ARE YOU DENSE? EXPLAINABILITY IN CLUSTERING CONSIDERING DENSITY-BASED STRUCTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Explainable clustering methods have focused almost entirely on centroid-based reference clusterings with mostly convex cluster shapes. However, many real-world datasets contain clusters of non-convex, arbitrary shapes that are typically found with density-based clustering methods. We address this gap with our novel explainability framework **LUCID** (exp**L**aining cl**U**ster stru**C**tures v**I**a **D**ensity-aware trees). LUCID explains a given density-based clustering by fitting an interpretable tree over its density-connectivity structure, combining sparse feature rules with a case-based explanation for structure no such simple rule can separate. We demonstrate LUCID's applicability in extensive experiments, highlighting its necessity for density-based reference clusterings and datasets containing density-connected structures.

1 INTRODUCTION

Clustering methods group data by similarity without access to ground truth labels. Among clustering paradigms, density-based methods such as DBSCAN (Ester et al., 1996) and HDBSCAN (Campello et al., 2013) play an important role: unlike centroid-based methods, they recover non-convex shapes, do not require the number of clusters to be known and fixed in advance, and explicitly label data objects that cannot be assigned to any cluster as noise. These properties make density-based methods particularly relevant for domains like geospatial analysis (Kamh et al., 2024), astrophysics (Zhang, 2019), or single-cell biology (Petegrosso et al., 2020), where clusters are data objects in densely connected areas that are separated by sparser areas, allowing for arbitrary shapes. However, like other clustering approaches, density-based clusterings provide no inherent explanation for their assignments: a data object's cluster membership emerges from the density-connectivity structure of the data rather than from any single interpretable rule, leaving practitioners with a labeling but no reasoning for why the assignment was produced.

Several paradigms have been proposed to explain a clustering post hoc, including rule-based (Saisubramanian et al., 2020; Carrizosa et al., 2023), prototype-based (Carrizosa et al., 2022; Davidson et al., 2024), convex-polyhedral (Lawless & Gunluk, 2023; Chen et al., 2016), and multi-hyperrectangle (Zeng et al., 2024) descriptions. Among these, decision-tree-based explanations (Moshkovitz et al., 2020) are the dominant paradigm, since a tree provides an inherently hierarchical, traceable explanation whose reading cost is bounded by its depth. Moshkovitz et al. (2020) and follow-up work (Laber & Murtinho, 2021; Makarychev & Shan, 2021; Gamlath et al., 2021; Esfandiari et al., 2022; Gupta et al., 2023) establish tight bounds on the price of explainability, the unavoidable cost gap between a reference clustering and its best tree-induced approximation, for centroid-based objectives such as k-means and k-median, and SpEx (Argov & Wagner, 2025) extends this to spectral clustering via a general graph-conductance criterion. Both settings optimize a global objective, whether their clusters are convex cells around representative centers (centroid-based) or possibly non-convex partitions of a similarity graph (spectral), and it is this objective that makes the price of explainability well-defined. Density-based clustering has no such objective: its clusters are defined by density-reachability rather than by distance to a representative, can take arbitrary shapes, and may leave points unassigned as noise. Consequently, none of the existing guarantees carry over.

Density-based clustering has so far been explained only at the level of individual data objects (*local explanation*): ExDBSCAN (Matthews et al., 2026) generates counterfactual explanations that move an individual DBSCAN inlier or outlier across a density-connected graph to a target cluster, and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

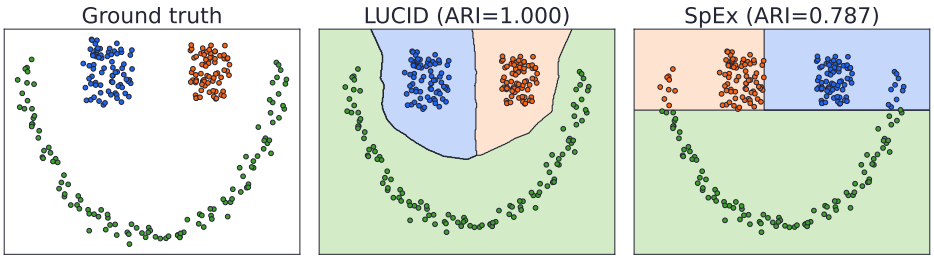


Figure 1: **Ground truth vs. LUCID vs. SpEx explanations on the Zelnik3 benchmark.** A smiley face consisting of a curved cluster (green) and two centroid-based clusters (blue, red). LUCID separates the three perfectly (ARI$=1.000$); SpEx fails to explain the bent shape of the arc and instead merges parts of it into both other clusters (ARI$=0.787$).

similarly DCFO (Amico et al., 2026) for Local Outlier Factor outliers via locally smooth density regions. Neither says anything about the clustering as a whole. The *global* explanation of density-based clustering, that is, describing the entire clustering through a compact, interpretable structure, has, to the best of our knowledge, not been studied at all.

We close this gap with LUCID (exp**L**aining cl**U**ster stru**C**tures v**I**a **D**ensity-aware trees), our novel explanation framework that grounds explanations in density-connectivity rather than in centroids or generic graphs. LUCID works with any given reference clustering and fits a single global decision tree whose leaves approximate the given labeling. To find the tree's splits, we study the reference clustering labels under the density-connectivity distance (dc-dist) (Beer et al., 2023), an ultrametric on the mutual-reachability graph. For each node, LUCID selects the simplest explanation that faithfully realizes the cut: feature-space rules, linear or signomial, or a case-based split defined directly in terms of density-connectivity. Figure 1 illustrates the difference to the state-of-the-art: even SpEx, which is designed explicitly for non-convex structure, cannot follow a density-connected arc due to its reliance on axis-aligned cuts, whereas LUCID's rules explain the reference labeling perfectly.

Our main contributions are:

- We introduce LUCID, the first density-based explainability framework that captures and explains clusters of arbitrary shape.
- LUCID explains a clustering through an interpretable decision tree whose splits are derived from both the data structure and the reference labels. Each split follows an interpretability hierarchy: it prefers simple feature rules, either linear or signomial, and falls back to density-connectivity only when no feature rule is expressive enough to realize the cut.
- In extensive experiments across 67 datasets and five reference clusterings, we demonstrate that LUCID achieves significantly higher alignment with reference clusterings than existing explainable-clustering methods.

2 Related Work

Explainable Clustering Existing approaches for explainable clustering fall into rule-based methods Saisubramanian et al. (2020); Carrizosa et al. (2023), prototype-based methods, Carrizosa et al. (2022); Davidson et al. (2024), convex-polyhedral methods, Lawless & Gunluk (2023); Chen et al. (2016), and multi-hyperrectangle descriptions Zeng et al. (2024). We focus on the most dominant paradigm, decision-tree based methodsMoshkovitz et al. (2020), where the tree offers an inherently hierarchical structure, with traceable explanations, where the number of conditions a user must read to understand cluster assignments is tightly bounded by the depth of the tree. For a broader overview, see Hu et al. (2026).

Decision-Tree based. Tree-based explainable clustering works can be grouped into two lines of works (a) axis-aligned, which build trees that split on a single feature threshold at each node, and (b) oblique cuts, which split on (typically sparse) linear combinations of features.

Axis-aligned. Moshkovitz et al. (2020) introduce explainable k-means/ k-medians via axis-aligned trees, with subsequent work sharpening the cost of explainability bounds and algorithms in this setting (Laber & Murtinho, 2021; Makarychev & Shan, 2021; Gamlath et al., 2021; Esfandiari et al., 2022; Gupta et al., 2023). These guarantees rely on the reference cost being convex over Euclidean cluster centers, and do not carry over to a density-connected reference, which might feature elongated clusters or clusters that are not pairwise linearly separable. Other works in this line expand the tree to more than k leaves (Frost et al., 2020), treat the construction as a Mixed-Integer-Optimization problem (Bertsimas et al., 2021), fit the clustering and the explainable tree jointly rather than post-hoc (Hwang & Whang, 2023), or extend it to kernel clustering (Fleissner et al., 2024). SpEx (Argov & Wagner, 2025) keeps the same axis-aligned, single coordinate split but replaces the fixed reference objective with a general graph-conductance criterion.

Oblique. Lawless et al. (2022) construct polytopes around each cluster via a Mixed-Integer Non-Linear Program, constraining the polytopes to be axis-parallel or sparse with integer coefficients. Gabidolla & Carreira-Perpiñán (2022) train sparse oblique trees with Tree Alternating Optimization (TAO). However, their approach is limited to clustering methods that are defined by an optimization problem of a cost function.

Density-connectivity. Density-connectivity has been used for explanations, but only at a different level for local explanations: Matthews et al. (2026) (ExDBSCAN) generate counterfactual explanations for individual DBSCAN inliers/outliers directly on a density-connected weighted graph, and Amico et al. (2026) (DCFO) similarly for Local Outlier Factor outliers via locally smooth density regions. Both answer the counterfactual question "why is this data object noise" at the level of a single data object rather than at the global level of the entire clustering.

3 LUCID - DENSITY-AWARE EXPLANATION TREES

The following subsections present LUCID, our framework for explaining a given density-based clustering. LUCID takes a fixed reference clustering $\mathscr{C}_{\text{ref}}$, e.g., from DBSCAN, and constructs an interpretable explanation tree that approximates it. Each node of this tree splits the data according to its density-connectivity structure and provides an explanation for that split.

After introducing notation and formalizing the problem (Section 3.1), we describe how we derive these splits from the density-connectivity dendrogram (Section 3.2). In short, we fit a minimum spanning tree (MST) on the mutual-reachability distance, as in density-based methods such as OPTICS or HDBSCAN. Wherever possible, each split in our explanation tree follows a cut in the single-linkage hierarchy induced by this MST. Each node then receives one of three explanation types (Section 3.3): (1) linear rules, in which features contribute additively; (2) signomial rules, i.e., hyperplanes in log-space, in which features combine multiplicatively, capturing ratios and power laws; and (3) connectivity rules, which explain by density-connectivity to a real example, for complex arbitrarily shaped clusters. Figure 2 visualizes our approach. Section 3.4 presents the algorithm that assembles these nodes into a tree, and Section 3.5 discusses complexity and limitations.

3.1 NOTATION AND PROBLEM STATEMENT

Let $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ be a finite dataset with $n \in \mathbb{N}$ data objects of dimensionality $d \in \mathbb{N}$ and $\mathscr{C}_{\text{ref}} = \{C_1, \ldots, C_k, \mathcal{N}\}$ a fixed reference clustering with k clusters and potential noise objects $\mathcal{N}$, produced by some density-based method, e.g., DBSCAN (Ester et al., 1996) or HDBSCAN Campello et al. (2013). Let $X' = X \setminus \mathcal{N}$ and $\hat{C}_x \in \{C_1, \ldots, C_k\}$ the reference cluster of $x \in X'$. Following (Moshkovitz et al., 2020), our goal is to identify an *explanation tree* $\mathcal{T}^*$: a binary tree whose internal nodes partition the data by a single rule and whose leaves correspond to clusters. places each non-noise data object $x \in X'$ in exactly one leaf $\text{leaf}(x) \in \text{leaves}(\mathcal{T})$ and induces a labeling $\ell_{\mathcal{T}} : \text{leaves}(\mathcal{T}) \to \{C_1, \ldots, C_k\}$ that assigns each leaf its plurality cluster, i.e., the reference cluster to which most of its data objects belong. For fixed leaves, this labeling minimizes the cost below. We write $\mathcal{T}_k$ for the set of all explanation trees with at most k leaves.

Definition 1 (Explanation Tree)**.** For a given reference clustering $\mathscr{C}_{\text{ref}}$, the explanation tree $\mathcal{T}^*$ is a tree with at most k leaves inducing a clustering that matches $\mathscr{C}_{\text{ref}}$ best. The cost of a tree $\mathcal{T}$, is the

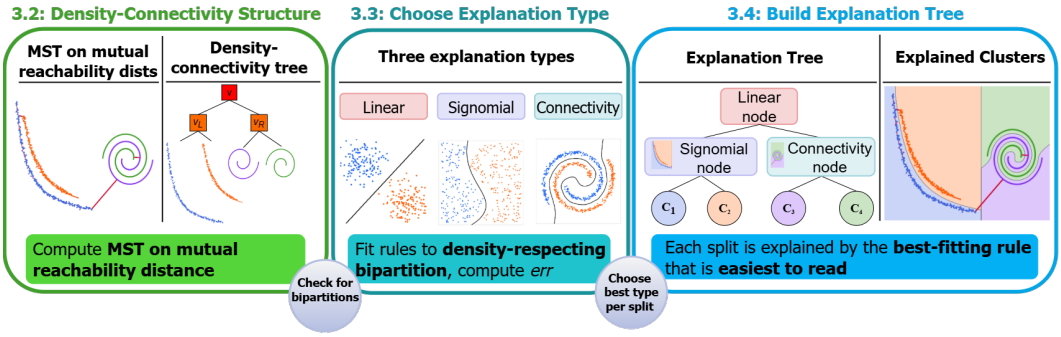


Figure 2: **Overview of LUCID.** (1) We fit a minimum spanning tree on the data to get the density-connectivity hierarchy $\mathcal{D}$. (2) A bipartite cut on this hierarchy defines one split of the explanation tree that can be linear, signomial, or connectivity-based. (3) For each split, LUCID chooses the explanation that best reproduces it, preferring linear over signomial over connectivity nodes among equally accurate ones. Repeating steps (2) and (3) yields the explanation tree (right), whose leaves correspond to the reference clusters.

fraction of data objects whose leaf label differs from their reference cluster,

$$\mathrm{cost}(\mathcal{T}, \mathscr{C}_{\mathrm{ref}}) = \frac{1}{|X'|} \sum_{x \in X'} \mathbb{I}[\hat{C}_x \neq \ell_{\mathcal{T}}(\mathrm{leaf}(x))], \tag{1}$$

and the explanation tree is $\mathcal{T}^* = \arg\min_{\mathcal{T} \in \mathcal{T}_k} \mathrm{cost}(\mathcal{T}, \mathscr{C}_{\mathrm{ref}})$.

Each node v in $\mathcal{T}$ holds the data objects routed to it, $X_v \subseteq X'$, and $\mathscr{C}_v$ denotes the reference clusters present in X_v.

Noise Handling. Density-based clustering algorithms label data objects that cannot be assigned to any cluster as noise. Noise data objects cannot be represented faithfully within a leaf, so we filter noise $X' = X \setminus \mathcal{N}$ before computing. Explaining noise objects specifically is out of scope here; we refer to methods like ExDBSCAN (Matthews et al., 2026) or DCFO (Amico et al., 2026) that explain noise objects locally rather than trying to find a potentially misleading global explanation.

3.2 THE DENSITY-CONNECTIVITY DENDROGRAM

The structure of our explanation tree is fit to the density-based structure of the data. For this, we regard the MST on the complete graph on X' whose edge weights are the pairwise mutual reachability distances (mrd), where

$$\mathrm{mrd}(x, y) = \max(\kappa_\mu(x), \kappa_\mu(y), \mathrm{dist}(x, y))$$

for a given $\mu \in \mathbb{N}$ and where κ_μ is the core-distance, i.e., the distance of a data object to its $(\mu - 1)$-th nearest neighbor and dist is the Euclidean distance. This MST is originally described in HDBSCAN (Campello et al., 2013), and the minmax path distances on it describe an ultrametric introduced in Beer et al. (2023) as the density-connectivity distance d^{μ}_{dc} (dc-dist). This ultrametric induces the density-connectivity hierarchy (dendrogram) $\mathcal{D}$ on X', and we write $\mathcal{D}_v$ for its restriction to the data objects X_v at node v. $\mathcal{D}_v$ is the basis for any split in our explanation tree $\mathcal{T}$.

Density-respecting bipartitions. Each node v in $\mathcal{T}$ with $|\mathscr{C}_v| \geq 2$ separates the reference clusters present at v, $\mathscr{C}_v$, into two groups: $\mathscr{S}_v$ and its complement $\overline{\mathscr{S}_v}$. Where possible, this bipartition is obtained by a cut in $\mathcal{D}_v$, which corresponds to removing an edge from the MST. We call the bipartition *density-respecting* if every $C_i \cap X_v$ with $C_i \in \mathscr{C}_v$ lies entirely on one side of the cut. Equivalently, $X_{\mathscr{S}_v} = \{x \in X_v : \hat{C}_x \in \mathscr{S}_v\}$ is itself a node of $\mathcal{D}_v$, and write $\mathcal{B}_v$ for the set of density-respecting bipartitions at v.

Choosing the cut. If several density-respecting bipartitions exist at v, we choose the one that is highest in the cluster hierarchy $\mathcal{D}$ (i.e., that removes the longest edge in the MST). Ties are broken deterministically (Appendix A.1). If $\mathcal{B}_v = \varnothing$, i.e., no density-respecting bipartition exists at a node v, every node of $\mathcal{D}_v$ that separates clusters also cuts through at least one of them, so no cut in the hierarchy matches the reference. So instead of following a cut in $\mathcal{D}_v$, we choose $\mathscr{S}_v$ s.t. it corresponds exactly to the largest cluster in the current node: $\mathscr{S}_v = \arg\max_{C_i \in \mathscr{C}_v} |C_i \cap X_v|$ and, thus, $\overline{\mathscr{S}_v}$ contains all remaining clusters occurring in $\mathscr{C}_v$. Either way, the bipartition $B_v^* = (\mathscr{S}_v, \overline{\mathscr{S}_v})$ that we choose divides the reference clusters present at v into two groups, where $\mathscr{S}_v$ is a set of clusters, not necessarily a single one.

Nodes at lower levels reuse the dc-dists without recomputation, since their restriction to any subset is again an ultrametric.

3.3 Types of node-wise explanations

At each node v with $|\mathscr{C}_v| \geq 2$, LUCID explains the chosen bipartition B_v^* by a readable rule r_v. As density-based clusters are often non-convex, the typical axis-aligned thresholds used by other methods are insufficient, whereas richer rules are harder to read. LUCID therefore offers three rule types of decreasing directness (Table 1): *linear* rules (lin), in which features contribute additively; *signomial* rules (sig), adapted from the explainable classifiers of Lumadjeng et al. (2026), in which features combine multiplicatively, capturing power laws, ratios, and roots common in scientific equations; and *connectivity* rules (con), which explain by density-connectivity to a real example rather than by features. All three are fit against the same per-object target, and the most accurate one is kept, preferring simpler rules among equally accurate ones.

Target and error. To find explanations for a node v that reproduce the chosen bipartition $\mathcal{B}_v^*$, we fit a decision rule to the objects rather than the full clusters. For this, we translate the chosen cut into a label for every data object that serves as the target of an oracle and is simply given by the side of the cut in which the data object falls (left resp. right for $\mathscr{S}_v$ resp. $\overline{\mathscr{S}_v}$).

Definition 2 (Oracle target)**.** Given a bipartition $B_v^* = (\mathscr{S}_v, \overline{\mathscr{S}_v})$ at node v, the oracle target maps each data object $x \in X_v$ to

$$y_v(x) = \mathbb{I}[\hat{C}_x \in \mathscr{S}_v] = \begin{cases} 1, & \hat{C}_x \in \mathscr{S}_v, \\ 0, & \hat{C}_x \in \overline{\mathscr{S}_v}. \end{cases} \tag{2}$$

By construction, y_v is constant on each cluster and takes both values on X_v whenever $|\mathscr{C}_v| \geq 2$. If B_v^* is density-respecting, y_v reproduces the density-based cut exactly; otherwise, it separates the largest cluster from the rest (Section 3.2).

Each rule r splits X_v into the data objects that satisfy it and the remaining ones, and sends one group to the left child v_L and the other to the right child v_R. We choose the assignment that agrees better with y_v. Since data objects with $y_v(x) = 1$ belong to v_L, the *error* of r is the fraction of X_v that ends up in the wrong child,

$$\mathrm{err}_v(r) = \frac{\big|\{x \in X_{v_L} : y_v(x) = 0\}\big| + \big|\{x \in X_{v_R} : y_v(x) = 1\}\big|}{|X_v|}. \tag{3}$$

A rule is *valid* if X_{v_L} and X_{v_R} are both non-empty. To keep linear and signomial rules simple and explainable, users can set a maximum number of features $j_{\max}$ as an upper limit to the size of the set of attributes J that can be used for the rule: $|J| \leq \min(j_{\max}, d)$.

Linear rules. The most direct explanation is a sparse hyperplane $\sum_{j \in J} \beta_j x_j \leq c$ over at most $j_{\max}$ features J, where x_j denotes the j-th feature of a data object x (not the j-th data object). We fit it by logistic regression against the oracle target $y_v(x)$:

$$(\omega_J, \beta_J) = \arg \min_{\omega, \beta:\, \beta_j = 0 \ (j \notin J)} \sum_{x \in X_v} \mathcal{L}_{\log}\big(y_v(x),\, \omega + \beta^\top u(x)\big) + \tfrac{\lambda_0}{2}\|\beta\|_2^2, \tag{4}$$

with $u(x) = x$ and a small fixed λ_0, which makes the fit unique. The score threshold τ_J is placed at the midpoint between consecutive sorted scores that minimizes the misclassification rate $\mathrm{err}(J)$ against y_v, and $c = \tau_J - \omega_J$.

Table 1: Explanation types LUCID offers at each node, from most to least direct. For lin, $c = \tau - \omega$; for sig, $c = e^{\tau-\omega}$ (intercept ω, score threshold τ).

Type	Rule reads as	Explains by
linear (lin)	$\sum_{j\in J} \beta_j x_j \le c$	few features, additive
signomial (sig)	$\prod_{j\in J} x_j^{\beta_j} \le c$	few features, multiplicative
connectivity (con)	$d^{\mu}_{\mathrm{dc}}(x, \mathbf{z}) < \varepsilon$	density-connectivity to a real example

Signomial rules. To capture multiplicative relationships such as ratios and power laws, we fit Equation (4) with $u(x) = \log x$ instead. Then, a sig score $\omega + \beta^\top \log x$ is the logarithm of a single-term power law $e^\omega \prod_j x_j^{\beta_j}$, in the spirit of the explainable classifiers of Lumadjeng et al. (2026). Negative exponents express inverse relationships, and fractional exponents express roots. Since signomial rules operate in log-space, all feature rules are fit on a per-feature affine rescaling $X^+ \subset [1, 10]^d$ of the data (Appendix B).

Sparse support selection. For both rule types, the set of used attributes J is chosen by FORWARDSELECT (Algorithm 2): starting from $J = \varnothing$, add the feature whose fit has the lowest err (ties broken by smallest index), up to a fixed cap $j_{\max}$. Let $t = |J|$ be the support size after t such rounds, so $t \in \{1, \ldots, \min(j_{\max}, d)\}$. Among these nested supports, we return the smallest t with $\mathrm{err} = 0$; if none exists, we fall back to the Bayesian Information Criterion (BIC) (Schwarz, 1978), the standard fit-vs-complexity trade-off: the t minimizing $2L^{(t)} + (t+1)\ln|X_v|$, where $L^{(t)}$ is the negative log-likelihood, summed over all data objects in X_v, of the fit at that support, and $t+1$ counts its fitted parameters – the t selected coefficients plus the intercept. Concretely, adding one more feature is only worth it if it lowers $2L^{(t)}$ by more than $\ln|X_v|$, the cost BIC charges per extra parameter. Cardinality is thus limited by a cap rather than a tuned penalty, and never trades against exactness: an exact fit at any size $t \le j_{\max}$ on the greedy path always wins over a larger or inexact one.

Connectivity rules. When FORWARDSELECT finds no sparse feature rule that reproduces y_v, e.g., for nested or entangled clusters, we explain by example: instead of thresholding a coordinate score, a connectivity node (con) asks whether a data object is close enough in the d^{μ}_{dc}-metric to one fixed real example $\mathbf{z}$. To find such a rule, we search over every node h of the density-connectivity hierarchy $\mathcal{D}_v$. LUCID takes the node $h^\star$ that minimizes err_v (Equation (3)), ties broken by node with fewer data objects $|X_h|$, then smallest member index, and reports the rule "$d^{\mu}_{\mathrm{dc}}(x, \mathbf{z}) < \varepsilon$" where $\mathbf{z}$ is the d^{μ}_{dc}-medoid of all nodes in the subtree with $h^\star$ as a root and ε is the height at which $h^\star$ merges with its siblings in $\mathcal{D}_v$. Note that $h^\star$ is independent of the chosen example $\mathbf{z} \in X_h$ because of the ultrametric properties in $\mathcal{D}$ (Lemma 1, Appendix A.3). We report the medoid as the most central real example.

Proposition 1 (Connectivity realizes any density-respecting target)**.** *If $\mathcal{B}_v \neq \varnothing$, then $X_{\mathscr{S}_v} = \{x \in X_v : \hat{C}_x \in \mathscr{S}_v\}$ is exactly given by a node h in $\mathcal{D}_v$ on which the rule "$x \in X_h$" coincides with y_v; hence the connectivity rule attains* $\mathrm{err}_v = 0$.

Proposition 1 (Proof in Appendix A.3) makes the rule type con a safety net: whenever $\mathcal{B}_v \neq \varnothing$, zero error is always attainable at v, and a feature rule is preferred only if it attains it as well.

Selecting the type. At each node, LUCID fits lin, sig, and con against y_v and keeps the valid rule minimizing, lexicographically,

$$\big(\mathrm{err}_v,\ \mathrm{size}\big), \tag{5}$$

where the size is $|J|$ for a feature rule and $j_{\max} + 1$ for a connectivity rule; a remaining tie between lin and sig is broken in favor of lin. Error comes first, so a less direct rule is used only when it reproduces y_v strictly better. Whenever $\mathcal{B}_v \neq \varnothing$, the best achievable error is 0 by Proposition 1. At such nodes, con is chosen exactly when FORWARDSELECT finds no exact rule with at most $j_{\max}$ features, neither in the original features nor in log-space. Connectivity nodes are thus diagnostic: they mark splits that are clean in the density hierarchy but not expressible by a sparse feature rule. If no type is valid, we fall back to a logistic fit on all features, or make the node a leaf. However, this never occurred in our experiments.

3.4 THE LUCID TREE

LUCID constructs the tree with the identified node-wise rules. The root holds all non-noise data objects X', and a node v is split in three steps: (1) choose the bipartition B_v^* of the clusters present at v (Section 3.2), (2) derive the target y_v and select the rule r_v with the lowest error err_v (Section 3.3), and (3) split X_v into the children v_L and v_R according to r_v.

LUCID grows the tree best-first. A leaf is *splittable* if it contains data objects of at least two reference clusters (a rule always exists, if necessary the ridge fallback of Section 3.3). When a node v is created, its rule r_v and error err_v are computed once, and splittable leaves are kept in a priority queue ordered by err_v, with ties broken by creation order. Splitting the leaf with the smallest error first spends the budget of $k-1$ clean splits before inexact ones. Growth stops when the tree has k leaves or no leaf is splittable. Each leaf is labeled with its plurality cluster, i.e., the reference cluster to which most of its data objects belong.

Exact recovery.

Corollary 1 (Exact recovery). If every reference cluster C_i is a node of $\mathcal{D}$, LUCID returns a tree with exactly k leaves and $\mathrm{cost}(\mathcal{T}, \mathscr{C}_{\mathrm{ref}}) = 0$.

Every split is then density-respecting, so Proposition 1 guarantees an exact rule, and exact rules pass whole clusters to their children, rendering $k-1$ splits sufficient (proof in Appendix A.3). The condition holds, in particular, for DBSCAN* clusterings of X' in which every data object is core (Lemma 2, Appendix A.3).

3.5 DISCUSSION

Complexity. The distance matrix d_{dc}^{μ} is computed once in $O(n^2 d)$ time and $O(n^2)$ memory, and each node v only slices it. With $n_v = |X_v|$, the per-node cost is dominated by building the single-linkage dendrogram and slicing the dc-dist matrix to X_v, both $O(n_v^2)$; the feature rules, oracle, and connectivity search cost only $\tilde{O}(n_v)$ up to factors in $j_{\max}$, d, and $|\mathscr{C}_v|$. Since the active sets at a given level partition X', each level costs $O(n^2)$, and the full tree costs $O(height \cdot n^2)$ for $height \leq k-1$. As usually $height \ll n$, this is effectively quadratic in n. Pseudocode is provided in Appendix A.2.

Limitations. First, the signomial cuts require $\log$ to be well-defined, thus the provided data needs to be rescaled to be positive. While linear rules translate back to X-space exactly (Section 3.3), the power-law reading of a signomial rule is exact only in X^+-space, since the additive shift of the rescaling keeps $\log X^+$ from reducing to $\log X$ plus a constant. LUCID's exactness guarantee (Corollary 1) covers references whose clusters are given by nodes of $\mathcal{D}$, e.g., DBSCAN* clusterings without noise. For other references, nodes with $\mathcal{B}_v = \varnothing$ fall back to separating the largest cluster, and no rule type is then guaranteed to reproduce this target exactly. Beyond the exact-recovery case, we give no guarantee on how close LUCID's tree comes to the optimal explanation tree $\mathcal{T}^*$. A price-of-explainability bound in the sense of Moshkovitz et al. (2020)'s remains open. Their analysis relies on a center-based objective such as k-medians, whose cost is a sum over data objects, so that the extra cost of a tree can be charged to the data objects its cuts separate from their reference center. Density-based clusterings optimize no such objective, and under our misclassification cost the reference itself has cost zero; an analogous bound would first need a density-based clustering objective.

4 EXPERIMENTS

We evaluate LUCID across a range of datasets, reference clusterings, competitors, and metrics. We provide our code online: `https://anonymous.4open.science/r/LUCID-2424/README.md`. Additional experiments are provided in Appendix D.

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate LUCID on 67 synthetic and real-world datasets, listed in Table 13 in the appendix. Synthetic data is generated with DENSIRED (Jahn et al., 2024), a density-based data

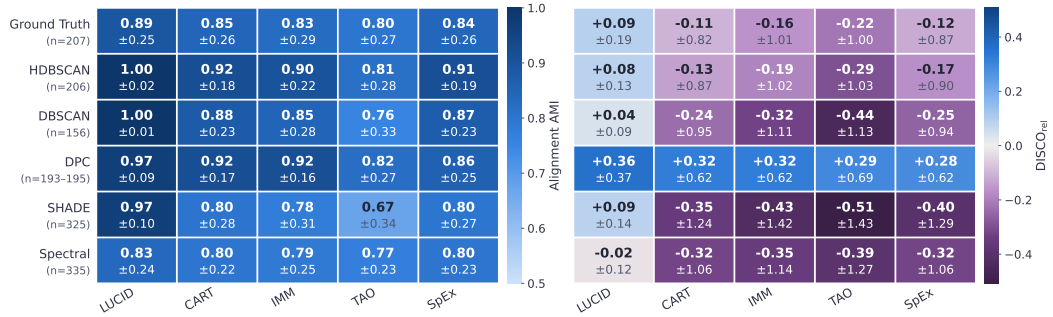


Figure 3: **Alignment to Reference Clustering and Density-Connectivity Structure.** All scores per explainability method and reference clustering are averaged over all datasets and seeds. **Left:** Mean AMI. Darker blue indicates higher alignment, with the color scale starting at 0.5. **Right:** Mean relative DISCO deviation $\mathrm{DISCO}_{\mathrm{rel}}$. Blue indicates that the method's labeling has higher DISCO than the reference, purple indicates lower, and white indicates no change.

generator using presets for uniform-density, varying-density, and noisy/unbalanced clusters. We additionally include 21 datasets from the Deric benchmark datasets as well as 9 classic shape datasets such as two moons, nested rings, and anisotropic or multi-density blobs. The real-world datasets span tabular data (e.g., Iris, Ecoli, Breast Cancer, Dry Beans), images (e.g., Coil-20, MNIST, CIFAR-10), and SBERT-embedded 20 Newsgroups text. MNIST, CIFAR-10, and 20 Newsgroups are subsampled to 3,000 data objects, and MNIST and CIFAR-10 are PCA-reduced to 50 dimensions. Synthetic datasets are regenerated for each of five seeds, and all results are averaged over seeds.

Reference clusterings. Besides the ground-truth labels, we explain five density-based reference clusterings: DBSCAN (Ester et al., 1996), HDBSCAN (Campello et al., 2013), SHADE (Beer et al., 2024), Density Peaks (DPC) (Rodriguez & Laio, 2014), and Spectral Clustering (Von Luxburg, 2007). To avoid biasing the references toward the structure LUCID exploits, none is tuned per dataset; each uses a fixed, data-driven heuristic (Appendix B).

Competitors. We compare against axis-aligned and oblique explainable-clustering trees: SpEx (Argov & Wagner, 2025), IMM (Moshkovitz et al., 2020), and CART (Breiman et al., 1984) repurposed for clustering, all axis-aligned, and TAO (Gabidolla & Carreira-Perpiñán, 2022) with sparse linear-oblique splits. Every method, including LUCID, is fit to the reference's non-noise data objects, with the number of leaves set to the reference's number of clusters.

Hyperparameters. LUCID uses $\mu = 5$, the default of the original dc-distance Beer et al. (2023), and a maximum split cardinality of $j_{\max} = 3$ features per split. $j_{\max}$ can be lowered by a user who wants sparser explanations. We ablate μ in Appendix C.

Metrics. We measure alignment with the reference using the Adjusted Mutual Information (AMI) and Adjusted Rand Index (ARI), excluding noise objects. To assess whether the induced labelings preserve density structure, we additionally compute the density-aware internal validity index DISCO (Beer et al., 2026) and report DBCV (Moulavi et al., 2014) in the appendix.

4.2 Alignment to Reference Clustering

Figure 3 shows the mean AMI between each reference clustering and each method's induced labeling, averaged over all datasets and seeds. LUCID achieves the highest alignment for every reference. Note that the alignment is higher for the reference methods that include some assessment of density, while the ground truth, which need not follow any density notion, is hardest to explain (0.866). The strongest competitors, CART, IMM, and SpEX-Clique, reach 0.78-0.92, while TAO aligns least (0.67-0.82). The gap is largest for SHADE, a deep-density-based clustering method, where LUCID leads the best competitors, CART and SpEx, by 0.17 AMI. We confirm that LUCID's advantage over every competitor (pooled and for the density-based references individually) is statistically significant in Table 9. Table 11 in the appendix gives the corresponding alignment ARI broken down per dataset

(averaged over references and seeds instead). LUCID attains the highest, or tied-highest, alignment of the five methods on 54 of 67 datasets (81%). It trails by 0.05 or more only on eight datasets, mostly those whose ground-truth classes are not density-separated (Table 13).

4.3 ALIGNMENT TO DENSITY-CONNECTIVITY STRUCTURE

To assess whether the induced labeling preserves the density-connected structure of the reference, we compare the DISCO score of each method's labeling to that of the reference clustering it explains. Because DISCO is upper-bounded by 1, we report the relative DISCO deviation $\mathrm{DISCO}_{\mathrm{rel}} = (\mathrm{DISCO}(\mathrm{method}) - \mathrm{DISCO}(\mathrm{reference}))/(1 - \mathrm{DISCO}(\mathrm{reference}))$, i.e., the fraction of the reference's remaining headroom to the optimum that the method gains (positive) or loses (negative). As shown in Figure 3 on the right, LUCID is the only method with a positive mean $\mathrm{DISCO}_{\mathrm{rel}}$ for the ground truth, HDBSCAN, DBSCAN, and SHADE references (0.09, 0.08, 0.04, and 0.09), while all competitors degrade the density structure, most strongly TAO (down to -0.51 for SHADE). DPC relies on density, but not on density-connectivity, thus, it leaves the most space for improvement, leading to high DISCO improvements for all competitors (0.28-0.32), topped by LUCID with 0.36. Spectral clusterings' DISCO scores are only decreased very slightly by LUCID (-0.02), while our competitors destroy the density-connectivity structure, leading to a high relative DISCO loss of 0.345 on average. DBCV is reported in Figure 4 in the appendix and is mostly consistent with DISCO: LUCID's labelings have the best results for every reference clustering method.

4.4 SELECTION OF NODE TYPES AND FALLBACK

Table 2 reports the rule types LUCID selects across all 5,382 internal nodes, including all datasets and references. Linear rules dominate (78.22%), and most use a single feature (54.07% of all splits), so most explanations are as simple as axis-aligned thresholds. Signomial rules are rare (4.55%) and almost always combine two or three features, capturing products or ratios of features that no sparse linear rule matches. Connectivity rules account for the remaining 17.22%. For most nodes (3,727), the target is density-respecting, and the selected rule separates it exactly. For the other 1655 nodes (30.75%), the reference clusters are interleaved, so no node of the density-connectivity dendrogram matches a union of reference clusters, and LUCID falls back to splitting off the plurality cluster, see also Table 10.

Table 2: **Distribution of Node Cuts across all datasets.** Counts are counted over every internal node of the learned LUCID trees. k is the number of features in a split's rule.

Rule Type	Cardinality	Count	% of splits
linear	$k = 1$	2910	54.07%
	$k = 2$	974	18.10%
	$k = 3$	326	6.06%
	total	4210	78.22%
signomial	$k = 1$	5	0.09%
	$k = 2$	76	1.41%
	$k = 3$	164	3.05%
	total	245	4.55%
connectivity	—	927	17.22%
total		5382	100.00%

5 CONCLUSION

We introduced LUCID, which explains a given density-based clustering with an interpretable decision tree grounded in density-connectivity structure. At each node, the density-connectivity hierarchy determines which clusters to separate, and LUCID selects the rule that reproduces this split most faithfully. It prefers sparse linear or signomial feature rules and falls back to a connectivity rule of the form "density-connected to example z" only when no short feature rule reproduces the split equally well. Whenever the reference clusters are nodes of this hierarchy, e.g., for DBSCAN* clusterings without noise, LUCID recovers the reference exactly.

In our experiments, LUCID matches the reference clustering more closely than all axis-aligned and oblique baselines, with statistically significant gains for every density-based reference and the largest lead on the deep density-based clustering SHADE. When explaining ground-truth labels, which need not be density-based, it performs comparably to CART, IMM, and SpEx. Moreover, LUCID is the only method whose explanations preserve the density structure of the reference, whereas all baselines degrade it. Natural next steps for future work are the extension of LUCID to categorical and mixed data, however, this is not trivial as it requires defining rules for categorical data.

AI USE STATEMENT

Required disclosures. AI tools were used to assist in implementing the proposed method. All code was reviewed, tested, and validated by the authors, who take full responsibility for its correctness. The design of the method, the experimental methodology, and the interpretation of results were carried out entirely by the authors.

Recommended disclosures. AI tools were also used to (i) write and edit code for running experiments and for generating the figures presented in the paper, (ii) identify potentially relevant literature that we may have overlooked; all suggested works were read and assessed by the authors, and the related work section was written entirely by the authors, and (iii) edit the manuscript to improve readability and language. All scientific content, claims, and conclusions are the authors' own.

The authors take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

We provide an anonymous repository containing our code online `https://anonymous.4open.science/r/LUCID-2424/README.md`. Additionally, we describe implementation details, such as software used for the implementation, parameter choices, or compute resources in Appendix B.

REFERENCES

Tommaso Amico, Pernille Matthews, Lena Krieger, Arthur Zimek, and Ira Assent. Dcfo: Density-based counterfactuals for outliers. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 37–48, 2026.

Tal Argov and Tal Wagner. Spex: A spectral approach to explainable clustering. *arXiv preprint arXiv:2511.00885*, 2025.

Anna Beer, Andrew Draganov, Ellen Hohma, Philipp Jahn, Christian MM Frey, and Ira Assent. Connecting the dots–density-connectivity distance unifies dbscan, k-center and spectral clustering. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 80–92, 2023.

Anna Beer, Pascal Weber, Lukas Miklautz, Collin Leiber, Walid Durani, Christian Böhm, and Claudia Plant. Shade: Deep density-based clustering. In *2024 IEEE International Conference on Data Mining (ICDM)*, pp. 675–680. IEEE, 2024.

Anna Beer, Lena Krieger, Pascal Weber, Martin Ritzert, Ira Assent, and Claudia Plant. Internal evaluation of density-based clusterings with noise. In *International Conference on Learning Representations*, volume 2026, pp. 86149–86172, 2026.

Dimitris Bertsimas, Agni Orfanoudaki, and Holly Wiberg. Interpretable clustering: an optimization approach. *Machine Learning*, 110(1):89–138, 2021.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pp. 160–172. Springer, 2013.

Emilio Carrizosa, Kseniia Kurishchenko, Alfredo Marín, and Dolores Romero Morales. Interpreting clusters via prototype optimization. *Omega*, 107:102543, 2022.

Emilio Carrizosa, Kseniia Kurishchenko, Alfredo Marín, and Dolores Romero Morales. On clustering and interpreting with rules by means of mathematical optimization. *Computers & Operations Research*, 154:106180, 2023.

Junxiang Chen, Yale Chang, Brian Hobbs, Peter Castaldi, Michael Cho, Edwin Silverman, and Jennifer Dy. Interpretable clustering via discriminative rectangle mixture model. In *2016 IEEE 16th international conference on data mining (ICDM)*, pp. 823–828. IEEE, 2016.

Ian Davidson, Michael Livanos, Antoine Gourru, Peter Walker, and Julien Velcin SS Ravi. An exemplars-based approach for explainable clustering: Complexity and efficient approximation algorithms. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pp. 46–54. Society for Industrial and Applied Mathematics, 2024.

Hossein Esfandiari, Vahab Mirrokni, and Shyam Narayanan. Almost tight approximation algorithms for explainable clustering. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2641–2663. SIAM, 2022.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, volume 96, pp. 226–231, 1996.

Maximilian Fleissner, Leena Chennuru Vankadara, and Debarghya Ghoshdastidar. Explaining kernel clustering via decision trees. *arXiv preprint arXiv:2402.09881*, 2024.

Nave Frost, Michal Moshkovitz, and Cyrus Rashtchian. Exkmc: Expanding explainable k-means clustering. *arXiv preprint arXiv:2006.02399*, 2020.

Magzhan Gabidolla and Miguel Á Carreira-Perpiñán. Optimal interpretable clustering using oblique decision trees. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 400–410, 2022.

Buddhima Gamlath, Xinrui Jia, Adam Polak, and Ola Svensson. Nearly-tight and oblivious algorithms for explainable clustering. *Advances in Neural Information Processing Systems*, 34:28929–28939, 2021.

Jana Gauss, Fabian Scheipl, and Moritz Herrmann. Dcsi–an improved measure of cluster separability based on separation and connectedness. *arXiv preprint arXiv:2310.12806*, 2023.

Anupam Gupta, Madhusudhan Reddy Pittu, Ola Svensson, and Rachel Yuan. The price of explainability for clustering. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1131–1148. IEEE, 2023.

Lianyu Hu, Mudi Jiang, Junjie Dong, Xinying Liu, and Zengyou He. Interpretable clustering: A survey. *ACM Computing Surveys*, 58(8):1–21, 2026.

Hyunseung Hwang and Steven Euijong Whang. Xclusters: explainability-first clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7962–7970, 2023.

Philipp Jahn, Christian MM Frey, Anna Beer, Collin Leiber, and Thomas Seidl. Data with density-based clusters: A generator for systematic evaluation of clustering algorithms. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 3–21. Springer, 2024.

Hussien Kamh, Saleh H Alyami, Afaq Khattak, Mana Alyami, and Hamad Almujibah. Exploring road traffic accidents hotspots using clustering algorithms and gis-based spatial analysis. *IEEE Access*, 13:60944–60954, 2024.

Eduardo S Laber and Lucas Murtinho. On the price of explainability for some clustering problems. In *International Conference on Machine Learning*, pp. 5915–5925. PMLR, 2021.

Connor Lawless and Oktay Gunluk. Cluster explanation via polyhedral descriptions. In *International conference on machine learning*, pp. 18652–18666. PMLR, 2023.

Connor Lawless, Jayant Kalagnanam, Lam M Nguyen, Dzung Phan, and Chandra Reddy. Interpretable clustering via multi-polytope machines. In *Proceedings of the aaai conference on artificial intelligence*, volume 36, pp. 7309–7316, 2022.

Adia C. Lumadjeng, S. Ilker Birbil, and Erman Acar. ECSEL: explainable classification via signomial equation learning. In *International conference on machine learning*. PMLR, 2026.

Konstantin Makarychev and Liren Shan. Near-optimal algorithms for explainable k-medians and k-means. In *International Conference on Machine Learning*, pp. 7358–7367. PMLR, 2021.

Pernille Matthews, Lena Krieger, Tommaso Amico, Arthur Zimek, Thomas Seidl, and Ira Assent. Exdbscan: Explaining dbscan with counterfactual reasoning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 3703–3714, 2026.

Michal Moshkovitz, Sanjoy Dasgupta, Cyrus Rashtchian, and Nave Frost. Explainable k-means and k-medians clustering. In *International conference on machine learning*, pp. 7055–7065. PMLR, 2020.

Davoud Moulavi, Pablo A Jaskowiak, Ricardo JGB Campello, Arthur Zimek, and Jörg Sander. Density-based clustering validation. In *Proceedings of the 2014 SIAM international conference on data mining*, pp. 839–847. SIAM, 2014.

Raphael Petegrosso, Zhuliu Li, and Rui Kuang. Machine learning and statistical methods for clustering single-cell rna-sequencing data. *Briefings in bioinformatics*, 21(4):1209–1223, 2020.

Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496, 2014.

Sandhya Saisubramanian, Sainyam Galhotra, and Shlomo Zilberstein. Balancing the tradeoff between clustering value and interpretability. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 351–357, 2020.

Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pp. 461–464, 1978.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Tao Zeng, Caiming Zhong, and Tiejun Pan. Clustering explanation based on multi-hyperrectangle. *Scientific Reports*, 14(1):30260, 2024.

Mingrui Zhang. Use density-based spatial clustering of applications with noise (dbscan) algorithm to identify galaxy cluster members. In *IOP conference series: earth and environmental science*, volume 252, pp. 042033. IOP Publishing, 2019.

A METHOD DETAILS

A.1 TIE-BREAKING AND DEGENERATE NODES

All tie-breaking rules only make LUCID deterministic; none affects its guarantees. If several density-respecting bipartitions are equally high in $\mathcal{D}_v$, we prefer fewer clusters in $\mathscr{S}_v$, then the lexicographically smallest cluster-id tuple. In FORWARDSELECT, ties between features are broken by the smallest feature index. Among connectivity candidates with equal err_v, we prefer the node h of $\mathcal{D}_v$ with fewer data objects $|X_h|$, then the node containing the data object with the smallest index. This covers, in particular, the two children of a root of $\mathcal{D}_v$ that has exactly two children, since both induce the same bipartition. If lin and sig still tie after comparing $(\mathrm{err}_v, \mathrm{size})$, lin is preferred (Section 3.3). In the priority queue of the tree, ties in err_v are broken by creation order.

If no rule type is valid at a node, i.e., every fitted rule leaves one child empty, a dense ridge fit on all features is used as a last resort (Appendix B). If this fit is not valid either, v becomes a leaf labeled by plurality. This only arises for degenerate data, e.g., duplicate data objects, and never occurred in our experiments (0 of 3,815 splits).

A.2 PSEUDOCODE

Algorithms 1 and 2 summarize the method detailed in Section 3.

Algorithm 1 LUCID

Require: X', reference clustering $\mathscr{C}_{\mathrm{ref}}$, μ, $j_{\max}$
compute d^{μ}_{dc} on X' once; rescale features to $[1, 10]$
$k \leftarrow |\{\hat{C}_x : x \in X'\}|$
leaves $\leftarrow \{\mathrm{root}\}$; queue $\leftarrow \varnothing$
procedure FITCUT(v)
  if $|\mathscr{C}_v| = 1$ **then return**
  end if
  $B^*_v = (\mathscr{S}_v, \overline{\mathscr{S}_v}) \leftarrow$ highest bipartition (Section 3.2)
  $y_v \leftarrow$ oracle target (Def. 2) of B^*_v
  $r^{\mathrm{lin}} \leftarrow$ FORWARDSELECT$((x^+)_{x \in X_v},\ y_v,\ j_{\max})$
  $r^{\mathrm{sig}} \leftarrow$ FORWARDSELECT$((\log x^+)_{x \in X_v},\ y_v,\ j_{\max})$
  $r^{\mathrm{con}} \leftarrow$ "$d^{\mu}_{\mathrm{dc}}(x, \mathbf{z}) < \varepsilon$" for the node $h^\star$ of $\mathcal{D}_v$ minimizing err_v (Section 3.3)
  $r_v \leftarrow$ valid rule among $r^{\mathrm{lin}}, r^{\mathrm{sig}}, r^{\mathrm{con}}$ minimizing $(\mathrm{err}_v, \mathrm{size})$; if none is valid, dense ridge fit
  if r_v is valid **then** push v onto queue with priority $\mathrm{err}_v(r_v)$
  else return
  end if
end procedure
FITCUT(root)
while $|\mathrm{leaves}| < k$ and queue is non-empty **do**
  pop v with smallest priority from queue
  split X_v into X_{v_L}, X_{v_R} by r_v; leaves $\leftarrow$ (leaves $\setminus \{v\}$) $\cup \{v_L, v_R\}$
  FITCUT(v_L); FITCUT(v_R)
end while
label every leaf in leaves with its plurality cluster

A.3 PROOFS

Lemma 1 (Connectivity rules are dendrogram nodes)**.** *Let h be a node of $\mathcal{D}_v$ other than its root, and let $\varepsilon_h > 0$ be the height of its parent. For every $\mathbf{z} \in X_h$, $\{x \in X_v : d^{\mu}_{\mathrm{dc}}(x, \mathbf{z}) < \varepsilon_h\} = X_h$. Conversely, for every set $\mathbf{z} \in X_v$ and $\varepsilon > 0$, the ball $X^{\varepsilon} = \{x \in X_v : d^{\mu}_{\mathrm{dc}}(x, \mathbf{z}) < \varepsilon\}$ equals X_h for some node h of $\mathcal{D}_v$ whenever $X^{\varepsilon} \neq X_h$.*

Proof. In an ultrametric hierarchy, any $x \notin X_h$ first joins h at or above the parent of h, so $d^{\mu}_{\mathrm{dc}}(\mathbf{z}, x) \geq \varepsilon_h$. Data objects within X_h merge strictly below ε_h because equal-height merges are collapsed, so

Algorithm 2 FORWARDSELECT

Require: transformed data objects $u(x)$ for $x \in X_v$, with $u(x) = x^+$ for lin or $u(x) = \log x^+$ for sig, target y_v, cap $j_{\max}$

$J \leftarrow \varnothing$; $t_{\max} \leftarrow \min(j_{\max}, d)$
for $t = 1, \ldots, t_{\max}$ **do**
  for each $j \in \{1, \ldots, d\} \setminus J$ **do**
    fit Eq. 4 on $J \cup \{j\}$; place $\tau_{J \cup \{j\}}$; $\mathrm{err}_j \leftarrow \mathrm{err}_v$ of the resulting rule
  end for
  $j^\star \leftarrow \arg\min_j \mathrm{err}_j$ (ties: smallest j)
  $J \leftarrow J \cup \{j^\star\}$; $\mathrm{err}^{(t)} \leftarrow \mathrm{err}_{j^\star}$; $L^{(t)} \leftarrow$ negative log-likelihood of the fit on J
end for
return the rule fitted on the first t selected features, for the smallest t with $\mathrm{err}^{(t)} = 0$, else for $\arg\min_t \; 2L^{(t)} + (t+1)\ln|X_v|$

$d^\mu_{\mathrm{dc}}(\mathbf{z}, x) < \varepsilon_h$ for $x \in X_h$. The converse is the ball property of ultrametrics: every open ball is the set of data objects of a node of the hierarchy (cf. Beer et al. (2023)). □

Proof of Proposition 1. If $\mathcal{B}_v \neq \varnothing$, the chosen B^*_v is density-respecting (Section 3.2), so $X_{\mathscr{S}_v} = X_h$ for a node h of $\mathcal{D}_v$ other than its root, and h is thus a candidate of the connectivity search. Since $y_v(x) = \mathbb{I}[x \in X_h]$, the rule "$x \in X_h$" has error $\mathrm{err}_v = 0$. □

Proof of Corollary 1. Since $\mathcal{D}_v$ is the restriction of $\mathcal{D}$, $C_i \cap X_v$ is a node of $\mathcal{D}_v$ for every v and every cluster present. Nodes of a hierarchy are nested or disjoint, so if $|\mathscr{C}_v| \geq 2$, no child of the root of $\mathcal{D}_v$ can lie strictly inside one cluster; each child is a union of whole clusters, hence $\mathcal{B}_v \neq \varnothing$. By Proposition 1 the selected rule has $\mathrm{err}_v = 0$, so both children again contain only whole clusters. By induction, every split separates whole clusters, $k-1$ splits yield k single-cluster leaves, and the plurality labeling makes no mistakes. □

The condition of Corollary 1 holds, in particular, for DBSCAN* clusterings of $X' = X \setminus \mathcal{N}$, since these clusters are exactly the ε-neighborhoods in the dc-dist. (DBSCAN* refers to DBSCAN where *all* points with fewer than μ points in their ε-neighborhood are assigned to noise, effectively removing the concept of border points).

Lemma 2 (DBSCAN* clusters are ε-neighborhoods in the dc-dist)**.** *Let $\{C_1, \ldots, C_k\}$ be the DBSCAN*(μ, ε) *clusters of X' with the same μ as d^μ_{dc}, such that every data object of X' is a core object within X', i.e., has at least $\mu - 1$ other data objects of X' within distance ε. The clusters are then the connected components of the ε-neighborhood graph on X', and for $a, b \in X'$, $d^\mu_{\mathrm{dc}}(a, b) \leq \varepsilon$ if a, b lie in the same cluster and $d^\mu_{\mathrm{dc}}(a, b) > \varepsilon$ otherwise. In particular, every C_i is a node of $\mathcal{D}$.*

Proof. Every data object is *core*, i.e., it has at least $\mu - 1$ other data objects within distance ε. Thus, its $(\mu-1)$-th nearest neighbor, and hence its core distance κ_μ, is at most ε. If a, b lie in the same cluster, they are joined by a path in X' with consecutive Euclidean distances at most ε, so every edge (x, y) on it has $\mathrm{mrd}(x, y) \leq \varepsilon$, since both its length and the core distances of its endpoints are at most ε. Otherwise, every path from a to b contains an edge (x, y) with Euclidean distance $> \varepsilon$ (else a, b would be in one component), and hence with $\mathrm{mrd}(x, y) > \varepsilon$. Thus even the minimax path has an edge (x, y) with $\mathrm{mrd}(x, y) > \varepsilon$, and for any $a \in C_i$, $C_i = \{x \in X' : d^\mu_{\mathrm{dc}}(a, x) \leq \varepsilon\}$ is the ε-neighborhood of a in the dc-dist, i.e., a node of $\mathcal{D}$. □

The condition refers to $\mathcal{D}$ on X', since LUCID computes the dc-dist after removing noise (Section 3.1). It holds, e.g., for a DBSCAN* clustering without noise, i.e., on X'.

B Implementation Details

Rescaling. Signomial rules take $\log$ of the features and therefore require strictly positive inputs. We map each feature affinely to $[1, 10]$,

$$x_j^+ = 1 + 9\,\frac{x_j - \min_i x_{ij}}{\max_i x_{ij} - \min_i x_{ij}},$$

and fit all feature rules on X^+. The lower bound 1 keeps $\log x_j^+ \geq 0$ and away from the singularity at 0; the range $[1, 10]$ spans one order of magnitude, so signomial exponents stay on a comparable scale across features. Constant features ($\max_i x_{ij} = \min_i x_{ij}$) carry no information for any split and are dropped. Because the map is invertible and affine per feature, a linear rule $\beta^\top x^+ \leq c$ translates back to the original features exactly, with β_j divided by the feature's scale factor and c shifted accordingly. The power-law reading of signomial rules, in contrast, holds in X^+ only, since the additive shift keeps $\log x_j^+$ from reducing to $\log x_j$ plus a constant. At test time, points outside the training range are clipped to $[1, 10]$ before $\log$ is applied.

Reference method parameters. HDBSCAN uses $\text{min_cluster_size} = \max(\mu, 5)$. DBSCAN is vanilla Euclidean DBSCAN with $\text{min_samples} = \mu$ and ε set to the knee of the sorted Euclidean μ-nearest-neighbor distance curve, found via Kneedle. DPC has no self-determined cluster count, so, as is common in benchmarks, it is given the number of ground-truth clusters and takes the points with the largest product of density and distance to a denser point as centers; it is run with halo assignment, so data objects it does not assign to a density peak's basin are treated as noise. SHADE trains an autoencoder whose embedding preserves the dc-distances of the (standardized) input and extracts stable density-based clusters from a dc-tree on the learned embedding; we use the authors' default configuration with $\text{min_points} = \mu$. SHADE shares the dc-distance with LUCID, but only as a training signal: its clusters are extracted in the learned embedding, not on the input, so it does not coincide with the dc-distance hierarchy LUCID builds on. Spectral clustering runs on a symmetrized 10-nearest-neighbor graph with all other parameters at their defaults; like DPC, it is given the number of ground-truth clusters, and it labels no point as noise. Only the reference sees this number; every explainable method is fit to the reference labels alone. Reference clusterings with fewer than two non-noise clusters are excluded from the evaluation. This affects DBSCAN on 51 of 207 dataset–seed pairs, DPC on 12 of 207, HDBSCAN on 1 of 207, SHADE on 10 of 335, and spectral clustering on none.

Software. All methods are implemented in Python 3.14 with NumPy 2.5, SciPy 1.18, scikit-learn 1.9, and pandas 3.0; SHADE additionally uses PyTorch 2.13. We implemented LUCID: the dc-distance is computed from the minimum spanning tree of the mutual-reachability graph, and the logistic node fits (Eq. 4, $\lambda_0 = 10^{-6}$) are solved with a vectorized Newton method over all candidate features at once.

Implementations. Our code can be found online: `https://anonymous.4open.science/r/LUCID-2424/README.md`. Wherever possible, we use the original authors' code. SpEx (Argov & Wagner, 2025) runs in its clique-graph variant from the official repository, with one Numba-compiled counting routine replaced by an equivalent NumPy call. IMM (Moshkovitz et al., 2020) runs through the authors' ExKMC package (version 0.0.3) restricted to k leaves; since its compiled extension does not build on current Python and NumPy versions, we use the package's own pure-Python cut finder, which yields identical trees on all non-degenerate references of the first seed. SHADE (Beer et al., 2024) is the authors' implementation in `clustpy` 0.0.3, used with its default configuration (embedding size 10, batch size 500, 100 training epochs, no pretraining) on z-score standardized input and $\text{min_points} = \mu$. DISCO (Beer et al., 2026), DBCV (Moulavi et al., 2014), and DCSI (Gauss et al., 2023) come from the DISCO authors' repository, with only NumPy-2 compatibility fixes. HDBSCAN uses the `hdbscan` package (0.8.44), and Kneedle uses the `kneed` package (0.8.6); DBSCAN, spectral clustering, CART, ARI, and AMI use scikit-learn; DENSIRED data is generated with the `densired` package (1.2.0). Methods without a usable official version are reimplemented from their papers: TAO (Gabidolla & Carreira-Perpiñán, 2022), whose official release covers classification and regression but not the clustering variant (our decision nodes use the same solver as that release), and DPC (Rodriguez & Laio, 2014), whose available packages are unmaintained.

Table 3: **Ablation of μ.** Mean alignment over the 1,424 runs in which all three values succeed, share of runs with the same alignment as for $\mu = 5$, and distribution of the applied splits over rule types ("1 feature" is linear splits with a single feature, a subset of the "linear" column, not an additional category), with the share of splits at which LUCID falls back to the plurality target. The reference clusterings are fixed.

	Alignment			Rule type (% of splits)				
μ	ARI	AMI	same as $\mu = 5$	linear	1 feature	signomial	connectivity	fallback
2	0.921	0.938	84.8%	78.8	53.2	4.5	16.8	23.6
5	0.915	0.933	—	78.2	54.1	4.6	17.2	30.8
10	0.908	0.927	83.8%	77.4	52.0	5.2	17.4	38.7

Table 4: **Ablation of μ per reference.** Mean alignment ARI of LUCID for each reference clustering, over the runs in which all three values of μ succeed; the reference clusterings are fixed.

μ	Ground Truth	HDBSCAN	DBSCAN	DPC	SHADE	Spectral
2	0.873	0.999	1.000	0.972	0.981	0.776
5	0.866	0.997	0.999	0.965	0.973	0.771
10	0.853	0.991	0.997	0.962	0.969	0.758

Data preprocessing. Real-world datasets are z-score standardized. MNIST and CIFAR-10 are subsampled to 3,000 data objects and reduced to 50 dimensions by PCA, and 20 Newsgroups is subsampled to 3,000 documents and embedded with `all-MiniLM-L6-v2` (384 dimensions). Synthetic datasets are regenerated with seeds 0–4; SHADE, the only stochastic reference clustering, is seeded with the same seeds.

Hardware and runtime. All experiments ran on a single Apple M4 Pro (12 CPU cores, 24 GB RAM) under macOS 26, on the CPU only.

C ABLATION

The following sections analyse LUCID's behavior against another choice of parameter μ, and when the noise filtering is done after the distance computations rather than before, as stated in the method section.

C.1 ABLATION OF THE μ-PARAMETER

LUCID's only parameter is μ, the neighborhood size of the dc-distance, which defaults to $\mu = 5$. To isolate its effect, we refit LUCID with $\mu = 2$ and $\mu = 10$ on exactly the reference clusterings of the main experiments.

LUCID is robust to μ (Table 3). Compared to $\mu = 5$, 84.8% of the runs give an identical alignment for $\mu = 2$ and 83.8% for $\mu = 10$. The remaining runs show a small but consistent trend: a smaller μ slightly improves alignment and a larger one slightly lowers it. Per reference, the mean ARI changes by at most 0.013 (Table 4) and the mean AMI by at most 0.012 (Table 5), and LUCID remains best or tied with the best competitor in 84.4%, 83.5%, and 81.7% of the runs. The trees keep their size (mean depth 3.17, 3.21, 3.23) and use the same number of features per split (1.93, 1.93, 1.94; Table 6).

The rule types shift only slightly with μ: a larger μ yields marginally fewer linear and more signomial and connectivity splits. The plurality fallback, however, depends clearly on μ and rises from 23.6% of the splits for $\mu = 2$ to 38.7% for $\mu = 10$ (Table 7). A larger μ smooths the dc-distance, so reference clusters are more often interleaved in its dendrogram, and LUCID more often has to fall back to splitting off the plurality cluster instead of a density-respecting target. That the alignment barely changes suggests that LUCID's rules still separate most of these fallback targets well. We keep $\mu = 5$ as the default, as it is the standard choice for the dc-distance and matches the neighborhood size of the reference clusterings.

Table 5: **Ablation of μ per reference (AMI).** Mean alignment AMI of LUCID for each reference clustering, over the runs in which all three values of μ succeed; the reference clusterings are fixed.

μ	Ground Truth	HDBSCAN	DBSCAN	DPC	SHADE	Spectral
2	0.902	0.999	1.000	0.973	0.976	0.835
5	0.893	0.996	0.998	0.967	0.968	0.834
10	0.888	0.990	0.994	0.965	0.963	0.823

Table 6: μ **ablation: explanation quality**, mean over the same 1,424 runs.

μ	Alignment ARI	ARI vs. GT	DISCO	DBCV	Mean ℓ_0
2	0.921	0.763	0.596	0.361	1.93
5	0.915	0.758	0.559	0.355	1.93
10	0.908	0.753	0.502	0.334	1.94

C.2 NOISE FILTERING

LUCID removes the reference's noise data objects before computing the dc-distance, so that core distances describe the density of the data objects the tree has to explain (filter, then distance). The alternative computes the dc-distance on all data objects and only then restricts it to the non-noise data objects (distance, then filter), so that noise data objects still shape the core distances of their neighbors.

Both orders coincide whenever the reference labels no point as noise, which holds for 735 of the 1,424 runs of Section 4, including all runs on spectral clustering. We refit LUCID with the distance-then-filter order on the remaining 689 runs, on the same reference clusterings and with all other settings unchanged. The order has no meaningful effect (Table 8). In 91.6% of the 689 runs, both orders yield identical alignment; of the remaining 58 runs, distance-then-filter is better in 38 and worse in 20, and the difference is not significant (mean ARI 0.987 vs. 0.985, Wilcoxon signed-rank test on the non-tied runs, $p = 0.16$). Per reference, the mean ARI over the runs with noise differs by at most 0.004 (at most 0.003 over all runs, Table 8), and for the ground truth all runs coincide. The trees are also structurally similar: over the 2,426 applied splits, 85.4% vs. 84.0% are linear, 2.1% vs. 1.7% signomial, and 12.6% vs. 14.3% connectivity rules, and the plurality fallback occurs at 20.8% vs. 19.1% of the splits. We therefore keep filtering first: it describes the density of exactly the data objects the tree explains, and it computes the dc-distance on fewer data objects.

D ADDITIONAL RESULTS

This section complements the experiments of Section 4 with further metrics, a robustness check, a breakdown of the rule fit by oracle target, and the full list of datasets.

D.1 STATISTICAL SIGNIFICANCE

Table 9 tests LUCID against each competitor on alignment AMI. LUCID's advantage is statistically significant against every competitor when pooled over all references ($\Delta\mathrm{AMI} = +0.08$ to $+0.17$, rank-biserial $r = 0.48$–0.60), for each density-based reference separately, and for the ground truth (Holm-corrected $p < .05$ throughout). The effects are largest on the density-based references, where LUCID wins or ties in almost every run ($r = 0.80$–0.87 for HDBSCAN, DBSCAN, and SHADE); on DPC they are smaller but still clear ($r = 0.51$–0.67). Spectral clustering, the one reference that is not density-based, is the exception: LUCID's lead is significant only against TAO ($+0.06$ AMI); against CART, IMM, and SpEx ($+0.03$ to $+0.05$) wins and losses are nearly balanced and the difference is not significant (Holm-corrected $p = .58$).

Table 7: μ **ablation: oracle and rule-fitting outcomes** over the applied splits of the same 1,424 dataset×reference×seed runs, identical across all three μ.

μ	Splits	Density-respecting	Exact (err $= 0$)	BIC	Dense fallback
2	5382	76.37%	85.62%	14.38%	0.00%
5	5382	69.25%	82.79%	17.21%	0.00%
10	5382	61.33%	78.58%	21.42%	0.00%

D.2 Cut and Node Analysis

Table 10 breaks the 5,382 splits of LUCID's trees down by the kind of oracle target (Theorem 2) and by how well the selected rule reproduces it. At 69.2% of the splits, the target is density-respecting, and in every one of these 3,727 splits the selected rule reproduces it exactly (err $= 0$), as Proposition 1 guarantees: whenever a density-respecting target exists, at least the connectivity rule attains zero error. At the remaining 30.8% of the splits, no density-respecting target exists and LUCID falls back to splitting off the plurality cluster; there is no guarantee in this case, yet 729 of these 1,655 fallback targets (44%) are still reproduced exactly. Only the other 926 splits (17.2% of all splits) use an approximate rule, whose number of features is chosen by BIC among the forward-selected candidates. The dense fallback, a logistic fit on all features used when no rule type yields two non-empty children, never occurs.

D.3 Density-connectivity Quality

Figure 4 scores the labeling induced by each method with DBCV, complementing the DISCO analysis of Figure 3. While DBCV is the more commonly used CVI, it was recently shown to be non-deterministic (Beer et al., 2026). Under DBCV, LUCID's labelings are the most density-valid for every reference, by a wide margin: its mean DBCV is 0.38–0.66, against at most 0.33 for any competitor, and its lead over the best competitor ranges from 0.12 (DPC) to 0.43 (SHADE); for SHADE, three of the four competitors even reach a negative mean DBCV.

D.4 Alignment related to Connectivity

We next investigate whether references with a cleaner density structure are easier to explain. For every dataset–reference pair (381 pairs, averaged over seeds), we relate the reference's own DISCO to each method's alignment AMI, and confidence intervals come from a bootstrap that resamples datasets, since the references of one dataset are not independent.

LUCID's alignment increases with the reference's DISCO (Spearman $\rho = 0.49$, 95% CI $[0.43, 0.56]$), whereas no competitor shows a clear relation ($\rho = 0.12$–0.19, all intervals include 0). Figure 5 shows why: on references in the top DISCO quartile, LUCID reconstructs the reference essentially perfectly (mean ARI 1.00), while all competitors drop to 0.67–0.75, since clean density-based clusters are often non-convex and cannot be separated by a few axis-aligned or oblique cuts. Consequently, LUCID's lead over the best competitor grows from -0.04 AMI in the lowest to $+0.23$ in the highest quartile (difference 0.28, 95% CI $[0.12, 0.42]$). On references with the weakest density structure, LUCID

Table 8: **Noise-ordering ablation.** Runs whose reference contains noise, with the median share of noise objects, the share of runs with identical alignment under both orders, and the mean alignment ARI for filter-then-distance (LUCID's default) and distance-then-filter.

Reference	Runs	Median noise share	Identical	Filter first	Distance first
Ground Truth	75	10.0%	100.0%	1.000	1.000
HDBSCAN	146	4.9%	83.6%	0.996	0.999
DBSCAN	120	0.8%	95.8%	0.999	0.999
DPC	141	56.8%	97.2%	0.989	0.992
SHADE	223	6.0%	88.8%	0.961	0.963
All	705	—	91.8%	0.985	0.987

Table 9: **Improvement of LUCID over each competitor in alignment AMI.** One-sided paired Wilcoxon signed-rank tests (H_1: LUCID $>$ competitor), Holm–Bonferroni corrected separately within the pooled family and each per-reference family. ΔAMI is the mean paired difference; r is the rank-biserial effect size (pooled in the r column). Per-reference cells are shaded by their own r only if significant (Holm-corrected $p < .05$), with intensity linear in r from 0 to 1.

LUCID	Pooled		Per reference (ΔAMI)					
compared to	ΔAMI	r	Ground Truth	HDBSCAN	DBSCAN	DPC	SHADE	Spectral
CART	+0.08	+0.48	+0.05	+0.08	+0.12	+0.05	+0.17	+0.03
IMM	+0.10	+0.50	+0.07	+0.09	+0.15	+0.05	+0.19	+0.05
TAO	+0.17	+0.60	+0.10	+0.18	+0.24	+0.15	+0.30	+0.06
SpEx	+0.10	+0.50	+0.05	+0.08	+0.13	+0.11	+0.17	+0.04

Table 10: **Rule fit by oracle target type**, over the applied splits of LUCID's trees, pooled over all datasets, references, and seeds. "Exact": the selected rule reproduces the target with $\mathrm{err} = 0$; "BIC-selected": no rule is exact and the number of features is chosen by BIC; "dense fallback": no rule type yields two non-empty children.

Oracle target	Rule fit	Count	% of splits
density-respecting ($\mathcal{B}_v \neq \varnothing$)	exact, err $= 0$	3727	69.25%
	BIC-selected	0	0.00%
	dense fallback	0	0.00%
	total	3727	69.25%
plurality fallback ($\mathcal{B}_v = \varnothing$)	exact, err $= 0$	729	13.55%
	BIC-selected	926	17.21%
	dense fallback	0	0.00%
	total	1655	30.75%
total		5382	100.00%

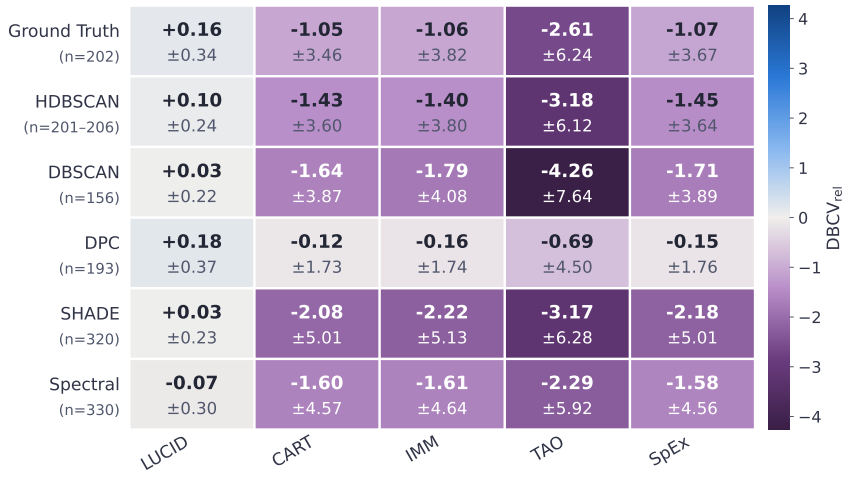


Figure 4: **Mean relative DBCV deviation** $\mathrm{DBCV}_{\mathrm{rel}}$ per explainability method and reference clustering, averaged over all datasets and seeds. Blue indicates that the method's labeling has higher DBCV than the reference, purple indicates lower, and white indicates no change.

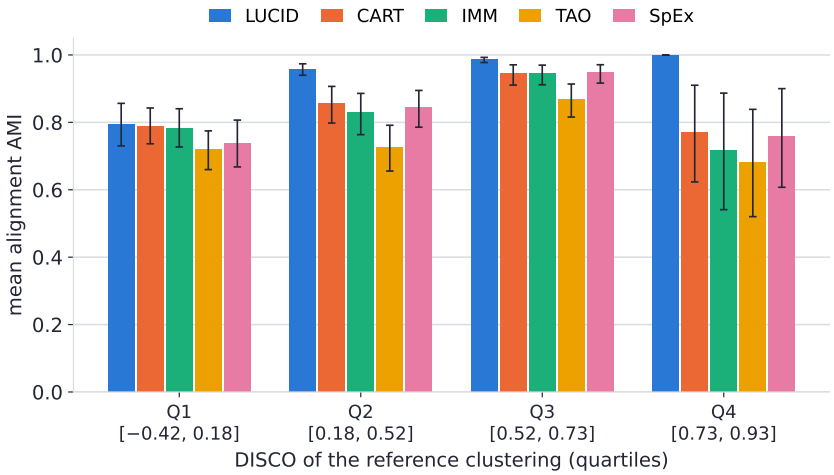


Figure 5: **Alignment AMI by the density quality measured with DISCO of the reference.** Dataset–reference pairs are binned into quartiles of the reference's DISCO (range in brackets); bars show the mean AMI per method, error bars are 95% bootstrap intervals over datasets.

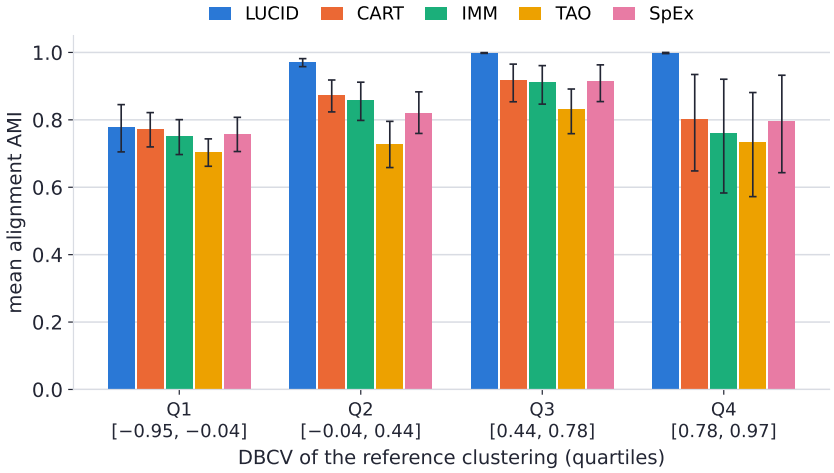


Figure 6: **Alignment AMI by the density quality measured with DBCV of the reference.** Dataset–reference pairs are binned into quartiles of the reference's DBCV (range in brackets); bars show the mean AMI per method, error bars are 95% bootstrap intervals over datasets.

is only on par with CART and IMM (0.79 vs. 0.79 and 0.78): without a clear density-connected target, LUCID has no structural advantage to exploit. This relation is expected: a high DISCO means that the reference clusters are well separated in the dc-distance, which approaches the setting in which LUCID provably recovers them exactly. Because DISCO itself builds on the dc-distance, we repeat the analysis with DBCV (Figure 6), which does not (379 pairs, as DBCV is undefined for two references): the results agree ($\rho = 0.65$ for LUCID vs. 0.28–0.33 for the competitors; lead growing from -0.03 to $+0.20$, difference 95% CI $[0.09, 0.40]$).

D.5 ALIGNMENT MEASURED BY ARI

Figure 7 shows the same experiment as Figure 3 on the left but with ARI instead of AMI. The conclusions are similar: LUCID aligns best with most references, reaching 1.0 on DBSCAN and HDBSCAN, 0.97 on DPC and SHADE, and 0.87 resp. 0.77 on the ground truth resp. Spectral Clustering. It thus outperforms other explainability methods for DBSCAN, HDBSCAN, DPC, and,

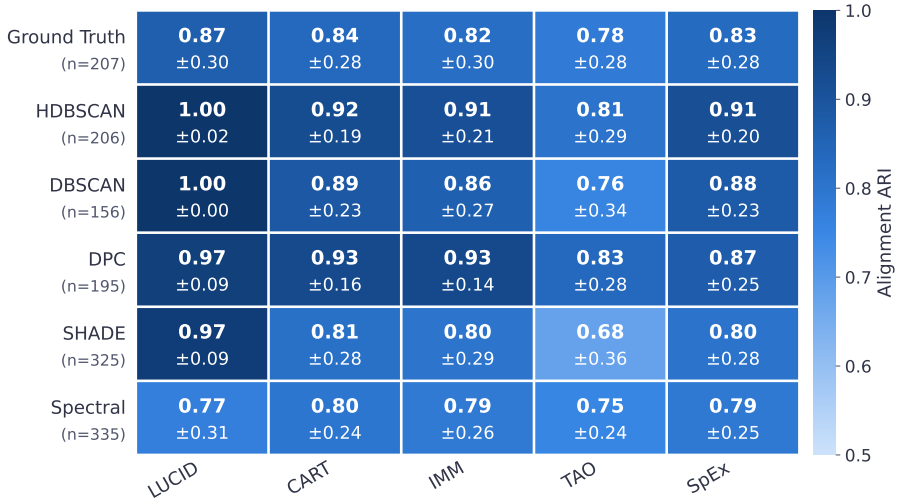


Figure 7: **Mean ARI per explainability method and reference clustering**, averaged over all datasets and seeds (references with fewer than two non-noise clusters excluded). Darker blue indicates higher alignment; the color scale starts at 0.5.

most clearly, the deep clustering method SHADE. On ground truth labelings, LUCID is only slightly better than the competitors. On Spectral Clustering, where LUCID leads regarding AMI, it is only comparable to the competitors for ARI. This can be attributed to the different definitions and foci of the two evaluation measures (counting pairs vs mutual reachability).

D.6 ALIGNMENT PER DATASET

Table 11 extends Figure 3 from the main paper. It shows the alignment of the labelings induced by the explainable methods to the reference labeling per dataset, averaged over the reference methods.

Table 11: **Alignment AMI per dataset.** AMI scores averaged over all reference clusterings and seeds. Datasets are sorted by descending ground-truth DISCO, as in Table 13. Colors use the same colormap and scale as Figure 3; — marks datasets with no successful run for that method, i.e., clusterings with less than two non-noise clusters.

Dataset	LUCID	CART	IMM	TAO	SpEx
densired_uniform	1.00	1.00	1.00	0.98	1.00
cb_banana	1.00	0.51	0.51	0.37	0.52
cb_spiral	1.00	0.08	0.08	0.03	0.08
densired_varying_density	1.00	0.99	0.99	0.97	0.99
cb_target	0.91	0.53	0.34	0.56	0.50
densired_lowdim_uniform_noiseless	1.00	1.00	1.00	0.98	1.00
cb_chainlink	1.00	0.39	0.39	0.13	0.39
densired_highdim_uniform_noiseless	1.00	1.00	1.00	0.95	0.99
nested_rings	1.00	0.37	0.16	0.08	0.36
densired_highdim_uniform_noisy	0.98	0.98	0.97	0.94	0.98
densired_lowdim_uniform_noisy	0.96	0.97	0.97	0.92	0.97
cb_atom	0.97	0.32	0.07	0.26	0.32
densired_highdim_uniform_veryNoisy	0.94	0.93	0.92	0.91	0.93
densired_lowdim_varying_noiseless	1.00	0.93	0.93	0.83	0.93
densired_meddim_varying_noiseless	1.00	0.99	0.99	0.94	0.99
jain	1.00	0.90	0.89	0.74	0.90
densired_noisy_unbalanced	0.96	0.94	0.95	0.90	0.95
densired_lowdim_varying_noisy	0.98	0.92	0.89	0.75	0.90
densired_meddim_varying_noisy	0.98	0.96	0.96	0.89	0.94
densired_meddim_uniform_noiseless	1.00	0.99	0.99	0.93	0.99

continued on next page

Table 11 – continued from previous page

Dataset	LUCID	CART	IMM	TAO	SpEx
densired_noisy_balanced	0.98	0.95	0.95	0.86	0.94
cb_hepta	1.00	0.90	0.90	0.89	0.90
densired_highdim_varying_noiseless	1.00	0.99	0.99	0.97	0.97
cb_zelnik3	1.00	0.79	0.81	0.68	0.80
densired_meddim_uniform_noisy	0.97	0.96	0.96	0.94	0.96
densired_highdim_varying_noisy	0.96	0.91	0.93	0.89	0.89
densired_highdim_varying_veryNoisy	0.94	0.90	0.90	0.86	0.87
densired_meddim_uniform_veryNoisy	0.94	0.91	0.91	0.89	0.87
densired_meddim_varying_veryNoisy	0.94	0.92	0.91	0.86	0.91
varied_density_blobs	0.98	0.99	0.99	0.96	0.99
densired_lowdim_uniform_veryNoisy	0.98	0.94	0.89	0.80	0.91
cb_r15	0.96	0.99	1.00	0.94	1.00
densired_lowdim_varying_veryNoisy	0.99	0.90	0.87	0.78	0.88
cb_lsun	1.00	1.00	1.00	0.80	1.00
multi_density_blobs	0.99	0.99	0.99	0.91	0.99
two_moons	0.97	0.51	0.49	0.25	0.51
flame	1.00	0.99	0.98	0.91	0.99
cb_jain	0.99	0.75	0.76	0.57	0.74
cb_wingnut	1.00	0.98	0.97	0.62	0.98
cb_complex8	0.95	0.87	0.77	0.66	0.82
anisotropic	0.92	0.76	0.75	0.71	0.75
cb_complex9	1.00	0.88	0.69	0.66	0.74
cb_compound	0.98	0.95	0.94	0.90	0.95
cb_aggregation	1.00	0.97	0.97	0.89	0.97
coil20	0.96	0.81	0.78	0.69	0.79
compound	0.95	0.96	0.95	0.82	0.96
pathbased	0.91	0.97	0.97	0.81	0.97
cb_d31	0.90	0.61	0.63	0.54	0.68
iris	0.99	0.91	0.91	0.89	0.91
cb_pathbased	0.79	0.77	0.82	0.76	0.81
cb_twodiamonds	0.96	0.85	0.87	0.78	0.85
pendigits	0.79	0.60	0.44	0.49	0.53
cb_flame	0.92	0.49	0.49	0.63	0.51
cb_tetra	0.95	0.97	0.97	0.79	0.97
cb_ds3c3sc6	0.79	0.92	0.92	0.81	0.92
breast_cancer	0.89	0.74	0.74	0.74	0.74
drivface	0.93	0.84	0.85	0.73	0.86
olivetti	0.81	0.79	0.78	0.64	0.81
wine	0.81	0.74	0.79	0.86	0.65
digits	0.54	0.64	0.58	0.67	0.50
newsgroups_3000	0.61	0.65	0.62	0.48	0.52
optdigits	0.55	0.59	0.57	0.46	0.56
mnist_3000	0.54	0.52	0.49	0.55	0.51
ecoli	0.78	0.89	0.89	0.83	0.81
beans_1500	0.78	0.88	0.91	0.78	0.89
beans	0.76	0.90	0.92	0.86	0.90
cifar10_3000	0.39	0.47	0.44	0.33	0.45

D.7 DATASETS

Synthetic clusters were generated with DENSIRED (Jahn et al., 2024), which builds each cluster as a chain of local data object-cores placed by a momentum-biased random walk through the ambient space and then samples data objects around each core, producing non-convex clusters rather than isotropic blobs. We used its four primary presets (uniform density; varying per-cluster density via `dens_factors`; noisy with unbalanced cluster sizes via `clu_ratios`; noisy and balanced) together with a systematic $3 \times 2 \times 3$ grid crossing dimensionality ($d \in 2, 5, 10$), density uniformity (uniform vs. varying), and background-noise ratio ($0\%/5\%/20\%$ via `ratio_noise`), for 22 DENSIRED datasets in total.

Datasets denoted with cb are taken from the Deric benchmark `https://github.com/deric/clustering-benchmark`.

Table 12 lists all 67 datasets with their size, dimensionality, number of ground-truth clusters, and how well each reference clustering recovers the ground truth. Table 13 provides the DISCO of the ground truth and of each reference, in the same row order.

Table 12: **Per-dataset overview.** Size n, dimensionality d, number of ground-truth clusters k, and each reference clustering's agreement with the ground truth (AMI, mean over seeds). Shading is on a fixed scale from 0 (lightest) to 1 (darkest), so shades are comparable across columns. Datasets are sorted by descending ground-truth DISCO, as in Table 13; — marks references with no successful run.

				AMI vs. ground truth				
Dataset	n	d	k	HDBSCAN	DBSCAN	DPC	SHADE	Spectral
densired_uniform	900	2	3	1.00	0.98	0.77	1.00	1.00
cb_banana	4811	2	2	1.00	0.98	0.21	1.00	1.00
cb_spiral	1000	2	2	1.00	1.00	1.00	1.00	1.00
densired_varying_density	1500	2	3	0.99	0.99	0.71	1.00	1.00
cb_target	770	2	6	0.99	0.99	0.41	0.99	0.57
densired_lowdim_uniform_noiseless	1200	2	4	1.00	1.00	0.90	1.00	1.00
cb_chainlink	1000	3	2	1.00	1.00	0.16	1.00	1.00
densired_highdim_uniform_noiseless	1200	10	4	1.00	1.00	0.96	0.94	1.00
nested_rings	1000	2	2	1.00	0.95	0.19	1.00	1.00
densired_highdim_uniform_noisy	1200	10	4	1.00	0.91	0.97	0.87	0.96
densired_lowdim_uniform_noisy	1200	2	4	1.00	0.99	0.85	0.96	0.88
cb_atom	800	3	2	1.00	0.99	0.21	1.00	1.00
densired_highdim_uniform_veryNoisy	1200	10	4	1.00	0.24	0.96	0.94	0.96
densired_lowdim_varying_noiseless	1200	2	4	1.00	1.00	0.88	1.00	1.00
densired_meddim_varying_noiseless	1200	5	4	1.00	1.00	0.92	0.99	1.00
jain	373	2	2	1.00	0.91	0.37	1.00	1.00
densired_noisy_unbalanced	1200	2	4	1.00	0.79	0.99	1.00	0.94
densired_lowdim_varying_noisy	1200	2	4	1.00	0.44	0.85	0.70	0.92
densired_meddim_varying_noisy	1200	5	4	1.00	1.00	0.93	0.99	0.96
densired_meddim_uniform_noiseless	1200	5	4	1.00	1.00	0.94	0.91	1.00
densired_noisy_balanced	1200	2	4	1.00	0.80	0.91	0.97	0.96
cb_hepta	212	3	7	1.00	1.00	1.00	0.79	1.00
densired_highdim_varying_noiseless	1200	10	4	1.00	1.00	0.85	0.98	1.00
cb_zelnik3	266	2	3	1.00	0.93	0.58	1.00	1.00
densired_meddim_uniform_noisy	1200	5	4	1.00	0.75	0.97	0.99	0.96
densired_highdim_varying_noisy	1200	10	4	1.00	1.00	0.85	0.90	0.96
densired_highdim_varying_veryNoisy	1200	10	4	1.00	0.35	0.89	0.89	0.87
densired_meddim_uniform_veryNoisy	1200	5	4	1.00	0.34	0.75	0.85	0.93
densired_meddim_varying_veryNoisy	1200	5	4	1.00	0.54	0.90	0.85	0.96
varied_density_blobs	2000	2	5	0.94	0.89	0.98	0.93	0.98
densired_lowdim_uniform_veryNoisy	1200	2	4	1.00	0.95	0.93	0.84	0.93
cb_r15	600	2	15	0.96	0.73	0.98	0.65	0.99
densired_lowdim_varying_veryNoisy	1200	2	4	1.00	0.92	0.74	0.65	0.75
cb_lsun	400	2	3	0.98	0.97	0.46	1.00	1.00
multi_density_blobs	1500	2	3	0.73	0.72	0.94	0.76	0.96
two_moons	1000	2	2	0.92	0.95	0.09	0.97	0.58
flame	240	2	2	0.96	0.96	0.81	0.88	0.99
cb_jain	373	2	2	0.77	0.78	0.25	0.88	0.66
cb_wingnut	1016	2	2	0.70	–	0.61	0.92	0.98
cb_complex8	2551	2	8	0.86	0.94	0.36	0.68	0.82
anisotropic	1500	2	3	0.70	0.70	0.73	0.73	0.71
cb_complex9	3031	2	9	0.53	0.89	0.38	0.53	0.93
cb_compound	399	2	6	0.80	0.83	0.55	0.81	0.72
cb_aggregation	788	2	7	0.89	0.89	0.83	0.89	0.98
coil20	1440	1024	20	0.87	0.12	0.59	0.49	0.87
compound	798	2	6	0.76	0.58	0.78	0.65	0.85

continued on next page

Table 12 – continued from previous page

Dataset	n	d	k	AMI vs. ground truth HDBSCAN	DBSCAN	DPC	SHADE	Spectral
pathbased	300	2	3	0.55	0.74	0.68	0.64	0.57
cb_d31	3100	2	31	0.83	0.29	0.44	0.08	0.96
iris	150	4	3	0.71	0.68	0.63	0.71	0.68
cb_pathbased	300	2	3	0.58	–	0.58	0.70	0.57
cb_twodiamonds	800	2	2	0.22	–	0.02	0.35	1.00
pendigits	10992	16	10	0.72	0.02	0.22	0.01	0.81
cb_flame	240	2	2	0.41	–	0.16	0.47	0.35
cb_tetra	400	3	4	0.73	–	0.56	0.65	1.00
cb_ds3c3sc6	905	2	6	0.52	–	0.50	0.60	0.72
breast_cancer	569	30	2	0.20	–	–	–	0.66
drivface	606	14	3	0.12	0.08	0.05	0.09	0.10
olivetti	400	4096	40	0.32	–	0.23	0.22	0.68
wine	178	13	3	0.35	–	0.57	0.35	0.86
digits	1797	64	10	0.56	–	0.00	0.41	0.83
newsgroups_3000	3000	384	20	0.03	–	0.00	0.07	0.53
optdigits	5620	64	10	0.00	–	–	0.27	0.83
mnist_3000	3000	50	10	0.34	–	–	0.21	0.67
ecoli	336	7	8	0.10	–	0.13	0.38	0.57
beans_1500	1500	16	7	0.17	0.17	0.02	0.26	0.72
beans	13611	16	7	0.15	–	0.00	0.16	0.73
cifar10_3000	3000	50	10	0.03	–	–	0.02	0.09

Table 13: **Density-based validity (DISCO) per dataset.** For the ground truth (GT) and each reference clustering (mean over seeds). Datasets are sorted by descending GT DISCO. Shading is on a fixed scale over DISCO's range from -1 (lightest) to 1 (darkest), so shades are comparable across columns; — marks references with no successful run.

Dataset	DISCO GT	HDBSCAN	DBSCAN	DPC	SHADE	Spectral
densired_uniform	0.93	0.93	0.91	0.39	0.92	0.93
cb_banana	0.89	0.89	0.89	-0.09	0.89	0.89
cb_spiral	0.89	0.89	0.89	0.89	0.89	0.89
densired_varying_density	0.87	0.87	0.87	0.41	0.87	0.87
cb_target	0.86	0.87	0.87	-0.42	0.87	0.02
densired_lowdim_uniform_noiseless	0.85	0.85	0.84	0.45	0.86	0.85
cb_chainlink	0.84	0.84	0.84	0.10	0.84	0.84
densired_highdim_uniform_noiseless	0.83	0.83	0.83	0.52	0.82	0.83
nested_rings	0.81	0.81	0.75	0.03	0.81	0.81
densired_highdim_uniform_noisy	0.81	0.77	0.72	0.28	0.69	0.64
densired_lowdim_uniform_noisy	0.80	0.78	0.79	0.21	0.76	0.53
cb_atom	0.79	0.79	0.79	0.19	0.79	0.79
densired_highdim_uniform_veryNoisy	0.78	0.62	0.69	0.16	0.55	0.44
densired_lowdim_varying_noiseless	0.77	0.77	0.77	0.36	0.77	0.77
densired_meddim_varying_noiseless	0.76	0.76	0.76	0.37	0.75	0.76
jain	0.76	0.76	0.70	0.36	0.76	0.75
densired_noisy_unbalanced	0.75	0.72	0.60	0.63	0.71	0.66
densired_lowdim_varying_noisy	0.74	0.73	0.61	0.02	0.70	0.54
densired_meddim_varying_noisy	0.74	0.71	0.71	0.23	0.70	0.58
densired_meddim_uniform_noiseless	0.74	0.74	0.74	0.41	0.79	0.74
densired_noisy_balanced	0.74	0.74	0.67	0.35	0.70	0.58
cb_hepta	0.74	0.74	0.74	0.74	0.19	0.74
densired_highdim_varying_noiseless	0.72	0.72	0.72	0.29	0.71	0.72
cb_zelnik3	0.72	0.72	0.53	0.36	0.72	0.72
densired_meddim_uniform_noisy	0.72	0.69	0.69	0.39	0.67	0.56
densired_highdim_varying_noisy	0.72	0.70	0.72	0.16	0.67	0.56
densired_highdim_varying_veryNoisy	0.70	0.61	0.53	0.07	0.50	0.30

continued on next page

Table 13 – continued from previous page

	DISCO					
Dataset	GT	HDBSCAN	DBSCAN	DPC	SHADE	Spectral
densired_meddim_uniform_veryNoisy	0.69	0.56	0.68	0.11	0.52	0.36
densired_meddim_varying_veryNoisy	0.66	0.57	0.64	0.05	0.48	0.38
varied_density_blobs	0.63	0.67	0.75	0.64	0.67	0.64
densired_lowdim_uniform_veryNoisy	0.61	0.71	0.71	0.33	0.67	0.53
cb_r15	0.61	0.61	0.86	0.60	0.55	0.61
densired_lowdim_varying_veryNoisy	0.54	0.64	0.63	0.11	0.59	0.49
cb_lsun	0.53	0.54	0.54	0.22	0.53	0.53
multi_density_blobs	0.48	0.82	0.80	0.48	0.78	0.48
two_moons	0.47	0.47	0.48	-0.01	0.48	0.32
flame	0.46	0.52	0.55	0.43	0.47	0.50
cb_jain	0.44	0.46	0.45	0.04	0.43	0.33
cb_wingnut	0.39	0.31	–	0.37	0.36	0.39
cb_complex8	0.39	0.35	0.41	-0.16	0.42	0.27
anisotropic	0.36	0.47	0.73	0.25	0.58	0.33
cb_complex9	0.36	0.75	0.37	-0.13	0.75	0.30
cb_compound	0.35	0.60	0.60	-0.26	0.58	0.27
cb_aggregation	0.31	0.67	0.67	0.05	0.64	0.31
coil20	0.29	0.29	0.17	-0.07	0.14	0.27
compound	0.29	0.34	0.54	0.14	0.43	0.27
pathbased	0.27	0.26	0.41	0.19	0.18	0.15
cb_d31	0.24	0.24	0.33	-0.19	0.65	0.24
iris	0.24	0.62	0.62	0.23	0.62	0.25
cb_pathbased	0.15	0.20	–	0.13	0.13	0.13
cb_twodiamonds	0.12	0.03	–	-0.15	0.03	0.12
pendigits	0.11	0.07	0.38	-0.15	0.46	0.13
cb_flame	0.11	0.06	–	0.11	0.08	0.03
cb_tetra	0.11	0.12	–	0.08	0.10	0.11
cb_ds3c3sc6	0.09	0.04	–	0.06	0.15	0.06
breast_cancer	0.08	0.22	–	–	–	0.12
drivface	0.06	0.32	0.39	0.31	0.33	0.27
olivetti	0.00	0.07	–	-0.22	-0.02	0.00
wine	-0.00	0.12	–	-0.30	0.08	0.02
digits	-0.01	-0.01	–	-0.01	0.01	-0.01
newsgroups_3000	-0.01	0.06	–	-0.06	0.01	-0.01
optdigits	-0.01	0.72	–	–	0.03	-0.03
mnist_3000	-0.01	0.04	–	–	-0.02	-0.07
ecoli	-0.02	0.82	–	0.07	0.13	-0.02
beans_1500	-0.03	0.68	0.68	0.01	0.24	-0.04
beans	-0.04	0.39	–	-0.02	0.53	-0.07
cifar10_3000	-0.05	0.31	–	–	0.01	-0.04