SCUT : Spectral Clustering for Unsupervised classification Trees

Anonymous authors
Paper under double-blind review

Abstract

The lack of annotated data often limits the training of machine learning models. In addition, during the labelling process, some data points may remain unlabelled. While unsupervised methods such as clustering can reveal the underlying structure of the data, they are typically unsuitable to place new samples into existing clusters. Here, we propose Spectral Clustering for Unsupervised decision Tree (SCUT), a novel hierarchical clustering method based on algebraic connectivity that can position new data points appropriately within the clustering structure. By leveraging a feature-splitting approach, SCUT also enables straightforward extraction of *ante-hoc* explanations for its clustering decisions. Formally, SCUT works by recursively splitting the data through the solution of the Normalized Cut (NCUT) problem—a graph-partitioning formulation that seeks to split a graph into balanced subsets while minimizing the total connection strength between them—on a bipartite graph. We demonstrate, both visually and quantitatively, that SCUT captures the intrinsic structure of data more effectively than existing methods, while offering competitive performance compared to common hierarchical clustering algorithms.

1 Introduction

Clustering is the task of grouping similar object together. Given a set of data points (also called observations) and a pairwise distance (or similarity) measure, clustering consists of forming homogeneous groups of observations. In most clustering problems, data is unlabelled, making clustering primarily an unsupervised learning technique widely used to gain insights into the underlying structure of the data. This structure can be hierarchical: for instance, in a dataset of movies, films can be grouped by genre and further subdivided into subgenres. Hierarchical clustering algorithms aim to capture such multilevel structures within a single dendrogram.

Methods for hierarchical clustering are typically divided into two categories: agglomerative and divisive. Agglomerative approaches, such as Ward's method, proceed bottom-up, recursively merging the best pair of clusters until only one remains. Divisive approaches work top-down, splitting clusters into subclusters until each cluster contains a single element. Let m denote the number of features and n the number of observations. Most research has focused on agglomerative methods, as merging requires examining only $\Theta(n^2)$ pairs of clusters, while splitting a cluster into two subclusters involves considering $\Theta(2^n)$ possibilities. Nevertheless, agglomerative methods typically have a time complexity of at least $\Theta(mn^2)$— often $\Theta(mn^3)$— which limits their scalability for large datasets.

In this work, we propose a novel divisive hierarchical clustering method called SCUT, for Spectral Clustering for Unsupervised decision Trees, with a worst-case complexity of $\mathcal{O}(mn^2)$ and best-case complexity of $\Omega(mn \log n)$, offering an efficient alternative to classical approaches. We also propose a simple modification of SCUT to make its worst case complexity $\mathcal{O}(mn \log n)$ at a slight cost of accuracy. Our method also yields improved clustering performance compared to traditional divisive techniques. SCUT seeks an optimized solution to the NCUT problem on a bipartite graph and extracts algebraic connectivity—the second-smallest eigenvalue of the graph Laplacian—as the basis for an efficient divisive hierarchical clustering strategy. Be-

yond efficiency, the structure of our approach naturally supports the extraction of *ante-hoc* explanations for its clustering decisions, making it particularly well-suited to applications where interpretability is essential.

2 Material and Methods

We begin by defining the notation and framework—previously introduced in several studies such as Von Luxburg (2007)—used to describe our novel method, SCUT.

2.1 Spectral clustering

Let $G = (V, E)$ be a weighted graph and w a positive weight function for edges in G. The volume of X is defined as

$$vol(X) = \sum_{u \in X, v \in V} w(u, v)$$

where X is a subset of V. Intuitively, the volume measures the weight of the connections in X. If V is partitioned into two disjoint sets (X, Y), that is $X \cup Y = V$ and $X \cap Y = \emptyset$, the *cut* of the partition (X, Y) is defined as

$$cut(X, Y) = \sum_{u \in X, v \in Y} w(u, v)$$

Intuitively, the cut of a partition measures the total weight of the connections between subsets X and Y. Note that $cut(X, Y) = cut(Y, X)$ and $cut(X, Y) \leq vol(X)$.

In clustering, the goal is to partition a set into subsets that are as homogeneous as possible. In our case, this means identifying subsets that minimize the *cut* value. As noted in Shi & Malik (2000), directly minimizing this function for partitioning often results in highly unbalanced partitions. To address this issue, they introduced the notion of *normalized cut*:

$$Ncut(X, Y) = \frac{cut(X, Y)}{vol(X)} + \frac{cut(X, Y)}{vol(Y)}$$

Since the edges that define $cut(X, Y)$ are a subset of the edges defining $vol(X)$ and $vol(Y)$, it follows that $cut(X, Y) < vol(X)$ and $cut(X, Y) < vol(Y)$. Intuitively, minimizing $Ncut$ means to search for a nearly equally-sized partition (X, Y) where $vol(X)$ and $vol(Y)$ are comparable, and X and Y are weakly connected.

Formally, using Theorem 3 from Dhillon (2001), the $Ncut(X, Y)$ objective function can be conveniently rewritten using the unnormalized graph Laplacian $D - A$ as:

$$Ncut(X, Y) = \frac{q^\intercal (D - A) q}{q^\intercal A q}$$

where D is a $n \times n$ diagonal matrix with $D_{ii} = vol(\{u_i\})$ and $D_{ij} = 0$ for $i \neq j$, A is a $n \times n$ weight matrix with $A_{ij} = w(u_i, v_j)$ if $i \neq j$ and $A_{ii} = 0$, and q is a partitioning vector representing (X, Y):

$$q_i = \begin{cases} +\sqrt{\frac{vol(Y)}{vol(X)}} & \text{for } u_i \in X \\ -\sqrt{\frac{vol(X)}{vol(Y)}} & \text{for } u_i \in Y \end{cases}$$

The normalized cut problem becomes:

$$\min_q \frac{q^\intercal (D - A) q}{q^\intercal A q}$$

This is a discrete optimization problem as the entries of the partitioning vector q are only allowed to take two particular values. Most importantly, this problem is NP-complete Shi & Malik (2000). Spectral

graph partitioning is an effective heuristic to find solutions avoiding as much as possible local minima barriers Donath & Hoffman (1972); Shi & Malik (2000); Ng et al. (2001). It discards the discreteness condition and instead it allows that the q_i's take arbitrary values in $\mathbb{R}$. This leads to the relaxed optimisation problem

$$\min_{q \in \mathbb{R}^n} q^{\mathsf{T}}(D-A)q \qquad \text{subject to } q \perp \mathbb{1}, \parallel q \parallel = \sqrt{n}$$

By the Rayleigh-Ritz Theorem Lütkepohl (1997), it can be seen that the solution of this problem is given by the vector q which is the eigenvector corresponding to the second smallest eigenvalue of $D-A$ (recall that the smallest eigenvalue of $D-A$ is 0 with eigenvector $\mathbb{1}$), also called Fiedler vectorFiedler (1973). So we can approximate a minimizer of $Ncut(X,Y)$ by the second eigenvector of the Laplacian $D-A$. However, in order to obtain a partition of the graph, we need to re-transform the real-valued solution vector q of the relaxed problem into a discrete partitioning vector. This can be done by using the sign of q as partition function, that is by chosing $X = \{u_i \in V, q_i \geq 0\}$ and $Y = \{u_i \in V, q_i < 0\}$.

2.2 Spectral Clustering with Bipartite Graphs

In many practical applications, data is represented in the form of a tabular matrix. Specifically, let W be an $n \times m$ tabular matrix, where n is the number of observations and m the number of features. Typically, the number of observations exceeds the number of features ($n > m$). Applying clustering directly to the n observations is equivalent to constructing a graph with n nodes and $\frac{n(n-1)}{2}$ edges, which becomes computationally expensive when n is large. An alternative is to model the data as a bipartite graph consisting of n observation nodes and m feature nodes, with edge weights defined by the entries of a matrix W. Although this approach may seem more costly at first—since the matrices A and D would be of size $(n+m) \times (n+m)$—the bipartite nature of the graph allows for a more efficient representation. In particular, matrix A takes the form of a symmetric block matrix:

$$A = \begin{bmatrix} 0 & W \\ W^{\mathsf{T}} & 0 \end{bmatrix}$$

This structure enables several computational optimizations. In fact, it has been shown Dhillon (2001); Zha et al. (2001) that clustering the n observations (and/or the m features) can be performed without explicitly constructing the full matrix A; only the original matrix W is needed for the computation. More formally, we consider a bipartite graph $\mathrm{G} = (\mathrm{U} \cup \mathrm{V}, \mathrm{E})$, where $U \cap V = \emptyset$ and $\forall (a,b) \in E,\ (a \in U \wedge b \in V) \vee (a \in V \wedge b \in U)$, and let $n = |U|$, $m = |V|$. G has weighted edges and it is represented by a matrix $W \in \mathbb{R}^{n \times m}$ where each entry $W_{i,j} = w(u_i, v_j)$, for $i = 1 \ldots n$ and $j = 1 \ldots m$, denotes the weight of the edge between node u_i, an observation, and node v_j, a feature. For bipartite graphs G, the diagonal matrix D becomes the symmetric block matrix:

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}$$

where $D_1(i,i) = vol(\{u_i\})$ and $D_1(i,k) = 0$ for $i \neq k$, and $D_2[j,j] = vol(\{v_j\})$ and $D_2[k,j] = 0$ for $j \neq k$. In practice, only the diagonals of the matrices D_1 and D_2 are needed for the computations.

As shown in (Dhillon, 2001; Zha et al., 2001), we can find the Fiedler vector f of G by using a singular value decomposition (SVD) of $D_1^{-1/2} W D_2^{-1/2}$, where $D_h^{-1/2}[k,k]$ denotes $\sqrt{D_h[k,k]}$ for $h = 1, 2$. The Fiedler vector f is the concatenation of the second largest left and right singular vectors, respectively denoted f_u, f_v. Based on them, the partition is defined as $X_U = \{u_i \in U, f_u \geq 0\}$ and $Y_U = \{u_i \in U, f_u < 0\}$, and $X_V = \{v_i \in V, f_v \geq 0\}$ and $Y_V = \{v_i \in V, f_v < 0\}$ (Dhillon, 2001; Zha et al., 2001). Namely, f_u gives a bipartitioning of the observations and f_v of the features.

2.3 The SCUT method

In this section, we introduce SCUT (Spectral Clustering for Unsupervised classification Trees), a novel method for hierarchical clustering of tabular data. SCUT consists of two main components: tree construction and inference. The tree is constructed by recursively applying a modified version of the spectral

clustering algorithm proposed in Dhillon (2001); Zha et al. (2001), resulting in a dendrogram that hierarchically partitions the data. At each recursive step, the algorithm computes and stores a linear projection that fully characterizes the corresponding partition. During inference, these learned linear projections are used to assign new data points to appropriate leaves of the decision tree.

SCUT introduces several technical innovations that lead to important improvements over existing spectral clustering approaches:

- At each recursive step, SCUT clusters samples while retaining all original features, avoiding feature reduction.
- It interprets the Fiedler vector f_u as samples from a one-dimensional distribution, which can be analyzed based on its density.
- It introduces an original way (Algorithm 2) to assign new samples to a previously constructed tree.
- SCUT leverages a physical analogy to interpret f_v, offering *ante-hoc* explanations for clustering decisions Zha et al. (2001).
- Complexity analysis shows that SCUT achieves a best-case time complexity of $\Omega(mn \log n)$ and a worst case complexity of $\mathcal{O}(mn^2)$, offering improvements over other methods.

As in Dhillon (2001); Zha et al. (2001), SCUT operates on non-negative tabular data. Various preprocessing techniques, such as min-max scaling or sigmoid transformations, can be applied to convert real-valued data into non-negative form. It is worth noting that categorical data can be handled through one-hot encoding.

2.3.1 Split characterisation from properties of the SVD

We use properties of SVD to learn a linear projection that characterizes a split. Let $W_N = D_1^{-1/2} W D_2^{-1/2}$ be the normalized matrix. Its SVD is given by:

$$W_N = U \Sigma V^\intercal$$

where U and V are the matrices whose columns are the left and right singular vectors, respectively, and Σ is a diagonal matrix containing the singular values $\sigma = \text{diag}(\Sigma)$ in decreasing order. Hence, for the second-largest singular value σ_2 of W_N, we have

$$W_N V[2] = \sigma_2 U[2] \qquad \text{and} \qquad W_N^\intercal U[2] = \sigma_2 V[2]$$

where $U[2]$ and $V[2]$ denote the second columns of U and V respectively. Defining $f_u = U[2]$ and $f_v = V[2]$, we identify f_u and f_v as the left and right singular vectors associated with σ_2. The above relations can then be written compactly as

$$W_N f_v = \sigma_2 f_u \qquad \text{and} \qquad W_N^\intercal f_u = \sigma_2 f_v.$$

Given a new observation x to classify, we can use the learned parameters σ_2, f_v, $D_2^{-1/2}$ and the threshold th (discussed below) to determine the partition in which x falls. We define the scaling $T : x \mapsto x_N$ by

$$x_N \;=\; \frac{x}{\sqrt{\sum_i x_i}}$$

T is a scaling defined for a given point, that, when applied to the whole W matrix, becomes $T(W) = D_1^{1/2} W$. Note that we do not need to define an additional mapping for $D_2^{1/2}$, since it is specified for each given feature and is therefore "learned" directly from W. We then compute the projection score

$$s(x) = \frac{f_v^\intercal x_N D_2^{-1/2}}{\sigma_2},$$

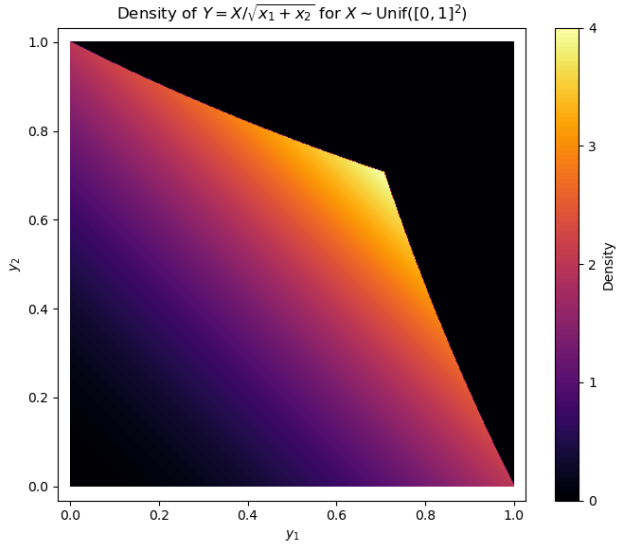


Figure 1: **Visualisation of the distortion induced by the function T mapping x to x_N.** Assuming $X \sim Unif([0,1]^2)$, and $Y = T(X)$, we plot f_Y, the density of Y.

and assign x to one side of the split according to whether $s(x) > th$ or $s(x) \leq th$.

A linear classifier can be defined as $\text{clf} : \mathbb{R}^m \to [-1, 1]$ that associates to x_n the predicted class $\hat{y}$ given by

$$\hat{y} = \text{sign}(\frac{f_v^\intercal x_N D_2^{-1/2}}{\sigma_2} - th)$$

where th is a threshold parameter.

While the scaling T is necessary for extracting clusters, as previously established Dhillon (2001); Zha et al. (2001), it also compresses data points near the set $\{x \in \mathbb{R}^m | x = T(x)\}$, as illustrated in two dimensions in Figure 1.

2.3.2 SCUT hierarchical partitioning

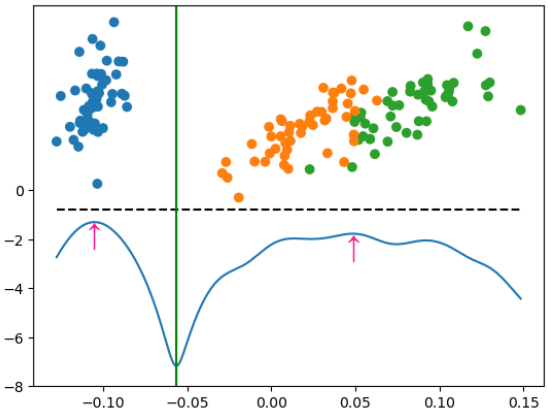
W is a non-negative weight matrix with n observations and m features. In this setting, the goal is to cluster the observations while retaining all m features at each recursive step of the algorithm. To build the hierarchical partitioning, SCUT recursively splits the data by extracting the Fiedler vector f_u from a matrix W_N, which is updated and recomputed at each step of the recursion with respect to the m features using the SVD. Rather than splitting the data solely based on the sign of the components of f_u—a common practice that can lead to suboptimal or unstable clustering—SCUT introduces a flexible, data-driven strategy for determining a threshold th that defines the partition.

Since the components of f_u represent a linear projection of the n observations (sampled from an m-dimensional space) onto a one-dimensional space, the vector f_u can be interpreted as a sample from a one-dimensional distribution with probability density function p. In this context, regions of high density in p are likely to correspond to meaningful clusters. Therefore, SCUT selects a threshold such that the cut:

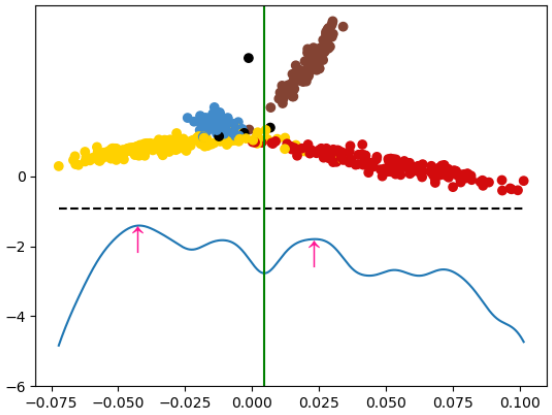
- occurs at a point of low density, and
- separates distinct high-density regions.

Locations that verify these conditions are usually called "valleys". The threshold th is defined as:

$$th = \arg\max_z \left[\left(\max_{x<z} \log \frac{p(x)}{p(z)} \right) \cdot \left(\max_{x>z} \log \frac{p(x)}{p(z)} \right) \right]$$



(a) First threshold selection on the Iris dataset



(b) First threshold selection on the French National Assembly dataset

Figure 2: **Illustrations of threshold selection.** (a): the Iris dataset. The x-axis represents the one-dimensional space containing $f_u = U[2]$. On the y-axis, below the dashed horizontal line: estimated log-density of f_u; above the dashed horizontal line: scatter plot of $(U[2], U[3])$, where $U[3]$ is used for visualization purposes only. Colors indicate species (blue: *I. setosa*, orange: *I. versicolor*, green *I. virginica*). The vertical green line marks the selected threshold th, and pink arrows highlight the maximum density peaks on the left and right of th. (b): the French National Assembly dataset. x-axis, y-axis, dashed horizontal line, vertical green line, and pink arrows as on left. Colors correspond to political alignment, "Non-attached" are in black.

where, $\max_{x<z} p(x)$ and $\max_{x>z} p(x)$ correspond to the highest peaks of the density to the left and right of z, respectively. Intuitively, SCUT seeks a threshold located in a valley between two peaks, where the drop in log-density on both sides is significant. Note that valleys might not exist; in such cases, the above formula will yield a low density location. This is not necessarily undesirable, since valleys do not occur when attempting to split a homogeneous cluster. Let

$$h : z \to \left[\max_{x<z} \log \frac{p(x)}{p(z)}\right] \cdot \left[\max_{x>z} \log \frac{p(x)}{p(z)}\right]$$

where p denotes the estimated density. We define a confidence metric

$$\text{conf} = \max_z \left[1 - \frac{1}{e^{\frac{\sqrt{h(z)}}{2}}}\right]$$

which takes values in $[0, 1]$ and measures how well defined the clusters are. Intuitively, conf measures the quality of the split based on density: 0 indicates a poor split, and 1 indicates a perfect one. Note that the 2 in the denominator of the exponential serves as a scale parameter, controlling the slope of conf. Empirically, we found that using 2 worked well with our other hyperparameters. conf has the following properties:

- $\text{conf} = 1$ when $p(th) \to 0$ (i.e., a deep valley).
- $\text{conf} = 0$ when $p(th) = min(\max_{x<z} p(x), \max_{x>z} p(x))$ (i.e., no drop in density at threshold th)
- as a function of $p(z)$, in $[0, min(\max_{x<z} p(x), \max_{x>z} p(x))]$, conf is strictly increasing, being the composition of strictly monotone functions.

SCUT uses the log-density rather than the raw density because differences in log-density reflect relative changes—i.e., density ratios—which are more meaningful for identifying significant separations. Taking the product of the two differences emphasizes that both sides of the threshold should show strong contrast with the surrounding peaks; this models the logical “and” condition needed to detect a meaningful cut.

Algorithm 1 HierarchicalSpectralClustering

Require: A positive data matrix W and root node t
Ensure: A hierarchical clustering of W, starting at the node t.
if W has only 1 row **then**
  t.is_leaf $\leftarrow True$
  return t
end if
Compute D_1 and D_2 from W
$W_N \leftarrow D_1^{-1/2} W D_2^{-1/2}$
$U, \sigma, V \leftarrow \text{SVD}(W_N)$
$f_u \leftarrow U[2]$
Consider f_u as a series of 1-d samples from a 1-d probability distribution :
$p \leftarrow$ a density estimation given f_u
$th \leftarrow \arg\max_z [[\max_{x<z} \log \frac{p(x)}{p(z)}] \cdot [\max_{x>z} \log \frac{p(x)}{p(z)}]]$
$l \leftarrow \{i \in U, f_u[i] < th\}$
$r \leftarrow \{i \in U, f_u[i] \geq th\}$
$t.left \leftarrow$ HierarchicalSpectralClustering$(W[l], t.left)$
$t.right \leftarrow$ HierarchicalSpectralClustering$(W[r], t.right)$
return t

The selection of the threshold is illustrated on Figure 2. On the Iris dataset, shown in Figure 2a, a threshold of 0 would misclassify some *I. versicolor* as *I. setosa*, whereas choosing a value at the density valley prevents such errors. In the French National Assembly dataset, shown in Figure 2b, multiple density valleys appear because the distribution contains several peaks—one for each political ideology, with multiple peaks within the left-aligned cluster. Selecting the threshold th according to our criterion yields a split that correctly separates deputies from the majority and the opposition.

Once the optimal threshold th is selected, SCUT defines the two partitions:

$$X = \{u_i \in V \mid f_u[i] \geq th\} \quad \text{and} \quad Y = \{u_i \in V \mid f_u[i] < th\}$$

The same partitioning procedure is then recursively applied to the submatrices $W[\{i \mid u_i \in X\}]$ and $W[\{i \mid u_i \in Y\}]$ until each resulting matrix contains only a single observation. The complete recursive procedure is detailed in Algorithm 1.

2.3.3 SCUT inference for new observations

New data is frequently discovered, and researchers may wish to incorporate it into an existing hierarchical clustering. However, most hierarchical clustering methods do not support such updates and require the entire clustering to be recomputed—a process that is computationally expensive and may result in significant changes to the structure of the dendrogram. In contrast, SCUT enables the insertion of new observations into an existing dendrogram with minimal structural modifications. To support this, during the tree construction phase, SCUT stores a linear classifier clf_t at each internal node t of the tree. Given a new data point x, it traverses the tree recursively: at each node t, it evaluates the classifier $clf_t(x)$. If $clf_t(x) < 0$, the point is assigned to the left child t.`left`; otherwise, it proceeds to the right child t.`right`. This process is repeated until a leaf node is reached.

The complete inference procedure is presented in Algorithm 2. Note that this algorithm is structurally identical to standard decision tree inference, except that the branching decisions are based on learned linear classifiers rather than simple thresholding on individual features.

2.4 SCUT explainability

By construction, each node in the tree corresponds to a convex subspace of $\mathbb{R}^m$ that contains all data points associated with its descendant leaves. Note that this convexity holds in the space obtained after applying

Algorithm 2 UnsupervisedDecisionTreeInference

Require: x, t,
Ensure: A predicted leaf for input x.
1: **if** t.is_leaf **then**
2: **return** t
3: **end if**
4: $\hat{y} \Leftarrow clf_t(x)$
5: **if** $\hat{y} < 0$ **then**
6: **return** UnsupervisedDecisionTreeInference$(x, t.left)$
7: **else**
8: **return** UnsupervisedDecisionTreeInference$(x, t.right)$
9: **end if**

the scaling $T : x \to \frac{x}{\sqrt{\sum_i x_i}}$ (Figure 1). The root node covers the entire space $\mathbb{R}^m$. Let e be an internal node, and let $S \subset \mathbb{R}^m$ denote the associated subspace. The left child $e.\texttt{left}$ corresponds to the region $\{x \in S \mid clf_e(x) < 0\}$, while the right child $e.\texttt{right}$ corresponds to $\{x \in S \mid clf_e(x) \geq 0\}$. These regions are convex by construction, meaning that for any $x, y \in S$, the line segment between x and y lies entirely within S. This convexity ensures that the clusters are geometrically coherent and stable under interpolation. Proof of this claim is detailed in the Supplementary Material.

Further insight into the interpretability of SCUT can be gained from a physical analogy. Following the interpretation of spectral clustering as the analysis of a mass-spring system under transverse vibrations Demmel (1999); Pothen et al. (1990), the Fiedler vector f_u can be seen as describing the displacement of the "observation nodes" at the second lowest resonating frequency during vibration. While f_v describes the displacement of the "feature nodes." Since the positions of the feature nodes directly influence how strongly they "pull" on the observation nodes, this relationship offers an intuitive explanation of feature importance.

Given a new point x to be explained in terms of the clustering decision, and using the normalized form $x_n = \frac{x}{\sqrt{\sum_i x_i}}$ as defined previously, the element-wise product $f_v \odot (x_n D_2^{-1/2})$ quantifies the influence—or "pull"—each feature exerts on x during the partitioning step. This provides a mechanism for ante-hoc interpretability by revealing which features were most influential in assigning x to a particular branch of the tree.

2.5 Metrics to validate the predictions

Predictions have been compared to true labels using the accuracy and Matthews correlation coefficient (MCC) metrics. Let:

- c be the total of samples correctly predicted
- s the number of samples
- t_i the number of times class i truly occurred
- p_i the number of times class i was predicted

We define accuracy (ACC) and MCC as follows:

$$ACC = \frac{c}{s}; \qquad MCC = \frac{cs - t \cdot p}{\sqrt{s^2 - p \cdot p}\sqrt{s^2 - t \cdot t}}$$

2.6 Datasets

We compared our method to other hierarchical clustering algorithms, including the approach proposed in Zha et al. (2001). Experiments were conducted on four datasets spanning different data types: textual

data from the 20 Newsgroups dataset (retrieved using the scikit-learn library Pedregosa et al. (2011)), floral phenotype measurements from the Iris dataset Fisher (1936) (also retrieved using scikit-learn), political voting data from the French National Assembly[1], and user-item interaction from the Goodreads dataset Wan & McAuley (2018). For each dataset, we detail the preprocessing steps:

20 Newsgroups. We applied TF-IDF vectorization to the text corpus, and removed terms (columns) whose total TF-IDF score across all documents (rows) was less than 2. This produced a sparse matrix of size $18,846 \times 11,812$—corresponding to $18,846$ documents and $11,812$ retained terms.

Iris. We first scaled the four original features to the $[0, 1]$ range using min-max normalization. To enrich the feature space, we created four additional features by subtracting each of the normalized features from 1. The final dataset contains 8 features per observation, all running between 0 and 1.

French National Assembly. We collected voting records from the 16th legislature of the French National Assembly, spanning June 22, 2022 to June 9, 2024. We retained all deputies who voted at least once, and all legislative votes, excluding senators who only participated in the constitutional vote on March 4, 2024. This yielded a dataset of 605 deputies and 4,106 votes. We attributed political nuance to each of the political groups using the official attribution from 2023[2]. Each entry in the data matrix encodes a deputy's vote as follows: 1, if they voted in favor; 0, if they voted against; $\frac{0.5+r}{2}$, if they did not vote, where r is the result of the vote (1 if adopted, 0 if rejected). Since political alliances were prominent during this legislature, we assigned a political label to each deputy based on their party affiliation. Deputies were categorized into five groups: left-wing, center, right-wing, far-right, or non-attached.

Goodreads. The Goodreads dataset Wan & McAuley (2018)[3] comprises user-book interaction records collected from the Goodreads platform. Metadata are available for books, including user-generated genre annotations obtained via a simple keyword-matching procedure, whereas no metadata are provided for users. Due to the high sparsity of the interaction matrix, we applied core filtering to retain approximately 28,000 users and 13,000 books.

3 Results

The divisive hierarchical clustering can be represented as a binary tree, where each leaf corresponds to an observation. To compare two such trees built on the same dataset, we developed two evaluation methods: a top-down annotation method (TD) and a bottom-up annotation method (BU). Both approaches assign a "predicted" class to each leaf; given a leaf, the two methods find a subtree that contains it and predict a class for it by returning the most common label in the subtree (excluding the leaf itself).

BU selects the smallest subtree containing the given leaf that includes at least k leaves, where k is a user-defined hyperparameter. The predicted label is the most frequent class among the other leaves in this subtree. Note that we used the same k to test both SCUT and Zha et al. (2001) algorithm. Namely, we set $k = 5$ for the Iris dataset, $k = 15$ for the French National Assembly dataset, and $k = 20$ for the 20 newsgroups dataset.

In contrast, TD starts at the root of the tree and traverses downwards. At each node, it compares the label distribution of its full subtree to that of its child subtrees. If the Kullback-Leibler (KL) divergence between the overall distribution and that of the child containing the target leaf exceeds a threshold θ, the traversal continues to that child. This process is repeated recursively until the KL divergence falls below θ, and the label distribution of the final subtree is used to assign a predicted class to the leaf. This procedure create a partition of the leaves into clusters, some cluster might contain only a few leaves and as such are probably outliers. We consider clusters having less than k leaves to be outliers, using the same k as in BU. On each hierachical clustering, we searched for the best θ to maximize ACC given k.

[1] `https://data.assemblee-nationale.fr/archives-16e/votes`
[2] `https://www.legifrance.gouv.fr/circulaire/id/45472`
[3] `https://cseweb.ucsd.edu/~jmcauley/datasets/goodreads.html`

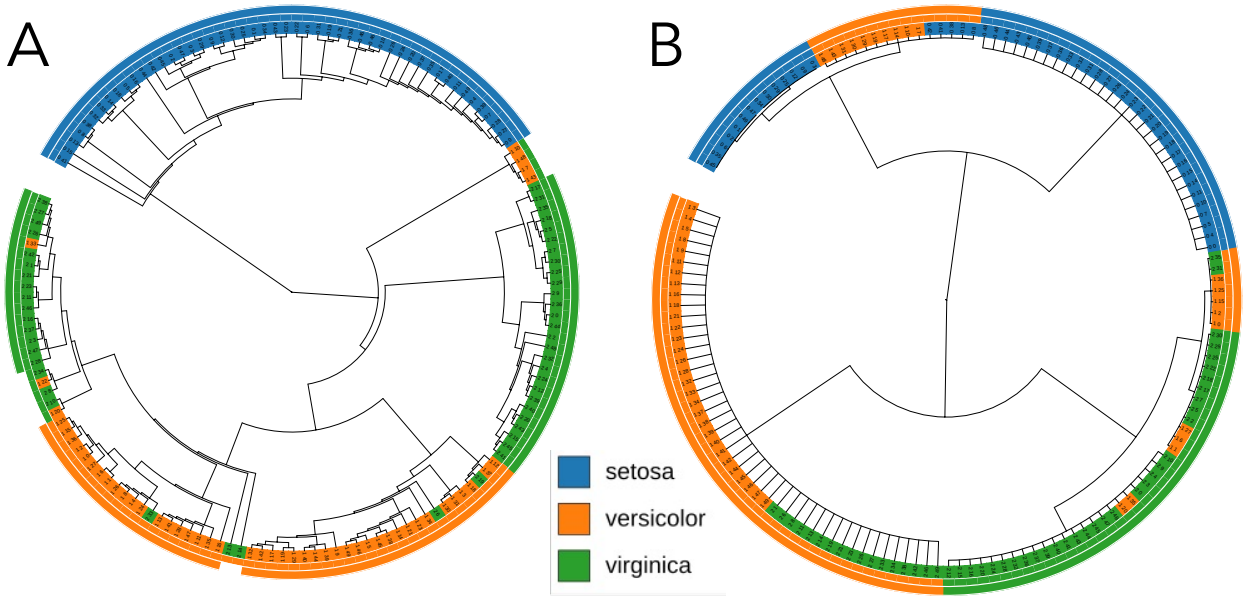


Figure 3: **Hierarchical clustering of the Iris dataset.** Comparison of hierarchical clusterings produced by the SCUT algorithm (A) and the Zha et al. Zha et al. (2001) algorithm (B) on the Iris dataset, which includes setosa (blue), versicolor (orange) and virginica (green) Iris flowers. In both dendograms, the three concentric rings represent the ground-truth class labels (inner ring), BU predictions (middle ring), and TD predictions (outer ring).

3.1 SCUT on the Iris dataset

The Iris dataset Fisher (1936) is one of the most widely used benchmarks in machine learning. It contains measurements of 150 Iris flowers, each belonging to one of the three species: *Iris setosa*, *Iris versicolor*, *Iris virginica*. Each flower is described by four features: petal length, petal width, sepal length, and sepal width. We applied min-max normalization to scale all features to the $[0, 1]$ range. To enrich the representation, we introduced four additional features by subtracting each normalized feature from 1. This transformation offers a complementary interpretation of each feature. For example, the original "petal length" feature yields two perspectives: "long petal" and "short petal".

SCUT produces a tree that perfectly clusters the *I. setosa* class (Figure 3A). While some mixing occurs between *I. versicolor* and *I. virginica*, SCUT largely recovers the underlying structure of the dataset. This mixing appears because the *I. versicolor* and *I. virginica* clusters almost overlap, with no density valley separating them. In contrast, the algorithm of Zha et al. Zha et al. (2001) splits not only observations but also features, and since there are only eight features, the resulting tree is influenced by too few of them. This limits expressiveness and leads to poor clustering (Figure 3B).

Intuitively, BU and TD scores reflect how well the clustering captures local and global structures, respectively. For BU and TD, we set the hyperparameter setting the minimal number of leaves in a subtree k at 5 (see above).

On this toy dataset, both the quantitative metrics in Table 1 (top) and the dendrogram visualizations in Figure 3 demonstrate that SCUT more effectively captures both the global and local structure of the data compared to Zha et al. algorithm.

Note that in the TD approach, Iris clusters should have ≥ 5 leaves and this leaves some leaves unlabelled in the tree of Figure 3.

Method Annotation	SCUT	Zha Bottom-Up	Ward	SCUT	Zha Top-Down	Ward
			The Iris dataset			
MCC	0.88	0.72	–	0.96 (0.86)	0.72	–
ACC	0.92	0.81	–	0.97 (0.90)	0.81	–
			The French National Assembly dataset			
MCC	0.94	0.88	–	0.94	0.88	–
ACC	0.96	0.92	–	0.96	0.92	–
			The 20newsgroups dataset			
MCC	0.73	0.56	0.60	0.71 (0.64)	0.44 (0.42)	0.51 (0.45)
ACC	0.78	0.65	0.68	0.76 (0.70)	0.55 (0.54)	0.60 (0.53)

Table 1: **Comparative performance across three datasets of varying classification difficulty.** Classification performance summary for the SCUT algorithm, the method of Zha et al. (2001) and Ward's method Ward Jr (1963) over the two annotation strategies TD and BU. The SCUT(TD) column shows scores without outliers and scores with outliers counted as misprediction in parenthesis. Top: the Iris dataset; the minimal size of cluster k is set to 5. Middle: the French National Assembly dataset; k is set to 15, corresponding to the minimal number of lawmakers in a parliamentary group; we also defined the class of outliers predicted by TD as "non-attached". Bottom: the 20Newsgroups dataset; k is set to 20.

3.2 SCUT on voting data from the French National Assembly

The voting behavior of French lawmakers offers a rich and nuanced dataset for clustering, as individual voting choices are shaped by both personal views and broader ideologies, party, and coalition alignments.

The 16^{th} legislature of the French National Assembly exemplifies this complexity. During the recent elective term, there were 577 parliamentary seats. Of these, 250 were aligned with the governing majority, distributed among three political groups: Renaissance (RE, 169 seats), Democratic Movement (DEM, 50), and Horizons (HOR, 31). The opposition held 320 seats, the majority of which belonged to the NUPES coalition—including France Insoumise (LFI, 75), Socialist Party (SOC, 31), Communist Party (GDR, 22), and Europe Ecology – The Greens (ECO, 21). The remaining opposition seats were held by the National Rally (RN, 88), The Republicans (LR, 61), and the LIOT group (22). Note that although LR officially positions itself as an opposition party, its voting behavior frequently aligns with the majority. Seven seats were unassigned to any political group.

The dataset comprises 4,106 roll-call votes. For each vote, we record the position of every present lawmaker: in favor, against, or abstention. Due to substitutions and temporary replacements, the dataset includes 605 unique lawmakers, exceeding the number of official seats. Notably, in this legislature—characterized by the absence of a clear majority and the presence of fragile alliances—abstention or non-participation often served as a deliberate political signal. For example, a lawmaker might choose not to vote as a way of expressing disagreement with a proposed law without actively opposing its adoption.

To account for this subtlety, we represent abstentions or absences with a value close to the vote's outcome. Specifically, the data is encoded in a 605×4106 matrix where 0 indicates a vote against, 1 indicates a vote in favor, and $\frac{0.5+r}{2}$ is assigned to missing votes, with $r \in 0, 1$ denoting the result of the vote (0 for rejected, 1 for adopted).

For visualization and evaluation, each lawmaker is labeled according to the political leaning of their group: left-wing, center, right-wing, far-right.

Both clustering algorithms successfully recover the overall structure of the voting data. The initial split in both trees (Figure 4AB) clearly separates government-aligned groups from the opposition. While the LR group officially identifies as part of the opposition, many political analysts consider it generally aligned with the government—a nuance that is correctly reflected by both clustering methods. Within the opposition, both algorithms further distinguish between the far-right National Rally (RN) and the left-wing NUPES

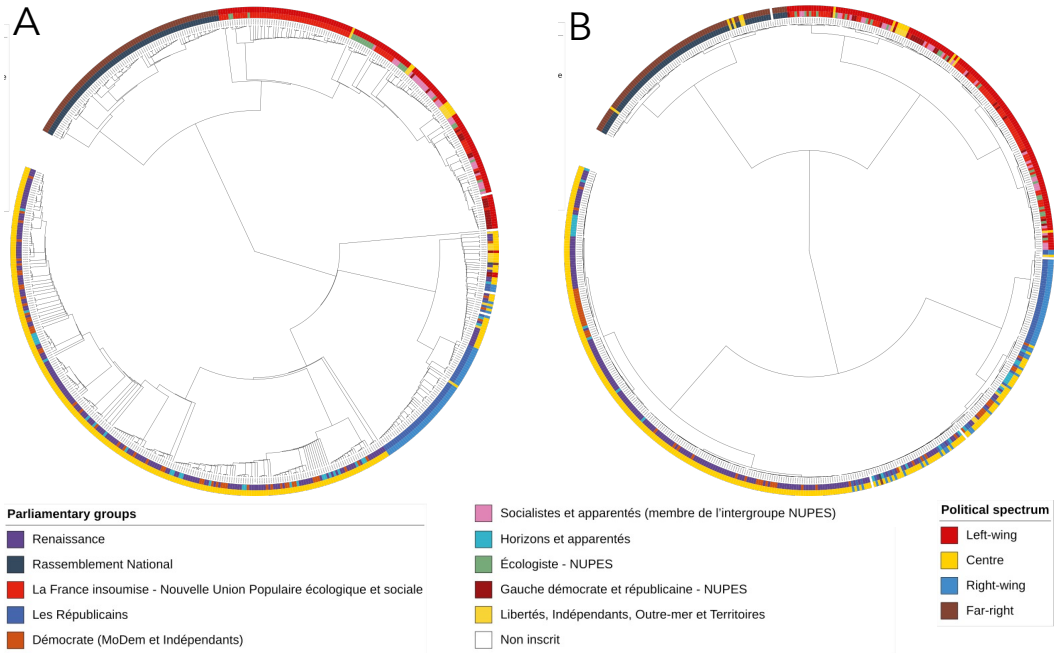


Figure 4: **Hierarchical clustering of the French National Assembly dataset.** Comparison of hierarchical clusterings produced by the SCUT algorithm (A) and the Zha et al. (2001) algorithm (B) on the French National Assembly dataset, comprising eleven Parliament parties/alliances organising four main political spectra. In both dendograms, the two concentric rings represent the ground-truth political alliances labels (inner ring) and the political leaning predictions (outer ring).

coalition. On the government side, both methods identify a distinct subcluster corresponding to LR, as well as a larger cluster representing the formal majority parties.

As shown by the BU and TD predictions scores (in Table 1, middle), SCUT effectively captures both the local and global structures in the voting data. Using the TD annotation method, SCUT is even able to correctly identify unaffiliated (non-attached) lawmakers based solely on their voting patterns. Moreover, SCUT's misclassifications tend to occur between politically adjacent groups-for example, many LIOT (centre) lawmakers are misclassified as left-leaning. Notably, only a single RN (far-right) lawmaker is incorrectly predicted to be a centrist. While the algorithm by Zha also captures much of the local and global structure, its performance is consistently below that of SCUT. It exhibits a higher rate of misclassification, including more severe errors-for instance, in some cases, centrist lawmakers are predicted to belong to the far-right, and vice-versa. As illustrated in Figure 4, a significant portion of this discrepancy in performance stems from Zha's difficulty in correctly classifying LR (right-leaning) lawmakers, many of whom are incorrectly assigned to the centrist group.

3.3 20Newsgroup dataset

The 20newsgroups dataset is a collection of $18,846$ documents, each corresponding to a post from one of 20 Usenet newsgroups. These categories can be grouped into broader thematic areas; fror example, the baseball and hockey newsgroups both fall under the general theme of sports.

To construct a weight matrix representing a bipartite graph between documents and terms, we applied TF-IDF vectorization to the text corpus. The resulting matrix was high-dimensional and sparse, with many terms appearing in only a few documents. To reduce noise and dimensionality, we removed columns (i.e., terms) whose total TF-IDF score across all documents (rows) was less than 2. This filtering step yielded a final matrix containing $11,812$ terms.

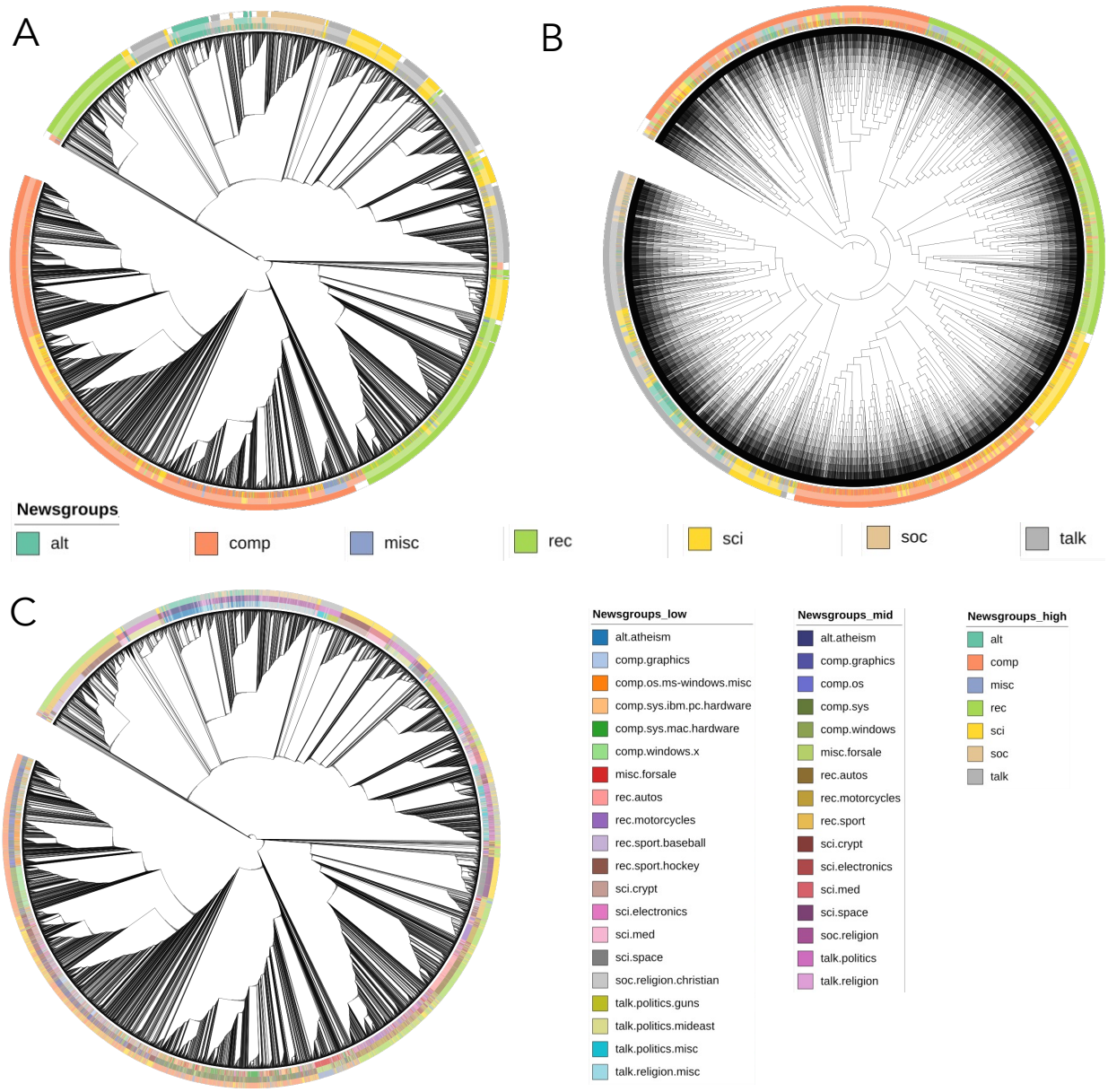


Figure 5: **Hierarchical clustering of the 20Newsgroups dataset.** Comparison of hierarchical clusterings produced by the SCUT algorithm (A) and the Zha et al. Zha et al. (2001) algorithm (B) on the 20Newsgroup dataset, comprising seven broadest categories. In both dendograms, the three concentric rings represent the ground-truth broadest categories (inner ring), the bottom-up predictions (middle ring), and the top-down predictions (outer ring). (C) Hierarchical clusterings produced by the SCUT algorithm as in A, where the three rings correspond to 20 categories (inner ring), 16 broader themes (middle ring) and 7 broadest themes (outer ring). The discussion topics considered by the analysis are: computer-related topics (comp), scientific subjects (sci) recreational activities (e.g. games and hobbies; rec), socializing and social issues (soc), contentious issues such as religion and politics (talk), anything which does not fit in the other hierarchies (misc), alternative discussion (created after the ones above; alt).

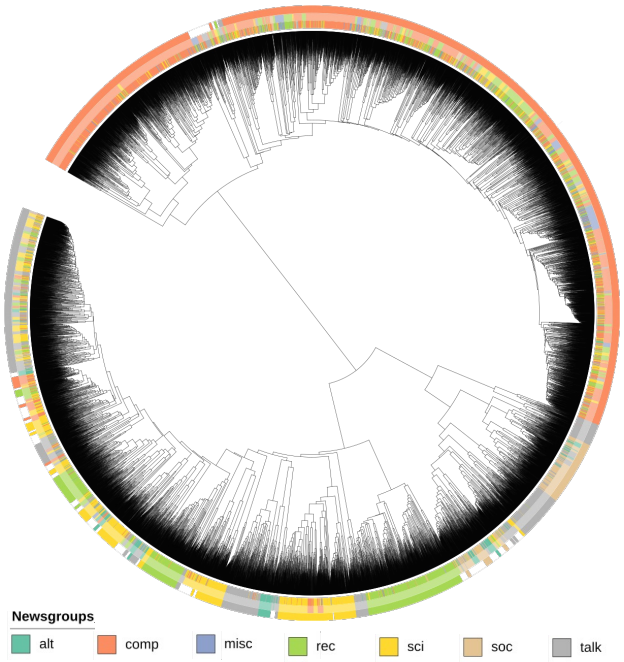


Figure 6: **Agglomerative hierarchical clustering of the 20Newsgroups dataset using Ward's metrics.** The three outer circles are colored with labels described in the inset legend. See Legend in Figure 5.

For this dataset, predictions were made on the seven broadest Usenet categories, corresponding to the original seven top-level hierarchies (excluding `news.`) along with the `alt.` hierarchy (see legend in Figure 5AB). Zha's algorithm clearly struggles with this task: using the TD annotation method, it fails to assign a single document to the `misc`, `alt`, or `soc` categories (Figure 5B). While SCUT also fails to predict any documents as `misc` (Figure 5A), it achieves significantly better performance overall, both in TD and BU scores (Table 1).

Examining the topology of the SCUT dendrogram in Figure 5A, we observe that the `sci` and `talk` categories appear globally dispersed across the tree. However, within local regions, there is minimal mixing with other categories, suggesting that SCUT captures coherent substructures despite global overlap.

As shown in Figure 5C, many of SCUT's misclassifications originate from a large `sci` (science) subtree embedded within a broader `comp` (computers) branch. A substantial number of the documents in this `sci` subtree belong to the `sci.crypt` (cryptography) and `sci.electronics` newsgroups. These misclassifications are intuitively reasonable, as both topics are closely related to computing and often overlap in subject matter. More generally, the topology of the SCUT-generated hierarchy appears semantically meaningful. For example, the `hockey` and `baseball` subtrees are positioned adjacent to one another, forming a coherent "sports" branch. Likewise, within the larger `comp` cluster, one subtree contains almost exclusively documents from `comp.windows.x`, and it is located next to a subtree corresponding to the `comp.os.ms-windows.misc` newsgroup—another thematically consistent placement.

On this dataset, we also evaluated the clustering obtained using Ward's method, one of the most commonly used hierarchical clustering algorithms. While Ward's method performs better than Zha's method, it still underperforms compared to SCUT (Table 1, Figure 6). This performance gap may be attributed to its reliance on Euclidean distance, which is not well-suited for the type of data used here, where the goal is to capture "similarity" between terms and documents. In this setting, most terms are absent from most documents, yielding a TF-IDF score of zero. In Euclidean space, however, two documents that both lack a term are treated as equally similar as two documents that share a highly specific term— and the similarity may even be exaggerated when many zero entries are present. Furthermore, Ward's method assigns equal

weight to all terms, regardless of whether they are highly domain-specific or very generic, limiting its ability to capture meaningful structures in the data.

We also examined the interpretability of the clusters built using SCUT. For example, in Figure 5.C, the "sport" cluster, located between the 10 and 11 o'clock sector of the inner ring, naturally splits into subclusters corresponding to hockey and baseball topics.

Baseball	**Score**	**Hockey**	**Score**
duke	0.1052	buffalo	-0.0861
fls	0.0958	leafs	-0.0692
econ	0.0906	hockey	-0.0691
mattingly	0.0739	hammerl	-0.0659
adobe	0.0728	espn	-0.0626
gant	0.0693	ca	-0.0615
bonds	0.0671	wings	-0.0614
sherri	0.0651	pens	-0.0583
snichols	0.0640	detroit	-0.0566
braves	0.0638	nhl	-0.0563

Table 2: Top 10 baseball and hockey terms with their relevance scores, positive score indicates closeness to baseball, and negative to hockey.

We extracted the 10 most influential terms for both subclusters (Table 2). For the "hockey" subcluster, several terms correspond to the email addresses of the most prominent users (e.g. "hammerl", "ca"). Most of the remaining terms are strongly related to the domain of hockey, including the term "hockey" itself, as well as team-related terms such as "buffalo" (Buffalo Sabres), "leafs" (Toronto Maple Leafs), "detroit" and "wings" (Detroit Red Wings), and "pens" (Pittsburgh Penguins). Additional terms include general hockey-related vocabulary, such as "nhl" (National Hockey League) and "espn", a sports network that covers both hockey and baseball. Notably, "espn" appeared almost ten times more frequently in the hockey newsgroup than in the baseball one.

For the "baseball" subcluster, several terms again reflect prominent users (e.g., "duke", "fls", "econ", "adobe", "sherri", "snichols"). Interestingly "sherri" and "snichols" refer to Sherri Nichols, a well-known baseball statistician and active contributor. Domain-specific terms include names of players and teams, such as "mattingly" (Don Mattingly), "gant" (Ron Gant), "bonds" (Barry Bonds), and "braves" (Atlanta Braves).

Overall, for both clusters, the most relevant terms are closely aligned with the corresponding topics, while a smaller set reflects highly active users in each group. It is also noteworthy that "ca" (the country code top-level domain for Canada) appears among the top terms for hockey, consistent with hockey's prominent status as Canada's national sport. These results demonstrate that SCUT effectively captures the underlying semantic structure across diverse topics.

3.4 Analysis on the Goodreads dataset

The Goodreads dataset Wan & McAuley (2018) is a large-scale collaborative filtering dataset comprising interactions between approximately 900,000 users and 2.4 million books. Each interaction corresponds to a user-assigned rating, taking integer values from 1 to 5, with 5 indicating the highest rating. In total, the dataset contains around 100 million such ratings.

Given the extreme sparsity of the full interaction matrix, we applied core filtering to obtain a denser submatrix comprising 9.7 million interactions between 28,000 users and 13,000 books. SCUT was then applied to this filtered matrix to construct hierarchical trees for both users and books. These trees were subsequently used to reorder the interaction matrix, thereby highlighting interaction patterns discovered by SCUT.

Book leaves in the resulting hierarchy were annotated using the genre metadata provided with the dataset Wan & McAuley (2018). These annotations were obtained through keyword matching in users' bookshelves (user-defined lists of books), as described in the original dataset documentation.

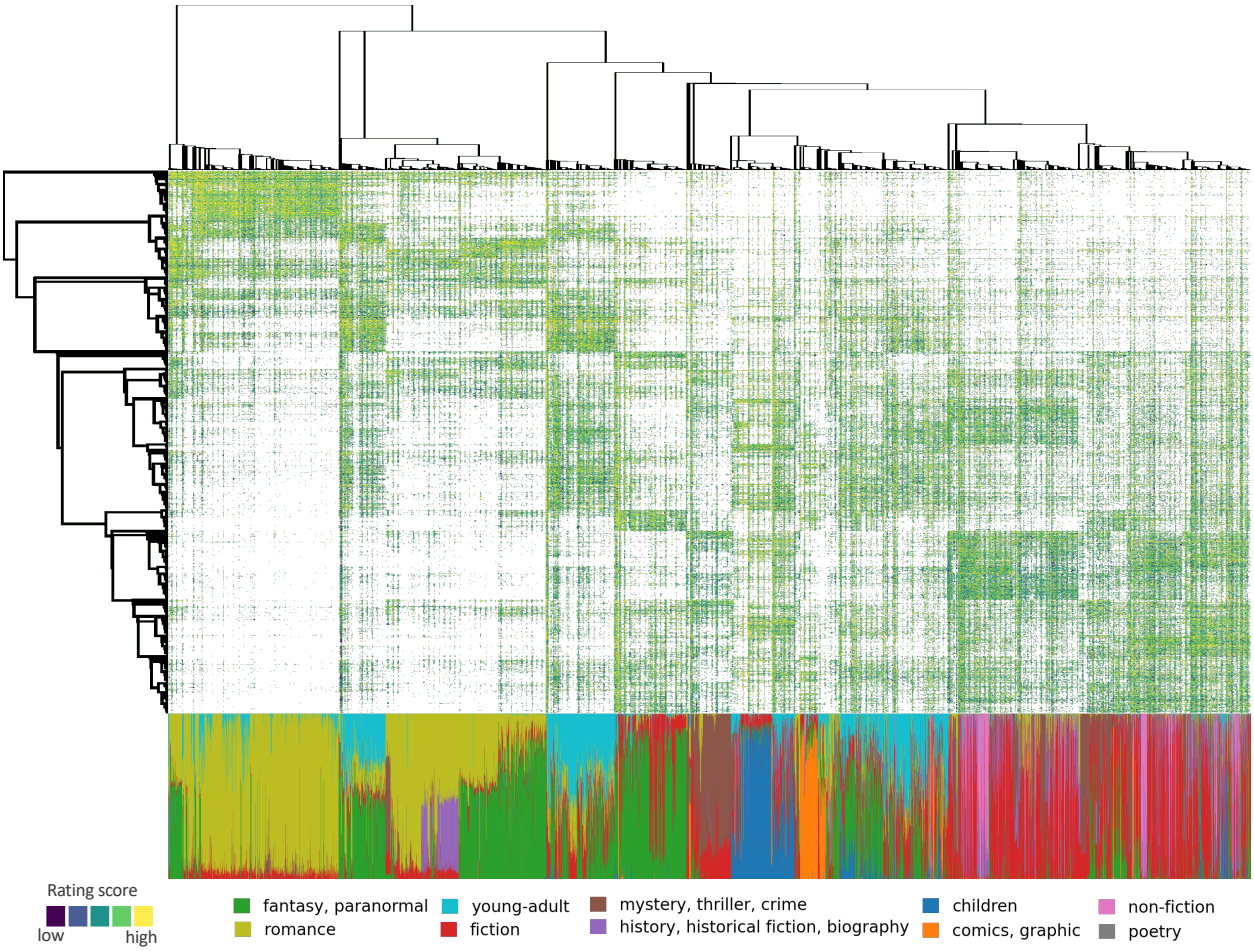


Figure 7: **Interaction patterns highlighted by SCUT.** The Goodreads interaction matrix (rows represent users, columns represent books) is ordered according to the SCUT trees. Each dot corresponds to a rating, colored using the viridis colormap (bright yellow = highest rating, dark blue = lowest rating). White cells indicate missing interactions. Books are labeled using user's reported tags: for example, if a book is tagged 20 times as 'children' and 10 times as 'fiction', its label bar is 2/3 blue (children) and 1/3 red (fiction). Tree branches are sorted by the number of leaves—at each node, the left child has fewer leaves than the right—to improve visual clarity. Because the original matrix ($13k \times 28k$) is too large to render directly, the figure uses a scatter-based representation of its nonzero entries. Note that overplotting may make the matrix appear denser than it truly is.

SCUT effectively extracts meaningful book genres using only the user-book interaction data. As illustrated in Figure 7, the book tree reveals well-defined clusters corresponding to distinct genre groupings. From left to right in the figure, the clusters include: a group dominated by romance titles; a smaller cluster combining fantasy/paranormal and young-adult books; a mixed cluster of romance works overlapping with other genres; a predominantly young-adult cluster with fewer fantasy/paranormal annotations than the preceding one; and a cluster composed mainly of fantasy/paranormal and fiction titles. This is followed by a smaller cluster rich in mystery/thriller/crime books with some fiction overlap. Next appears a large "young reader" cluster, which subdivides into three subgroups corresponding to children's books, comics, and a less clearly defined set. The remaining rightmost clusters encompass books spanning multiple genres—primarily fiction, non-fiction, and mystery/thriller/crime. Although this final cluster appears heterogeneous, the corresponding user tree reveals that the associated user groups exhibit broad yet internally consistent reading preferences. While external annotations for users were unavailable, qualitative inspection of the user clusters suggests that users within each cluster tend to share similar reading patterns and genre affinities.

3.5 Complexity analysis

Let the input matrix $W \in \mathbb{R}^{n \times m}$. First, note that computing $W_N = D_1^{-1/2} W D_2^{-1/2}$ can be done efficiently in $\mathcal{O}(mn)$ operations. To obtain a fast approximation of the truncated SVD of W_N, one may use the method proposed in Halko et al. (2011), which achieves a complexity of $\mathcal{O}(mnk + k^2(n+m))$ flops, where k is the number of singular values retained. For small k, this is commonly written as $\mathcal{O}(mnk)$. In our case, we set $k = 2$—since only the Fiedler vector is required— reducing the complexity to $\mathcal{O}(mn)$. Alternatively, the IRLBA method Baglama & Reichel (2005) provides another efficient approach, running in $\mathcal{O}(mnk)$ flops.

The threshold th is selected by evaluating the density function p at s evenly spaced points between the minimum and maximum of the Fiedler vector f_u. This step requires $\Theta(ns)$ operations using kernel density estimation. We treat s as a constant in the rest of the analysis. A more sophisticated approach to finding th, for example using algorithms from black-box optimization, could further reduce the overall computational time. In SCUT, in the worst case, each recursive step isolates a single data point, resulting in a time complexity of $\mathcal{O}(mn^2)$. In the best case, each recursive step divides the dataset into two balanced subclusters, leading to a complexity of $\Omega(mn \log n)$. This best-case and worst-case behavior is reminiscent of the Quicksort algorithm. However, unlike Quicksort, we cannot analytically derive an average-case complexity for SCUT, as the size of each recursive cut depends on the data distribution and is not independent of the input structure.

The SCUT method can be modified to ensure a time complexity of $\Theta(mn \log n)$ in both the best and worst cases. This is achieved by constraining the size of the cuts during the recursive partitioning process. Specifically, during the threshold selection step (line 10 in Algorithm 1), if the smaller of the two resulting clusters contain less than $\frac{n}{c}$ elements-where $2 \leq c < n$ is a hyperparameter-the threshold th is reselected such that the smaller cluster contains exactly $\lceil \frac{n}{c} \rceil$ elements. This reselection can be efficiently performed in linear time using the Introselect algorithm. With this modification, the worst case behavior is improved: at each recursive step, the algorithm guarantees that at least a $1/c$-fraction of the data is separated, ensuring that the recursion depth remains logarithmic. Consequently, the overall time complexity becomes $\mathcal{O}(mn \log n)$ in both the best and worst cases.

4 Limitations and possible improvements

Qualitatively, the main limitation of SCUT lies on its linear cuts, which restrict it to concave clusters. While it offers better time complexity than other hierarchical clustering methods, the current Python implementation is not well-optimized, resulting in runtimes about twice as slow as optimized C implementations Virtanen et al. (2020) of hierarchical clusters, on large datasets. Nonetheless, clustering tasks such as 20newsgroup were completed in about an hour. SCUT could benefit significantly from improved implementation, notably SVD computation Halko et al. (2011) can be run on GPU. Additionally, Algorithm 1 is easily parallelizable, as the computations in lines 14 and 15 are entirely independent.

5 Conclusion

We introduced SCUT, a novel hierarchical clustering approach based on spectral clustering. We demonstrated its effectiveness across three diverse datasets, showing that SCUT outperforms existing hierarchical clustering methods in capturing both global and local data structures. On the textual dataset, we further illustrated the interpretability of SCUT by analyzing the cluster splits: the most influential terms identified are strongly aligned with the underlying topics of each cluster. Finally, using a fourth dataset, we showed how SCUT can extract meaningful cluster labels in a multi-label setting.

In addition, we showed that SCUT offers a lower theoretical best-case time complexity compared to widely used agglomerative clustering methods, while maintaining the same worst-case complexity. We also proposed minor modifications that further reduce the worst-case complexity from quadratic to quasi-linear.

6 Data availability

SCUT code and the data to reproduce the analysis described in this article are available at: `https://anonymous.4open.science/r/SCUT-5B16/`.

References

James Baglama and Lothar Reichel. Augmented implicitly restarted lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing*, 27(1):19–42, 2005.

James Demmel. Lecture notes in applications of parallel computers, lecture 20, 1999. URL `https://people.eecs.berkeley.edu/~demmel/cs267/lecture20/lecture20.html`.

Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 269–274, 2001.

William E Donath and Alan J Hoffman. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. *IBM Technical Disclosure Bulletin*, 15(3):938–944, 1972.

Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2): 179–188, 1936.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Helmut Lütkepohl. *Handbook of matrices*. John Wiley & Sons, 1997.

Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Alex Pothen, Horst D Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM journal on matrix analysis and applications*, 11(3):430–452, 1990.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

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

Mengting Wan and Julian McAuley. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM conference on recommender systems*, pp. 86–94, 2018.

Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, pp. 25–32, 2001.

A Appendix

A.1 Proof of the convexity of SCUT clusters

First, let's show that $C = \{x \in \mathbb{R}^m | f(x) < 0\}$ where f is an affine function, is a convex subspace of $\mathbb{R}^m$.

Proof. The function f is affine so we can write for any $x \in \mathbb{R}^m$, $f(x) = w^\intercal x + b$, where $w \in \mathbb{R}^m$ is the weight vector and $b \in \mathbb{R}$ the bias.

Let x and y two points in C. For any $\lambda \in [0, 1]$, consider $z = \lambda x + (1 - \lambda)y$.

$$\begin{aligned} f(z) &= w^\intercal z + b \\ &= w^\intercal(\lambda x + (1-\lambda)y) + b \\ &= \lambda(w^\intercal x + b) + (1-\lambda)(w^\intercal y + b) \\ &= \lambda f(x) + (1-\lambda) f(y) \end{aligned}$$

Since $f(x) < 0$ and $f(y) < 0$, $f(z) < 0$. Hence $z \in C$, which proves C is convex. □

It can be similarly proven that $\{x \in \mathbb{R}^m | f(x) \geq 0\}$ is also convex.

Let's prove the convexity of all SCUT clusters. Given a non-leaf cluster t, we note its associated space S and $S.left$ ($S.right$) the space associated to $t.left$ ($t.right$).

We prove by induction the convexity of S for any t SCUT cluster.

Proof. **Basis step :** t is the root note, $S = \mathbb{R}^m$. By definition, S is convex.
Inductive step : Let t a non-leaf cluster, we note clf the affine function that splits t, assume S is convex.

$$\begin{aligned} S.left &= S \cap \{x \in \mathbb{R}^m | clf(x) < 0\} \\ S.right &= S \cap \{x \in \mathbb{R}^m | clf(x) \geq 0\} \end{aligned}$$

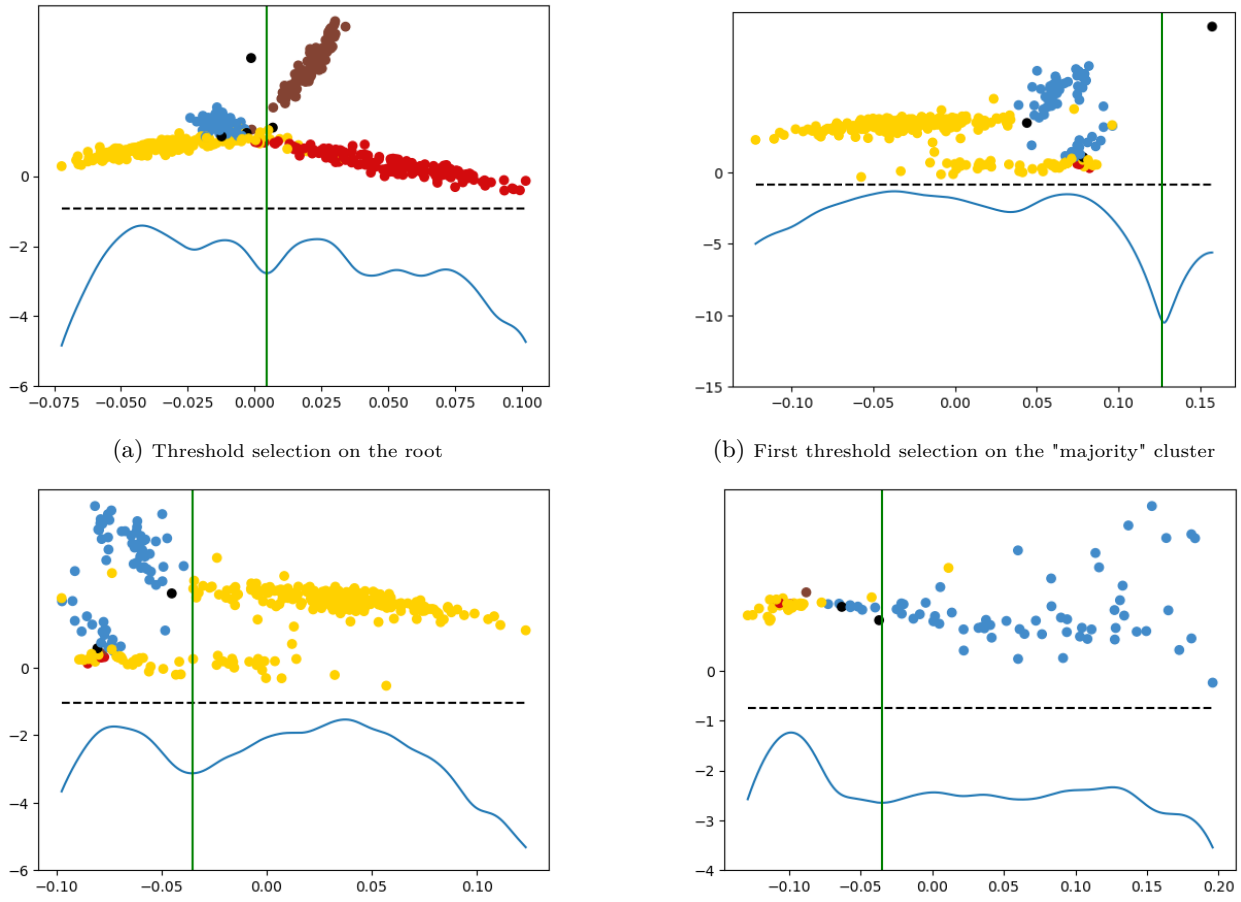
$S.left$ and $S.right$ are convex as the intersection of 2 convex sets. □

Note that the clusters are convex on the sets containing the observations after applying the scaling $T : \mathbb{R}^m \to \mathbb{R}^m$, which associate x to $\frac{x}{\sqrt{\sum_i x_i}}$.

A.2 Density analysis of French National Assembly clusters

Using the same method used to build Figure 2, we can analyze quantitatively the quality of the SCUT clusters built on the French National Assembly dataset. Figure 8 shows the first few "majority" clusters. The first split (Figure 8a) is quite hard to make as there are many deputies, from all part of the political spectrum, that did not vote much and ends up close to 0 in the f_u projection, anyway, SCUT still manages to make coherent clusters that split cleanly the data between "majority" and "opposition". Interestingly, while $U[2]$ encodes for "majority/opposition", $U[3]$ seems to encode for the "left/right" political spectrum. The next split (Figure 8b) singles out a single deputy that while non-attached was supported by the far-right. The last 2 splits (Figure 8cd) try to separate the "presidential majority" from "The Republicans".

The same analysis can be done on the opposition cluster (Figure 9) where the 2 clusters "NUPES" and "National Rally" are easily distinguishable.



(a) Threshold selection on the root

(b) First threshold selection on the "majority" cluster

(c) Second threshold selection on the "majority" cluster

(d) Third threshold selection on the "majority" cluster

Figure 8: Threshold selection on the "majority" cluters of the French National Assembly dataset: x-axis : one-dimensional space containing $f_U = U[2]$. y-axis below dashed line : estimated log-density. above dashed line : scatter plot of $(U[2], U[3])$, $U[3]$ is used for visualization purposes only, colors correspond political alignment, "Non-attached" are in black. Vertical green line corresponds to the selected threshold.

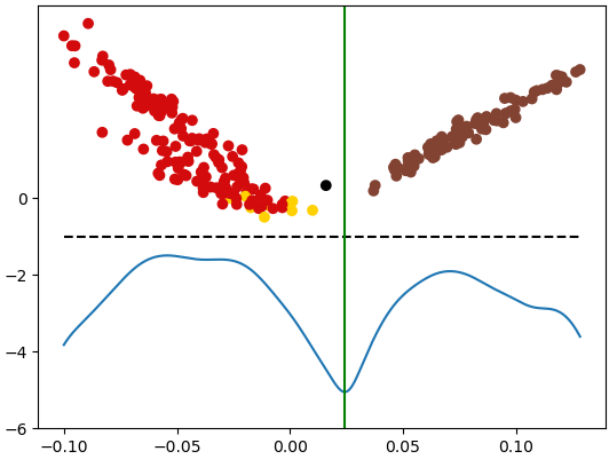


Figure 9: Threshold selection on the "opposition" cluster of the French National Assembly dataset : x-axis : one-dimensional space containing $f_U = U[2]$. y-axis below dashed line : estimated log-density. above dashed line : scatter plot of $(U[2], U[3])$, $U[3]$ is used for visualization purposes only, colors correspond political alignment, "Non-attached" are in black. Vertical green line corresponds to the selected threshold.