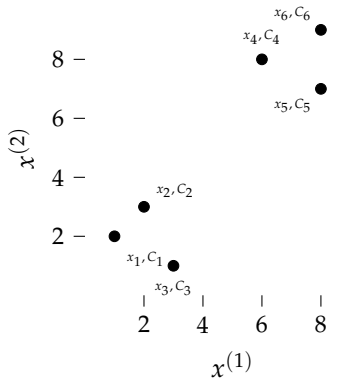



Figure 3:

INITIALIZATION: $n$=6 singleton clusters $C_i$=$\{x_i\}$ placed in the active set.

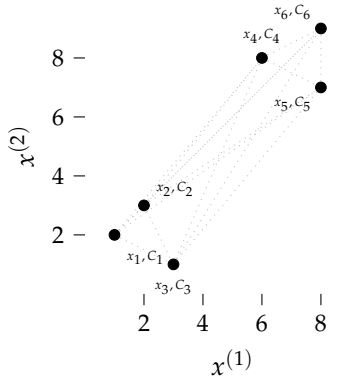



Figure 4:

COMPUTE $D[i,j]$=$d(x_i, x_j)$ for all $\binom{6}{2}$=15 pairs. Each dotted edge is one entry in the distance matrix. At every iteration the algorithm selects the globally shortest edge, $(i^*, j^*) = \arg\min_{i \neq j} D[i,j]$, where $i, j$ range only over the current active set. Once $C_{i^*}$ and $C_{j^*}$ are merged into $C_m$, they are removed from active and replaced by $C_m$; past merges are never revisited.

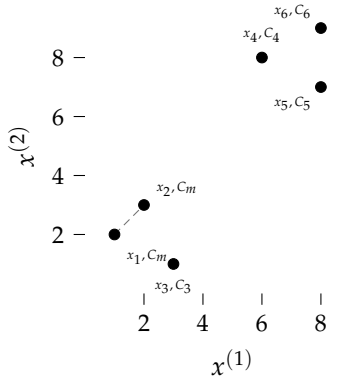



Figure 5:

MERGE 1: the globally smallest distance is $d(x_1, x_2)$. Singletons $\{x_1\}$ and $\{x_2\}$ merge into $C_m$=$\{x_1, x_2\}$, recorded as an internal node in $T$ with both singletons as children, labeled by $d(x_1, x_2)$: $T$ += $C_7 = (C_1, C_2, d(x_1, x_2))$.

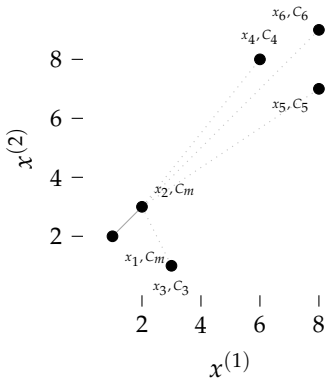



Figure 6:

UPDATE $D$: after forming $C_7 = \{x_1, x_2\}$, four rows and columns of $D$ are refreshed via $D[7,k] \leftarrow L(C_7, C_k)$. Under single linkage $L$ takes the nearest member: $D[7,k] = \min_{x \in C_7} d(x, C_k)$. Here $x_2$ is the nearest member of $C_7$ to all remaining singletons, so every new entry anchors at $x_2$ (dotted edges).

DISCUSS LINKAGE

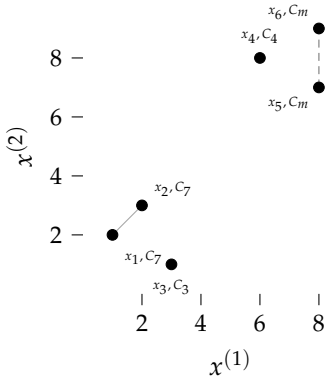



Figure 7:

MERGE 2: the next minimum in the updated $D$ is $d(x_5, x_6)$. Singletons $\{x_5\}$ and $\{x_6\}$ merge into $C_m = \{x_5, x_6\}$, recorded in $T$ as $T \mathrel{+}= C_8 = \big(C_5,\, C_6,\, d(x_5, x_6)\big)$.

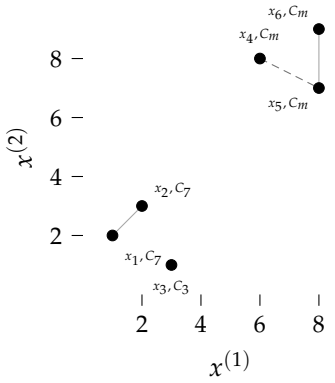



Figure 8:

MERGE 3: $D\big(C_8, x_4\big)$ is minimal under single linkage (distances to a merged cluster use the nearest member). $D\big(C_7, x_3\big)$ ties but $x_4$ merges first by index. Cluster $C_8$ absorbs $x_4$, linking via $x_5$: $T \mathrel{+}= C_9 = \big(C_8,\, C_4,\, D(C_8, C_4)\big)$.

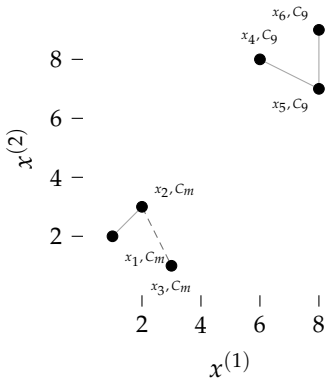



Figure 9:

MERGE 4: also at height $\sqrt{5}$, $D(C_7, x_3) = \sqrt{5}$. Cluster $C_7$ absorbs $x_3$, linking via $x_2$: $T \mathrel{+}= C_{10} = (C_7, C_3, D(C_7, C_3))$.

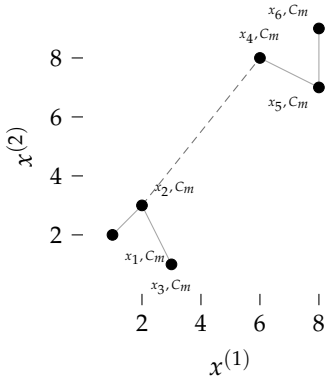



Figure 10:

MERGE 5: $C_{10}=\{x_1, x_2, x_3\}$ and $C_9=\{x_4, x_5, x_6\}$ merge at height $d(x_2, x_4)$, the nearest cross-cluster pair. All six points form one cluster; the dendrogram $T$ is complete.

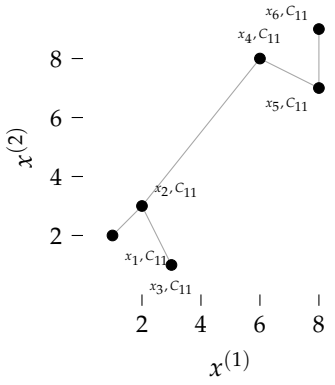



Figure 11:

TERMINATION: |active|=1. All $n$=6 points belong to $C_{11}$; the five merge edges span the complete single-linkage dendrogram. No further merges are possible.

TERMINATION. The algorithm halts when $|\text{active}| = 1$: exactly one cluster remains, containing all $n$ points. Termination is guaranteed after exactly $n - 1$ merges. At that point $T$ holds $n - 1$ internal nodes, one per merge, and is the complete dendrogram.

LINKAGE CRITERIA BEYOND TWO POINTS IS NON-TRIVIAL determine how the distance between two clusters is computed from the distances between their individual members. The choice of linkage profoundly shapes the dendrogram. Swap $L$ and you get a fundamentally different clustering.

ANSWERS FROM WHERE WE MEASURE THE DISTANCE. The distance then still decides what we are going to merge.

### SINGLE LINKAGE

$$D_{\text{single}}(C_i, C_j) = \min_{x \in C_i,\, y \in C_j} d(x, y)$$

The distance between two clusters is the distance between their nearest members. Single linkage admits an implementation based on Prim's minimal spanning tree (details later), which is fast. This also lets it detect elongated, non-convex clusters. This behavior is called chaining, and is also a weakness; a single close pair of points can bridge two otherwise distant clusters, pulling them into one.

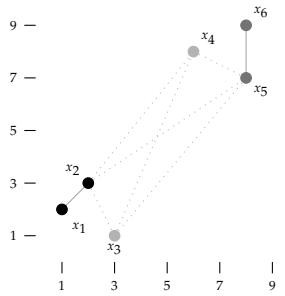



Figure 12: Single linkage: dotted lines show $d_{\text{min}}$ for each of the six active cluster pairs. The smallest, $d(x_2, x_3)$=$d(x_4, x_5)$, determines the next merge.

### COMPLETE LINKAGE

$$D_{\text{complete}}(C_i, C_j) = \max_{x \in C_i,\, y \in C_j} d(x, y)$$

The distance between two clusters is the distance between their furthest members. Complete linkage tends to produce compact, roughly equal-diameter clusters and is robust to chaining. It may, however, split naturally elongated shapes.

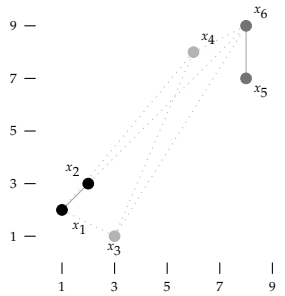



Figure 13: Complete linkage: dotted lines show $d_{\text{max}}$ for each active cluster pair. The largest determines which merge is most expensive.

### AVERAGE LINKAGE

$$D_{\text{average}}(C_i, C_j) = \frac{1}{|C_i|\,|C_j|} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y)$$

Also known as the Unweighted Pair Group Method with Arithmetic Mean. Is a compromise: less prone to chaining than single linkage, less aggressive at squashing elongated clusters than complete linkage. Average linkage averages all cross-cluster pairwise distances.

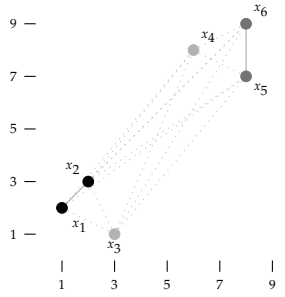



Figure 14: Average linkage: $D$ averages all pairwise cross-cluster distances. Every dotted line contributes to the mean.

### Ward's Method

Ward's criterion merges the two clusters whose union minimises the increase in total within-cluster variance:

$$D_{\text{Ward}}(C_i, C_j) = \frac{|C_i|\,|C_j|}{|C_i| + |C_j|}\,\|\mu_i - \mu_j\|^2$$

where $\mu_i$ and $\mu_j$ are the centroids of $C_i$ and $C_j$.

$\|\cdot\|^2$ is a squared norm, not a Euclidean distance. Ward's values are variance increases; single and average linkage values are distances. The scales are not comparable.

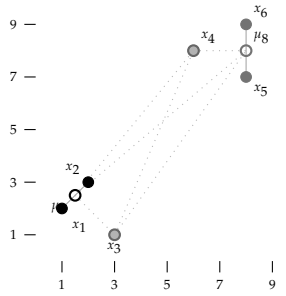
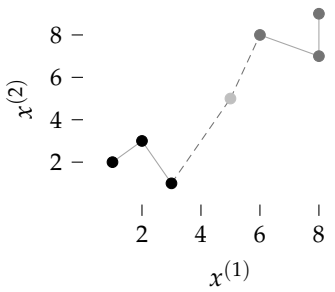



Figure 15: Ward's linkage: dotted lines connect centroids of all active cluster pairs; $D$ is proportional to $\|\mu_i - \mu_j\|^2$ weighted by cluster size.

When Hierarchical Clustering Fails the root cause is that the linkage criterions optimises locally with no global view nor understanding for scale. Every point gets clustered at some point in the agglomeration.




#### Chaining and bridging

Figure 16: Single-linkage chaining: one intermediate point (light) bridges two well-separated clusters. Within-cluster merges (solid) complete first; single linkage then greedily closes through the bridge (dashed), incorrectly joining both clusters into one.

Shape Bias Ward's linkage inherits K-Means's spherical bias exactly. The difference is recovery: K-Means re-assigns points each iteration and can partially correct a bad partition. Ward's cannot; the bias compounds irreversibly.

Irreversible greedy merges and local optima. At each step the algorithm selects the globally optimal merge under the current distance matrix: no local minimum trap exists within a single iteration. However, the greedy (no-reassignment) strategy is myopic. Once $C_{i^*}$ and $C_{j^*}$ are merged into $C_m$, that decision is permanent; no subsequent step can reverse it. A suboptimal early merge propagates through all future distance updates via $L$, and the resulting dendrogram may differ substantially from the globally optimal hierarchy.

Irrecoverability is what makes chaining and shape bias genuinely dangerous. A bad merge silently propagates through every subsequent distance update; the only remedy is to restart from scratch.

COMPUTATIONAL COMPLEXITY Memory is the hard wall. The distance matrix $D_{i,j}$ requires $O(n^2)$ space. At $n$=100,000 that is already ≈40 GB for float32: you either fit it in RAM or the algorithm cannot run.

$D_{i,j}$ has $\frac{n(n-1)}{2}$ entries; $O(n^2)$ space.

FLOAT32 AT SCALE. $n$=10,000 needs ≈400 MB; $n$=100,000 needs ≈40 GB.

COMPUTE IS THE SOFT WALL. Even when the matrix fits, filling it costs $O(n^2)$. The merge steps then scan and update distances after each of the $n-1$ merges, adding $O(n^2)$ for optimised linkages or $O(n^3)$ for naive implementations.

| **Algorithm** | **Time** | **Notes** |
|---|---|---|
| K-Means (baseline) | $O(n)$ | $K$, $T$, $d$ treated as constants |
| Naive agglomerative | $O(n^3)$ | Recomputes all distances each step |
| Single linkage (SLINK) | $O(n^2)$ | Optimal algorithm for single linkage |
| Average linkage | $O(n^3)$ | Without acceleration |

Table 1: Complexity of hierarchical clustering variants; K-Means is the linear baseline ($K$, $T$, $d$ constant). Naive agglomerative is $O(n^3)$: it rescans the full distance matrix at each of the $n-1$ merge steps.

SINGLE LINKAGE ACHIEVES $O(n^2)$ because its update rule is a simple minimum, as the minimal distance to a merged cluster is the minimum of the distances to its two constituent, deprecated clusters:

$$d(C_i \cup C_j,\, C_k) = \min\big(d(C_i, C_k),\, d(C_j, C_k)\big)$$

SLINK[2] exploits this as each new point requires only a single scan of the current distances; $O(n)$ per update step over $n-1$ merges gives $O(n^2)$ total time.

[2]

SCALABILITY BY COMPRESSION is regained by BIRCH[3], which breaks the $O(n^2)$ memory barrier by compressing data into a compressed tree in one pass, then running agglomeration on the leaf nodes rather than individual points reducing cost. A threshold $T$ controls compression: A small $T$ produces many fine-grained leaves (accurate but larger tree); a large $T$ merges more points per leaf (faster but coarser). Setting $T$ too large can merge points from different true clusters before agglomeration begins.

[3]

DENDROGRAM VOCABULARY borrows from genealogy and graph theory. The dendrogram $T$ is a binary tree with $n$ leaves and $n-1$ internal nodes. Each data point $x_i$ is a leaf $\ell_i$ at height $h=0$. Each internal node $v_k$ records a merge of two children $v_a, v_b$ at height $h_k = D[i^*, j^*]$; the merged cluster is $C_k = C_a \cup C_b$. The root $v_{n-1}$ sits at the highest merge height $h_{n-1}$ and contains all $n$ points. Following the tree upward from any node gives its ancestors; following downward gives its descendants.

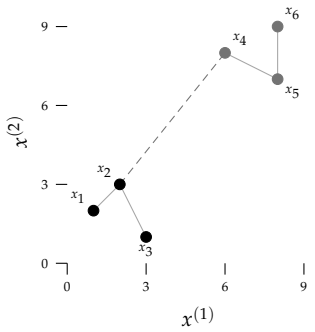
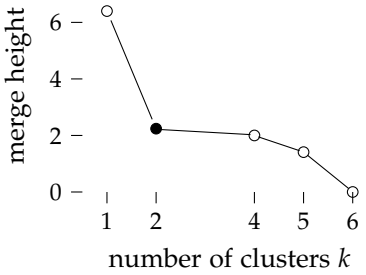



Figure 17: Six points displaying the cut cluster link executed in the dendrograms.

READING A DENDROGRAM requires only one rule: a horizontal cut at height $h$ partitions the data into as many clusters as there are vertical stems crossing that line. Cutting above $h_4 = 6.40$ gives one cluster; cutting between $h_3 = 2.24$ and $h_4 = 6.40$ gives two; cutting between $h_1 = 1.41$ and $h_2 = 2.00$ gives four; cutting below $h_1$ gives $n$ singletons. A large gap between consecutive merge heights is the dendrogram equivalent of the elbow: it suggests a natural number of clusters. The kink at $k=2$ (filled dot) marks the jump from 2.24 to 6.40, confirming two natural clusters. Notice the missing value at $k = 3$.



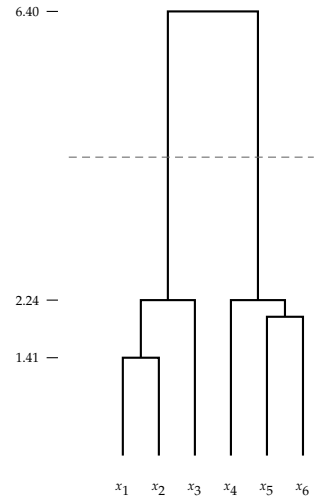




Figure 18: Single-linkage dendrogram on true scale. Heights are Euclidean distances. The jump from 2.24 to 6.40 (dashed cut) yields $C_1 = \{x_1, x_2, x_3\}$, $C_2 = \{x_4, x_5, x_6\}$.

WARD'S DENDROGRAM for the same six points is shown alongside. Heights are no longer Euclidean distances but variance increases: each merge height records the rise in total within-cluster variance caused by that merge.

COMPARE THE GAPS. Single linkage gave a gap from 2.24 to 6.40 (ratio $\approx 2.9\times$). Ward's gives a gap from 3.00 to 96.6 (ratio $\approx 32\times$). Ward's tends to amplify the gap signal, making cluster number selection more pronounced.

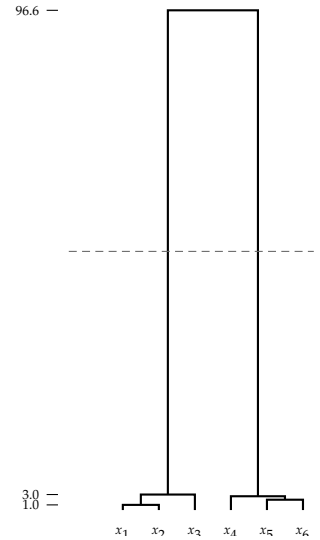



Figure 19: Ward's dendrogram on true scale (Δvar). Four merges pack below 3.0; the final merge towers at 96.6. The gap dwarfs the one in the single-linkage dendrogram above.

How do we infer new data points? The dendrogram is a static structure built on the original data; it does not directly provide a way to assign new points to clusters. But, given a cut of the dendrogram into $K$ clusters, we can assign a new point $x_{\text{new}}$ to the cluster whose centroid is closest under the same distance metric used for clustering. Or we assign it to a cluster based on the closest leaf or a k-nearest neighbor approach.

*Examples & Exercises*

GIVEN THREE POINTS $x_1$=(1, 1), $x_2$=(3, 1), $x_3$=(8, 7), run single-linkage agglomerative clustering step by step. These are the same points you clustered with K-Means already, but this time there is no centroid initialization, no random seed, no choice of $K$. Your tasks: compute the initial pairwise distance table, identify the minimum pair at each step, record each merge, update distances using single linkage, and construct the dendrogram.

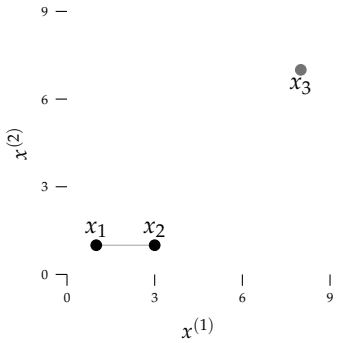



Figure 20: The same three points from the K-Means exercise. No centroids this time; the algorithm needs only pairwise distances.

REUSE BY DESIGN. Working on the same three points lets you compare the two algorithms directly: K-Means needed an initialization and an iterative loop; hierarchical clustering needs neither.

INITIAL DISTANCES. For each pair $(x_i, x_j)$ compute the euclidean distance. With $n$=3 there are only $\binom{3}{2}$=3 pairs. Here is the full calculation for $d(x_1, x_2)$:

$$\begin{aligned} d(x_1, x_2) &= \sqrt{(3-1)^2 + (1-1)^2} \\ &= \sqrt{4+0} \\ &= 2 \end{aligned}$$

Compute the remaining two distances yourself, then verify against the table below. We use symmetries to reduce compute, so only the upper triangle of the distance matrix is filled; the diagonal is empty since $d(x_i, x_i) = 0$ is trivial and not needed for merges.

| | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| $x_1$ | – | **2.00** | 8.49 |
| $x_2$ | – | – | 7.81 |
| $x_3$ | – | – | – |

Table 2: Initial pairwise Euclidean distances. The global minimum $d(x_1, x_2)$=2 (bold) triggers the first merge.

FIRST MERGE. The minimum entry is $d(x_1, x_2)$=2. Merge $x_1$ and $x_2$ into cluster $\{x_1, x_2\}$ at height 2. Update the distance to the remaining point using single linkage. Remember that you do not have to recompute the full distance matrix; only the distances to the new cluster $\{x_1, x_2\}$, repurposing the already computed distances to $x_1$ and $x_2$ and solving for the minimum:

$$\begin{aligned} d(\{x_1, x_2\},\, x_3) &= \min(d(x_1, x_3),\, d(x_2, x_3)) \\ &= \min(8.49,\, 7.81) \\ &= 7.81 \end{aligned}$$

SECOND MERGE. Only two active clusters remain: $\{x_1, x_2\}$ and $\{x_3\}$. Their distance is 7.81, so they merge at height 7.81. The dendrogram is complete.

| Step | Clusters merged | Height | Updated distances (single linkage) |
|---|---|---|---|
| 1 | $x_1,\ x_2$ | 2.00 | $d(\{x_1, x_2\},\ x_3)$=7.81 |
| 2 | $\{x_1, x_2\},\ x_3$ | 7.81 | Final merge; algorithm complete |

Table 3: Single-linkage merge history for $x_1, x_2, x_3$. The gap from 2.00 to 7.81 confirms two natural groups, matching the K-Means result from Chapter 2.

GUARANTEED TERMINATION. How many merges did you just perform? Could there have been more? Fewer? Why?

EACH MERGE REMOVES TWO CLUSTERS from the active set and adds one, reducing |active| by exactly one. Starting from $n$ singletons, the algorithm reaches |active|=1 after exactly $n-1$ steps.

- For $n$=3: exactly 2 merges.
- For $n$=1,000: exactly 999 merges.

This also means the dendrogram always has $n$ leaves and $n-1$ internal nodes. Verify this on your two-merge trace: 3 leaves, 2 internal nodes, one complete binary tree $T$.

CONTRAST WITH K-MEANS. K-Means has no guaranteed iteration count. It can converge in 2 iterations or 200, depending on initialization. Hierarchical clustering always finishes in $n-1$ steps, with no restarts and no randomness.

READING THE DENDROGRAM. Sketch the dendrogram for the three-point trace. It has two horizontal bars: one at height 2.00 (merging $x_1, x_2$) and one at height 7.81 (merging everything). Now answer the following:

- Cut at $h$=5. How many clusters do you get? Which points are in each?
- Cut at $h$=1. How many clusters? What does this tell you?
- What is the largest gap between consecutive merge heights? What $K$ does it suggest?
- How does the spread of the nodes in the dendrogram relate to an elbow plot over $K$?

LEAF ORDERING TRAP. A dendrogram with $n$ leaves has $2^{n-1}$ valid leaf orderings: at each internal node you can swap left and right children without changing the hierarchy. Two points adjacent in the plot are not necessarily similar; only the merge height tells you when they joined.

THE GAP is $7.81 - 2.00 = 5.81$: the final merge costs nearly four times as much as the first. A cut anywhere in this gap yields $K$=2 with $C_1$=$\{x_1, x_2\}$, $C_2$=$\{x_3\}$, exactly the partition K-Means found.

LINKAGE COMPARISON WITH CHAINING. Add a bridge point $x_4{=}(5,4)$ to the three points. This point sits between the two natural groups in a region where no cluster really lives.

Compute the initial $\binom{4}{2}{=}6$ pairwise distances. Then trace single linkage to completion. Based on our previous trace, reuse the distances you already calculated, prior to the merge of $x_1, x_2$ at height 2.00 and calculate the missing distances to $x_4$. You should find the second merge of $\{x_1, x_2\}, x_4$ at height $\approx 3.61$. Finally merge $\{x_1, x_2, x_4\}, x_3$ at height $\approx 4.24$

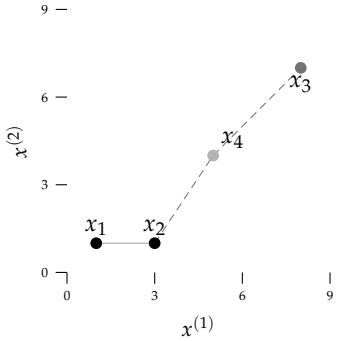



Figure 21: Bridge point $x_4$ in the gap between the two natural groups. Dashed lines show the chain that single linkage will follow.

THE BRIDGE POINT chains the two groups together through nearest-neighbour links: $x_2 \to x_4 \to x_3$. What happens with our gap signal? The first merge is still $x_1, x_2$ at height 2.00, but the second merge is now $\{x_1, x_2\}, x_4$ at height 3.61 deluting our signal, and the final merge is $\{x_1, x_2, x_4\}, x_3$ at height 4.24.

NOW REPEAT WITH COMPLETE LINKAGE. Replace each min in the update rule with max. After step 1 (still $x_1, x_2$ at height 2.00), the updated distance from $\{x_1, x_2\}$ to $x_4$ under complete linkage becomes:

$$\begin{aligned} d_{\text{comp}}(\{x_1, x_2\},\ x_4) &= \max(d(x_1, x_4),\ d(x_2, x_4)) \\ &= \max(5.00,\ 3.61) \\ &= 5.00 \end{aligned}$$

Check whether $x_4$ still merges with $\{x_1, x_2\}$ next, or whether a different pair is closer. Trace the remaining merges and compare the complete-linkage dendrogram to the single-linkage one. Which linkage is more vulnerable to the bridge point? Why?

WHY SINGLE LINKAGE IS FAST. Look back at the distance update you performed in the single-linkage trace. When $C_i$ and $C_j$ merge into $C_m$, the new distance to any remaining cluster $C_k$ is:

$$D_{\text{single}}(C_m,\, C_k) = \min\big(D(C_i,\, C_k),\ D(C_j,\, C_k)\big)$$

One comparison, two table lookups, no access to the original data points. The update depends only on values already in the distance matrix. This means the entire algorithm can run on the $n \times n$ matrix alone; once initialized, the raw data is never touched again.

SLINK exploits exactly this property. Because single linkage only needs nearest-neighbour information, the merge sequence is equivalent to a minimum spanning tree. Prim's algorithm builds that tree in $O(n^2)$, matching SLINK's optimal complexity.

NOW COMPARE WITH WARD'S UPDATE. Recall that Ward's distance measures the increase in total within-cluster variance caused by a merge. For two clusters $C_a$ and $C_b$ with centroids $\mu_a, \mu_b$ and sizes $n_a, n_b$:

$$D_{\text{Ward}}(C_a, C_b) = \frac{n_a \cdot n_b}{n_a + n_b} \|\mu_a - \mu_b\|^2$$

To update the distance after a merge, you must recompute the centroid of the merged cluster and then apply the definition again. This means going back to the raw data, unlike single linkage which never looks beyond the distance matrix.

SHORTCUT. The Lance-Williams formula expresses Ward's update purely in terms of existing table entries and cluster sizes, avoiding the detour through centroids. The merge loop still runs in $O(n^2 \log n)$ at best and $O(n^3)$ in the naive implementation, compared to $O(n^2)$ for single linkage.

VERIFY THE DIFFERENCE. For the three-point trace, compute the Ward distance update for $D_{\text{Ward}}(\{x_1, x_2\}, x_3)$ from the definition and confirm that it gives a different value than the single-linkage update. Ward's initial distance between two singletons reduces to $\frac{1}{2}\|x_i - x_j\|^2$. The first merge is still $x_1, x_2$ (at Ward height $\frac{1}{2} \cdot 4 = 2.00$). After merging, the centroid of $\{x_1, x_2\}$ is $\mu_{12} = (2, 1)$. Now apply the definition to compute the distance to the remaining singleton $x_3$:

$$\begin{aligned}
D_{\text{Ward}}(\{x_1, x_2\}, x_3) &= \frac{n_{12} \cdot n_3}{n_{12} + n_3} \|\mu_{12} - x_3\|^2 \\
&= \frac{2 \cdot 1}{2 + 1} \left\| \begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} 8 \\ 7 \end{pmatrix} \right\|^2 \\
&= \frac{2}{3} \left\| \begin{pmatrix} -6 \\ -6 \end{pmatrix} \right\|^2 \\
&= \frac{2}{3}\left((-6)^2 + (-6)^2\right) \\
&= \frac{2}{3}(36 + 36) \\
&= \frac{2}{3} \cdot 72 \\
&= 48.00
\end{aligned}$$

Single linkage gave 7.81 (a Euclidean distance); Ward's gives 48.00 (a variance increase). The numbers are not comparable across linkage criteria, but within each criterion they determine the merge order.

BACK-OF-ENVELOPE: SCALABILITY. The distance matrix stores $\binom{n}{2}$ entries. For $n$=50,000 with 64-bit floats, that is $\binom{50000}{2} \times 8 \approx 9.3$ GB. For $n$=100,000: roughly 37 GB.

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

BACK AT THE COFFEETABLE. Same 1 797 coffee aroma fingerprints, ten brewing methods, two coordinates. Compute the pairwise distance matrix, implement single, complete, and average linkage from

scratch, then run all four methods and compare the truncated dendrograms. Cut each tree at $K=3$, colour the scatter plot, and compare with $K$-Means at $K=3$.

WHAT TO OBSERVE. Single linkage chains into one dominant cluster. Complete and Ward yield balanced partitions; the $K$-Means result most closely resembles Ward's, because both favour compact, spherical clusters. One hierarchical run gives you every $K$ from 1 to $n$.

## SELF-REFLECTION AND RECAP

SELF-REFLECTION questions to guide your thinking:

- What does the merge height in a dendrogram represent, and why does a large gap between consecutive heights signal a natural number of clusters?
- Why does single linkage produce chaining while complete linkage does not? What property of each formula causes this difference?
- In what situations would you prefer Ward's linkage over single or complete linkage? When would Ward's fail?
- You compute a dendrogram and find that the top-level merge height is only marginally larger than the second-to-top merge height. What does this tell you about the cluster structure?
- What is the computational bottleneck of hierarchical clustering, and what approximate methods exist to overcome it at scale?
- Why is a bad early merge more damaging in hierarchical clustering than a bad initial assignment in K-Means? What mechanism does K-Means have that the dendrogram lacks?

RECAP of Key Concepts:

- Agglomerative clustering greedily merges the two closest clusters at each step, producing a dendrogram that encodes all flat partitions from $K=1$ to $K=n$ in a single structure
- The algorithm terminates in exactly $n-1$ merges; no initialization, no restarts, no convergence check
- Ward's linkage minimises the same within-cluster variance as K-Means; a single Ward dendrogram encodes all partitions from $K=1$ to $K=n$, each equivalent to a K-Means run for that $K$ but without

iterative re-assignment, so Ward's inherits the spherical bias and cannot correct it

- Single linkage detects elongated clusters but suffers from chaining; complete and Ward's linkage produce compact clusters and are more robust to sparse bridge points
- Single linkage is uniquely fast ($O(n^2)$ via SLINK) because its update rule reduces to a single min over existing table entries, equivalent to a minimum spanning tree
- A large gap between consecutive merge heights is the dendrogram signal for a natural number of clusters, analogous to the elbow in K-Means inertia curves
- Leaf adjacency in a dendrogram does not imply similarity; only the merge height matters

THE ROOT OF HIERARCHICAL FAILURE. Every linkage criterion measures distance between points or boundaries locally. No criterion has a concept of sparsity or of low-density regions. In the next Chapter we will introduce and leverage the concept of density. Starting from the single question that hierarchical clustering cannot answer; what are the characteristics of a space or neighbourhood?

TEASER How can we define a cluster without any centroid, any linkage criterion, or any choice of $K$, label outliers as noise automatically, and still recover clusters of arbitrary shape?

FEEDBACK