Information Sciences 689 (2025) 121453

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Interpretable sequence clustering

Junjie Dong [a], Xinyi Yang [a], Mudi Jiang [a], Lianyu Hu [a], Zengyou He [a,b,*]

[a] School of Software, Dalian University of Technology, Dalian, 116620, China
[b] Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian, 116024, China

ARTICLE INFO

Keywords:
Sequence clustering
Pattern tree
Interpretable clustering
Sequential pattern mining

ABSTRACT

Categorical sequence clustering is vital across various domains; however, the interpretability of cluster assignments presents considerable challenges. Sequences inherently lack explicit features, and existing sequence clustering algorithms heavily rely on complex representations, which complicates the explanation of their outcomes. To address this issue, we propose a method called Interpretable Sequence Clustering Tree (ISCT), which combines sequential patterns with a concise and interpretable tree structure. ISCT leverages $k-1$ patterns to generate k leaf nodes, corresponding to k clusters, which provides an intuitive explanation on how each cluster is formed. More precisely, ISCT first projects sequences into random subspaces and then utilizes the k-means algorithm to obtain high-quality initial cluster assignments. Subsequently, it constructs a pattern-based decision tree using a boosting strategy in which sequences are re-projected and re-clustered at each node before mining the top-1 discriminative splitting pattern. Experimental results on 14 real-world data sets demonstrate that our proposed method provides an interpretable tree structure while delivering fast and accurate cluster assignments.

1. Introduction

A categorical sequence is a series of symbols arranged in a specific order, where the symbols belong to a finite set of items. Categorical sequence clustering is an unsupervised machine-learning problem that aims to partition unlabeled sequences into homogeneous groups. This technique is widely used in several fields, including bioinformatics [1], natural language processing [2], and financial data analysis [3]. For instance, in bioinformatics, sequence clustering can aid in identifying gene families and determining the evolutionary relationships between different species.

Existing efforts mainly focus on obtaining accurate clustering results rather than interpreting identified clusters. For scientific purposes, it is crucial to understand the reasons why certain genes are grouped into particular familiar and what drives the model to make these divisions. However, the current sequence clustering methods [4–9] simply output the clusters of sequences without offering a succinct explanation for the cluster assignments, as detailed below.

A significant amount of algorithms for clustering sequences have been proposed during the last decades. These methods can be roughly divided into feature/pattern-based methods, partitional methods, hierarchical methods, and model-based methods. However, the cluster assignments in these methods are typically hard to explain in a concise manner that is comprehensible to humans.

* Corresponding author at: School of Software, Dalian University of Technology, Dalian, 116620, China.
E-mail addresses: jd445@qq.com (J. Dong), yxinyi1209@163.com (X. Yang), 792145962@qq.com (M. Jiang), hly4ml@gmail.com (L. Hu), zyhe@dlut.edu.cn (Z. He).

https://doi.org/10.1016/j.ins.2024.121453
Received 22 April 2024; Received in revised form 29 August 2024; Accepted 7 September 2024

Available online 12 September 2024

Firstly, feature/pattern-based methods [6,4] transform the sequence into a feature vector based on extracted patterns and then use vectorial data clustering methods like k-means for clustering. However, the process of transfer the sequences into feature vectors diminishes the interpretability. Although methods like k-means can be considered as prototype-based interpretable methods, their interpretability is significantly lower compared to rule-based and tree-based methods. In k-means, each input vector is assigned to a cluster based on its distance to the center of each cluster. The employment of centroids and the propensity to create spherical clusters assist in visualization. However, as each feature influences the calculation of distance, it becomes difficult to identify a clear rationale behind the decision-making process.

Partitional methods [9] divide sequences into several clusters by optimizing an objective function, while hierarchical methods [5] use distance measures to obtain a hierarchical structure. However, these methods do not produce a human-friendly structure that directly reveals the decision process. Model-based methods [7] assume that sequences from different clusters have different distributions, and then assign sequences to clusters accordingly. However, for those unfamiliar with statistical techniques, this approach makes it even more difficult to understand why certain sequences are allocated to the same cluster.

In recent years, there has been an increasing research interest in interpretable clustering methods for non-sequential data [10–19]. These methods can generally be classified into two main categories: *pre*-model and *post*-model methods. The *post*-model approach concentrates on providing the rationale for the decisions made by the black box model, that is, explaining the reasons for the model's data clustering. Conversely, the *pre*-model approach focuses on creating a succinct model that humans can easily comprehend or perceive, with a straightforward and intuitive structure.

Various conceptual descriptions have been provided to define interpretable clustering models. Generally, from the perspective of clustering results, interpretable clustering aims to characterize each cluster as precisely and distinctively as possible [16,13]. From a human comprehension standpoint, interpretable models emphasize on making the clustering results reliable and easy to debug, which can guide human decision-making and establish trust between humans and machines [20]. Therefore, the explanation of clusters should be human-readable and easy to understand.

There are three common forms of interpretable clustering models: decision trees, rules, and rectangles [20]. Evaluating these different methods using a uniform set of metrics can be challenging. Nevertheless, to address the complexity of decision-making processes, interpretability can be broadly assessed through model complexity. For instance, with decision trees, we consider the size of the tree and the complexity of the split nodes. For rule-based methods, we evaluate the number of rules and the length of the predicates within each rule.

The threshold tree [14] is an interpretable clustering method that divides the data set into several partitions with a binary tree structure. Each node in the tree iteratively partitions the data based on a threshold value derived from a specific feature. In this way, the data set can be divided into k partitions with $k-1$ nodes, which is highly straightforward and interpretable. One of the key reasons for this clarity is that the decision-tree clustering creates hyper-rectangular regions in space [11], making the threshold divisions on any dimension intuitive. In other words, the threshold tree provides a simple procedure using a few features to explain why any point belongs to its cluster [14]. Furthermore, the feature splits and decision paths of the tree help us to understand the distinctive characteristics of members within each leaf [12]. Unfortunately, the current state-of-the-art interpretable clustering algorithms are mainly focused on continuous vector data and cannot be directly applied to sequential data. This is mainly because the features of the sequential data are hidden and the potential dimensionality is very high.

Inspired by the threshold tree, we propose a new explainable clustering algorithm for sequential data based on patterns, which offers a concise way for humans to understand the partition. Interpretable Sequence Clustering Tree (ISCT) utilizes a binary tree structure, constructed in a top-down manner. The splitting criterion is based on the presence of certain discriminative sequential patterns, which are selected based on both their discriminative power and their support. The main steps for constructing an unsupervised pattern tree are as follows:

- Initial Cluster Assignment: We first use fast random projection to transfer the sequences into feature vectors and then adopt k-means to obtain initial cluster assignments.
- Discriminative Sequential Pattern Mining: We first mine the frequent sequential patterns and then identify one pattern that has the best discriminative ability with respect to the cluster label based on relative risk.
- Cluster Formation: We construct a refined cluster using sequences that contain the discriminative pattern chosen in the previous step as subsequences.
- Iterative Process: Repeat the above process for the remaining sequences until $k-1$ patterns have been reported, where k is the number of user-specified number of clusters.

Through the iterative process, we can obtain a hierarchical unsupervised pattern tree with k leaf nodes.
The contribution of this paper is as follows:

- A tree-based interpretable sequence clustering method named Interpretable Sequence Clustering Tree is proposed in this paper. To our knowledge, this is the first interpretable clustering method for sequential data.
- ISCT algorithm constructs a tree in a boosting manner by utilizing a small number of discriminative patterns, resulting in a clear and interpretable structure. This provides a concise and understandable way for humans to interpret the clustering results.
- The experimental results on 14 real-world data sets show that ISCT can achieve comparable performance to state-of-the-art methods while providing a clear and interpretable structure.

2. Related work

This section discusses previous works closely related to our approach. A brief overview of recent developments in interpretable clustering is provided in Section 2.1, and an outline of the newest findings in sequence clustering is detailed in Section 2.2. Additionally, Section 2.3 provides an introduction to some typical methods for discriminative sequential pattern mining.

2.1. Interpretable clustering

2.1.1. Fuzzy rule-based approach

Fuzzy rule-based methods aim to extract fuzzy IF-THEN rules from data, which take the form:

$$\text{Rule } R_i\text{: IF } x_1 \text{ is } \phi_{11} \text{ and } \ldots \text{ and } x_n \text{ is } \phi_{jj} \text{ THEN cluster } c_i.$$

Here, x_n represents the n-th feature of a sample x, and ϕ_{jj} is the fuzzy number that must be satisfied for that attribute. By discovering different rules that are satisfied by one or more samples within a cluster, the decision process of the clustering algorithm can be explicitly characterized with high interpretability. Fuzzy rule-based methods can be broadly classified into the adaptive partition and fixed grid partition based on whether the membership function is related to the data [20].

In the adaptive partition approach, the information of data set is considered throughout the partitioning process. For example, FRCGC [21] is a rapid fuzzy rule clustering method based on granular computing. It provides descriptions for all clusters by using exemplar descriptions selected from sample descriptions to guide data granulation, resulting in each cluster being depicted by a single fuzzy rule. For fixed grid partition, the information of the data set is not considered in the fuzzy partition process. Commonly used membership functions in this context are the trigonometric and trapezoidal ones [22,23].

2.1.2. Polytope-based approach

The polytope-based approach uses hyperplanes as the boundaries of the clusters, which are combined to form a polytope. The description of the hyperplanes is then used to construct rules that enable interpretable clustering. This approach is similar to Support Vector Machines (SVM) since they both use hyperplanes as the decision boundary. However, unlike SVM, the polytope-based clustering method does not rely on labeled data, making the task of finding the hyperplanes for the clustering boundary more challenging.

From this perspective, many different algorithms have been proposed. For example, in [24], hyperrectangle boxes are used as the clustering boundary. Initially, the hyperrectangles may overlap, and to obtain the final rectangles, a soft “tail” is introduced to calculate the membership of samples to different rectangles. The algorithm then calculates the final upper and lower boundaries using an alternating minimization strategy. The Discriminative Rectangle Mixture Model (DReaM) [25], on the other hand, is a more flexible approach as it allows for the specification of two types of features: rule-generating features and cluster-preserving features. Specifically, DReaM constructs rules for clustering using only the rule-generating features, while finding the clustering structure using the cluster-preserving features. Unlike the rectangle decision boundaries used in the previous two methods, [16] constructs polytopes around each cluster and translates the decision boundaries into rules to obtain interpretable clustering results.

2.1.3. Threshold tree-based approach

Tree-based methods are widely used in the fields of classification and clustering due to their simplicity and intuitive interpretability. In the field of clustering, threshold tree methods can be divided into two categories based on whether they require the label information of other clustering algorithms.

Four different attribute selection metrics and two data partitioning algorithms are proposed in [10], which introduces a hierarchical clustering method for generating interpretable tree structures. In contrast to the direct construction of the tree as done previously, CUBT [15] incorporates random auxiliary samples as a second class in the sample space to acquire the necessary labels for calculating information gain. To enhance the algorithm’s performance, the decision process is optimized by adjusting the splitting criterion. FDTC algorithm [19] integrates fuzzy rules with decision tree structures, employing these rules as the criteria for node splitting within the trees.

The tree construction process can also leverage pseudo-labels generated by conventional clustering methods [11,14,17]. These studies tackled this problem from a theoretical standpoint and evaluated cluster quality using the objective functions of k-means and k-medians. Especially, it has been demonstrated that a single threshold cut can achieve a constant factor approximation for both means and medians. Subsequent research efforts have focused on further improving the approximation ratio. Our method is similar to these studies in the sense that we also require the clustering result of k-means as the input for tree construction and the number of clusters is fixed in advance. In contrast to the aforementioned methods that depend on numerical values for splitting criteria, our method employs sequential patterns. This offers a more straightforward and intuitive comprehension of sequential data.

2.2. Categorical sequence clustering

2.2.1. Feature/pattern-based methods

There are primarily two different strategies for the feature or pattern-based methods. The first approach involves transforming sequences into feature vectors and then applying existing clustering methods such as k-means [26,6] or DBSCAN [27] to obtain the final partition results. The differences between these methods mainly arise from the process of converting sequences into feature

vectors. Popular approaches include using sequential patterns [6,28], k-mers [29], or embedding techniques [4,27]. Another strategy involves initially constructing a clustering result based on the discovered sequential patterns and subsequently expanding and optimizing these clusters to achieve the final refined clustering results [29]. Some clustering methods [30] also attempt to interpret the obtained clusters through rule extraction, which can indeed help improve our understanding of the clusters. However, these methods do not provide a clear and intuitive clustering process.

2.2.2. Partitional methods

Partitional methods [8,31] directly divide a set of sequences into multiple clusters. Instead of transforming the sequence vectors and applying traditional partitional clustering algorithms for vectorial data, these methods aim to obtain partition results directly without transforming sequences into feature vectors.

Note that one may argue that some methods can be both partitional method and feature-based method. This is because feature-based methods can use various clustering techniques such as those partitional methods like k-means after feature extraction. In our categorization, only those methods directly divide sequences into clusters without explicit feature extraction are considered to be partitional method. Based on this criteria, there will be no overlap between these two categories.

2.2.3. Model-based methods

Model-based methods are based on the assumption that sequences originating from distinct clusters follow separate probability distributions. Existing model-based approaches utilize either Markov models, constructing Markov chains that consider the occurrence of a sequence category given its preceding subsequences [7], or the Hidden Markov Model (HMM), which introduces hidden states to the sequences [32].

2.2.4. Hierarchical methods

Hierarchical methods primarily rely on the distances between all sequences. Typically, pairwise similarity matrices need to be computed as a first step. Subsequently, either agglomerative or divisive approaches are employed to progressively build the hierarchical structure. The main distinction between these methods lies in their utilization of different distance functions to obtain the similarity matrix. Commonly used distance functions include Kullback-Leibler distance [33], edit distance [5], and Jaccard similarity based on subsequences [34].

It is worth mentioning that this method already exhibits a certain level of interpretability, as the partition can be explained based on the distances between samples. However, this form of explanation may not be intuitive enough. The computation of distances is not straightforward for humans, as it involves complex combinations of individual elements within the sequences.

2.3. Discriminative sequential pattern mining

Discriminative sequential pattern mining is closely related to sequence classification [35]. Its objective is to discover patterns that exhibit strong correlations with specific labels. These mined patterns are characterized by their frequent occurrence in sequences with one label and their absence as subsequences in sequences with another label. The identification of discriminative patterns for sequence classification has been extensively investigated, with each study focusing on different pattern evaluation strategies [36,37]. The cSPADE method was enhanced by incorporating an interestingness measure that considers both the support and the window (cohesion) in which the constraint items occur [37]. Furthermore, BIDE was extended to BIDE-D(C) by incorporating information gain, providing a direct pattern mining approach to sequence classification [36].

3. Problem statements

3.1. Notation

Given a set of items $\mathcal{I} = \{e_1, e_2, \ldots, e_m\}$ with m distinct items, a discrete sequence $s = \langle \sigma_1, \sigma_2, \ldots, \sigma_l \rangle$ with length $|s| = l$ is an ordered list over $\mathcal{I}$ if each $\sigma_i \in \mathcal{I}$. A subsequence t of s is defined as a sequence that can be obtained by removing some elements from s while preserving the relative order of the remaining elements. Formally, t is a subsequence of s, denoted as $t \subseteq s$, if there exists a strictly increasing sequence of indices $1 \le i_1 < i_2 < \cdots < i_k \le l$ such that $t = \langle \sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k} \rangle$. We use a subsequence as the sequential pattern p.

3.2. Support

A sequence database $\mathcal{D}$ is a list of sequences, where the number of sequences in the database is denoted as $|\mathcal{D}|$. The number of occurrences of a given sequential pattern p in the database is denoted as $Occ(p, \mathcal{D})$. The support of a pattern p in $\mathcal{D}$ is defined as:

$$Supp(p, \mathcal{D}) = \frac{Occ(p, \mathcal{D})}{|\mathcal{D}|}. \tag{1}$$

Given a threshold α_{Supp}, a sequential pattern p can be regarded as a frequent sequential pattern if $Supp(p, \mathcal{D}) \ge \alpha_{Supp}$.

However, directly specifying the support threshold can be challenging in many cases. As a practical alternative, it is common to set a parameter k to obtain the top-k patterns with sufficiently high support.

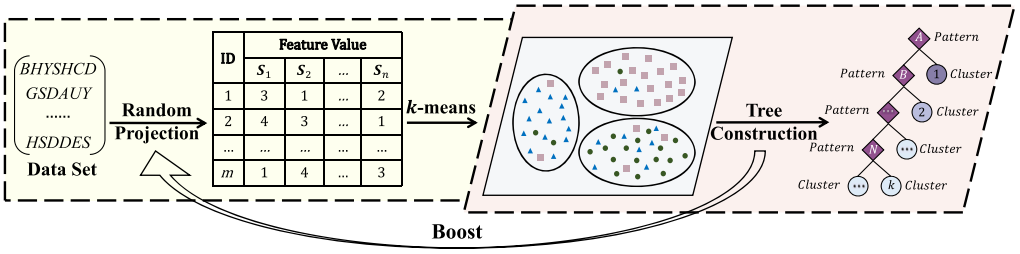

Fig. 1. The workflow of the Interpretable Sequence Clustering Tree. It mainly consists of two parts: (1) Clustering via random projection and (2) Tree construction.

Pattern	Sequence	Match	Value
$< bd >$	$< abddc >$	<*bd**>	2
$< cec >$	$< acaeadcd >$	$<* c * e ** c *>$	3
$< abbaa >$	$< acbbccac >$	$< a * bb ** a *>$	4

Fig. 2. Example of calculating the length of Longest Common Subsequence.

3.3. Problem formulation

For a given database $D = \{s_1, s_2, \ldots, s_n\}$ and an integer k, the sequence clustering method partitions the database D into k subsets $C = \{c_1, c_2, \ldots, c_k\}$. The objective is to construct an interpretable sequence clustering tree characterized by its compact and concise structure, facilitating an intuitive and thorough comprehension for human users. In detail, the clustering tree should exhibit the following characteristics:

- *Shallow*: the interpretability of the decision tree is mainly determined by its size. Similar to [14], our objective is to obtain a tree with exactly k leaves, one for each cluster, which means that the depth and the total number of non-leaf nodes is at most $k - 1$.
- *Concise*: the splitting criteria for trees should be concise, particularly in the case of sequential data where we aim to avoid incorporating distance functions or similarity matrices in the splitting process.
- *Accurate*: the interpretable clustering tree should produce accurate result that is comparable to those of non-interpretable clustering algorithms.

4. Interpretable sequence clustering tree

4.1. An overview of the algorithm

In order to construct an Interpretable Sequence Clustering Tree (ISCT) for a given sequence database, we can utilize cluster membership from another clustering algorithm as a pseudo label and apply various tree-based classification methods. However, using these tree-based methods directly often results in complex and non-interpretable tree structures, as these methods tend to generate large and complex trees. Even with control over splitting procedures or the use of pruning strategies, creating an intuitive and concise decision tree remains a challenging task in practice.

The most intuitive approach to separate two clusters in a sequence database is by utilizing the presence or absence of a specific sequential pattern [38]. Our objective is to construct a decision tree with k leaf nodes and $k - 1$ patterns, where each leaf node comprises sequences belonging to the same cluster. Using $k - 1$ patterns to describe k clusters is extremely compact in terms of information content, and the splitting criterion for nodes is the existence of patterns rather than specific values, which is human-understandable for sequential data. The workflow of our method is illustrated in Fig. 1. Our method consists of two main components. The first component of our method involves rapid vectorization by randomly projecting sequences into subspaces, facilitating the efficient utilization of the k-means algorithm for obtaining initial partition results. This component is elaborated on in Section 4.2. The second component centers around the tree construction algorithm, as explained in Section 4.3. Here, we present a new measure derived from relative risk for pinpointing discriminative patterns and outline the procedure for building the tree based on the top-1 discriminative pattern obtained.

4.2. Clustering via random projections

We propose a method that employs randomly generated sequential patterns for the transformation of sequences into vector representations. That is, by calculating the distance between each random pattern and the target sequence, each sequence in the database is projected into the corresponding subspace.

We employ the longest common subsequence (LCS) to compute the similarity between two sequences (the random sequential pattern and each sequence in the database). Fig. 2 provides several examples for calculating the length of LCS between patterns and

Algorithm 1 Random Projection Clustering.

Input: Sequence database $D = \{s_1, s_2, \ldots, s_n\}$, number of clusters k, pattern number $numP$, maximal random pattern length $maxS$.
Output: Clusters $C = \{c_1, c_2, \ldots c_k\}$

1: $\mathcal{I} \leftarrow readItem(D)$
2: $Pset \leftarrow \emptyset$
3: **for** $i \in [1, numP]$ **do**
4: $\quad l = random(1, maxS)$
5: $\quad p = randomSubsequence(l, \mathcal{I})$
6: $\quad Pset.add(p)$
7: **end for**
8: $feature \leftarrow lcsTransform(D, Pset)$
9: $feature \leftarrow PCA(feature, k)$
10: $C \leftarrow kmeans(feature, k)$
11: **return** C

sequences. To mitigate the impact of redundant patterns and enhance stability, we further employ PCA (Principal Component Analysis) for dimensionality reduction. This reduces the dimensionality of the feature vectors to a lower-dimensional space. Subsequently, we apply the k-means algorithm to cluster the transformed vectorial data.

The overall procedure is outlined in Algorithm 1. The main steps are as follows: First, we randomly generate a number of subsequences from the sequence database D (lines 3 to 6). Then, we transform the sequences into feature vectors using the LCS similarity (line 8). Subsequently, PCA is used for conducting the dimension reduction (line 9). Finally, we cluster the feature vectors using k-means to obtain the clustering result (line 10).

The time complexity of calculating the distance between a sequence s_n and a random sequential pattern p through dynamic programming is $O(|s_n| \cdot |p|)$. Consequently, the computational overhead for projecting all $|D|$ sequences over $numP$ random patterns is $O(|D| \cdot numP \cdot maxl \cdot maxS)$, where $maxl$ represents the maximum length of a sequence in the database. Additionally, the time complexity for executing the subsequent PCA algorithm across $numP$ features is $O(|D|^2 \cdot numP + numP^3)$ [39]. In the case of the final k-means algorithm, the complexity is $O(|D| \cdot t \cdot k \cdot numP))$, where k is the number of specified cluster number and t is the number of iterations after its convergence.

4.3. Tree construction

We can utilize the existence of a specific pattern as the splitting criterion for building the tree. Specifically, given initial clusters $C = \{c_1, c_2, \ldots, c_k\}$, we need to find representative patterns for each of the $k-1$ clusters in order to construct the tree. During the construction of the clustering tree, the database D is firstly divided into two sets D_p and $\overline{D_p}$ based on the existence of a sequential pattern p. The problem is then converted into finding the top-1 discriminative pattern p for tree splitting.

However, directly using traditional splitting criteria such as the Gini impurity may not perform well when dealing data sets with a large number of clusters. In this paper, we present a new split measure based on relative risk for identifying discriminative patterns.

4.3.1. Splitting criterion

The relative risk (RR) is a commonly used measure for evaluating the discriminative power of sequential patterns. The relative risk of pattern p is calculated as the ratio of the support in positive class D^+ to the support in negative class D^-:

$$RR = \frac{Supp(p, D^+)}{Supp(p, D^-)}, \tag{2}$$

where the positive class D^+ consists of the sequences in the target class, and D^- consists of all the remaining sequences from other classes. In addition to relative risk, the following internal similarity measure is employed to evaluate the goodness of a pattern p:

$$SIM = \frac{|p| \times |D^+|}{\sum_{j=1}^{|D^+|} |s_i|}, \tag{3}$$

where $|p|$ is the length of pattern p, and $|D^+|$ is the number of sequences in the positive class D^+. That is, we prefer longer patterns if they have the same discriminative capability.

However, the concept of relative risk can only be applied as a measure when there are two classes involved. For output k partitions, we need to extend the concept of relative risk to accommodate multiple classes. To address this requirement, D^+ is defined as:

$$D^+ = \{c_i \mid \arg\max_i Supp\left(p, c_i\right)\}. \tag{4}$$

We select the cluster with the highest support of pattern p as the positive class D^+ and obtain the negative class D^- using the one-vs-rest strategy, where $D^- = D \setminus D^+$. We try to identify a pattern that exhibits the highest discriminative power according to Equation (4). If multiple patterns have the same relative risk, we choose the pattern among them with the maximal SIM value according to Equation (3). The presence of this pattern serves as the basis for partitioning the data set.

Algorithm 2 ISCT Construction.

Input: Sequence database $D = \{s_1, s_2, \ldots, s_n\}$, number of clusters k, indicator of boosting $boost$, maximal pattern number $maxN$, minimum number of sequences $minS$.
Output: root of the pattern tree N

```
1: C = {c_1, c_2, ..., c_k} ← Algorithm 1(D, k)
2: N ← creatNode(D)
3: Built_Tree(N, D, C)
4: return N
1: function BUILT_TREE(N, D, C)
2:     if k < 2 or |D| ≤ min(K, minS) then
3:         return
4:     end if
5:     if boost = True then
6:         C ← Algorithm 1(D, k)
7:     end if
8:     candidatePatterns ← ∅
9:     for each cluster c_i ∈ C do
10:        freqPattern ← mineFreqPattern(c_i, maxN)
11:        candidatePatterns.add(freqPattern)
12:    end for
13:    p ← getTop1Pattern(candidatePatterns)
14:    if p = ∅ then
15:        return
16:    end if
17:    D_p ← {s ∈ D | p ⊆ s}
18:    D̄_p ← {s ∈ D | p ⊈ s}
19:    k ← k − 1
20:    N.left ← createNode(D̄_p)
21:    N.right ← createNode(D_p)
22:    Built_Tree(N.left, D̄_p, C)
23:    Built_Tree(N.right, D_p, C)
24: end function
```

4.3.2. Tree construction

The top-1 discriminative sequential pattern is used for splitting during tree construction, assigning sequences that contain this pattern to the same cluster. Based on this idea, we propose an algorithm for building an interpretable sequence clustering tree.

The construction process is described in Algorithm 2. The first step is to obtain k initial clusters based on Algorithm 1. After initializing the root node N, we invoke the $Built\_Tree$ to construct the pattern tree.

The tree construction follows a top-down approach with binary splits. In the first stage, the algorithm checks the stop criteria of the given node (line 2 to 4). The first criterion is $k < 2$, which determines whether the desired number of leaf nodes has been reached. The second criterion is $|D| \leq \min(k, minS)$, which determines whether to continue splitting based on the minimum number of sequences included.

After checking the stop criteria, a frequent sequential pattern mining algorithm is applied to extract frequent sequential patterns for each $c_i \in C$ (line 9 to 12). The top-1 discriminative pattern p is selected using different measures (line 13). If a sequence s in D contains pattern p, it is assigned to the right child node. On the other hand, if a sequence does not contain pattern p, it is assigned to the left child node and the algorithm continues building the tree by recursively calling the $Built\_Tree$ function (line 17 to 23). The process is completed once all tree nodes meet the stop criteria.

4.3.3. Boost construction

In our experiments, we have observed that this direct construction method often leads to inconsistencies between the reported clusters $\hat{C}$ and given partition C when dealing with data sets with more clusters. This inconsistency arises from the difference between the cluster $\hat{c_i}$ constructed using the splitting pattern and the expected cluster c_i given by Algorithm 1.

To address this problem, we propose a boosting strategy during the tree construction process. In other words, each C is re-generated using Algorithm 1 based on the current remaining cluster number k. Fig. 3 illustrates this problem and the boosting result.

Due to the limitations of random projection clustering, the resulting sequence clusters may not always be accurate. For instance, in Fig. 3, cluster 2 is not pure as it contains sequences from both cluster 1 and cluster 3. However, when we remove $\mathcal{D}_p$ from the database, some items that only appear in sequences of cluster 1 are also discarded. To utilize this information, we propose a boosting strategy to re-partition the database $\mathcal{D}$ by randomly re-projecting the sequences into different subspaces. This approach can partially calibrate our results and improve their consistency with the base sequence clustering method.

Example 1. Consider the toy example sequence database presented in Table 1, which consists of six sequences ($|D| = 6$) and a set of five items denoted as $\mathcal{I} = \{a, b, c, d, e\}$. The database is partitioned into three clusters. Fig. 4 illustrates the construction of a pattern tree for Example 1.

The first selected top-1 discriminative pattern is $p_1 = \langle bd \rangle$, which splits the data into two clusters. The first cluster $\hat{c_1}$ is obtained, containing sequences s_1 and s_2. The remaining sequences are then assigned to the left node. Since the stop criteria indicate that the

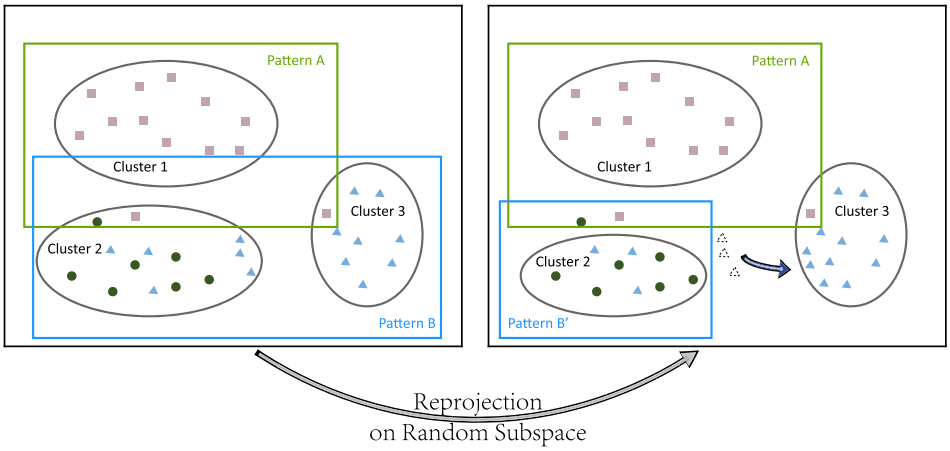

Fig. 3. The inconsistency between the reported clusters and given partitions and the boosting strategy.

Table 1
Toy sequence database and the clustering result.

ID	Sequence	Cluster	ID	Sequence	Cluster
s_1	$\langle abddc\rangle$	1	s_2	$\langle adbbded\rangle$	1
s_3	$\langle acdeeaa\rangle$	2	s_4	$\langle acaeadcd\rangle$	2
s_5	$\langle abbcaac\rangle$	3	s_6	$\langle acbbccaa\rangle$	3

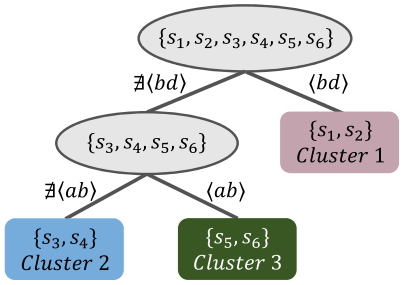

Fig. 4. The structure of the pattern tree: Each splitting node represents a sequential pattern.

left node should be further divided, the top-1 discriminative pattern $p_2 = \langle ab\rangle$ is selected. Eventually, we obtain a highly concise and interpretable clustering tree with three leaf nodes, represented by rectangles in the figure.

4.3.4. *Time complexity*

Similar to other feature-based clustering algorithms, ISCT requires the extraction of frequent sequential patterns. Consequently, the time complexity of ISCT is at least equivalent to that of algorithms used for discovering frequent sequential patterns. As described in [40], mining maximal frequent patterns from transactional data is an NP-hard problem. It is critical to note that the search space for sequential patterns is even more expansive. This complexity extends to the discovery of frequent sequential patterns from sequence databases, rendering this issue similarly NP-hard. However, the choice of a larger minimum support threshold enables us to prune the search space greatly and the practical computational complexity is acceptable although its analytical form is difficult to obtain. Hence, the symbol δ is employed herein to denote the number of operations associated with mining frequent sequential patterns in practice.

The analysis of the tree construction process can be articulated as follows: Initially, we need to mine frequent patterns for each cluster after the cluster analysis via of random projection, the time complexity here is $O(k \cdot \delta)$. For each pattern obtained, calculating the discriminative power requires at most $O(n)$. The operation to split the tree involves checking whether the sequences in a non-leaf node contain the discriminative pattern, with a time complexity of $O(n \cdot maxl \cdot maxS)$. In summary, the overall time complexity for constructing the tree is $O(k \cdot \delta + k \cdot \beta + k \cdot n \cdot maxl \cdot maxS)$, where β is the number of operations of cluster analysis via random projection as discussed in Section 4.2, which equals 1 in the absence of a boost operation.

Table 2
Some characteristics of the data sets used in the performance comparison.

Data sets	$\|D\|$	$\|\mathcal{I}\|$	min$l$	max$l$	#classes
Activity	35	10	12	43	2
Aslbu	424	250	2	54	7
Auslan2	200	16	2	18	10
Context	240	94	22	246	5
Epitope	2392	20	9	21	2
Gene	2942	5	41	216	2
News	4976	27884	1	6779	5
Pioneer	160	178	4	100	3
Question	1731	3612	4	29	2
Robot	4302	95	24	24	2
Skating	530	82	18	240	7
Reuters	1010	6380	4	533	4
Unix	5472	1697	1	1400	4
Webkb	3667	7736	1	20628	3

5. Experiments

We conduct extensive experiments on 14 real-world data sets to evaluate the performance of ISCT. Specifically, we aim to answer the following Research Questions (RQs):

- **RQ1:** How does our method compare to state-of-the-art approaches in terms of identification performance and interpretability?
- **RQ2:** What is the performance difference between using a naive tree-based approach and ISCT in terms of interpretability?
- **RQ3:** What are the advantages of using the Boost method in constructing interpretable clustering trees?

5.1. Experimental setup

5.1.1. Data sets

We used 14 real-world data sets to evaluate the performance of our algorithm: Activity [41], Aslbu [42], Auslan2 [42], Context [43], Epitope [44], Gene [45], News [46], Pioneer [42], Question, Robot [46], Skating [42], Reuters [46], Unix [46], and Webkb [46]. The details of these data sets are summarized in Table 2, where $|D|$ refers to the number of sequences, $|\mathcal{I}|$ is the number of items, min$l$ and max$l$ are the minimum and maximum length of the sequences, respectively.

5.1.2. Evaluation measures

The evaluation measures used in this study include Purity, NMI (Normalized Mutual Information), and F1-score.

5.1.3. Baseline methods

The following baseline methods are included in the performance comparison:

- Hierarchical clustering methods: HC [5] adopts the standard edit distance to obtain the pairwise dissimilarity and average linkage is used to compute the distance between clusters. For MCSC [7], we utilize the B_MCSC variant.
- Feature-based methods: For FB-LL [6], the minimum support is fixed at 0.05, and the length of the pattern is constrained between 3 and 5. For SGT [4], we set $kappa = 1$ and $lengthSensitive = False$.
- Partitional methods: kkmeans [47] uses the 3-spectrum string kernel. For k-Median [8], the edit distance is used as the pairwise dissimilarity measure. For MinDL [9], the penalty parameters α and γ are specified as 0.1 and 0, respectively.
- Tree-based methods: NTC [48] first divides the sequences into subsequences using a default window length to obtain a segmented matrix presentation, which is used as the input of an individual decision tree or a random forest. Then, the number of repetitions of each segmented subsequence in leaf nodes of decision trees (nTree = 10) are counted to create a numeric representation of the sequence. For RFSC [49], the approach first generates a set of fake sequences as negative samples and then applies a supervised random forest algorithm to produce multiple clustering results. Thereafter, a cluster ensemble procedure based on the bipartite partition is employed to generate the final consensus clusters. Here, the number of trees (q) and the number of random patterns ($numP$) are set to be 30 and 10, respectively.

5.1.4. Implementation

In the experiments, the number of final clusters k is set to be the number of ground-truth clusters except for MinDL. The experiments were performed on a PC with an AMD Ryzen 9 5900X CPU with 32 GB memory, and the results were obtained by averaging ten

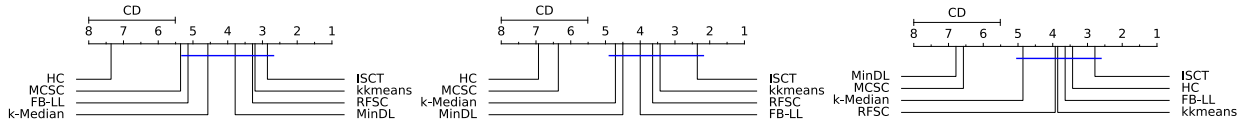

Fig. 5. Bonferroni-Dunn critical difference diagrams at a significance level of 0.05.

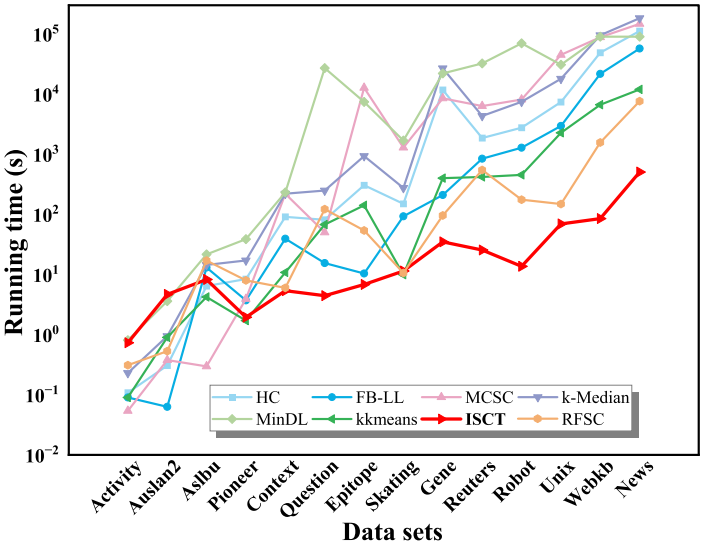

Fig. 6. Running time comparison of ISCT, HC, FB-LL, MCSC, *k*-Median, MinDL, kkmeans, and SGT.

trials. For our algorithm, we set the maximum pattern number to 2048, and the default maximum pattern length to 5 if min$l \leq 10$, maximum pattern length is min$l$. The number of top-k frequent patterns is set to 512. Our source code is available online.[1]

5.2. Investigation of RQ1

5.2.1. Performance

Our method offers a highly interpretable structure for the sequence clustering process. However, it is important to compare our algorithm with existing algorithms as well. The performance comparison results in terms of Purity, NMI, and F1-score are presented in Table 3. Due to memory limitations during the experiment, some results of SGT were not available. As for NTC, some results are missing because it reports operational errors on certain datasets. Fig. 6 shows the running time of the different methods on the 14 data sets.

Our proposed ISCT method achieves the best average performance across all the evaluation measures, indicating that it outperforms the other methods in terms of identification performance. To further evaluate ISCT's performance compared to other methods, Bonferroni-Dunn tests were conducted to validate the null hypothesis that ISCT performs similarly to the other methods. The significance testing results ($CD = 2.49$) are presented in Fig. 5. If there are no connections between our method and the existing algorithm, it indicates that our method is significantly better. From the results, it can be observed that while ISCT does not significantly outperform all other methods, it does exhibit better performance than HC. It is important to note that MinDL achieves the highest purity on certain data sets. However, it often generates a large number of clusters with only a few sequences in each cluster. For instance, on the "Pioneer" data set, MinDL produces 23 clusters, whereas the ground truth number of clusters is only 3. As for SGT, it performs well on many datasets but struggles with data sets containing large item sets due to its high storage space requirements. In comparison to two hierarchical clustering algorithms (HC and MCSC), our method consistently outperforms them on the majority of data sets.

When comparing ISCT with the feature-based method FB-LL, ISCT outperforms FB-LL in terms of purity and NMI on all data sets, and F1-score on 9 data sets. This improvement is mainly attributed to the pattern mining strategy employed by ISCT. In FB-LL, the feature vector is generated by mining frequent patterns. However, frequent patterns are often observed throughout the entire data set, which means that the transformed feature vectors for different sequences might be similar. As a result, the mined patterns rarely provide distinctive information for clustering. On the other hand, by employing random projection and a similarity measure based on the LCS, the transformed features become more informative for clustering. This approach enhances the discriminative power of the feature vectors, enabling better differentiation between different sequences during the clustering process.

[1] https://github.com/jd445/Interpretable-Sequence-Clustering-Tree.

Table 3
Performance comparison of ISCT, HC, FB-LL, MCSC, *k*-Median, MinDL, kkmeans, RFSC, SGT, and NTC.

Datasets	Evaluation	ISCT	HC	FB-LL	MCSC	k-Median	MinDL	kkmeans	RFSC	SGT	NTC
Activity	Purity	0.994	0.600	0.643	0.686	0.695	0.600	0.720	0.700	1	0.886
	NMI	0.973	0.038	0.129	0.118	0.231	0.078	0.225	0.248	1	0.565
	F1-score	0.989	0.652	0.613	0.537	0.620	0.515	0.612	0.664	1	0.808
Aslbu	Purity	0.491	0.380	0.501	0.373	0.441	0.507	0.469	0.507	0.432	N.A
	NMI	0.218	0.028	0.220	0.046	0.139	0.177	0.157	0.230	0.164	N.A
	F1-score	0.317	0.366	0.286	0.183	0.268	0.126	0.257	0.360	0.306	N.A
Auslan2	Purity	0.348	0.195	0.309	0.275	0.316	0.295	0.321	0.347	0.356	N.A
	NMI	0.331	0.159	0.336	0.221	0.315	0.245	0.316	0.312	0.355	N.A
	F1-score	0.265	0.173	0.246	0.153	0.251	0.193	0.256	0.209	0.265	N.A
Context	Purity	0.569	0.425	0.518	0.354	0.572	0.55	0.610	0.377	0.792	0.600
	NMI	0.557	0.466	0.407	0.071	0.515	0.300	0.593	0.201	0.655	0.538
	F1-score	0.551	0.487	0.444	0.235	0.523	0.211	0.573	0.352	0.699	0.535
Epitope	Purity	0.627	0.559	0.559	0.683	0.594	0.670	0.656	0.674	0.559	0.559
	NMI	0.114	0.044	0.096	0.093	0.054	0.060	0.126	0.105	0.142	0.142
	F1-score	0.587	0.671	0.635	0.585	0.550	0.277	0.582	0.568	0.620	0.620
Gene	Purity	1	0.511	1	0.519	0.935	0.965	0.999	0.977	1	1
	NMI	1	0.060	0.994	0.012	0.782	0.158	0.989	0.875	1	1
	F1-score	1	0.665	0.999	0.500	0.914	0.059	0.998	0.956	1	1
News	Purity	0.254	0.210	0.269	0.241	0.284	0.535	0.462	0.266	N.A	N.A
	NMI	0.020	0.002	0.106	0.007	0.036	0.249	0.226	0.023	N.A	N.A
	F1-score	0.262	0.253	0.329	0.218	0.263	0.256	0.344	0.282	N.A	N.A
Pioneer	Purity	0.798	0.656	0.652	0.644	0.662	0.713	0.807	0.790	0.806	0.881
	NMI	0.549	0.064	0.175	0.090	0.097	0.181	0.459	0.465	0.474	0.616
	F1-score	0.688	0.657	0.476	0.406	0.515	0.338	0.652	0.686	0.684	0.727
Question	Purity	0.656	0.518	0.518	0.745	0.604	0.570	0.560	0.656	N.A	0.637
	NMI	0.122	0.022	0.019	0.295	0.081	0.047	0.015	0.086	N.A	0.205
	F1-score	0.572	0.666	0.619	0.665	0.591	0.546	0.514	0.561	N.A	0.626
Reuters	Purity	0.579	0.255	0.321	0.347	0.448	0.287	0.699	0.528	N.A	0.450
	NMI	0.392	0.097	0.101	0.048	0.154	0.062	0.482	0.293	N.A	0.319
	F1-score	0.501	0.298	0.350	0.292	0.388	0.391	0.582	0.479	N.A	0.431
Robot	Purity	0.639	0.515	0.544	0.637	0.557	0.535	0.618	0.595	0.683	0.551
	NMI	0.071	0.046	0.035	0.056	0.017	0.019	0.068	0.032	0.107	0.032
	F1-score	0.551	0.655	0.592	0.538	0.521	0.522	0.564	0.522	0.585	0.643
Skating	Purity	0.208	0.183	0.222	0.221	0.220	0.232	0.228	0.221	0.227	0.215
	NMI	0.041	0.029	0.041	0.023	0.041	0.112	0.032	0.039	0.053	0.047
	F1-score	0.224	0.252	0.193	0.150	0.181	0.246	0.164	0.169	0.159	0.181
Unix	Purity	0.506	0.444	0.491	0.464	0.479	0.756	0.451	0.610	N.A	N.A
	NMI	0.110	0.004	0.094	0.031	0.055	0.224	0.018	0.235	N.A	N.A
	F1-score	0.385	0.484	0.422	0.308	0.371	0.227	0.438	0.430	N.A	N.A
Webkb	Purity	0.502	0.444	0.443	0.448	0.479	0.723	0.473	0.491	N.A	N.A
	NMI	0.094	0.020	0.055	0.019	0.072	0.199	0.126	0.078	N.A	N.A
	F1-score	0.452	0.421	0.455	0.378	0.426	0.341	0.403	0.473	N.A	N.A
Mean (Average Rank)	Purity	0.584 (2.86)	0.421 (7.36)	0.499 (5.14)	0.474 (5.36)	0.520 (4.57)	0.567 (3.79)	0.577 (3.21)	0.553 (3.29)	/	/
	NMI	0.328 (2.35)	0.077 (6.93)	0.201 (4.00)	0.081 (6.36)	0.185 (4.71)	0.151 (4.50)	0.274 (3.42)	0.230 (3.64)	/	/
	F1-score	0.525 (2.79)	0.479 (3.43)	0.476 (3.64)	0.368 (6.57)	0.456 (4.86)	0.303 (6.79)	0.496 (3.86)	0.479 (3.93)	/	/

When comparing ISCT with kkmeans and *k*-Median, ISCT demonstrates superior performance on the majority of data sets. Additionally, as depicted in Fig. 6, our method exhibits significantly improved efficiency. In summary, ISCT achieves comparable performance to state-of-the-art methods while delivering considerably faster computation times and ensuring interpretability.

5.2.2. Interpretability

In the field of clustering, there's still no clear quantitative standard for assessing the interpretability of an algorithm [12]. Typically, for rule-based methods, interpretability metrics are related to the number and length of rules [16]. Likewise, for a tree structure, our goal is to minimize the number of splitting nodes or patterns. We strive for clustering that utilizes a smaller tree, where the splitting criteria are as concise as possible.

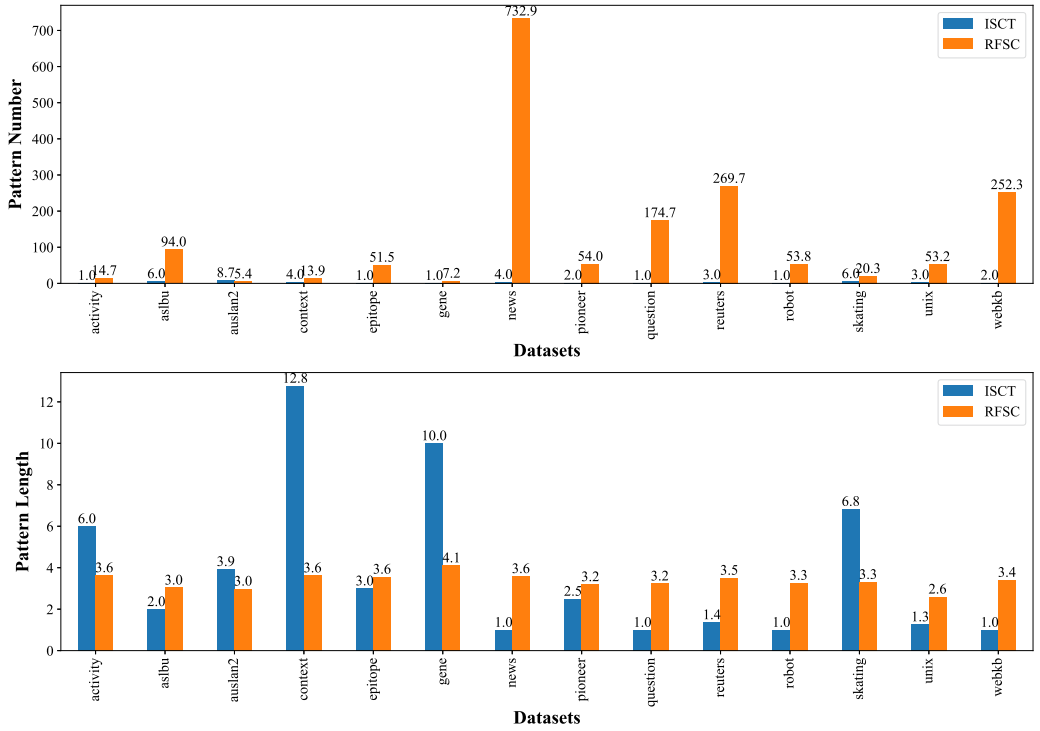

Fig. 7. Comparison of ISCT and RFSC in terms of utilized pattern number and pattern length per tree on average.

Therefore, our goal is to create a tree using fewer and shorter sequential patterns while maintaining performance. Other sequence clustering algorithms typically do not incorporate interpretability into their design, making them less suitable for direct comparison. For instance, in standard feature-based clustering algorithms, the pattern number is usually predetermined, often varying from several hundred to thousands. These methods focus on a comprehensive data representation, do not prioritize interpretability, which is a key consideration in our approach. Here we compare our method with RFSC [49], which is a sequence clustering algorithm based on random forest. This comparison focuses on the number and length of patterns used in each method. The comparative results, as shown in Fig. 7, reveal that ISCT significantly uses fewer patterns than RFSC, especially in larger datasets like “news”, “webkb”, “reuters”, and “question”, while maintaining competitive performance. In addition, the pattern length of ISCT is typically shorter than that of RFSC on most data sets. This demonstrates that our algorithm constructs significantly smaller trees while being competitive in pattern length usage compared to RFSC.

To concretely illustrate the interpretability of our approach, we employ the “pioneer” dataset as an example to plot its clustering tree. Detailed visualizations of clustering trees for all datasets are available in Supplementary Note 1. Furthermore, we compare the outcomes derived from our clustering tree with those obtained from the hierarchical clustering method. The comparative results are presented in Fig. 8.

Our method starts with k-means clustering results ($k = 3$) on features from random projections as pseudo-labels. Next, we identify the most discriminative sequential pattern (‘11’, ‘12’), grouping sequences hit by pattern into a cluster, as seen in cluster 3 in Fig. 8. Subsequently, we apply k-means again to the features (this time with $k = 2$), uncovering another discriminative sequential pattern (‘113’, ‘134’, ‘114’), and forming two more clusters, depicted as cluster 1 and cluster 2 in Fig. 8.

Notably, our approach achieves outstanding clustering performance, evidenced by purity, NMI, and F1-scores of 0.798, 0.549, and 0.688, respectively. Simultaneously, our clustering method maintains a high degree of interpretability, as it employs just two patterns and the length of each pattern is very short. This balance of identification performance and interpretability underscores the effectiveness of our methodology in data mining. In contrast, hierarchical clustering trees face significant challenges in understanding the decision-making process due to their reliance on a distance metric, particularly when interpreting the distance threshold between sequences, which is greater than 130 under the distance function LCS for the “pioneer” dataset. Then, obtaining interpretable results from this method is intricate. Furthermore, when assigning a new sample to existing clusters, ISCT can determine its cluster membership using only two patterns as well. In contrast, hierarchical clustering methods necessitate distance calculations for all data points, presenting a considerable efficiency and interpretability gap compared to ISCT.

5.3. Investigation of RQ2

The cluster membership can be viewed as pseudo labels. A naive approach to utilize cluster membership as the class label to train a classifier using popular tree-based methods. However, the interpretability of tree-based methods often falls short due to the

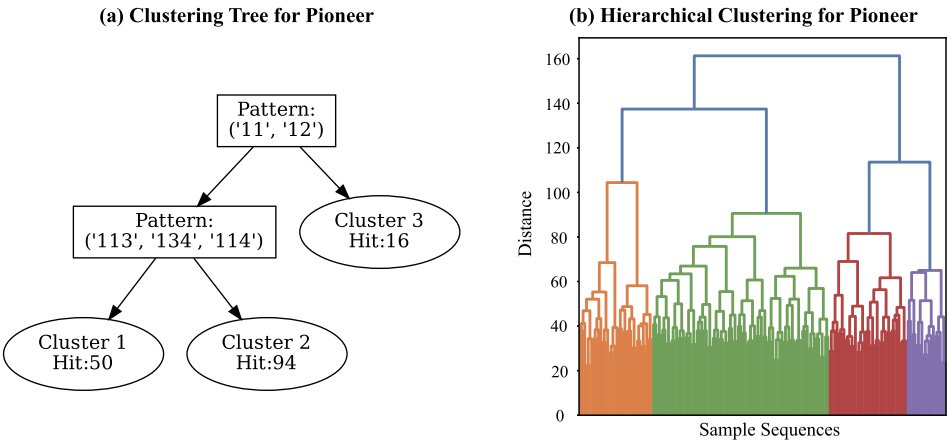

Fig. 8. Comparison of the decision process of sequence clustering tree and hierarchical clustering on "Pioneer".

Table 4
Comparison of average tree sizes between ISCT and SeqDT in terms of the number of leaf nodes with different Gini impurities (averaged over ten experiments).

Datasets	Ground Truth	ISCT	SeqDT_0.1	SeqDT_0.01
Activity	2	**2**	**2**	**2**
Aslbu	7	**7**	32.2	33.7
Auslan2	10	**9.5**	11.9	13.1
Context	5	**5**	9.5	10.1
Epitope	2	**2**	**2**	11.6
Gene	2	**2**	**2**	**2**
News	5	**5**	94.1	159.3
Pioneer	3	**3**	4.6	3.7
Question	2	**2**	6.8	4.1
Reuters	4	**4**	34.5	35.7
Robot	2	**2**	76.1	107.2
Skating	7	**7**	71.4	66.2
Unix	4	**4**	53.3	84.5
Webkb	3	**3**	132.1	185.6

complexity in controlling the size of the obtained tree. In practice, the tendency to produce large trees frequently hampers the clarity of the decision-making process.

In this section, we compare the interpretability of our ISCT method with SeqDT [38], as both methods use patterns for splitting. We aim to assess which method provides a more interpretable representation of the clustering process, of which the interpretability is measured with the constructed tree size. For SeqDT, we use different parameters on the threshold of Gini impurity to show its influence on the size of the tree.

The results of the tree size comparison in terms of the number of leaf nodes between ISCT and SeqDT with different Gini impurities are shown in Table 4. The ground truth column represents the actual number of clusters $|C|$ in each data set.

For most data sets, SeqDT with a threshold of 0.01 produces larger trees compared to SeqDT with a threshold of 0.1. This suggests that a lower threshold leads to more complex trees, which may be harder to interpret. However, even with a threshold of 0.1 and employing the PEP (Pessimistic Error Pruning) strategy, SeqDT still generates larger trees compared to ISCT, indicating its limited interpretability.

In contrast, ISCT consistently reports the same number of leaf nodes as the ground truth, except on the "Auslan2" data set where there are a significant number of duplicate sequences marked as different classes. Excluding this exceptional case, the consistent number of leaf nodes in ISCT indicates the effectiveness of our tree construction strategy. It demonstrates that ISCT can accurately capture the underlying clustering structure while maintaining a concise and intuitive decision tree structure. As a result, ISCT facilitates a better understanding and provides valuable insights into the clustering results.

5.4. Investigation of RQ3

Firstly, the clustering results obtained through random projection differ from those achieved by directly constructing the tree. Essentially, direct tree construction fits the initial clusters while restricting the tree's complexity. During the tree construction process, we introduce a boosting approach to facilitate the construction. In order to validate the efficacy of our proposed strategy, we perform experiments on datasets that consist of more than two categories.

Table 5
Performance comparison on different tree construction strategies in terms of average reduction rate with respect to Purity, NMI and F1-score.

Method	ICST(Boost)	ICST(Direct)
Purity	**-0.099**	-0.158
NMI	**-0.202**	-0.264
F1	**0.040**	0.039

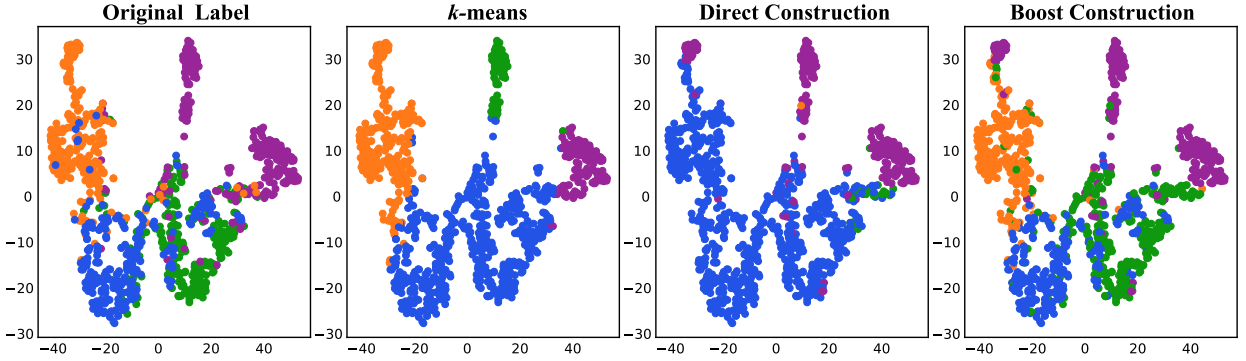

Fig. 9. T-SNE visualization of feature space with original label, k-means, direct and boost constructions.

Table 5 presents a comparison of the average performance reduction rates between different tree construction methods and the random projection clustering algorithm. The results demonstrate that the performance reduction rates of boosting strategy are superior to direct construction method.

To show the benefits of our construction strategy, we employ t-SNE to visualize the "Reuters" as an illustrative example. The visualization results are presented in Fig. 9. It is evident from the figure that the distinguishability of the four clusters in the random projection subspace is not particularly apparent. Consequently, this leads the k-means algorithm to erroneously merge the blue and green clusters into a single cluster. Constructing a tree based on this partition would result in the identification of inappropriate patterns. However, by leveraging our boosting strategy, we successfully extract patterns that accurately differentiate between the four distinct categories.

6. Conclusion

This paper introduces the Interpretable Sequence Clustering Tree (ISCT), which aims to construct a concise and interpretable decision tree for sequence clustering. The ISCT consists of k leaf nodes that correspond to k clusters, enabling a transparent and comprehensible decision-making process for sequence clustering. Our method is the first interpretable clustering algorithm specifically designed for sequence data. After obtaining sequence clustering results using k-means, we integrate a sequential pattern discovery algorithm to identify the characteristics of the clusters. Unlike existing threshold-based trees, our approach proposes the use of relative risk as a splitting criterion for discrete sequences and employs a boosting method to update the candidate patterns list. Experimental results on 14 real-world benchmark data sets demonstrate that our proposed ISCT method achieves state-of-the-art performance while ensuring interpretability.

Our objective is to determine a set of $k-1$ patterns and to construct a concise tree. However, the number of possible candidate patterns is vast, and there are numerous ways to construct the tree. Our current method is a two-stage heuristic approach, where the initial stage influences the subsequent stage. In the future, we aim to develop a joint optimization algorithm that operates as a single, unified process, which we believe will offer a more coherent and efficient solution.

CRediT authorship contribution statement

Junjie Dong: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Formal analysis, Conceptualization. **Xinyi Yang:** Writing – review & editing. **Mudi Jiang:** Writing – review & editing, Methodology, Investigation. **Lianyu Hu:** Writing – review & editing, Methodology, Formal analysis, Conceptualization. **Zengyou He:** Writing – original draft, Supervision, Methodology, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the data and code link in the paper.

Acknowledgement

This work has been supported by the Natural Science Foundation of China under Grant No. 62472064.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ins.2024.121453.

References

[1] Q. Zou, G. Lin, X. Jiang, X. Liu, X. Zeng, Sequence clustering in bioinformatics: an empirical study, Brief. Bioinform. 21 (1) (2020) 1–10.
[2] Y. Li, C. Luo, S.M. Chung, Text clustering with feature selection by using statistical data, IEEE Trans. Knowl. Data Eng. 20 (5) (2008) 641–652.
[3] A. Gupta, V. Dengre, H.A. Kheruwala, M. Shah, Comprehensive review of text-mining applications in finance, Financ. Innov. 6 (2020) 1–25.
[4] C. Ranjan, S. Ebrahimi, K. Paynabar, Sequence graph transform (sgt): a feature embedding function for sequence data mining, Data Min. Knowl. Discov. 36 (2) (2022) 668–708.
[5] R.J.C. Bose, W.M. Van der Aalst, Context aware trace clustering: towards improving process mining results, in: Proceedings of the 2009 SIAM International Conference on Data Mining, 2009, pp. 401–412.
[6] V. Guralnik, G. Karypis, A scalable algorithm for clustering sequential data, in: Proceedings of the 2001 IEEE International Conference on Data Mining, 2001, pp. 179–186.
[7] T.X. Society, S. Wang, Q. Jiang, J.Z. Huang, A novel variable-order Markov model for clustering categorical sequences, IEEE Trans. Knowl. Data Eng. 26 (10) (2013) 2339–2353.
[8] L.P. Dinu, R.T. Ionescu, Clustering based on median and closest string via rank distance with applications on dna, Neural Comput. Appl. 24 (2014) 77–84.
[9] Y. Chen, P. Xu, L. Ren, Sequence synopsis: optimize visual summary of temporal event data, IEEE Trans. Vis. Comput. Graph. 24 (1) (2017) 45–55.
[10] J. Basak, R. Krishnapuram, Interpretable hierarchical clustering by constructing an unsupervised decision tree, IEEE Trans. Knowl. Data Eng. 17 (1) (2005) 121–132.
[11] S. Bandyapadhyay, F.V. Fomin, P.A. Golovach, W. Lochet, N. Purohit, K. Simonov, How to find a good explanation for clustering?, Artif. Intell. 322 (2023) 103948.
[12] D. Bertsimas, A. Orfanoudaki, H. Wiberg, Interpretable clustering: an optimization approach, Mach. Learn. 110 (2021) 89–138.
[13] E. Carrizosa, K. Kurishchenko, A. Marín, D. Romero Morales, Interpreting clusters via prototype optimization, Omega 107 (2022) 102543, https://doi.org/10.1016/j.omega.2021.102543.
[14] M. Moshkovitz, S. Dasgupta, C. Rashtchian, N. Frost, Explainable k-means and k-medians clustering, in: Proceedings of International Conference on Machine Learning, 2020, pp. 7055–7065.
[15] R. Fraiman, B. Ghattas, M. Svarc, Interpretable clustering using unsupervised binary trees, Adv. Data Anal. Classif. 7 (2013) 125–145.
[16] C. Lawless, J. Kalagnanam, L.M. Nguyen, D. Phan, C. Reddy, Interpretable clustering via multi-polytope machines, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 7309–7316.
[17] K. Makarychev, L. Shan, Explainable k-means: don't be greedy, plant bigger trees!, in: Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing, 2022, pp. 1629–1642.
[18] M. Charikar, L. Hu, Near-optimal explainable k-means for all dimensions, in: Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2022, pp. 2580–2606.
[19] L. Jiao, H. Yang, Z. ga Liu, Q. Pan, Interpretable fuzzy clustering using unsupervised fuzzy decision trees, Inf. Sci. 611 (2022) 540–563.
[20] H. Yang, L. Jiao, Q. Pan, A survey on interpretable clustering, in: Proceedings of the 40th Chinese Control Conference (CCC), IEEE, 2021, pp. 7384–7388.
[21] X. Wang, X. Liu, L. Zhang, A rapid fuzzy rule clustering method based on granular computing, Appl. Soft Comput. 24 (2014) 534–542.
[22] E.G. Mansoori, Frbc: a fuzzy rule-based clustering algorithm, IEEE Trans. Fuzzy Syst. 19 (5) (2011) 960–971.
[23] C.-H. Hsieh, C.-H. Yan, C.-H. Mao, C.-P. Lai, J.-S. Leu, Gminer: rule-based fuzzy clustering for Google drive behavioral type mining, in: Proceedings of the 2016 International Computer Symposium (ICS), IEEE, 2016, pp. 98–103.
[24] D. Pelleg, A.W. Moore, Mixtures of rectangles: interpretable soft clustering, in: Proceedings of the International Conference on Machine Learning, 2001, pp. 401–408.
[25] J. Chen, Y. Chang, B. Hobbs, P. Castaldi, M. Cho, E. Silverman, J. Dy, Interpretable clustering via discriminative rectangle mixture model, in: Proceedings of the 2016 IEEE 16th International Conference on Data Mining, IEEE, 2016, pp. 823–828.
[26] L. Yuan, W. Wang, L. Chen, Two-stage pruning method for gram-based categorical sequence clustering, Int. J. Mach. Learn. Cybern. 10 (2019) 631–640.
[27] C. Li, X. Dong, W. Liu, S. Sheng, A. Qian, Ssrdvis: interactive visualization for event sequences summarization and rare detection, J. Vis. 23 (2020) 171–184.
[28] C. Li, Q. Yang, J. Wang, M. Li, Efficient mining of gap-constrained subsequences and its various applications, ACM Trans. Knowl. Discov. Data 6 (1) (2012) 1–39.
[29] M. Steinegger, J. Söding, Clustering huge protein sequence sets in linear time, Nat. Commun. 9 (1) (2018) 2542.
[30] W. Min, W. Liang, H. Yin, Z. Wang, M. Li, A. Lal, Explainable deep behavioral sequence clustering for transaction fraud detection, arXiv preprint, arXiv:2101.04285, 2021.
[31] A. Nadeem, S. Verwer, Secleds: sequence clustering in evolving data streams via multiple medoids and medoid voting, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2022, pp. 157–173.
[32] M. Bicego, V. Murino, M.A. Figueiredo, Similarity-based clustering of sequences using hidden Markov models, in: Proceedings of the Third International Conference on Machine Learning and Data Mining in Pattern Recognition, Springer, 2003, pp. 86–95.
[33] M. Ramoni, P. Sebastiani, P. Cohen, Bayesian clustering by dynamics, Mach. Learn. 47 (2002) 91–121.
[34] S.-J. Oh, J.-Y. Kim, A hierarchical clustering algorithm for categorical sequence data, Inf. Process. Lett. 91 (3) (2004) 135–140.
[35] Z. Xing, J. Pei, E. Keogh, A brief survey on sequence classification, ACM SIGKDD Explor. Newsl. 12 (1) (2010) 40–48.
[36] D. Fradkin, F. Mörchen, Mining sequential patterns for classification, Knowl. Inf. Syst. 45 (2015) 731–749.
[37] C. Zhou, B. Cule, B. Goethals, Pattern based sequence classification, IEEE Trans. Knowl. Data Eng. 28 (5) (2016) 1285–1298.
[38] Z. He, Z. Wu, G. Xu, Y. Liu, Q. Zou, Decision tree for sequences, IEEE Trans. Knowl. Data Eng. 35 (1) (2023) 251–263.
[39] V. Kadappa, A. Negi, Computational and space complexity analysis of subxpca, Pattern Recognit. 46 (8) (2013) 2169–2174.
[40] G. Yang, Computational aspects of mining maximal frequent patterns, Theor. Comput. Sci. 362 (1–3) (2006) 63–85.

[41] A. Asuncion, D. Newman, Uci Machine Learning Repository, 2007.
[42] C. Di Ciccio, M. Mecella, A two-step fast algorithm for the automated discovery of declarative workflows, in: Proceedings of the 2013 IEEE Symposium on Computational Intelligence and Data Mining, IEEE, 2013, pp. 135–142.
[43] F.M. Maggi, R.J.C. Bose, W.M. van der Aalst, Efficient discovery of understandable declarative process models from event logs, in: Proceedings of 24th International Conference on Advanced Information Systems Engineering, Springer, 2012, pp. 270–285.
[44] F.M. Maggi, M. Montali, C. Di Ciccio, J. Mendling, Semantical vacuity detection in declarative process mining, in: Proceedings of the 14th International Conference on Business Process Management, Springer, 2016, pp. 158–175.
[45] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014) 1–37.
[46] J. Zhang, Y. Wang, D. Yang, Ccspan: mining closed contiguous sequential patterns, Knowl.-Based Syst. 89 (2015) 1–13.
[47] K. Xu, L. Chen, S. Wang, A multi-view kernel clustering framework for categorical sequences, Expert Syst. Appl. 197 (2022) 116637.
[48] H. Jahanshahi, M.G. Baydogan, nTreeClus: a tree-based sequence encoder for clustering categorical series, Neurocomputing 494 (2022) 224–241.
[49] M. Jiang, J. Wang, L. Hu, Z. He, Random forest clustering for discrete sequences, Pattern Recognit. Lett. 174 (2023) 145–151.