Recovering the Apresjan Hierarchy Using Linkage-Based Clustering

Maximilien Dreveton[*]
LAMA, Université Gustave Eiffel
Marne-la-Vallée, France
maximilien.dreveton@univ-eiffel.fr

Daichi Kuroda[*]
EPFL
Lausanne, Switzerland
daichi.kuroda@epfl.ch

Matthias Grossglauser
EPFL
Lausanne, Switzerland
matthias.grossglauser@epfl.ch

Patrick Thiran
EPFL
Lausanne, Switzerland
patrick.thiran@epfl.ch

Abstract

Hierarchical clustering seeks to uncover nested structure in data by constructing a tree of clusters, whose deeper levels reveal increasingly fine-grained relationships. However, most hierarchical clustering methods always return a hierarchy, even when the data contain no meaningful hierarchical structure. Moreover, agglomerative linkage algorithms are highly sensitive to the choice of linkage function. To address these drawbacks, we study the Apresjan hierarchy, which consists of all clusters satisfying a strict separation condition. This hierarchy is not necessarily binary, and collapses to a star tree when no nontrivial cluster satisfies this separation condition. Our main contribution is to characterize when the Apresjan hierarchy can be recovered from agglomerative linkage algorithms. We propose a two-step procedure that first constructs a binary tree using a linkage method and then prunes all clusters that violate the separation condition. We give necessary and sufficient conditions on the linkage function under which this procedure exactly recovers the Apresjan hierarchy. Therefore, all linkage rules satisfying these conditions yield the same pruned hierarchy. In particular, single, complete, and average linkage satisfy the conditions, whereas Ward's linkage does not.

1 Introduction

Hierarchical clustering is the task of inferring a tree of clusters, where deeper levels of the hierarchy uncover finer-grained structures. Unlike flat clustering, which partitions the dataset into k groups, hierarchical clustering identifies nested clusters and captures their containment relationships. A hierarchy is represented by a rooted tree whose leaves correspond to data points, and internal vertices to the merge of smaller clusters into larger overarching groups.

Hierarchical clustering algorithms typically fall into two main categories: divisive (*top-down*) and agglomerative (*bottom-up*). Top-down algorithms (such as bisection k-means or recursive sparsest cut) construct the hierarchy by recursively splitting the data into two parts, working from the root downward (Caliński and Harabasz, 1974; Rousseeuw and Kaufman, 2009). In contrast, bottom-up algorithms, such as linkage, iteratively merge the two most similar clusters and recompute the dissimilarity between the newly formed cluster with the non-merged ones, thus building the hierarchy from the bottom upward (Florek et al., 1951; Johnson, 1967; Campello et al., 2015; Miyamoto, 2022).

[*]Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Two key advantages of hierarchical clustering are first that it does not require specifying the number of clusters in advance, which is a significant benefit when the true structure of the data is unknown, and second that it provides the user with a finer representation of the data than flat clustering.

Nonetheless, these hierarchical clustering algorithms have a major drawback: they always return a hierarchy, even when the data does not exhibit one. This can, in particular, lead to inferring hierarchical structures that do not exist. Moreover, the outcome of agglomerative algorithms is highly sensitive to the choice of linkage criterion, which defines how cluster dissimilarities are computed. Indeed, at each iteration, the dissimilarity between the newly formed cluster and the remaining ones can be defined in several ways, such as the minimum (single linkage), maximum (complete linkage), or average (average linkage) pairwise distance between their elements. Different linkage functions can yield drastically different trees on the *same* dataset, with no principled way to determine which is more appropriate. Together, these issues raise fundamental questions about the robustness and interpretability of traditional hierarchical clustering.

A starting point is the well-known correspondence between ultrametrics and dendrograms (Johnson, 1967; Hartigan, 1967; Benzécri et al., 1984; Rammal et al., 1986; Carlsson et al., 2010; Cohen-Addad et al., 2019). Given an ultrametric d (that is, a distance verifying $d(x, y) \leq \max\{d(x, z), d(y, z)\}$, a stronger version of the triangular inequality) one can associate a rooted tree in which the value $d(x, y)$ records the scale at which the points x and y first merge in the hierarchy. This correspondence provides a natural conceptual foundation for hierarchical clustering algorithms, whose goal is to recover the dendrogram associated with an ultrametric input.

However, only considering ultrametrics is restrictive: in particular, the stronger version of the triangular inequality imposes that every triangle must be isosceles, with its two largest side lengths equal.[2] This restriction suggests a natural direction for generalization: beyond the ultrametric setting, one may seek a broader notion of hierarchical structure that preserves the interpretability of hierarchies while accommodating more general dissimilarity data.

We pursue this direction by building on a classical notion of clusters that are strictly separated from their complement, studied under various names such as *nice clusters*, *strictly separated clusters*, and *K-clumps* (Ackerman and Dasgupta, 2014; Diatta and Fichet, 1994; Balcan et al., 2008; Bryant and Berry, 2001). In this paper, we call such clusters *valid clusters*. More precisely, given a dissimilarity, a cluster is *valid* if any two of its elements are closer to each other than either of them is to any element outside the cluster. Validity is a restrictive and local property, and it is simple to check whether a cluster is valid or not. In particular, it is natural to suppose that a valid cluster is a meaningful one, although the converse may not be (that is, there are settings where the validity condition is too strong, so that clusters meaningful for a given application may not all be valid).

The collection of all valid clusters is called the *Apresjan hierarchy* (Apresjan, 1966; Bertrand and Diatta, 2025). In particular, every hierarchy composed solely of valid clusters is a sub-hierarchy of the Apresjan hierarchy. Moreover, the Apresjan hierarchy need not be binary, and degenerates to a star tree (in which all elements connect directly to the root) when no nontrivial cluster is valid, which can indicate that the data exhibit no meaningful hierarchical structure. Finally, in the ultrametric case, the Apresjan hierarchy is equal to the usual dendrogram hierarchy (see Lemma 8 in Appendix A.1).

To construct a hierarchy consisting of only valid clusters, we consider a two-step algorithm. In the first step, we build a binary tree using a linkage method. Because the resulting tree may contain invalid clusters, the second step prunes it by removing the internal vertices that violate validity. Unlike traditional linkage methods, this two-step bottom-up procedure (i) does not necessarily yield a binary tree, and (ii) defaults to a star tree when no nontrivial cluster is valid. Hence, pruning the linkage tree preserves all the valid clusters of a hierarchy and discards all the invalid ones.

A result of Balcan et al. (2008) shows that the hierarchy returned by single linkage contains all valid clusters, although it may also contain additional clusters that are not valid. Therefore, pruning the single-linkage hierarchy by removing the clusters that violate the validity condition returns exactly the Apresjan hierarchy. In other words, the two-step procedure, when applied with single linkage, precisely returns the hierarchy consisting only of valid clusters. This naturally raises the question:

[2]To see that, let $a = d(x, y)$, $b = d(x, z)$, and $c = d(y, z)$. Assume without loss of generality that $a \geq b \geq c$. Then $d(x, y) \leq \max\{d(x, z), d(y, z)\}$ imposes $a \leq \max\{b, c\} = b$. Because $a \geq b$, this enforces $a = b$, and the two largest distances are equal.

Which linkage methods produce a hierarchy containing all valid clusters, and therefore recover the Apresjan hierarchy after pruning?

We call such linkage rules *certified*: on every input, the tree they return contains every valid cluster. So all certified linkage rules yield the same pruned hierarchy, which is the Apresjan hierarchy, regardless of their specific merging criteria. We answer the question above by establishing necessary and sufficient conditions on the linkage update rule for it to be certified. In particular, we show that classical linkage rules such as single, complete, and average linkage are certified, whereas Ward's linkage is not. This highlights a sharp conceptual distinction between these methods, with important practical implications for real-world applications.

Indeed, separating certified from non-certified linkage is useful in three ways. First, by definition, the hierarchy obtained by a certified linkage algorithm is, after pruning, exactly the Apresjan hierarchy, ensuring direct interpretability; in particular, obtaining a star tree after pruning can indicate the absence of a clear hierarchical structure. Second, because the *unpruned* tree obtained from a certified linkage contains all the valid clusters (together with some additional non-valid clusters) and is binary, this tree can be used for any downstream procedure that requires a binary tree (for example, coarse-to-fine search, hierarchical indexing, or tree-based dynamic programming). Third, we also prove in Proposition 7 that the set of certified linkage rules is closed under convex combinations. Hence, for a given application, an optimal linkage rule can be learned from a combination of certified linkage rules, and, whatever the learned coefficients, every valid cluster will belong to the hierarchy obtained.

Paper organization. Section 2 introduces the main definitions and notation concerning hierarchies, valid clusters, and the Apresjan hierarchy. Section 3 establishes the necessary and sufficient conditions under which a linkage-based pruning algorithm recovers the Apresjan hierarchy. Section 4 illustrates these results on real data, and Section 5 concludes the paper. Finally, we defer in the Appendix the discussion of related work and position our contributions within the existing literature (Section A) as well as the proofs (Sections B and C).

2 Preliminaries and Notations

2.1 Hierarchies and Trees

Throughout the paper, k is an integer and $\mathcal{X} = \{x_1, \cdots, x_k\}$ is a discrete set of k items. We denote $[k] = \{1, \cdots, k\}$, and let $\mathcal{P}(\mathcal{X})$ be the powerset of $\mathcal{X}$.

Denote by $\mathcal{T}(\mathcal{X})$ the set of rooted trees whose leaves are $\{x_1\}, \cdots, \{x_k\}$ and whose root is $\mathcal{X}$. For convenience, an element of $\mathcal{T}(\mathcal{X})$ is represented by a set of sets; it will be made precise in the following definition. We provide several examples in Figure 1, for illustration purposes.

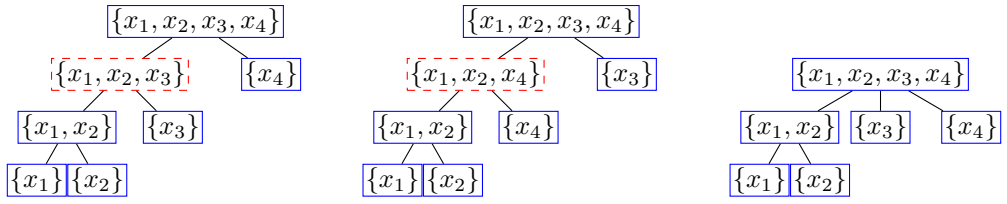
Definition 1. *A tree T on a finite set $\mathcal{X} = \{x_1, \cdots, x_k\}$ is a collection of nonempty subsets $T \subseteq \mathcal{P}(\mathcal{X}) \backslash \{\emptyset\}$ such that:*

1. ***Laminarity**: for all $u, v \in T$, we have $u \cap v \in \{\emptyset, u, v\}$;*
2. ***Root and leaves**: $\mathcal{X} \in T$ and $\{x_i\} \in T$ for all $i \in [k]$.*

Each $t \in T$ is called a vertex of the tree $T \in \mathcal{T}(\mathcal{X})$. Condition 1 (laminarity) ensures that any two vertices are either disjoint or nested, and the inclusion relation among them defines the ancestor–descendant structure of the tree. For example, if $t_1 \subseteq t_2$ and there is no t_3 with $t_1 \subseteq t_3 \subseteq t_2$, then t_2 is the parent of t_1. More generally, every non-leaf vertex t of a tree T is the union of at least two subsets that are the children of t. Condition 2 additionally requires that the root of $T \in \mathcal{T}(\mathcal{X})$ must be $\mathcal{X}$, and that the leaves of T are $\{x_1\}, \cdots, \{x_k\}$.

Moreover, any tree $T \in \mathcal{T}(\mathcal{X})$ defines a hierarchy among its leaves $\{x_1\}, \cdots, \{x_k\}$: from the bottom up, leaves of T merge into branches, and these branches merge further up into larger branches until they reach the root. Each vertex t of a tree is a cluster, which may or may not be valid (albeit the root $\{x_1, \cdots, x_k\}$ and the leaves $\{x_1\}, \cdots, \{x_k\}$ of any tree are always valid). As an example, we provide in Figure 1 three trees T_1, T_2 and T_3 belonging to $\mathcal{T}(\{x_1, \cdots, x_4\})$.

Finally, hierarchies can be compared by introducing the following partial order.



(a) T_1 (b) T_2 (c) T_3

Figure 1: Three trees T_1, T_2 and T_3 belonging to $\mathcal{T}(\{x_1, x_2, x_3, x_4\})$.
(a) $T_1 = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_1, x_2\}, \{x_1, x_2, x_3\}, \{x_1, x_2, x_3, x_4\}\}$,
(b) $T_2 = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_1, x_2\}, \{x_1, x_2, x_4\}, \{x_1, x_2, x_3, x_4\}\}$,
(c) $T_3 = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_1, x_2\}, \{x_1, x_2, x_3, x_4\}\}$.
We have $T_3 \subsetneq T_1$ and $T_3 \subsetneq T_2$, whereas T_1 and T_2 are incomparable. For the dissimilarity of Example 1, T_1 is the hierarchy obtained by single-linkage, T_2 the one obtained by complete- and average-linkage, and T_3 is the Apresjan hierarchy T_*. Dashed red boxes indicate invalid clusters, which are removed by pruning (Section 3.2).

Definition 2 (Partial order on $\mathcal{T}(\mathcal{X})$)**.** *For two trees $T, T' \in \mathcal{T}(\mathcal{X})$, we write*

$$T \subseteq T' \quad \Longleftrightarrow \quad \forall\, t \in T \colon t \in T'.$$

In words, $T \subseteq T'$ means that every cluster contained in T also appears in T'.

The binary relation $\subseteq$ is reflexive, antisymmetric, and transitive, and thus defines an order on the set of trees $\mathcal{T}(\mathcal{X})$. In the example of Figure 1, we have $T_3 \subseteq T_1$, and we say that T_1 *contains* T_3. Because neither $T_1 \subseteq T_2$ nor $T_2 \subseteq T_1$ holds, the relationship $\subseteq$ is only a *partial order*. Finally, we write $T \subsetneq T'$ if T' contains T but $T' \neq T$.

This order $\subseteq$ naturally allows to compare hierarchies according to the refinement of their cluster families. Intuitively, if $T \subseteq T'$, then T' represents a *finer* hierarchical structure, as it preserves all clusters of T while potentially adding new intermediate groupings.

2.2 Valid Clusters and Apresjan Hierarchy

Consider a function $d \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ that measures the dissimilarity between pairs of items in the finite set $\mathcal{X} = \{x_1, \ldots, x_k\}$. We only assume that (i) d is symmetric, *i.e.* $d(x, y) = d(y, x)$ for all $x, y \in \mathcal{X}$, and that (ii) $d(x, y) = 0$ if and only if $x = y$.

We refer to any subset of $\mathcal{X}$ as a *cluster*. A cluster is said to be *valid* with respect to the dissimilarity function d if the dissimilarity between any two items inside the cluster is strictly smaller than their dissimilarity with any item outside the cluster. In particular, all singletons $\{x_1\}, \ldots, \{x_k\}$ are valid clusters, as well as the full set $\mathcal{X}$ itself.

Definition 3 (Valid Cluster)**.** *A non-empty subset $C \subseteq \mathcal{X}$ is a* valid cluster *with respect to d if*

$$\forall x, y \in C,\ \forall z \in \mathcal{X} \setminus C \colon \quad d(x, y) < d(x, z).$$

By convention, we assume that $\mathcal{X}$ and every singleton cluster are valid.

At first glance, the validity requirement in Definition 3 may appear rather strong, as it enforces a strict separation between intra-cluster and inter-cluster dissimilarities. However, this condition captures a natural and intuitive notion of what a cluster should represent: items within a cluster are closer to each other than to any item outside the cluster. In fact, this notion of validity has appeared in prior work under different names: in Balcan et al. (2008) it is referred to as *strict separation*, while in Ackerman and Dasgupta (2014) it is called *nice clustering*. A more detailed discussion of these connections can be found in Section A.2. Importantly, we do not impose any structural assumptions on the dissimilarity function itself, which allows d to be defined in a highly flexible manner: for instance, via manifold-based dissimilarities, nearest-neighbor graphs, density-based proximity notions as in DBSCAN, kernel constructions, or dissimilarities induced by learned representations.

Definition 4. *A tree $T \in \mathcal{T}(\mathcal{X})$ is a* valid hierarchy *with respect to d if every cluster of T is valid.*

The definition of valid hierarchy imposes that the dissimilarity between two distinct items x and y that are close in the hierarchy defined by T is strictly smaller than the dissimilarity with a third item z located further away in the hierarchy. More precisely, denote by $\mathrm{lca}_T(x, y)$, or simply $\mathrm{lca}(x, y)$, the lowest common ancestor between two leaves x and y of a tree T, a tree T is a valid hierarchy if and only if $d(x, y) < \min\{d(x, z), d(y, z)\}$ whenever $\mathrm{lca}_T(x, y) \subsetneq \mathrm{lca}_T(x, z)$.

The following elementary lemma emphasizes an important structural property: two valid clusters cannot partially overlap, and thus any collection of valid clusters forms a laminar family.

Lemma 1 (Validity implies laminarity)**.** *Let $C_1, C_2 \subseteq \mathcal{X}$ be two valid clusters with respect to d. Then $C_1 \cap C_2 \in \{\emptyset, C_1, C_2\}$.*

Proof. Assume, for contradiction, that C_1 and C_2 overlap but that neither one contains the other. Then there exist $a \in C_1 \cap C_2$, $b \in C_1 \setminus C_2$, and $c \in C_2 \setminus C_1$. Because C_1 is valid, we have $d(a, b) < d(a, c)$, while the validity of C_2 yields $d(a, c) < d(a, b)$, a contradiction. Hence, the two sets C_1 and C_2 satisfy the laminarity property. □

Because two valid clusters cannot partially overlap, a set of valid clusters that includes the root and leaves is always a hierarchy, and is valid (as the root and leaves are also valid). Therefore, the collection of all valid clusters,

$$T_* := \left\{ C \subseteq \mathcal{X} : \min_{\substack{x,y \in C, \\ z \in \mathcal{X} \setminus C}} d(x, z) - d(x, y) \; > \; 0 \right\}, \tag{1}$$

is a valid hierarchy, called the *Apresjan hierarchy* (Apresjan, 1966).

Example 1. *Consider $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$ and the dissimilarity d represented by the matrix*

$$d = \begin{pmatrix} 0 & 1.1 & 2 & 2.2 \\ 1.1 & 0 & 3 & 2.2 \\ 2 & 3 & 0 & 2.5 \\ 2.2 & 2.2 & 2.5 & 0 \end{pmatrix},$$

which is a metric. The cluster $\{x_1, x_2\}$ is valid: $d(x_1, x_2) = 1.1$ is smaller than every dissimilarity from x_1 or x_2 to x_3 or x_4. In contrast, $\{x_1, x_2, x_3\}$ is not valid, since $d(x_2, x_3) = 3 > d(x_2, x_4) = 2.2$, and $\{x_1, x_2, x_4\}$ is not valid, since $d(x_1, x_4) = 2.2 > d(x_1, x_3) = 2$. Similarly, no other nontrivial cluster is valid, hence $T_*(\mathcal{X}, d) = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_1, x_2\}, \mathcal{X}\}$, which is not binary: the root has three children $\{x_1, x_2\}$, $\{x_3\}$ and $\{x_4\}$.

3 Recovering the Apresjan Hierarchy

3.1 Linkage Algorithms

A linkage algorithm starts from the singletons and repeatedly merges the two closest clusters, after which a linkage rule prescribes the dissimilarity between the new cluster and every other cluster. For instance, suppose that the items $\{x_1\}$ and $\{x_2\}$ are merged at the first step. The linkage rule then prescribes how to compute the dissimilarity between the new cluster $\{x_1, x_2\}$ and the remaining clusters $\{x_a\}$ for $a \in \{3, \ldots, k\}$. Many such linkage rules exist, and different rules typically produce different hierarchical structures. Hence, the choice of the rule has a direct impact on the recovered hierarchy.

To capture the impact of the choice of the linkage rule, we study a linkage defined by an update function $f \colon \mathbb{R}^3_{\geq 0} \times \mathbb{N}^3 \to \mathbb{R}$, such that when two clusters t_1 and t_2 are merged to form $t_1 \cup t_2$, the dissimilarity between $t_1 \cup t_2$ and any other cluster t_3 is computed as

$$d(t_1 \cup t_2, t_3) \; = \; f\big(d(t_1, t_2), d(t_2, t_3), d(t_3, t_1), |t_1|, |t_2|, |t_3|\big). \tag{2}$$

Thus, the update depends only on the pairwise dissimilarities among t_1, t_2, t_3 and on their sizes $|t_1|, |t_2|, |t_3|$. Observe that because $t_1 \cup t_2 = t_2 \cup t_1$, the function f must satisfy

$$f\big(d(t_1, t_2), d(t_2, t_3), d(t_3, t_1), |t_1|, |t_2|, |t_3|\big) \; = \; f\big(d(t_2, t_1), d(t_1, t_3), d(t_3, t_2), |t_2|, |t_1|, |t_3|\big).$$

Moreover, because linkage will not merge the (t_1, t_2) if $d(t_1, t_2) > \min_{1 \leq i < j \leq 3} d(t_i, t_j)$, Equation (2) is defined only for $d(t_1, t_2) \leq \min_{i,j} d(t_i, t_j)$. Finally, we will only assume that f is left-continuous with respect to its first coordinate.

Assumption 1. *The linkage update function f is left-continuous with respect to its first argument: for any $(q_1, q_2, q_3, n_1, n_2, n_3) \in \mathbb{R}_+^3 \times \mathbb{N}^3$, we have*

$$\lim_{\delta \to 0+} f(q_1 - \delta, q_2, q_3, n_1, n_2, n_3) \;=\; f(q_1, q_2, q_3, n_1, n_2, n_3).$$

Algorithm 1 describes the linkage algorithm. We note that the update rule (2) is closely related to the classical Lance–Williams formula (Murtagh and Contreras, 2017), which provides a unifying scheme for many hierarchical clustering methods. This formula expresses the new dissimilarity as a linear combination of $d(t_1, t_3)$, $d(t_2, t_3)$, and $d(t_1, t_2)$ with coefficients depending only on the cluster sizes. Our formulation generalizes this formula by allowing f to be an arbitrary function of the same quantities, and thus includes all Lance–Williams linkages as well as potentially broader classes of linkage rules. We refer to Section 3.4 for a more detailed discussion and examples.

Algorithm 1: Linkage

Input: Dissimilarity function $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$, update rule $f\colon \mathbb{R}^3 \times \mathbb{N}^3 \to \mathbb{R}$

Initialize $\hat{T}_{\text{linkage}} = \{\{x_1\}, \ldots, \{x_k\}\}$ and active clusters $\mathcal{C}_{active}^{(0)} = \{\{x_1\}, \ldots, \{x_k\}\}$.

for $m = 1$ **to** $k - 1$ **do**

  Select $t_1, t_2 \in \arg\min_{t_1' \neq t_2' \in \mathcal{C}_{active}^{(m-1)}} d(t_1', t_2')$

  Form $t = t_1 \cup t_2$, update $\hat{T}_{\text{linkage}} \leftarrow \hat{T}_{\text{linkage}} \cup \{t\}$

  Update active clusters: $\mathcal{C}_{active}^{(m)} \leftarrow \left(\mathcal{C}_{active}^{(m-1)} \setminus \{t_1, t_2\}\right) \cup \{t\}$

  For each $t_3 \in \mathcal{C}_{active}^{(m)} \setminus \{t\}$, define $d(t_1 \cup t_2, t_3)$ using (2).

Return: $\hat{T}_{\text{linkage}}$

3.2 Pruned Linkage

Because Algorithm 1 merges items strictly in pairs, it always produces a binary tree. Therefore, if the Apresjan hierarchy T_* is *not* binary, Algorithm 1 cannot directly recover it. More generally, the resulting binary hierarchy $\hat{T}_{\text{linkage}}$ is not guaranteed to be valid and may contain clusters that violate the validity condition given in Definition 3. We address this by trimming the vertices of $\hat{T}_{\text{linkage}}$ that violate the validity condition. This yields a valid hierarchy and is formalized in Algorithm 2.

Algorithm 2: Pruned Linkage

Input: Dissimilarity function $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$, update rule $f\colon \mathbb{R}^3 \times \mathbb{N}^3 \to \mathbb{R}$

Apply Algorithm 1 with input (d, f) to obtain $\hat{T}_{\text{linkage}}$

Denote $\hat{T}_{\text{linkage}}^{-} = \left\{ t \in \hat{T}_{\text{linkage}} \colon \min_{(x,y,z) \in t \times t \times (\mathcal{X} \setminus t)} d(x, z) - d(x, y) \leq 0 \right\}$

Return: $\hat{T} = \hat{T}_{\text{linkage}} \setminus \hat{T}_{\text{linkage}}^{-}$

Lemma 2. *Let $\hat{T}_{\text{linkage}} = \hat{T}_{\text{linkage}}(\mathcal{X}, d, f)$ and $\hat{T} = \hat{T}(\mathcal{X}, d, f)$ be the outputs of Algorithm 1 and Algorithm 2, respectively. Then, $\hat{T}$ is a valid hierarchy, and $\hat{T} = \hat{T}_{\text{linkage}} \cap T_*$.*

Proof. Pruning removes exactly those clusters that are invalid, so $\hat{T}$ contains only the valid clusters of $\hat{T}_{\text{linkage}}$. In particular, $\hat{T}$ contains the root $\mathcal{X}$ and the leaves $\{x_i\}$. The laminarity of the remaining clusters follows from Lemma 1, ensuring that $\hat{T}$ is a tree (and hence is indeed a valid hierarchy). Finally, because by definition T_* contains all the valid clusters, we also have $\hat{T} = \hat{T}_{\text{linkage}} \cap T_*$. □

Remark 1 (Computational cost of pruning)**.** *Pruning does not change the asymptotic cost of linkage. Bryant and Berry (2001) proposed an efficient way to prune so pruning can be computed in $O(k^2)$ time (Bryant and Berry, 2001, Corollary 2.1); see Appendix A.4 for details on the algorithm. Many linkage hierarchies can also be obtained with $O(k^2)$ operations: single linkage via SLINK (Sibson, 1973) or Prim's minimum-spanning-tree algorithm (Gower and Ross, 1969), and complete and*

average linkage via the nearest-neighbor chain (Murtagh, 1983; Müllner, 2011). Hence the whole procedure runs in $O(k^2)$ time for these linkages, and the pruning comes at no additional asymptotic cost. This is optimal: any algorithm taking a dissimilarity matrix as input must read its $k(k-1)/2$ entries, so its running time is $\Omega(k^2)$ (Müllner, 2011).

Example 2 (Example 1, continued)**.** *Consider again the dissimilarities given in Example 1. Single, complete, and average linkage all first merge $\{x_1, x_2\}$, the unique closest pair. The updated dissimilarities $d(\{x_1, x_2\}, x_3)$ are then 2, 3 and 2.5 respectively, while $d(\{x_1, x_2\}, x_4) = 2.2$ and $d(x_3, x_4) = 2.5$ in all three cases. Hence single linkage forms $\{x_1, x_2, x_3\}$, whereas complete and average linkage form $\{x_1, x_2, x_4\}$ (Figure 1). Both size-three clusters are invalid, so pruning removes them, and all three trimmed trees coincide with $T_*(\mathcal{X}, d)$. Every merge choice above is strict: the multifurcation arises from removing an invalid intermediate cluster, not from ties.*

3.3 Certified Linkage Rules

The previous section shows that, as long as f satisfies the very mild Assumption 1, $\hat{T}$ is a valid hierarchy. Because $\hat{T} = \hat{T}_{\text{linkage}} \cap T_*$, this hierarchy $\hat{T}$ will be equal to the Apresjan one T_* if and only if $T_* \subseteq \hat{T}_{\text{linkage}}$. In this section, we precisely characterize which linkages ensure that $T_* \subseteq \hat{T}_{\text{linkage}}$.

Definition 5. *A linkage rule f is* certified *if $T_*(\mathcal{X}, d) \subseteq \hat{T}_{\text{linkage}}(\mathcal{X}, d, f)$ for every input $(\mathcal{X}, d)$.*

In the following, we establish that f is certified if and only if the update rule f extending d by (2) satisfies the following two conditions.

Condition 1 (Monotonicity across merges)**.** *For any dissimilarity function d, the update rule f extends d such that for any subsets $t_1, t_2, t_3, t_4 \subset \mathcal{X}$,*

$$\forall i \in \{1,2\} \colon d(t_i, t_3) < \min\left(d(t_i, t_4), d(t_3, t_4)\right) \;\Rightarrow\; d(t_1 \cup t_2, t_3) < \min\left(d(t_1 \cup t_2, t_4), d(t_3, t_4)\right).$$

Condition 2 (Dominance preservation)**.** *For any dissimilarity function d, the update rule f extends d such that for any subsets $t_1, t_2, t_3, t_4 \subset \mathcal{X}$,*

$$d(t_1, t_2) < \min_{\substack{i \in \{1,2\} \\ j \in \{3,4\}}} d(t_i, t_j) \;\Rightarrow\; d(t_1, t_2) < \min_{i \in \{1,2\}} d(t_i, t_3 \cup t_4).$$

Condition 1 ensures that merging two clusters that are both closer to a third cluster does not increase this dissimilarity. Indeed, this condition ensures that merging clusters t_1 and t_2 preserves an existing smaller dissimilarity to a third cluster t_3 relative to a fourth cluster t_4. Similarly, Condition 2 ensures that close clusters should remain closer to each other than to any newly formed cluster that merges two further clusters (with respect to the focal clusters). Indeed, if two clusters t_1 and t_2 are closer to each other than to t_3 and t_4, then Condition 2 maintains this closeness after t_3 and t_4 are merged.

Lemma 3. *The linkage rule f is certified if and only if f satisfies Conditions 1 and 2.*

Proof. The full proof is lengthy and is given in Appendix B, and we summarize the key steps below. The equivalence splits into two complementary statements.

For sufficiency, assume that f satisfies Conditions 1 and 2. We show by induction on the linkage step that every valid cluster C (*i.e.,* every vertex of T_*) is formed at some point of the iterative merging process. At the initial step, the forest (the incomplete linkage tree that is in construction) consists of singletons (formed by each item), each of which is either contained in C or disjoint from C. By Conditions 1 and 2, the dissimilarity between any two clusters contained in C is strictly smaller than the dissimilarity between any cluster contained in C and any cluster in $\mathcal{X} \backslash C$. This ensures that clusters contained in C always merge with each other, internally in C, before any cluster outside C can become part of a merge. Hence, the algorithm eventually produces C. However, non-valid clusters can be created during the iterative process, and thus $\hat{T}_{\text{linkage}}$ is not necessarily a valid hierarchy.

For necessity, assume f violates one of the conditions. Then, we explicitly build a dissimilarity function d_1 over a subset of items $\mathcal{X}_1$ for which the linkage algorithm omits a valid cluster. This yields a counterexample where the linkage does not contain the Apresjan hierarchy. □

The following theorem follows from Lemmas 2 and 3.

Theorem 4 (Characterization of Linkages Recovering the Apresjan Hierarchy)**.** *Let $\hat{T}(\mathcal{X}, d, f)$ be the tree returned by Algorithm 2 with input $(\mathcal{X}, d, f)$. The following holds:*

$$\forall(\mathcal{X}, d)\colon\ \hat{T}(\mathcal{X}, d, f) \ = \ T_*(\mathcal{X}, d) \iff f \text{ satisfies Conditions 1 and 2.}$$

Balcan et al. (2008, Theorem 2) established that the hierarchy produced by single linkage contains all valid clusters. Lemma 3 generalizes this result by providing necessary and sufficient conditions on the linkage rule that guarantee that the inferred hierarchy contains all the valid clusters. Moreover, although Balcan et al. (2008) employ a hierarchical procedure, their goal is to recover a flat ground-truth valid partition. In contrast, our focus is on recovering the entire hierarchical structure that includes all valid clusters while excluding any invalid ones. The exclusion of invalid clusters by pruning is crucial to obtain Theorem 4.

3.4 Consequences of Theorem 4

Theorem 4 shows that, whenever the linkage update rule f satisfies Conditions 1 and 2, the pruning procedure exactly recovers $T_*(\mathcal{X}, d)$ for any dissimilarity function d. In particular, all certified linkage rules produce the *same* pruned hierarchy, although the unpruned hierarchy may differ greatly. However, any disagreement between the hierarchies obtained from different certified linkage rules is confined to invalid clusters, which pruning removes.

Examples of linkage satisfying Conditions 1 and 2 The formulation (2) together with Assumption 1 subsumes the following four classical linkage rules as special cases:

- *Single linkage:* $f(d_{12}, d_{23}, d_{13}, n_1, n_2, n_3) = \min\{d_{23}, d_{13}\}$.
- *Complete linkage:* $f(d_{12}, d_{23}, d_{13}, n_1, n_2, n_3) = \max\{d_{23}, d_{13}\}$.
- *Unweighted Average linkage:* $f(d_{12}, d_{23}, d_{13}, n_1, n_2, n_3) = \frac{n_1}{n_1+n_2} d_{13} + \frac{n_2}{n_1+n_2} d_{23}$.
- *Weighted Average linkage:* $f(d_{12}, d_{23}, d_{13}, n_1, n_2, n_3) = \frac{d_{13}+d_{23}}{2}$.

The next lemma shows these linkages all satisfy the Conditions 1 and 2.

Lemma 5. *Single, complete, weighted average, and unweighted average linkages satisfy Conditions 1 and 2.*

Examples of linkage violating Conditions 1 or 2 Other widely used linkage methods are Ward's minimum variance (Ward, 1963), Median, and Centroid linkages (Murtagh and Contreras, 2017). The update rule of these linkages can be expressed via the *Lance–Williams formula* (Murtagh and Contreras, 2017):

$$d(t_1 \cup t_2, t_3) \ = \ \eta_1 d(t_1, t_3) + \eta_2 d(t_2, t_3) + \beta d(t_1, t_2) + \gamma |d(t_1, t_3) - d(t_2, t_3)|,$$

where the value of the coefficients $(\eta_1, \eta_2, \beta, \gamma)$ depend on the linkage method, and are given in Table 3 in the Appendix A.3.

The following lemma shows that these linkages fail to satisfy Conditions 1 and 2. These linkages are therefore not certified: after pruning, they can return a hierarchy that does not contain all valid clusters. We provide such explicit numeric examples for Ward linkage in Appendix C.2.

Lemma 6. *Ward linkage does not satisfy Condition 1, while Median and Centroid linkages do not satisfy Condition 2.*

Closure under Convex Combinations of Linkage Functions A natural way to interpolate between linkage methods is to combine their update rules linearly. This approach has been studied, for instance, by Balcan et al. (2020, 2024), who consider learning the coefficients of such combinations from the data in order to improve clustering performance. The next proposition shows that our characterization is stable under this operation: any convex combination of linkage rules satisfying the two conditions again satisfies the two conditions.

Proposition 7. *Let $f_1, \cdots, f_m$ be linkage rules satisfying Conditions 1 and 2. Let $F = \sum_{j=1}^{m} \alpha_j f_j$ be a convex combination, where $\alpha_j \in [0, 1]$ for all j and $\sum_j \alpha_j = 1$. Then F also satisfies Conditions 1 and 2.*

4 Numerical Illustrations

This section illustrates some practical consequences of our theoretical contributions.[3] First, on real data sets the Apresjan hierarchy does not collapse to a star: although the validity condition may appear strong, valid clusters do exist in real data, and pruning removes most, but not all, internal clusters of a linkage hierarchy. Second, the hierarchies obtained by single, complete, and average linkage disagree substantially before pruning, even though each of them contains T_* (Lemma 3).

Metrics. Every binary tree on k leaves has $k-2$ non-root internal clusters. We report the proportion of these that are valid; as all three trimmed trees equal T_*, this fraction does not depend on the choice of linkage. We also report the fraction of non-root internal clusters common to the hierarchies obtained by single, complete, and average linkage before pruning (*shared*). By Lemma 3, every such tree contains T_*, so the shared fraction is always at least the proportion of clusters retained by pruning. Finally, for the data sets that come with ground-truth cluster labels (all real data sets below), *triplet satisfaction* is the percentage of triplets (x, y, z), with x, y in the same ground-truth cluster and z in a different one, for which $d(x, y) < d(x, z)$; in particular, a cluster is valid iff all its triplets are satisfied.

Benchmark datasets. Table 4 reports results on Iris (Fisher, 1936), Wine (Aeberhard and Forina, 1992), Breast Cancer (Wolberg et al., 1993), and Digits (Alpaydin and Kaynak, 1998), using Euclidean and cosine dissimilarities. Pruning retains between 22% and 38% of the internal clusters, so the Apresjan hierarchy does not degenerate to a star, and all pruned hierarchies are non-binary. Before pruning, only 26% to 52% of the internal clusters are shared by the three linkage trees: hence, most of the structure of a linkage tree depends on the linkage, and pruning removes exactly this dependence. With one exception (one Iris class under the cosine dissimilarity), the ground-truth clusters are not valid; nonetheless, between 72% and 95% of the triplet inequalities involved in the validity condition are satisfied.

Dataset	k	Dissimilarity	Triplet satisfaction (%)	Valid (%)	Shared (%)
Iris	149	Euclidean	93.35	46 (31.3)	43.5
		Cosine	95.37	49 (33.3)	38.8
Wine	178	Euclidean	77.46	67 (38.1)	51.7
		Cosine	72.06	57 (32.4)	39.8
Breast Cancer	569	Euclidean	82.00	190 (33.5)	40.2
		Cosine	80.07	166 (29.3)	35.1
Digits	1797	Euclidean	87.86	430 (24.0)	27.7
		Cosine	87.63	403 (22.5)	26.4

Table 1: Results on benchmark datasets. Valid: number (percentage) of the $k-2$ non-root internal clusters that satisfy the validity condition. Shared: percentage of non-root internal clusters common to the unpruned single-, complete-, and average-linkage hierarchies. Triplet satisfaction: percentage of triplets (x, y, z), with x, y in the same ground-truth class and z in a different class, satisfying $d(x, y) < d(x, z)$.

Hierarchical labels. CIFAR-100 (Krizhevsky, 2009) provides a two-level label hierarchy of 100 fine classes nested in 20 coarse classes. We use ten test images per fine class ($k = 1000$) and compare raw-pixel distances with Euclidean and cosine dissimilarities between frozen self-supervised DINOv2 ViT-S/14 features (Oquab et al., 2024). Table 4 reports the results on this dataset. Under raw pixels, only 5.4% of the internal clusters are valid; under DINOv2 features, about a quarter are, and fine classes satisfy 93–94% of their triplet inequalities, against 78–80% for coarse classes.

5 Conclusion

In this work, we studied hierarchical structure through the notion of valid clusters and the hierarchies they induce. The valid cluster condition enforces a strong point-wise separation between intra- and

[3] Code to reproduce all experiments is available at `https://github.com/daichikuroda/Recovering_Apresjan`.

Geometry	Fine triplets (%)	Coarse triplets (%)	Valid (%)	Shared (%)
Raw RGB Euclidean	59.04	54.98	5.4	5.9
DINOv2 cosine	93.87	80.20	25.7	31.8
DINOv2 Euclidean	93.37	78.52	24.7	30.0

Table 2: Results on CIFAR-100 using raw pixels and DINOv2 features. Valid: percentage of the $k-2$ non-root internal clusters that satisfy the validity condition. Shared: percentage of non-root internal clusters common to the unpruned single-, complete-, and average-linkage hierarchies. Fine/Coarse triplets: triplet satisfaction under the fine/coarse ground-truth labels, respectively.

inter-cluster dissimilarities, and therefore captures a stringent but intuitive notion of cluster structure. Moreover, the collection of all valid clusters always forms a hierarchy: the Apresjan hierarchy. This hierarchy may be non-binary and degenerates into a star tree when no nontrivial cluster is valid.

In the ultrametric setting, the Apresjan hierarchy coincides with the standard hierarchy encoded by the dendrogram (see Lemma 8 in Appendix A.1). Thus, valid clusters extend the classical ultrametric–dendrogram correspondence to arbitrary dissimilarity data. In this sense, the Apresjan hierarchy provides a conservative, certified notion of hierarchical structure beyond the ultrametric case: it records exactly those clusters that are unambiguously supported by the dissimilarities. Consequently, it is natural to favor methods whose output contains all valid clusters, since such methods do not discard any of the hierarchical structure that is explicitly present in the data.

This viewpoint also gives an axiomatic interpretation of our results, in the spirit of Cohen-Addad et al. (2019). In their framework, the optimal tree associated with an ultrametric dissimilarity d is required to contain the hierarchy induced by d. Our work extends this type of requirement from ultrametrics to general dissimilarities: rather than asking a method to contain the ultrametric hierarchy, we ask that the hierarchy built by a linkage method should contain all the valid clusters. Because valid clusters satisfy a strict separation property, their recovery is rigid and testable; at the same time, the initial linkage hierarchy may contain additional, non-valid clusters, which can be interpreted as approximate or heuristic structure and removed when one wants a conservative hierarchy. Within this framework, we characterize exactly which linkage methods satisfy the inclusion requirement and which do not.

On the algorithmic side, we showed that the Apresjan hierarchy can be recovered by a simple two-step procedure: first construct a binary hierarchy using a linkage method, then prune the clusters that violate validity. We gave necessary and sufficient conditions on the linkage rule under which this procedure returns the Apresjan hierarchy. In particular, although different linkage rules may produce very different binary dendrograms on the same dataset, these differences disappear after pruning: all certified linkage methods (*i.e.*, those satisfying Conditions 1 and 2) return the same Apresjan hierarchy. Thus, the dependence on the linkage choice comes only from invalid clusters introduced during the agglomerative process, which pruning removes.

More broadly, our results address two fundamental limitations of classical agglomerative hierarchical clustering. First, standard linkage methods without pruning always return a hierarchy, even when the data does not support one, which can lead to over-interpreting spurious structure. Second, their output is often highly sensitive to the linkage criterion. Our framework resolves both issues: validity prevents unsupported clusters from being retained, and pruning removes the linkage dependence at the level of the final hierarchy whenever the linkage satisfies the required conditions.

Limitations and future work. Unlike the ultrametric correspondence, validity requires no assumption on the dissimilarity d, but it is deliberately conservative: it retains only clusters that are unambiguously supported by d. We do not claim that the Apresjan hierarchy contains all meaningful clusters; clusters that are meaningful under other criteria may be invalid, and studying alternative or application-dependent certifications is an interesting open direction. Second, valid clusters with a small validity margin are sensitive to perturbations: only clusters with $\Delta_d(C) > 2\varepsilon$ are guaranteed to survive perturbations of size ε, which motivates the margin hierarchy T_γ (Section A.6). Third, our guarantees are deterministic, and statistical guarantees under a sampling model remain open. Finally, Proposition 7 opens the way to data-driven strategies that learn a linkage rule for a given application while preserving every valid cluster.

References

Ackerman, M. and S. Dasgupta (2014). Incremental clustering: The case for extra clusters. *Advances in Neural Information Processing Systems 27*.

Aeberhard, S. and M. Forina (1992). Wine. UCI Machine Learning Repository, `https://doi.org/10.24432/C5PC7J`.

Alpaydin, E. and C. Kaynak (1998). Optical recognition of handwritten digits. UCI Machine Learning Repository, `https://doi.org/10.24432/C50P49`.

Apresjan, J. D. (1966). An algorithm for constructing clusters from a distance matrix. *Mashinnyi perevod: prikladnaja lingvistika 9*(3-18), 1.

Arias-Castro, E. and E. Coda (2025). An axiomatic definition of hierarchical clustering. *Journal of Machine Learning Research 26*(10), 1–26.

Balcan, M.-F., A. Blum, and S. Vempala (2008). A discriminative framework for clustering via similarity functions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, New York, NY, USA, pp. 671–680. Association for Computing Machinery.

Balcan, M.-F., T. Dick, and M. Lang (2020). Learning to link. In *International Conference on Learning Representation*.

Balcan, M.-F., C. Seiler, and D. Sharma (2024). Accelerating ERM for data-driven algorithm design using output-sensitive techniques. *Advances in Neural Information Processing Systems 37*, 72648–72687.

Benzécri, J.-P. et al. (1984). *L'analyse des données: la taxinomie*, Volume 1. Dunod Paris.

Bertrand, P. and J. Diatta (2025). A note on Apresjan hierarchies and linkage-based hierarchical clustering.

Blundell, C., Y. W. Teh, and K. A. Heller (2010). Bayesian rose trees. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, UAI 2010*.

Bryant, D. and V. Berry (2001). A structured family of clustering and tree construction methods. *Advances in Applied Mathematics 27*(4), 705–732.

Caliński, T. and J. Harabasz (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods 3*(1), 1–27.

Campello, R. J., D. Moulavi, and J. Sander (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pp. 160–172. Springer.

Campello, R. J., D. Moulavi, A. Zimek, and J. Sander (2015). Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD) 10*(1), 1–51.

Carlsson, G. E., F. Mémoli, et al. (2010). Characterization, stability and convergence of hierarchical clustering methods. *Journal of Machine Learning Research 11*(Apr), 1425–1470.

Cohen-Addad, V., V. Kanade, F. Mallmann-Trenn, and C. Mathieu (2019). Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM (JACM) 66*(4), 1–42.

Dasgupta, S. (2016). A cost function for similarity-based hierarchical clustering. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, New York, NY, USA, pp. 118–127. Association for Computing Machinery.

Diatta, J. and B. Fichet (1994). From apresjan hierarchies and bandelt-dress weak hierarchies to quasi-hierarchies. In *New approaches in classification and data analysis*, pp. 111–118. Springer.

Dreveton, M., D. Kuroda, M. Grossglauser, and P. Thiran (2026). When does bottom-up beat top-down in hierarchical community detection? *Journal of the American Statistical Association 0*(0), 1–12.

Fisher, R. A. (1936). Iris. UCI Machine Learning Repository, https://doi.org/10.24432/C56C76.

Florek, K., J. Łukaszewicz, J. Perkal, H. Steinhaus, and S. Zubrzycki (1951). Sur la liaison et la division des points d'un ensemble fini. *Colloquium Mathematicum 2*(3-4), 282–285.

Ghahramani, Z., M. Jordan, and R. P. Adams (2010). Tree-structured stick breaking for hierarchical data. *Advances in Neural Information Processing Systems 23*.

Gower, J. C. and G. J. Ross (1969). Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics) 18*(1), 54–64.

Gray, A., A. Modell, P. Rubin-Delanchy, and N. Whiteley (2023). Hierarchical clustering with dot products recovers hidden tree structure. *Advances in Neural Information Processing Systems 36*, 31752–31779.

Hartigan, J. A. (1967). Representation of similarity matrices by trees. *Journal of the American Statistical Association 62*(320), 1140–1158.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika 32*(3), 241–254.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Li, T., L. Lei, S. Bhattacharyya, K. Van den Berge, P. Sarkar, P. J. Bickel, and E. Levina (2022). Hierarchical community detection by recursive partitioning. *Journal of the American Statistical Association 117*(538), 951–968.

Miyamoto, S. (2022). *Theory of Agglomerative Hierarchical Clustering*, Volume 15. Springer Singapore.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

Murtagh, F. (1983). A survey of recent advances in hierarchical clustering algorithms. *The computer journal 26*(4), 354–359.

Murtagh, F. and P. Contreras (2017). Algorithms for hierarchical clustering: an overview, II. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 7*(6), e1219.

Oquab, M., T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski (2024). Dinov2: Learning robust visual features without supervision.

Pan, Y., B. Fan, P. Long, and F. Zheng (2025). An information-theoretic perspective of hierarchical clustering on graphs. In *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, Volume 286 of *Proceedings of Machine Learning Research*, pp. 3322–3345. PMLR.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research 12*, 2825–2830.

Rammal, R., G. Toulouse, and M. A. Virasoro (1986). Ultrametricity for physicists. *Reviews of Modern Physics 58*(3), 765.

Rousseeuw, P. J. and L. Kaufman (2009). *Finding groups in data: An introduction to cluster analysis*. John Wiley & Sons Hoboken, New Jersey.

Sibson, R. (1973). Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal 16*(1), 30–34.

Teh, Y., H. Daume III, and D. M. Roy (2007). Bayesian agglomerative clustering with coalescents. *Advances in Neural Information Processing Systems 20*.

Virtanen, P., R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods 17*(3), 261–272.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association 58*(301), 236–244.

Wolberg, W., O. Mangasarian, N. Street, and W. Street (1993). Breast cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, `https://doi.org/10.24432/C5DW2B`.

A Discussion and Related Work

A.1 Ultrametrics and Dendrograms

A dissimilarity $d\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is an *ultrametric* if, for every distinct $x, y, z \in \mathcal{X}$,

$$d(x, y) \;\le\; \max\{d(x, z), d(y, z)\}. \tag{3}$$

This condition, often called the strong triangle inequality, implies that among the three distances

$$d(x, y), \qquad d(x, z), \qquad d(y, z),$$

the largest value is attained at least twice. Thus every triangle in an ultrametric space is isosceles, with the two largest sides equal. This local structure induces a nested family of balls and, equivalently, a rooted tree representation.

More precisely, every ultrametric d induces a unique rooted tree $T_u(\mathcal{X}, d)$ with leaf set $\mathcal{X}$, together with a height function $h\colon T_u(\mathcal{X}, d) \to \mathbb{R}_+$, such that

$$d(x, y) = h\big(\mathrm{lca}(x, y)\big),$$

where $\mathrm{lca}(x, y)$ denotes the lowest common ancestor of x and y in the tree $T_u(\mathcal{X}, d)$. The pair $(T_u(\mathcal{X}, d), h)$ is called a *dendrogram*, or a dated tree. Conversely, every dendrogram defines an ultrametric through the same formula. Hence ultrametrics on $\mathcal{X}$ are in bijection with real-valued dendrograms on $\mathcal{X}$, a correspondence that has been extensively studied in the literature (Johnson, 1967; Hartigan, 1967; Benzécri et al., 1984; Rammal et al., 1986; Carlsson et al., 2010; Cohen-Addad et al., 2019).

The following lemma shows that, in the ultrametric case, the Apresjan hierarchy recovers exactly the usual dendrogram hierarchy.

Lemma 8. *For any ultrametric d on $\mathcal{X}$, we have $T_*(\mathcal{X}, d) = T_u(\mathcal{X}, d)$.*

Proof. We identify $T_u(\mathcal{X}, d)$ with the hierarchy formed by the balls of the ultrametric space. Namely,

$$T_u(\mathcal{X}, d) \;=\; \{B(x, r) : x \in \mathcal{X},\ r \ge 0\}, \qquad B(x, r) := \{y \in \mathcal{X} : d(x, y) \le r\}.$$

We first show that every cluster in $T_u(\mathcal{X}, d)$ is valid. Let $C = B(a, r)$ be a ball. If $x, y \in C$, then the ultrametric inequality gives

$$d(x, y) \le \max\{d(x, a), d(a, y)\} \le r.$$

On the other hand, if $z \notin C$, then $d(a, z) > r$. For any $x \in C$, the ultrametric inequality implies

$$d(a, z) \le \max\{d(a, x), d(x, z)\}.$$

Since $d(a, x) \le r < d(a, z)$, we must have $d(x, z) = d(a, z) > r$. Therefore, for all $x, y \in C$ and all $z \notin C$,

$$d(x, y) \le r < d(x, z),$$

so C is a valid cluster.

Conversely, let C be a valid cluster. Fix any $a \in C$ and define

$$r_a := \max_{y \in C} d(a, y).$$

We claim that $C = B(a, r_a)$. Clearly, every $y \in C$ satisfies $d(a, y) \le r_a$, so $C \subseteq B(a, r_a)$. Now let $z \notin C$. Since C is valid, for every $y \in C$,

$$d(a, y) < d(a, z).$$

Taking the maximum over $y \in C$ gives

$$r_a < d(a, z),$$

and therefore $z \notin B(a, r_a)$. Hence $B(a, r_a) \subseteq C$, proving that $C = B(a, r_a)$.

Thus every valid cluster is an ultrametric ball, and every ultrametric ball is valid. Therefore the Apresjan hierarchy coincides with the dendrogram hierarchy $T_u(\mathcal{X}, d)$. □

A.2 Related Axiomatic, Optimization, and Probabilistic Perspectives

A rich line of research has sought to characterize hierarchical clustering procedures through axioms or optimization principles. Our validity-based perspective is closely connected to these efforts, yet it differs in scope: rather than prescribing a specific algorithm, it formalizes what it means for a hierarchy to be compatible (or valid) with a given similarity function.

Axiomatic characterizations. The seminal work of Carlsson et al. (2010) proposed a set of axioms for hierarchical clustering viewed as a map from metric spaces to ultrametric spaces. Imposing invariance under isometries and stability with respect to perturbations of the input metric, they showed that the only admissible method satisfying their axioms is *single linkage*. Their setting, however, treats hierarchical clustering as a procedure producing an ultrametric rather than a hierarchy, and it merges all pairs, achieving the current maximal similarity simultaneously, yielding possibly non-binary trees. In contrast, our framework provides an intrinsic definition of when a given tree is consistent with the similarities, without assuming any particular hierarchical clustering algorithm.

More recently, Arias-Castro and Coda (2025) propose an axiomatic notion of hierarchical clustering at the population level: the input is a density f, and the axioms are formulated in terms of the geometry of its density level sets. Under mild regularity conditions, their axioms characterize Hartigan's cluster tree, i.e., the hierarchy of connected components of the upper level sets $\{f \geq \lambda\}$. However, this framework presumes access to the underlying density f; in applications, one only observes a finite sample, and reliably estimating f and its level-set connectivity across scales can be challenging. In contrast, our framework is intrinsically finite and relational: it takes an arbitrary pairwise similarity s on a discrete set and defines validity via strict separation inequalities over triples, without assuming a notion of boundary or any generative density model.

Objective-based formulations. An alternative perspective, initiated by Dasgupta (2016) and further developed by Cohen-Addad et al. (2019), formulates hierarchical clustering as an optimization problem over all possible trees. Most recently, Pan et al. (2025) proposes an information-theoretical hierarchical clustering objective function, called structural entropy, which is shown to be equivalent to optimizing a cost function closely related to Dasgupta's cost.

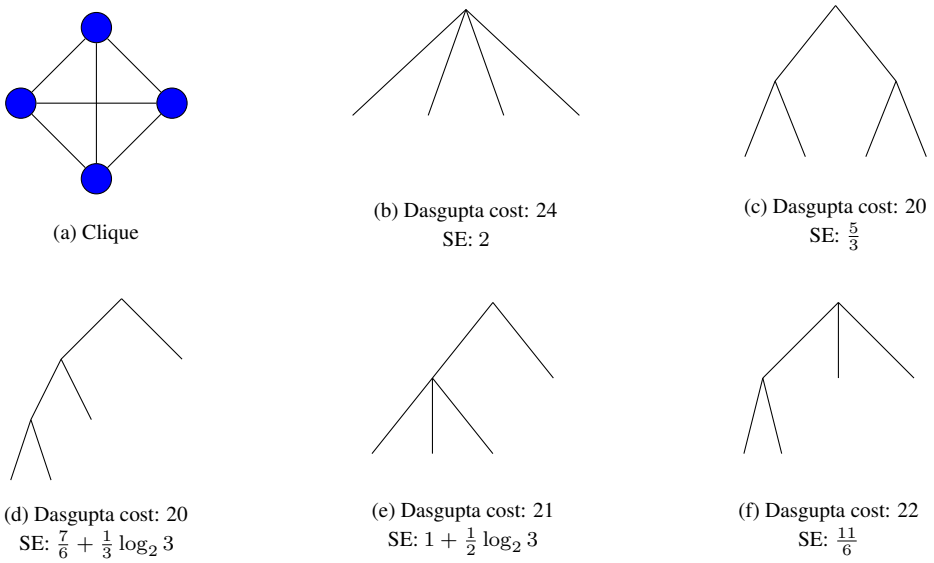
In that line of work, the goal is to find a hierarchy minimizing (or maximizing) a global cost functional depending on the pairwise similarities. However, the optimal tree minimizing the cost function is always binary. This issue can be seen in a simple example: a clique with unit edge weights (Figure 2(a)). Because the complete graph is fully symmetric, there is no structural basis for introducing intermediate clusters; the natural hierarchical representation is therefore the star-shaped tree shown in Figure 2(b), where all vertices attach directly to the root. However, as illustrated in Figure 2, both the Dasgupta cost and the objective proposed by Pan et al. (2025) favor binary hierarchies (Figure 2(c) instead, and for the Dasgupta cost also Figure 2(d)), even in this maximally symmetric setting.[4] Under both objectives, the intuitively appropriate star tree actually incurs the highest cost.[5]

In contrast, our approach establishes the necessary and sufficient conditions ensuring the *feasibility* of a hierarchy: a hierarchy is not optimized but required to satisfy a local set of validity inequalities. The resulting family of valid hierarchies could thus be viewed as the feasible region underlying those global optimization formulations.

Relation to our analysis. In summary, previous axiomatic and objective formulations identify either a particular algorithm (e.g., single linkage) or a preferred hierarchy optimizing a cost. Our contribution complements these perspectives by providing a foundational notion of *validity* that characterizes the entire space of hierarchies consistent with the similarity function. This structural viewpoint isolates the combinatorial core of hierarchical consistency, and it can accommodate both algorithmic and axiomatic specializations as particular instances.

[4] Dasgupta proves that, for a clique with uniform edge weights, every binary tree achieves the same optimal value under his objective; this "all trees have equal cost" property is later adopted as an axiom for desirable hierarchical clustering objectives in Cohen-Addad et al. (2019). In contrast, Pan et al. (2025) prove that only balanced binary trees are optimal under their objective.

[5] Moreover, Pan et al. (2025) also propose a heuristic that selects a target tree height and compresses the hierarchy accordingly, thereby producing a non-binary hierarchy. However, this procedure depends on the specific sequence of trees generated by their algorithm, offers no optimality guarantee, and in fact strictly increases the objective value.



(a) Clique

(b) Dasgupta cost: 24
SE: 2

(c) Dasgupta cost: 20
SE: $\frac{5}{3}$

(d) Dasgupta cost: 20
SE: $\frac{7}{6} + \frac{1}{3}\log_2 3$

(e) Dasgupta cost: 21
SE: $1 + \frac{1}{2}\log_2 3$

(f) Dasgupta cost: 22
SE: $\frac{11}{6}$

Figure 2: The complete unit weight graph with 4 nodes (a), and all possible trees (b)–(f) with 4 leaves associated with the Dasgupta cost, and the structural entropy (SE) introduced in Pan et al. (2025).

Probabilistic tree inference and network hierarchy models. Beyond axiomatic and cost-based frameworks, a significant body of work treats hierarchical clustering as the inference of tree structures under probabilistic or model-selection criteria. For instance, Blundell et al. (2010) propose the Bayesian Rose Trees approach, a greedy-agglomerative algorithm exploring a posterior over trees with arbitrary branching. Extensions of this idea include non-parametric priors over hierarchies such as the tree-structured stick-breaking process (Ghahramani et al., 2010) and coalescent-based Bayesian clustering models (Teh et al., 2007). In the network domain, recent work considers hierarchical community detection in random graphs via linkage or recursive partitioning methods (Li et al., 2022; Dreveton et al., 2026; Gray et al., 2023). These probabilistic and network-driven methodologies complement our validity-based framework: whereas they typically select or infer a single best tree under a likelihood or prior, our approach focuses on characterizing the entire feasible set of hierarchies consistent with a given similarity function. In this sense, our notion of validity may serve as a constraint or prior domain for tree-inference methods, suggesting a promising direction for future integration between combinatorial validity and probabilistic modeling.

A.3 Definitions of Linkages

Table 3: Lance–Williams coefficients for the linkages mentioned in the text.

Linkage	η_1	η_2	β
Ward	$\frac{\lvert t_1\rvert + \lvert t_3\rvert}{\lvert t_1\rvert + \lvert t_2\rvert + \lvert t_3\rvert}$	$\frac{\lvert t_2\rvert + \lvert t_3\rvert}{\lvert t_1\rvert + \lvert t_2\rvert + \lvert t_3\rvert}$	$-\frac{\lvert t_3\rvert}{\lvert t_1\rvert + \lvert t_2\rvert + \lvert t_3\rvert}$
Centroid	$\frac{\lvert t_1\rvert}{\lvert t_1\rvert + \lvert t_2\rvert}$	$\frac{\lvert t_2\rvert}{\lvert t_1\rvert + \lvert t_2\rvert}$	$-\frac{\lvert t_1\rvert\lvert t_2\rvert}{(\lvert t_1\rvert + \lvert t_2\rvert)^2}$
Median	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$

A.4 Computational Complexity of Pruning the Hierarchy

Bryant and Berry (2001) showed that Algorithm 2 admits a more efficient implementation. Given a hierarchy T, let $\hat{T}^-$ denote the collection of invalid clusters in T. Then the pruned hierarchy $\hat{T} = T \setminus \hat{T}^-$ can be computed in $O(k^2)$ time, where $|\mathcal{X}| = k$. The procedure identifies and removes the invalid clusters of T as follows.

1. **Compute intra-cluster maxima.** For each cluster v of T and each element $x \in v$, compute $M[v, x] := \max_{y \in v} d(x, y)$. These quantities can be computed bottom-up, from the leaves toward the root. If u is the child of v containing x, then

$$M[v, x] = \max\left(M[u, x], \max\{d(x, y) : \mathrm{lca}(x, y) = v\}\right),$$

where u is the child of v containing x.

2. **Compute inter-cluster minima.** For each cluster $v \in T$ and each element $x \in v$, compute $m[v, x] := \min_{z \notin v} d(x, z)$. These quantities can be computed top-down, from the root toward the leaves. If u is a child of v and $x \in u$, then

$$m[u, x] = \min\big(m[v, x], \min\{d(x, z) : \mathrm{lca}(x, z) = v\}\big).$$

3. **Prune invalid clusters.** A cluster t is valid if and only if, for every $x \in t$, the farthest point from x inside t is closer than the nearest point to x outside t, namely

$$M[t, x] < m[t, x] \qquad \text{for all } x \in t.$$

Therefore, pruning the linkage hierarchy to

$$\left\{ t \in T \colon \min_{x \in t} \big(m[t, x] - M[t, x]\big) > 0 \right\}$$

yields exactly $\hat{T}$.

A.5 Connection to HDBSCAN

HDBSCAN (Campello et al., 2013, 2015) starts from the mutual-reachability dissimilarity

$$d'_\ell(x, y) \;=\; \max\{\mathrm{core}_\ell(x), \mathrm{core}_\ell(y), d(x, y)\} \qquad (x \neq y),$$

where $\mathrm{core}_\ell(x)$ is the distance from x to its ℓ-th nearest neighbor, x itself counting as its first neighbor (the convention of scikit-learn), so that $d'_1 = d$. Its uncondensed hierarchy is the single-linkage tree $T_{\mathrm{SL}}(\mathcal{X}, d'_\ell)$. Because single linkage satisfies Conditions 1 and 2, pruning this tree returns $T_*(\mathcal{X}, d'_\ell)$ (Theorem 4).

HDBSCAN simplifies $T_{\mathrm{SL}}(\mathcal{X}, d'_\ell)$ with a different criterion. Given a minimum cluster size s, a split creates two new condensed branches only if both children contain at least s points; a child with fewer than s points is treated as points falling out of the parent, and the other child continues the parent's branch. A flat clustering is finally selected from the condensed tree by a stability (persistence) score. Neither step tests validity, and the two simplifications are incomparable: condensation can remove a valid cluster because $|C| < s$ can happen, and can keep a large branch that is not valid, whereas validity pruning ignores cardinality.

On the boundary case $s = 1$, the connection and difference between $T_*(\mathcal{X}, d'_\ell)$ and HDBSCAN $T_{\mathrm{SL}}(\mathcal{X}, d'_\ell)$ becomes most visible. Every child then meets the size threshold, so no branch is contracted and the condensed tree is $T_{\mathrm{SL}}(\mathcal{X}, d'_\ell)$ itself. So, $T_*(\mathcal{X}, d'_\ell) \subseteq T_{\mathrm{SL}}(\mathcal{X}, d'_\ell)$ holds.

A.6 Robustness to Perturbations

Validity is defined by strict inequalities, and the margin in these inequalities quantifies how robustly a cluster is valid.

Definition 6 (Validity margin)**.** *For a nonempty cluster $C \subsetneq \mathcal{X}$, the validity margin of C with respect to d is*

$$\Delta_d(C) \;=\; \min_{x \in C} \left(\min_{z \notin C} d(x, z) \;-\; \max_{y \in C} d(x, y) \right).$$

For $\gamma \geq 0$, define the margin hierarchy $T_\gamma(\mathcal{X}, d)$ *consists of $\mathcal{X}$, the singletons, and all clusters $C \subsetneq \mathcal{X}$ with $\Delta_d(C) > \gamma$.*

By Definition 3, C is valid if and only if $\Delta_d(C) > 0$; hence $T_0(\mathcal{X}, d) = T_*(\mathcal{X}, d)$, and more generally $T_\gamma(\mathcal{X}, d) \subseteq T_*(\mathcal{X}, d)$.

Proposition 9 (Robustness of Trimmed Linkage)**.** *Let f satisfy Conditions 1 and 2, and let d and d' be two dissimilarities on $\mathcal{X}$ with $\|d'-d\|_\infty = \max_{x,y} |d'(x,y)-d(x,y)| \le \varepsilon$. Denote by $\hat{T}(\mathcal{X}, d', f)$ the output of Algorithm 2 applied to d'. Then, every cluster C with $\Delta_d(C) > 2\varepsilon$ belongs to $\hat{T}(\mathcal{X}, d', f)$. Moreover, if $|\Delta_d(C)| > 2\varepsilon$ for every $C \subsetneq \mathcal{X}$ with $|C| \ge 2$, then $\hat{T}(\mathcal{X}, d', f) = T_*(\mathcal{X}, d)$.*

Proof. Fix $x \in C$. Replacing d by d' changes each of $\min_{z \notin C} d(x,z)$ and $\max_{y \in C} d(x,y)$ by at most ε, so their difference changes by at most 2ε; taking the minimum over $x \in C$ gives $|\Delta_{d'}(C) - \Delta_d(C)| \le 2\varepsilon$. Hence $\Delta_d(C) > 2\varepsilon$ implies $\Delta_{d'}(C) > 0$, and under the assumption of the second part, $\Delta_{d'}(C)$ and $\Delta_d(C)$ have the same sign for every C with $|C| \ge 2$. Thus C is valid for d' if it satisfies $\Delta_d(C) > 2\varepsilon$, and $T_*(\mathcal{X}, d') = T_*(\mathcal{X}, d)$ under the assumption of the second part. Both claims follow since $\hat{T}(\mathcal{X}, d', f) = T_*(\mathcal{X}, d')$ by Theorem 4. □

The factor 2ε cannot be improved: increasing $d(x,y)$ and decreasing $d(x,z)$ by ε, for the triplet attaining the margin, lowers $\Delta_d(C)$ by exactly 2ε. Moreover, because the pruning procedure of Section 3.2 computes $\Delta_{d'}(t)$ for every cluster t of the linkage tree, one can retain only the clusters with $\Delta_{d'}(t) > \gamma$ for a chosen $\gamma > 0$ at no extra cost; by the same argument, the result contains every cluster with $\Delta_d(C) > \gamma + 2\varepsilon$.

Planted hierarchy under noise. To illustrate Proposition 9, we consider $k = 36$ points forming nine fine clusters of four points, nested in three coarse clusters of three fine clusters each (Figure 3, left). The dissimilarity d_0 equals 2 within a fine cluster, 6 between fine clusters of the same coarse cluster, and 12 between coarse clusters. Then d_0 is an ultrametric, and $T_*(\mathcal{X}, d_0)$ is the planted hierarchy: it has 12 non-root internal clusters, has three- and four-way multifurcations, and every planted cluster has margin at least 4. A binary linkage tree has 34 non-root internal clusters, so pruning removes 22 of them (64.7%) and recovers the planted hierarchy exactly. For a noise level $\rho \ge 0$, we set $d_\rho = d_0 + 4\rho U$, where U is symmetric with zero diagonal and independent entries $U_{ij} \sim \mathrm{Unif}[-1, 1]$ for $i < j$ (entries are floored to keep d_ρ positive). As $\|d_\rho - d_0\|_\infty \le 4\rho$, Proposition 9 guarantees that every planted cluster is recovered whenever $\rho < 1/2$. Figure 3 reports the results over 50 independent draws per noise level: all planted clusters are recovered in every trial for $\rho < 1/2$. Beyond this worst-case threshold, recovery degrades gradually.

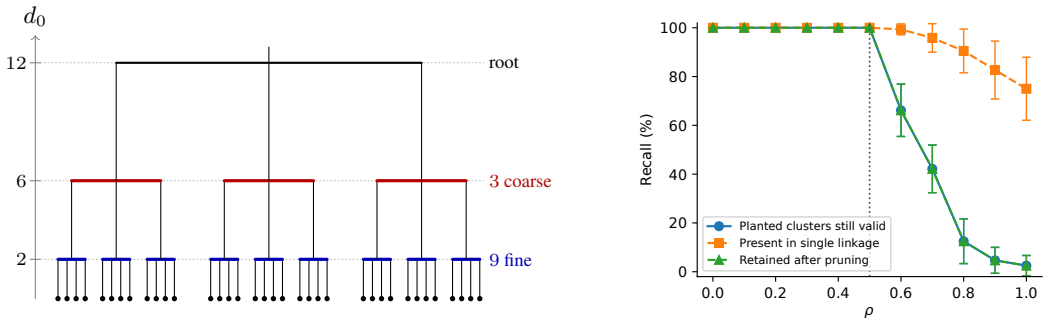


Figure 3: Left: the planted hierarchy $T_*(\mathcal{X}, d_0)$, drawn as the dendrogram of the ultrametric d_0. The height of a merge is the value of d_0 between the merged groups. Its 12 non-root internal clusters are the 9 fine clusters (blue, four-way merges at height 2) and the 3 coarse clusters (red, three-way merges at height 6). Right: planted-cluster recall as a function of ρ (mean $\pm$ one standard deviation over 50 draws).

B Proof of Lemma 3

B.1 Similarity formulation of the linkage conditions

In the main text, Lemma 3 is stated for a dissimilarity function d on the set $\mathcal{X}$. For the proof, it will be more convenient to work with the equivalent similarity formulation. This setting is also natural

in many applications, especially in graph-based settings, where edge weights typically measure the similarity between their endpoints.

We call similarity a function $s\colon \mathcal{X} \times \mathcal{X}$ such that (i) s is symmetric, *i.e.* $s(x,y) = s(y,x)$ for all $x, y \in \mathcal{X}$, and that (ii) $s(x,x) \geq s(x,y) \geq 0$ for all $x \neq y$. In the similarity setting, the corresponding notion of valid cluster is obtained by reversing the inequality of Definition 3: a subset $C \subseteq \mathcal{X}$ is valid with respect to s if

$$\forall x, y \in C,\ \forall z \in \mathcal{X} \setminus C\colon \quad s(x,y) > s(x,z).$$

We now formulate the analogues of Conditions 1 and 2 in terms of similarities.

Condition 3 (Monotonicity across merges)**.** *For any similarity function s, the update rule f extends s such that for any subsets $t_1, t_2, t_3, t_4 \subset \mathcal{X}$, we have*

$$\forall i \in \{1,2\}\colon s(t_i, t_3) > \max\left(s(t_i, t_4), s(t_3, t_4)\right) \ \Rightarrow\ s(t_1 \cup t_2, t_3) > \max\left(s(t_1 \cup t_2, t_4), s(t_3, t_4)\right).$$

Condition 4 (Dominance preservation)**.** *For any similarity function s, the update rule f extends s such that for any subsets $t_1, t_2, t_3, t_4 \subset \mathcal{X}$, we have*

$$s(t_1, t_2) > \max_{\substack{i \in \{1,2\} \\ j \in \{3,4\}}} s(t_i, t_j) \ \Rightarrow\ s(t_1, t_2) > \max_{i \in \{1,2\}} s(t_i, t_3 \cup t_4).$$

The analogue of Assumption 1 in the similarity setting is obtained by requiring the update rule f to be right-continuous with respect to its first argument. The linkage algorithm is simply modified by merging the most similar clusters.

We prove Lemma 3 in this similarity formulation. The proof in the dissimilarity formulation follows analogously, with the corresponding reversal of the inequalities.

B.2 Proof of Lemma 3: Sufficiency of Conditions 3 and 4

Let f be a linkage update rule satisfying Conditions 3 and 4. Let s be an arbitrary similarity function over a finite set $\mathcal{X}$. Denote by $\hat{T}_{\text{linkage}} = \hat{T}_{\text{linkage}}(\mathcal{X}, s, f)$ the hierarchy returned by Algorithm 1, which is a binary tree, and let $T \in \mathcal{H}(\mathcal{X}, s)$ be a valid hierarchy with respect to $(\mathcal{X}, s)$. We prove that $T \subseteq \hat{T}_{\text{linkage}}$.

Step 1. Setting and notation Throughout the proof, we use the notations defined in Algorithm 1, in particular for the sets $\mathcal{C}^{(m)}_{active}$ that result from the successive aggregations of vertices. In particular, for each iteration $m \in [k-1]$, we let $\mathcal{C}^{(m)}_{active}$ be the set of active clusters maintained by Algorithm 1. We also denote by $m_{\text{b}}(v)$ the creation time of cluster $v \in \hat{T}_{\text{linkage}}$, *i.e.,* $m_{\text{b}}(v) = m$ if and only if $v \in \mathcal{C}^{(m)}_{active}$ and $v \notin \mathcal{C}^{(m-1)}_{active}$.

Initially, $\mathcal{C}^{(0)}_{active} = \{\{x_1\}, \cdots, \{x_k\}\}$. Moreover, at all times m, we have $\bigcup_{y \in C^{(m)}_{active}} y = \mathcal{X}$ and

$$y \cap y' = \emptyset \quad \forall y \neq y' \in C^{(m)}_{active}. \tag{4}$$

Finally, for any $m \in [k-1]$, we define the property $\mathcal{P}(m)$:

$$\forall t_1, t_2, t_3 \in \mathcal{C}^{(m)}_{active},\ \text{if} \quad \exists t \in T\colon t_1, t_2 \subsetneq t, t_3 \subseteq \mathcal{X} \backslash t \Rightarrow s(t_1, t_2) > \max\left(s(t_1, t_3), s(t_2, t_3)\right). \tag{5}$$

Recall that T is a valid hierarchy. Hence, if t_1, t_2 and t_3 are three leaves such that $t_1, t_2 \subsetneq t$ and $t_3 \subseteq \mathcal{X} \backslash t$, then the condition $s(t_1, t_2) > \max\left(s(t_1, t_3), s(t_2, t_3)\right)$ is automatically verified because of (1), independently of the linkage function. But, when t_1, t_2 and t_3 are not all leaves, the condition $s(t_1, t_2) > \max\left(s(t_1, t_3), s(t_2, t_3)\right)$ is not guaranteed anymore. Indeed, the similarity between non-leaves elements depends on the linkage function, and only assumptions on the linkage function can ensure that $s(t_1, t_2) > \max\left(s(t_1, t_3), s(t_2, t_3)\right)$.

Step 2. If $\mathcal{P}(m)$ holds for every $m \in [k-1]$, then $T \subseteq \hat{T}_{\text{linkage}}$. Assume that $\mathcal{P}(m)$ holds for every $m \in [k-1]$. We will show that $T \subseteq \hat{T}_{\text{linkage}}$.

By contradiction, assume that $T \not\subseteq \hat{T}_{\text{linkage}}$. This is equivalent to the existence of a cluster $t \in T$ such that $t \notin \hat{T}_{\text{linkage}}$. For this cluster t, define

$$\mathcal{B}_t \;=\; \{v \in \hat{T}_{\text{linkage}} \colon v \cap t \notin \{\emptyset, v, t\}\}.$$

Because $\hat{T}_{\text{linkage}}$ is a binary tree and $t \notin \hat{T}_{\text{linkage}}$, Lemma 10 asserts that $\mathcal{B}_t$ is non-empty.

Because $\mathcal{B}_t$ is non-empty, we can define $u = \arg\min_{v \in \mathcal{B}_t} m_{\text{b}}(v)$, and let $m' = m_{\text{b}}(u)$ be the iteration in which the first element of $\mathcal{B}_t$ is created.

Because $C^{(0)}_{active}$ is composed of the leaves (which are the singletons $\{x_i\}$), for any $v \in C^{(0)}_{active}$ we have $v \cap t \in \{\emptyset, v\} \subseteq \{\emptyset, v, t\}$. Hence $|u| \geq 2$ and u is created at an iteration $m' \geq 1$ by merging two clusters u_1 and u_2, defined by

$$\{u_1, u_2\} \;\in\; \underset{u'_1 \neq u'_2 \in \mathcal{C}^{(m'-1)}_{active}}{\arg\max} s(u'_1, u'_2). \tag{6}$$

Moreover, because of the definition of u, we have $u_1, u_2 \notin \mathcal{B}_t$, otherwise, $u \neq \arg\min_{v \in \mathcal{B}_t} m_{\text{b}}(v)$, hence $u_i \cap t \in \{\emptyset, u_i, t\}$ ($i \in \{1, 2\}$). Therefore, we have one of the following five cases:

(i) $u_1 \cap t = t$ or $u_2 \cap t = t$;
(ii) $u_1 \cap t = \emptyset$ and $u_2 \cap t = \emptyset$;
(iii) $u_1 \cap t = u_1$ and $u_2 \cap t = \emptyset$;
(iv) $u_1 \cap t = \emptyset$ and $u_2 \cap t = u_2$;
(v) $u_1 \cap t = u_1$ and $u_2 \cap t = u_2$.

First, we can rule out Case (i). Indeed, if $u_i \cap t = t$ for some $i \in \{1, 2\}$, then we would have $t \cap u = t \cap (u_1 \cup u_2) = t$, which is impossible because $u \in \mathcal{B}_t$.

Case (ii) implies that $(u_1 \cup u_2) \cap t = \emptyset$. However, this is again impossible; indeed, $u = u_1 \cup u_2 \in \mathcal{B}_t$ implies that $(u_1 \cup u_2) \cap t \notin \{\emptyset, u_1 \cup u_2, t\}$. Similarly, case (v) is also impossible because it would imply $u = u_1 \cup u_2 \subseteq t$, again an impossibility due to $u \in \mathcal{B}_t$. As a result, only cases (iii) and (iv) are possible.

Consider case (iii) where $u_1 \cap t = u_1$ and $u_2 \cap t = \emptyset$, and notice that the symmetric case (iv) can be treated similarly. Because case (i) is impossible, $u_1 \neq t$. Together with $\bigcup_{y \in C^{(m'-1)}_{active}} y = \mathcal{X}$, it therefore implies the existence of a set $u'_1 \in C^{(m'-1)}_{active}$ such that $u'_1 \neq u_1$ and $u'_1 \cap t \neq \emptyset$.

Because $m_{\text{b}}(u'_1) \leq m' - 1 < m' = m_{\text{b}}(u)$ and because of the definition of u, we have $u'_1 \notin \mathcal{B}_t$. Combined with $u'_1 \cap t \neq \emptyset$, it implies that $u'_1 \cap t \in \{u'_1, t\}$. Moreover, because $u_1, u'_1 \in \mathcal{C}^{(m'-1)}_{active}$ are disjoint by (4), we have $(t \cap u_1) \cap u'_1 = \emptyset$. Combined with $u_1 \cap t = u_1 \neq \emptyset$, this rules out $u'_1 \cap t = t$ and forces $u'_1 \cap t = u'_1$.

To summarize, we have $u_1, u'_1, u_2 \in C^{(m'-1)}_{active}$ such that $u_1, u'_1 \subsetneq t$ and $u_2 \subseteq \mathcal{X} \setminus t$. Therefore, the condition $\mathcal{P}(m'-1)$ implies that $s(u_1, u'_1) > s(u_1, u_2)$. This contradicts (6) and therefore proves that $T \subseteq \hat{T}_{\text{linkage}}$.

Step 3. Inductive proof that $\mathcal{P}(m)$ holds for every $m \in [k-1]$. We proceed by induction on m.

Let $m = 0$. We have $\mathcal{C}^{(0)}_{active} = \{\{x_1\}, \cdots, \{x_k\}\}$. Let $t_1, t_2, t_3 \in \mathcal{C}^{(0)}_{active}$ and suppose there exists $t \in T$ such that $t_1, t_2 \subsetneq t$ and $t_3 \subseteq \mathcal{X} \backslash t$. Note that $\mathcal{C}^{(0)}_{active} = \mathcal{X}$ and thus $\mathcal{P}(0)$ corresponds to the following statement:

$$\forall x_1, x_2, x_3 \in \mathcal{X} \colon \exists t \in T \colon x_1, x_2 \in t, x_3 \in \mathcal{X} \backslash t \Rightarrow s\left(x_1, x_2\right) > \max\left(s\left(x_1, x_3\right), s\left(x_2, x_3\right)\right),$$

which we can rewrite as

$$\forall t \in T \colon \forall x_1, x_2 \in t, x_3 \in \mathcal{X} \backslash t \Rightarrow s\left(x_1, x_2\right) > \max\left(s\left(x_1, x_3\right), s\left(x_2, x_3\right)\right). \tag{7}$$

Because T is a valid hierarchy, the statement (7) holds. This establishes $\mathcal{P}(0)$.

Let $m \geq 0$ and suppose that $\mathcal{P}(m')$ holds for any $m' \in \{0, \cdots, m\}$. We prove that $\mathcal{P}(m+1)$ holds. Let $\{t'_1, t'_2\} \in \arg\max_{v_1 \neq v_2 \in \mathcal{C}^{(m)}_{active}} s(v_1, v_2)$ and denote $t_{new} = t'_1 \cup t'_2$. Recall that, by definition,

$\mathcal{C}_{active}^{(m+1)} \cup \{t_1', t_2'\} = \mathcal{C}_{active}^{(m)} \cup \{t_{new}\}$. We consider vertex $t \in T$ and a triplet of three different sets $t_1, t_2, t_3 \in \mathcal{C}_{active}^{(m+1)}$ such that $t_1, t_2 \subseteq t$ and $t_3 \subseteq \mathcal{X} \backslash t$. Because $\mathcal{C}_{active}^{(m+1)} \setminus \mathcal{C}_{active}^{(m)} = \{t_{new}\}$, we have the following four cases to consider.

- If $t_1, t_2, t_3 \in \mathcal{C}_{active}^{(m)}$, then the induction property $\mathcal{P}(m)$ implies that
$$s(t_1, t_2) > \max\left(s(t_1, t_3), s(t_2, t_3)\right).$$
- Suppose that $t_1 = t_{new} = t_1' \cup t_2'$. Because $t_1, t_2 \subseteq t$, we have $t_1', t_2', t_2 \subseteq t$. Moreover, $t_1', t_2' \in \mathcal{C}_{active}^{(m)}$, hence the induction property $\mathcal{P}(m)$ implies that
$$\forall i \in \{1, 2\}\colon \quad s(t_i', t_2) > \max\left(s(t_i', t_3), s(t_2, t_3)\right).$$
Thus, Condition 3 implies that
$$s(t_1' \cup t_2', t_2) > \max\left(s(t_1' \cup t_2', t_3), s(t_2, t_3)\right).$$
Because $t_1 = t_1' \cup t_2'$, this is equivalent to $s(t_1, t_2) > \max\left(s(t_1, t_3), s(t_2, t_3)\right)$.
- The case $t_2 = t_{new}$ can be treated similarly to the previous case, and leads again to $s(t_1, t_2) > \max\left(s(t_1, t_3), s(t_2, t_3)\right)$.
- Finally, suppose that $t_3 = t_{new}$. Because $t_3 \subseteq \mathcal{X} \backslash t$, we also have $t_1' \subseteq \mathcal{X} \backslash t$. Moreover, because $t_1' \in \mathcal{C}_{active}^{(m)}$, the induction property $\mathcal{P}(m)$ implies that
$$s(t_1, t_2) > \max\left(s(t_1, t_1'), s(t_2, t_1')\right),$$
Because we also have $t_2' \subseteq \mathcal{X} \backslash t$ and $t_2' \in \mathcal{C}_{active}^{(m)}$, we establish similarly that
$$s(t_1, t_2) > \max\left(s(t_1, t_2'), s(t_2, t_2')\right).$$
Hence, by combining the last two inequalities, we have
$$s(t_1, t_2) > \max\left(s(t_1, t_1'), s(t_2, t_1'), s(t_1, t_2'), s(t_2, t_2')\right).$$
Condition 4 therefore implies that $s(t_1, t_2) > \max\left(s(t_1, t_1' \cup t_2'), s(t_2, t_1' \cup t_2')\right)$. Because $t_3 = t_1' \cup t_2'$, this is equivalent to $s(t_1, t_2) > \max\left(s(t_1, t_3), s(t_2, t_3)\right)$.

The combination of these four cases proves $\mathcal{P}(m+1)$.

B.3 Proof of Lemma 3: Necessity of Conditions 3 and 4

We call an update rule f *valid* if it satisfies

$$\forall (\mathcal{X}, s)\colon \quad T_*(\mathcal{X}, s) \subseteq \hat{T}_{\text{linkage}}(\mathcal{X}, s, f). \tag{8}$$

We will show that every valid update rule f must satisfy Conditions 3 and 4. By contraposition, it suffices to prove the following statement: *Any update rule f that violates Condition 3 or Condition 4 is not valid.*

Case (i): Violation of Condition 3 A violation of Condition 3 means that there exist four disjoint clusters $t_1, \cdots, t_4$ such that t_1 and t_2 are merged at some step, but

$$\forall i \in \{1, 2\}\colon s(t_i, t_3) \;>\; \max\left(s(t_i, t_4), s(t_3, t_4)\right) \tag{9}$$

and

$$f(s_{12}, s_{23}, s_{31}, n_1, n_2, n_3) \;\leq\; \max\left(f(s_{12}, s_{24}, s_{41}, n_1, n_2, n_4), s_{34}\right), \tag{10}$$

where we use the shorthand $s_{ij} = s(t_i, t_j)$ and $n_i = |t_i|$. Because t_1 and t_2 are the clusters merged by the algorithm, we also have $(t_1, t_2) \in \arg\max_{i \in [4]} s(t_i, t_j)$ and thus

$$s(t_1, t_2) \geq s(t_i, t_j) \quad \text{for all } i, j \in [4], i \neq j. \tag{11}$$

Construction of a counterexample. Suppose f violates Condition 3 but is valid. We will construct a dataset $(\mathcal{X}_1, s_1)$ for which $T_*(\mathcal{X}_1, s_1) \not\subseteq \hat{T}_{\text{linkage}}(\mathcal{X}_1, s_1, f)$, contradicting (8) and thus the validity of f.

Let t'_1, t'_2, t'_3, t'_4 be disjoint sets with $|t'_i| = n_i$, and define $\mathcal{X}_1 = \bigcup_{i\in[4]} t'_i$. Define the similarity function $s_1 \colon \mathcal{X}_1 \times \mathcal{X}_1 \to \mathbb{R}_+$ by

$$s_1(x,y) \;=\; \begin{cases} s_{ij}, & \text{if } x \in t'_i,\ y \in t'_j \text{ for some } i \neq j \in \{1,2,3,4\}, \\ s_{12}+1, & \text{if } x, y \in t'_i \text{ for some } i \in \{1,2,3,4\}. \end{cases}$$

By (9) and (11), for each $i \in \{1,2\}$, we have

$$s_{12} \geq s_{i3} > \max(s_{i4}, s_{34}).$$

Hence, the union $t'_1 \cup t'_2 \cup t'_3$ forms a valid cluster with respect to s_1, *i.e.,*

$$t'_1 \cup t'_2 \cup t'_3 \in T_*(\mathcal{X}_1, s_1).$$

By Lemma 12, the clusters $t'_1, \cdots, t'_4$ appear as active clusters before any cross-merging occurs, so for all $1 \leq i < j \leq 4$,

$$s_1(t'_i, t'_j) \;=\; s_{ij}.$$

Moreover, since t'_1 and t'_2 are merged first (by (11)), the similarity between the cluster $t'_1 \cup t'_2$ formed by their merger and t'_3 satisfies

$$\begin{aligned} s_1(t'_1 \cup t'_2, t'_3) \;&=\; f(s_{12}, s_{23}, s_{31}, n_1, n_2, n_3) \\ &\leq\; \max\left(f(s_{12}, s_{24}, s_{41}, n_1, n_2, n_4), s_{34}\right) \qquad (12) \\ &=\; \max\left(s_1(t'_1 \cup t'_2, t'_4), s_1(t'_3, t'_4)\right), \end{aligned}$$

where (12) follows from (10). If the inequality (12) is strict, then the algorithm at the next step merges either the pair $(t'_1 \cup t'_2, t'_4)$ or the pair (t'_3, t'_4), preventing the formation of the true cluster $t'_1 \cup t'_2 \cup t'_3$. Even if equality holds, the similarity of the pair $(t'_1 \cup t'_2, t'_3)$ merely ties with the similarity of another pair, and tie-breaking may again exclude the valid cluster.

In both cases,

$$T_*(\mathcal{X}_1, s_1) \not\subseteq \hat{T}_{\text{linkage}}(\mathcal{X}_1, s_1, f),$$

contradicting (8). Thus, a valid f cannot violate Condition 3.

Case (ii): Violation of Condition 4 A violation of Condition 4 means that there exist four disjoint clusters t_1, t_2, t_3, t_4 such that t_3 and t_4 are merged, i.e.

$$s(t_3, t_4) \geq s(t_i, t_j) \quad \text{for all } i, j \in [4], i \neq j, \qquad (13)$$

and simultaneously

$$s_{12} > \max_{i\in\{1,2\},\, j\in\{3,4\}} s_{ij} \quad \text{and} \quad s_{12} \leq \max_{i\in\{1,2\}} f(s_{34}, s_{4i}, s_{3i}, n_3, n_4, n_i). \qquad (14)$$

Construction of a counterexample. Assume again that f is valid but violates Condition 4.

Let t'_1, t'_2, t'_3, t'_4 be disjoint sets with $|t'_i| = n_i$, and define $\mathcal{X}_1 = \bigcup_{i\in[4]} t'_i$. Define the similarity function $s_1 \colon \mathcal{X}_1 \times \mathcal{X}_1 \to \mathbb{R}_+$ by

$$s_1(x,y) \;=\; \begin{cases} s_{ij}, & \text{if } x \in t'_i,\ y \in t'_j \text{ for some } i \neq j \in \{1,2,3,4\}, \\ s_{34}+1, & \text{if } x, y \in t'_i \text{ for some } i \in \{1,2,3,4\}. \end{cases}$$

From (14), s_{12} is strictly larger than $\max_{i\in\{1,2\},\, j\in\{3,4\}} s_{ij}$, hence

$$t'_1 \cup t'_2 \in T_*(\mathcal{X}_1, s_1).$$

By Lemma 12, $t'_1, \cdots, t'_4$ appear as active clusters before any inter-cluster merge, and

$$s_1(t'_i, t'_j) \;=\; s_{ij} \quad \forall i < j.$$

By (14), we have

$$\begin{aligned} s_1(t'_1, t'_2) \;=\; s_{12} \;&\leq\; \max_{i\in\{1,2\}} f(s_{34}, s_{4i}, s_{3i}, n_3, n_4, n_i) \qquad (15) \\ &=\; \max_{i\in\{1,2\}} s_1(t'_3 \cup t'_4, t'_i). \end{aligned}$$

Therefore, the linkage procedure either merges one of the spurious pairs $(t'_3 \cup t'_4, t'_i)$ or ties with the valid pair (t'_1, t'_2), preventing the formation of the true cluster $t'_1 \cup t'_2$. Hence

$$T_*(\mathcal{X}_1, s_1) \not\subseteq \hat{T}_{\text{linkage}}(\mathcal{X}_1, s_1, f),$$

contradicting (8). Therefore a valid f cannot violate Condition 4.

Remark 2. *Without Assumption 1, we can only show $f(q^+, q, q, \cdots) = q$ for any $q^+ > q$ (Lemma 12) and $f(q, q, q, \cdots) = q$ might not hold. This forbids us from using the ultrametric similarities (i.e., $s(x, y)$ that takes the same value for any $x, y \in t'_i$). However, careful constructions of $(\mathcal{X}_1, s_1)$ that serve as the counterexamples that are similar to those in the above proof (but more complex), are still possible but require more tedious arguments.*

B.4 Auxiliary Lemmas

B.4.1 Auxiliary Lemmas for Appendix B.2

The following technical lemma is used to prove the sufficiency of Conditions 3 and 4 in Lemma 3.

Lemma 10. *Let $v \subseteq \mathcal{X}$. Let $T_{bin} \in \mathcal{T}(\mathcal{X})$ be a binary tree. The following holds:*

$$v \notin T_{bin} \quad \Rightarrow \quad \exists u \in T_{bin} \colon u \cap v \notin \{\emptyset, u, v\}. \tag{16}$$

Proof. Let $v \subseteq \mathcal{X}$ such that $v \notin T_{bin}$. We show that there exists $u_* \in T_{bin}$ such that $u_* \cap v \notin \{\emptyset, u_*, v\}$.

Because $v \notin T_{bin}$ while $\mathcal{X} \in T_{bin}$, we have $v \subsetneq \mathcal{X}$. Denote by $\mathcal{B}$ the collection of ancestors of v in the tree

$$\mathcal{B} \;=\; \{u \in T_{bin} \colon v \subsetneq u\}.$$

Because $\mathcal{X} \in \mathcal{B}$, the set $\mathcal{B}$ is not empty. Let

$$u_0 \;=\; \arg\min\{|u| \colon u \in \mathcal{B}\}$$

be the smallest cluster of T_{bin} strictly containing v.

Because $v \subsetneq u_0$, u_0 is not a leaf. Because T_{bin} is a binary tree, u_0 has exactly two children u_1 and $u_2 \in T_{bin}$ satisfying $u_1 \cup u_2 = u_0$ and $u_1 \cap u_2 = \emptyset$.

By minimality of u_0, neither child fully contains v; otherwise that child would belong to $\mathcal{B}$ with smaller cardinality that u_0. Hence,

$$v \not\subseteq u_1 \quad \text{and} \quad v \not\subseteq u_2.$$

Moreover, at least one child is not contained in v, *i.e.,*

$$u_1 \not\subseteq v \text{ or } u_2 \not\subseteq v.$$

Indeed, by contradiction, suppose that both $u_1 \subseteq v$ and $u_2 \subseteq v$ hold. Then, $u_0 = u_1 \cup u_2 \subseteq v$, contradicting $v \subsetneq u_0$.

Without loss of generality, suppose that $u_1 \not\subseteq v$, and let $u_* = u_1$. We have $u_* \in T_{bin}$. Because u^* neither contains v nor is disjoint from it, we have

$$u_* \cap v \;\notin\; \{\emptyset, u_*, v\}.$$

This ends the proof. □

B.4.2 Auxiliary Lemmas for Appendix B.3

We first introduce the following notations, used throughout this section.

Notations.

- For any subset $\mathcal{X}_1 \subseteq \mathcal{X}$, denote by $s[\mathcal{X}_1]$ the restriction of the similarity function s to $\mathcal{X}_1$, that is,
$$s[\mathcal{X}_1] \colon (x, y) \in \mathcal{X}_1 \times \mathcal{X}_1 \mapsto s(x, y).$$
- Denote by $\mathcal{C}^{(m)}_{active}(\mathcal{X}, s)$ the active set at iteration m of the linkage process applied to $(\mathcal{X}, s)$. When unambiguous, we simply write $\mathcal{C}^{(m)}_{active}$.

We present two auxiliary lemmas used in Appendix B.3.

Lemma 11 (Bounds on updated similarities)**.** *Let f be a valid update function, and let t_1, t_2, t_3, t_4, t_5 be disjoint clusters. Then the following statements hold for the linkage process defined by f.*

1. *Suppose there exists $(\mathcal{X}, s)$ such that $t_1, t_2, t_3, t_4 \subsetneq \mathcal{X}$, the union $t_1 \cup t_2 \cup t_3$ is a valid cluster, and for some iteration m,*

$$t_1 \cup t_2,\ t_3,\ t_4 \in \mathcal{C}^{(m)}_{active}(\mathcal{X}, s).$$

Then necessarily,

$$s(t_1 \cup t_2, t_3) > s(t_3, t_4).$$

2. *Suppose there exists $(\mathcal{X}, s)$ such that $t_1, t_2, t_3, t_5 \subsetneq \mathcal{X}$, the union $t_3 \cup t_5$ is a valid cluster, and for some iteration m,*

$$t_1 \cup t_2,\ t_3,\ t_5 \in \mathcal{C}^{(m)}_{active}(\mathcal{X}, s).$$

Then necessarily,

$$s(t_1 \cup t_2, t_3) < s(t_3, t_5).$$

3. *Combining the previous two statements, if there exists $(\mathcal{X}, s)$ such that $(t_1 \cup t_2 \cup t_3 \cup t_4, s[t_1 \cup t_2 \cup t_3 \cup t_4])$ satisfies the assumptions of point 1 and $(t_1 \cup t_2 \cup t_3 \cup t_5, s[t_1 \cup t_2 \cup t_3 \cup t_5])$ satisfies those of point 2, then*

$$s(t_3, t_4) < s(t_1 \cup t_2, t_3) < s(t_3, t_5).$$

Proof.

1. Because $t_1 \cup t_2 \cup t_3$ is a valid cluster, the algorithm must merge t_3 with $t_1 \cup t_2$ before any merge involving t_4. If $s(t_1 \cup t_2, t_3) \leq s(t_3, t_4)$, the merge between t_3 and t_4 could occur first, violating the validity of $t_1 \cup t_2 \cup t_3$. Hence, $s(t_1 \cup t_2, t_3) > s(t_3, t_4)$.
2. Because $t_3 \cup t_5$ is a valid cluster, the linkage must merge t_3 and t_5 before any merge involving $t_1 \cup t_2$. If $s(t_1 \cup t_2, t_3) \geq s(t_3, t_5)$, the pair $(t_1 \cup t_2, t_3)$ could merge first, violating the validity of $t_3 \cup t_5$. Thus, $s(t_1 \cup t_2, t_3) < s(t_3, t_5)$.
3. Follows directly from combining the inequalities in points 1 and 2.

□

The following lemma formalizes a key property of any valid update function f: when both its second and third arguments are equal to q, the function must output q, i.e., $f(q^+, q, q, \cdots) = q$ for all $q^+ \geq q$.

Lemma 12. *Let f be a valid update function. Consider any $(\mathcal{X}, s)$ and disjoint clusters t_1, t_2 such that $s(x, y) = q$ for all $x \in t_1, y \in t_2$. If there exists an iteration m such that $t_1, t_2 \in \mathcal{C}^{(m)}_{active}(\mathcal{X}, s)$, then*

$$s(t_1, t_2) = q.$$

Moreover, for any $q^+ \geq q$ and any $n_1, n_2, n_3 \in \mathbb{N}$,

$$f(q^+, q, q, n_1, n_2, n_3) = q.$$

Proof. We prove the claim by induction on the cluster sizes n_1, n_2.

Claim. For any n_1, n_2 and any $q \in \mathbb{R}$, there exists m such that if $t_1, t_2 \in \mathcal{C}^{(m)}_{active}$ with $|t_1| = n_1$, $|t_2| = n_2$, and $s(x, y) = q$ for all $x \in t_1, y \in t_2$, then $s(t_1, t_2) = q$.

Base case ($n_1 = n_2 = 1$). Trivial, since the similarity between singletons equals $s(x, y) = q$ by definition.

Inductive hypothesis. Assume the claim holds for all $n_1 \leq b_1$ and $n_2 \leq b_2$. Without loss of generality, assume $b_1 \geq b_2$ (since s is symmetric).

Inductive step. We prove the claim for all (n_1, n_2) with $n_1 \leq \max(b_1, 2b_2)$ and $n_2 \leq \min(b_1, 2b_2)$.

Let t_1, t_2, t_3, t_4, t_5 be disjoint clusters such that $|t_1|, |t_2|, |t_4|, |t_5| \leq b_2$ and $|t_3| \leq b_1$. Fix arbitrary real numbers $q_0 > q^+ > q + \delta$ for some $\delta > 0$. Define $\mathcal{X} = t_1 \cup t_2 \cup t_3 \cup t_4 \cup t_5$ and a similarity function $s : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ by

$$s(x,y) \;=\; \begin{cases} q_0, & (x,y) \in t_i \times t_i \text{ for some } i \in [5], \\ q^+, & (x,y) \in t_1 \times t_2, \\ q, & (x,y) \in (t_1 \cup t_2) \times (t_3 \cup t_4), \\ q+\delta, & (x,y) \in t_3 \times t_4, \\ q-\delta, & (x,y) \in (t_1 \cup t_2 \cup t_3 \cup t_4) \times t_5. \end{cases}$$

Observe that all t_i are valid clusters with respect to s. By the inductive hypothesis, at every step of the linkage process:

$$s(u,v) \;=\; \begin{cases} q_0, & u, v \subsetneq t_i, \\ q^+, & u \subseteq t_1, v \subseteq t_2, \\ q, & u \subsetneq (t_1 \cup t_2),\ v \subsetneq (t_3 \cup t_4), \\ q+\delta, & u, v \subsetneq t_3 \times t_4, \\ q-\delta, & u \subsetneq (t_1 \cup t_2 \cup t_3 \cup t_4),\ v \subseteq t_5. \end{cases}$$

Hence, there exist m_1, m_2 such that

$$t_1, t_2, t_3, t_4 \in \mathcal{C}^{(m_1)}_{active}(t_1 \cup t_2 \cup t_3 \cup t_4, s[t_1 \cup t_2 \cup t_3 \cup t_4]),$$

and

$$t_1, t_2, t_3, t_5 \in \mathcal{C}^{(m_2)}_{active}(t_1 \cup t_2 \cup t_3 \cup t_5, s[t_1 \cup t_2 \cup t_3 \cup t_5]).$$

Moreover, by the construction of s,

$$s(t_i, t_j) \;=\; \begin{cases} q^+, & (i,j) = (1,2), \\ q, & i \in \{1,2\}, j \in \{3,4\}, \\ q+\delta, & (i,j) = (3,4), \\ q-\delta, & i \in [4], j = 5. \end{cases}$$

Because $s(t_1, t_2)$ is the unique maximum among all pairwise similarities $s(t_i, t_j)$, we have:

$$t_1 \cup t_2, t_3, t_4 \in \mathcal{C}^{(m_1+1)}_{active}(t_1 \cup t_2 \cup t_3 \cup t_4, s[t_1 \cup t_2 \cup t_3 \cup t_4]),$$

and

$$t_1 \cup t_2, t_3, t_5 \in \mathcal{C}^{(m_2+1)}_{active}(t_1 \cup t_2 \cup t_3 \cup t_5, s[t_1 \cup t_2 \cup t_3 \cup t_5]).$$

Moreover, $t_3 \cup t_4$ is a valid cluster with respect to $(t_1 \cup t_2 \cup t_3 \cup t_4, s[t_1 \cup t_2 \cup t_3 \cup t_4])$, and $t_1 \cup t_2$ and $t_1 \cup t_2 \cup t_3$ are valid clusters with respect to $(t_1 \cup t_2 \cup t_3 \cup t_5, s[t_1 \cup t_2 \cup t_3 \cup t_5])$. Applying Lemma 11 (point 3) with the roles of t_4 and t_5 exchanged yields

$$s(t_3, t_5) \;=\; q - \delta \;<\; s(t_1 \cup t_2, t_3) \;<\; s(t_3, t_4) \;=\; q + \delta.$$

Because this holds for any $\delta > 0$, we conclude that

$$s(t_1 \cup t_2, t_3) \;=\; q.$$

Thus the claim holds for all relevant cluster sizes, completing the induction.

Finally, this implies that

$$f(q^+, q, q, n_1, n_2, n_3) = q$$

for all $q^+ > q$. By the right-continuity of f (Assumption 1), the equality also holds for $q^+ = q$. $\square$

C Additional Proofs

C.1 Proof of Lemma 5

We prove Lemma 5 separately for unweighted and weighted average linkages (Lemma 13), and single and complete linkages (Lemma 14).

Lemma 13. *Weighted and unweighted average linkages satisfy Conditions 3 and 4.*

Proof of Lemma 13. First, we show that weighted and unweighted average linkage satisfy Condition 3. Consider vertices t_1, t_2, t_3, t_4 such that $\forall i \in \{1,2\},\ s(t_i, t_3) > \max\left(s(t_i, t_4), s(t_3, t_4)\right)$. Denote $\eta = \frac{|t_1|}{|t_1|+|t_2|}$ for unweighted average and $\eta = 1/2$ for weighted average. We have

$$\begin{aligned}
&s(t_1 \cup t_2, t_3) - \max\left(s(t_1 \cup t_2, t_4), s(t_3, t_4)\right) \\
&= \eta s(t_1, t_3) + (1-\eta) s(t_2, t_3) - \max\left(\eta s(t_1, t_4) + (1-\eta) s(t_2, t_4), s(t_3, t_4)\right) \\
&\geq \eta\left(s(t_1, t_3) - \max\left(s(t_1, t_4), s(t_3, t_4)\right)\right) + (1-\eta)\left(s(t_2, t_3) - \max\left(s(t_2, t_4), s(t_3, t_4)\right)\right) \\
&> 0,
\end{aligned}$$

and thus Condition 3 is verified.

Now, we show that these average linkages also satisfy Condition 4. Consider vertices t_1, t_2, t_3, t_4 such that $s(t_1, t_2) > \max_{i\in\{1,2\}, j\in\{3,4\}} s(t_i, t_j)$. Then, for these vertices, we have

$$\begin{aligned}
s(t_1, t_2) - \max_{i\in\{1,2\}} s(t_i, t_3 \cup t_4) &= \min_{i\in\{1,2\}} \eta \cdot \left(s(t_1, t_2) - s(t_i, t_3)\right) + (1-\eta) \cdot \left(s(t_1, t_2) - s(t_i, t_4)\right) \\
&> 0,
\end{aligned}$$

where $\eta = \frac{|t_3|}{|t_3|+|t_4|}$ for unweighted average and $\eta = 1 - \eta = 1/2$ for weighted average. □

Lemma 14. *Single and complete linkages satisfy Conditions 3 and 4.*

Proof of Lemma 14. We first show that complete and single linkages satisfy Condition 3. Consider vertices t_1, t_2, t_3, t_4 such that $\forall i \in \{1,2\},\ s(t_i, t_3) > \max\left(s(t_i, t_4), s(t_3, t_4)\right)$. Moreover, without loss of generality, assume that $s(t_1, t_3) > s(t_2, t_3)$. Then, with complete linkage, we have

$$\begin{aligned}
s(t_1 \cup t_2, t_3) - s(t_3, t_4) &= \min\left(s(t_1, t_3), s(t_2, t_3)\right) - s(t_3, t_4) \\
&= s(t_2, t_3) - s(t_3, t_4) > 0,
\end{aligned}$$

and

$$\begin{aligned}
s(t_1 \cup t_2, t_3) - s(t_1 \cup t_2, t_4) &= \min\left(s(t_1, t_3), s(t_2, t_3)\right) - \min\left(s(t_1, t_4), s(t_2, t_4)\right) \\
&= s(t_2, t_3) - \min\left(s(t_1, t_4), s(t_2, t_4)\right) \\
&= \begin{cases} s(t_2, t_3) - s(t_1, t_4) & \text{if } s(t_1, t_4) \leq s(t_2, t_4) \\ s(t_2, t_3) - s(t_2, t_4) & \text{if } s(t_1, t_4) > s(t_2, t_4) \end{cases} \\
&\geq s(t_2, t_3) - s(t_2, t_4) > 0.
\end{aligned}$$

Thus, complete linkage satisfies Condition 3.

Similarly, for single linkage, we have

$$\begin{aligned}
s(t_1 \cup t_2, t_3) - s(t_3, t_4) &= \max\left(s(t_1, t_3), s(t_2, t_3)\right) - s(t_3, t_4) \\
&= s(t_1, t_3) - s(t_3, t_4) > 0,
\end{aligned}$$

and

$$\begin{aligned}
s(t_1 \cup t_2, t_3) - s(t_1 \cup t_2, t_4) &= \max\left(s(t_1, t_3), s(t_2, t_3)\right) - \max\left(s(t_1, t_4), s(t_2, t_4)\right) \\
&= s(t_1, t_3) - \max\left(s(t_1, t_4), s(t_2, t_4)\right) \\
&= \begin{cases} s(t_1, t_3) - s(t_2, t_4) & \text{if } s(t_1, t_4) \leq s(t_2, t_4) \\ s(t_1, t_3) - s(t_1, t_4) & \text{if } s(t_1, t_4) > s(t_2, t_4) \end{cases} \\
&\geq \begin{cases} s(t_2, t_3) - s(t_2, t_4) & \text{if } s(t_1, t_4) \leq s(t_2, t_4) \\ s(t_1, t_3) - s(t_1, t_4) & \text{if } s(t_1, t_4) > s(t_2, t_4) \end{cases} \\
&> 0.
\end{aligned}$$

Hence, single linkage satisfies Condition 3.

We now establish that complete and single linkages also satisfy Condition 4. Consider vertices t_1, t_2, t_3, t_4 such that $s(t_1, t_2) > \max_{i\in\{1,2\}, j\in\{3,4\}} s(t_i, t_j)$. For complete linkage, we have

$$\begin{aligned} s(t_1,t_2) - \max_{i\in\{1,2\}} s(t_i, t_3 \cup t_4) &= s(t_1,t_2) - \max_{i\in\{1,2\}} \min_{j\in\{3,4\}} s(t_i,t_j) \\ &\geq s(t_1,t_2) - \max_{i\in\{1,2\}} \max_{j\in\{3,4\}} s(t_i,t_j) \\ &> 0. \end{aligned}$$

Similarly, for single linkage, we have

$$\begin{aligned} s(t_1,t_2) - \max_{i\in\{1,2\}} s(t_i, t_3 \cup t_4) &= s(t_1,t_2) - \max_{i\in\{1,2\}} \max_{j\in\{3,4\}} s(t_i,t_j) \\ &> 0. \end{aligned}$$

Therefore, complete and single linkage also satisfy Condition 4. □

C.2 Proof of Lemma 6

We prove that Ward violates Condition 1 in Lemma 15, and that Centroid and Median linkages violate Condition 2 in Lemma 16. We note that even if the dissimilarity function is restricted to distance, the following arguments still hold.

Lemma 15. *Ward does not satisfy Condition 1.*

Proof. Assume the case where $|t_3| > |t_4|$. Then

$$\begin{aligned} &d(t_1 \cup t_2, t_4) - d(t_1 \cup t_2, t_3) \\ &\quad= \frac{|t_1|+|t_4|}{|t_1|+|t_2|+|t_4|} d(t_1,t_4) + \frac{|t_2|+|t_4|}{|t_1|+|t_2|+|t_4|} d(t_2,t_4) - \frac{|t_4|}{|t_1|+|t_2|+|t_4|} d(t_1,t_2) \\ &\quad\quad - \frac{|t_1|+|t_3|}{|t_1|+|t_2|+|t_3|} d(t_1,t_3) - \frac{|t_2|+|t_3|}{|t_1|+|t_2|+|t_3|} d(t_2,t_3) + \frac{|t_3|}{|t_1|+|t_2|+|t_3|} d(t_1,t_2) \\ &\quad= \frac{|t_1|+|t_3|}{|t_1|+|t_2|+|t_3|} \left(d(t_1,t_4) - d(t_1,t_3)\right) + \frac{|t_2|+|t_3|}{|t_1|+|t_2|+|t_3|} \left(d(t_2,t_4) - d(t_2,t_3)\right) \\ &\quad\quad - \frac{\left(|t_3|-|t_4|\right)\left(|t_2|\left(d(t_1,t_4) - d(t_1,t_2)\right) + |t_1|\left(d(t_2,t_4) - d(t_1,t_2)\right)\right)}{\left(|t_1|+|t_2|+|t_3|\right)\left(|t_1|+|t_2|+|t_4|\right)}. \end{aligned}$$

Here, observe that $|t_3| > |t_4|$, $d(t_1,t_4) - d(t_1,t_2) > 0$, and $d(t_1,t_2)$ can be very close to 0, while $d(t_1,t_3)$ and $d(t_2,t_3)$ can be arbitrarily close to $d(t_1,t_4)$ and $d(t_2,t_4)$, respectively. Therefore $d(t_1 \cup t_2, t_4) - d(t_1 \cup t_2, t_3)$ can take non-positive values. Moreover,

$$\begin{aligned} &d(t_3,t_4) - d(t_1 \cup t_2, t_3) \\ &\quad= d(t_3,t_4) - \eta_1 d(t_1,t_3) - \eta_2 d(t_2,t_3) - \beta d(t_1,t_2) \\ &\quad= \eta_1 \left(d(t_3,t_4) - d(t_1,t_3)\right) + \eta_2 \left(d(t_3,t_4) - d(t_2,t_3)\right) + \beta \left(d(t_3,t_4) - d(t_1,t_2)\right), \end{aligned}$$

where $\eta_1 = \frac{|t_1|+|t_3|}{|t_1|+|t_2|+|t_3|}$, $\eta_2 = \frac{|t_2|+|t_3|}{|t_1|+|t_2|+|t_3|}$, and $\beta = -\frac{|t_3|}{|t_1|+|t_2|+|t_3|}$. Observe that $\beta < 0$, $d(t_3,t_4) - d(t_1,t_2) > 0$, and $d(t_1,t_2)$ can be arbitrarily close to 0 while $d(t_1,t_3)$ and $d(t_2,t_3)$ can be arbitrarily close to $d(t_3,t_4)$. Therefore $d(t_3,t_4) - d(t_1 \cup t_2, t_3)$ can take non-positive values. Consequently, Ward does not always respect Condition 1.

□

Lemma 16. *Median and Centroid linkages do not satisfy Condition 2.*

Proof. Without loss of generality, assume $d(t_1, t_3 \cup t_4) < d(t_2, t_3 \cup t_4)$

$$\begin{aligned} \min_{i\in\{1,2\}} d\left(t_i, t_3 \cup t_4\right) - d(t_1,t_2) &= d(t_1, t_3 \cup t_4) - d(t_1,t_2) \\ &= \eta d(t_1,t_3) + (1-\eta) d(t_1,t_4) + \beta d(t_3,t_4) - d(t_1,t_2) \\ &= \eta \left(d(t_1,t_3) - d(t_1,t_2)\right) + (1-\eta)\left(d(t_1,t_4) - d(t_1,t_2)\right) + \beta d(t_3,t_4), \end{aligned}$$

where $\eta = \frac{|t_3|}{|t_3|+|t_4|}$ ($1-\eta = \frac{|t_4|}{|t_3|+|t_4|}$), and $\beta = -\frac{|t_3||t_4|}{(|t_3|+|t_4|)^2}$ for Centroid method, $\eta = 1-\eta = 1/2$ and $\beta = -1/4$ for Median. Here, as $\beta < 0$, and $d(t_1, t_3)$ and as $d(t_1, t_4)$ can be very close to $d(t_1, t_2)$, $\min_{i\in\{1,2\}} d(t_i, t_3 \cup t_4) - d(t_1, t_2)$ can be non-positive. As a result, Median and Centroid do not always respect Condition 2. □

Finally, Figure 7 provides an explicit numeric example where Ward fails to return the finest valid hierarchy. Note that d satisfies triangular inequality, and thus is a distance function.

	x_1	x_2	x_3	x_4	x_5
x_1	0	4	15	15	16
x_2	4	0	15	15	16
x_3	15	15	0	3	21
x_4	15	15	3	0	21
x_5	16	16	21	21	0

Figure 4: Dissimilarity d

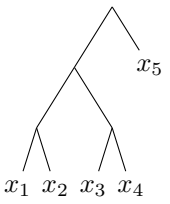


Figure 5: $T_*(\mathcal{X}, d)$

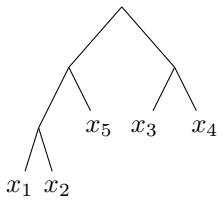


Figure 6: $T_{ward}(\mathcal{X}, d)$

Figure 7: Figure 4 gives a dissimilarity function over a set of 5 items. The valid hierarchy is provided in Figure 5, and Figure 6 shows the tree constructed by Ward linkage.

C.3 Proof of Proposition 7

Fix the current dissimilarity d (in a run of the linkage algorithm with rule F, this is the dissimilarity produced by F so far). For disjoint clusters t_1, t_2, t_3, let $d_j(t_1 \cup t_2, t_3) = f_j\big(d(t_1, t_2), d(t_2, t_3), d(t_3, t_1), |t_1|, |t_2|, |t_3|\big)$ be the one-step update by f_j, evaluated on these common inputs. Since $F = \sum_j \alpha_j f_j$, the one-step update by F is $d_F(t_1 \cup t_2, t_3) = \sum_{j=1}^m \alpha_j d_j(t_1 \cup t_2, t_3)$. Note that d_F is in general not the average of the dissimilarities obtained by running the linkage algorithm separately with each f_j.

Assume the premise of Condition 1 holds *i.e.,* $\forall i \in \{1, 2\}\colon d(t_i, t_3) < \min(d(t_i, t_4), d(t_3, t_4))$. Because every f_j satisfies Condition 1, we have

$$\forall j \in \{1, 2, \cdots, m\}\colon d_j(t_1 \cup t_2, t_3) < \min\left(d_j(t_1 \cup t_2, t_4), d(t_3, t_4)\right).$$

This implies two inequalities for each $j \in \{1, 2, \cdots, m\}$:

1. $d_j(t_1 \cup t_2, t_3) < d_j(t_1 \cup t_2, t_4)$;
2. $d_j(t_1 \cup t_2, t_3) < d(t_3, t_4)$.

Multiplying (1) by α_j and summing over j yields

$$d_F(t_1 \cup t_2, t_3) < \sum_{j=1}^m \alpha_j\, d_j(t_1 \cup t_2, t_4) = d_F(t_1 \cup t_2, t_4).$$

Multiplying (2) by α_j and summing over j yields:

$$d_F(t_1 \cup t_2, t_3) < \sum_{j=1}^m \alpha_j\, d(t_3, t_4) = d(t_3, t_4).$$

Combining these two inequalities yields

$$d_F(t_1 \cup t_2, t_3) < \min(d_F(t_1 \cup t_2, t_4), d(t_3, t_4)).$$

Therefore, F satisfies Condition 1.

A similar argument also holds for Condition 2, and hence F also satisfies Condition 2.

C.4 Experimental details

Benchmarks. Iris, Wine, Breast Cancer Wisconsin (Diagnostic) and Digits are loaded from scikit-learn (Pedregosa et al., 2011). We remove exact duplicate feature vectors, which is needed because our assumptions on d exclude $d(x, y) = 0$ for $x \neq y$; this removes one point from Iris ($k = 149$) and none from the other datasets. Features are used without rescaling. The cosine dissimilarity is $1 - \cos(x, y)$, a strictly decreasing function of the cosine similarity, which therefore has the same valid clusters.

Linkage and ties. Linkage trees are computed with `scipy.cluster.hierarchy.linkage` (Virtanen et al., 2020), which breaks ties deterministically. By Theorem 4, the pruned hierarchy does not depend on how ties are broken for single, complete and average linkage.

Planted hierarchy. $\rho \in \{0.1a : a \in [0, 10]\}$, 50 draws per level, seed 20260801. The same 50 draws $U^{(1)}, \ldots, U^{(50)}$ are used at every noise level, so only the amplitude 4ρ changes with ρ. Perturbed entries are floored at machine precision, which is only active for $\rho \geq 1/2$. The unperturbed d_0 has ties (all within-fine-cluster distances equal 2, etc.), so the binary tree at $\rho = 0$ is SciPy's tie-broken single-linkage tree.

CIFAR-100. Ten images per fine class are drawn without replacement from the test split with seed 20260801 ($k = 1000$). Raw pixels are scaled to $[0, 1]$ and flattened (3072 dimensions). DINOv2 ViT-S/14 features (Oquab et al., 2024) are computed with the official pretrained weights after resizing to 224×224 (bicubic, antialiased) and ImageNet mean/std normalization; we use the output of the model's forward pass, i.e. the final-layer class token after layer normalization. Cosine dissimilarities use ℓ_2-normalized features and Euclidean distances use unnormalized ones.

Assets and licenses. Iris (Fisher, 1936), Wine (Aeberhard and Forina, 1992), Breast Cancer Wisconsin (Diagnostic) (Wolberg et al., 1993) and Optical Recognition of Handwritten Digits (Alpaydin and Kaynak, 1998) are distributed by the UCI Machine Learning Repository under the CC BY 4.0 license; we load them through scikit-learn (BSD-3-Clause license), whose Digits data is the test split of the UCI dataset ($k = 1797$). CIFAR-100 (Krizhevsky, 2009) is distributed by its authors without an explicit license, and we use its test split for research purposes only. The DINOv2 code and pretrained weights (Oquab et al., 2024) are released under the Apache 2.0 license (`https://github.com/facebookresearch/dinov2`).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and the introduction clearly state all the results of the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 contains a dedicated paragraph, "Limitations and future work", which discusses that validity is deliberately conservative (meaningful clusters may be invalid), that valid clusters with a small margin are sensitive to perturbations (Appendix A.6), that our guarantees are deterministic rather than statistical, and that learning linkage rules is left open.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All theorems are carefully stated and the assumptions are also explained and discussed.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides all detail and data sets to reproduce our experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All data sets and the pretrained DINOv2 weights are publicly available, and the code reproducing all experiments is available at `https://github.com/daichikuroda/Recovering_Apresjan` (see Section 4).

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, we do specify all the details necessary to understand and to reproduce our experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, we do report the error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Our experiments were run on a standard laptop.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work fully conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: We do not directly discuss these impacts in the paper, as our main contribution is mostly a theoretic one.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and the pretrained model are cited, and their licenses are stated in Appendix C.4: the four UCI datasets are under CC BY 4.0 (loaded via scikit-learn, BSD-3-Clause), DINOv2 code and weights are under Apache 2.0, and CIFAR-100 is distributed without an explicit license and used for research only.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components. We only used LLM for spell-checking and writing improvement.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.