Information Sciences 704 (2025) 121972

Contents lists available at ScienceDirect

Information Sciences

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

Significance-based interpretable sequence clustering

Zengyou He *, Lianyu Hu, Jinfeng He, Junjie Dong, Mudi Jiang, Xinying Liu

School of Software, Dalian University of Technology, Tuqiang Road, Dalian, 116024, Liaoning, China

ARTICLE INFO

Keywords:
Sequential data clustering
Interpretable clustering
Unsupervised decision trees
Hypothesis testing
Multiple testing correction
p-value combination

ABSTRACT

Recently, many interpretable clustering algorithms have been proposed, which focus on characterizing the clustering outcome in terms of explainable models such as trees and rules. However, existing solutions are mainly developed for handling standard vectorial data and how to obtain interpretable clustering results for complicated non-vector data such as sequences and graphs is still in the infant stage. In this paper, we present a significance-based interpretable clustering algorithm for discrete sequences, which has the following key features. Firstly, instead of using a third-party clustering method to obtain the initial clusters, we directly extract cluster-critical sequential patterns to describe potential clusters. Secondly, without needing to specify the number of clusters, we guide the growth of the decision tree through a hypothesis testing procedure. As a result, not only the final clustering result is explainable but also the tree construction process is statistically interpretable. Experimental results on real-world sequential data sets show that our algorithm achieves comparable performance to state-of-the-art methods in both cluster quality and interpretability.

1. Introduction

Cluster analysis is a fundamental research issue in the field of machine learning and data mining. The specific goal of a clustering algorithm is to find clusters of data samples, where samples within each cluster are more similar to each other while samples from different clusters should be as dissimilar as possible. For decades, numerous clustering algorithms have been developed from different viewpoints based on different problem formulations [1].

Standard clustering algorithms are developed for partitioning vectorial data into different groups, where each sample in the data set is characterized by a fixed number of features. However, in many real applications, we have to conduct cluster analysis on more complicated non-vectorial data types such as sequences and graphs. In this paper, we focus on the discrete sequence clustering problem, where each sequence in the sequential data set is composed of an ordered list of items from an alphabet. Such sequence clustering algorithms are highly demanded in many fields such as bioinformatics [2] and process mining [3].

To date, many fast and accurate clustering algorithms for discrete sequences have been developed (e.g. [4,5]). These clustering algorithms can be categorized into different classes according to their underlying algorithmic principles: feature-based algorithms, hierarchical algorithms, partitional algorithms and model-based algorithms. The basic ideas and recent advances for different types of clustering methods would be introduced and discussed in the related work section. However, due to the discrete nature and the

* Corresponding author.
E-mail addresses: zyhe@dlut.edu.cn (Z. He), hly4ml@gmail.com (L. Hu), 3014484478@qq.com (J. He), jd445@qq.com (J. Dong), 792145962@qq.com (M. Jiang), 72317011@mail.dlut.edu.cn (X. Liu).

https://doi.org/10.1016/j.ins.2025.121972
Received 21 November 2024; Received in revised form 11 February 2025; Accepted 11 February 2025

Available online 13 February 2025

complex order-related semantics of sequences, current methods still lack a clear and direct way to produce final clustering results that are easy to understand. Here we would like to re-emphasize the following fact [6]: existing sequence clustering algorithms mainly focus on how to yield an accurate clustering outcome quickly, ignoring the interpretability issue of clusters. In other words, explaining why a set of sequences forms a cluster and characterizing each cluster in a human-understandable manner remains an open issue.

More recently, much attention has been paid to the development of interpretable clustering methods for standard vectorial data (e.g. [7,8]). Essentially, these methods try to employ interpretable models such as decision trees and rules to explain and characterize clusters. The interpretable clustering model is constructed based on two different strategies. On one hand, we can first run an existing clustering method and then use the clustering output as input to guide the generation of explainable models. On the other hand, without the reliance on a third-party clustering method, we may construct the interpretable clustering model from the target data set directly.

Despite the success of recent advances in interpretable clustering algorithms, transferring these methods to the domain of sequence clustering remains a non-trivial task for the following reasons. Firstly, since no explicit features are present in the sequential data, the extraction of interpretable and cluster-critical features is a challenging issue. Secondly, the potential feature space for sequential data is vast, which poses challenges for constructing the subsequent interpretable clustering model. Finally, due to the discrete nature of sequences, those existing interpretable algorithms for numeric vector data cannot be directly deployed.

To solve the interpretable sequence clustering issue, we have made a pilot study in [6]. In [6], the sequential patterns are employed as features and the decision tree is chosen as the interpretable clustering model for explaining clusters. The proposed algorithm iteratively performs discriminative pattern mining and tree growth to obtain a clustering decision tree. However, several limitations remain that need to be addressed, as discussed below.

First of all, the algorithm in [6] implicitly requires a third-party clustering method to guide the decision tree construction. The use of different clustering algorithms may affect the final clustering result. Secondly, the number of ground-truth clusters should be specified as input, which is typically unknown in practice. Finally, the algorithm in [6] can only ensure that the clustering result is explainable in terms of decision trees, failing to guarantee the interpretability of clustering process. That is, the split or non-split decision at each node is not statistically explainable.

Motivated by the above observations, we present a new tree-based interpretable clustering algorithm for sequences, which is named as SigISC (**Sig**nificance-based **I**nterpretable **S**equence **C**lustering). To alleviate the drawbacks of existing algorithms, the following novel ideas are introduced in SigISC. First, a new sequential pattern mining algorithm is presented, which is capable of extracting sequential patterns that are highly correlated with the underlying clustering structure. Second, the hypothesis testing method is employed for assessing the goodness of candidate split point, enabling that the split or non-split decision is at least explainable in a statistical sense. Finally, the significance testing results, expressed as p-values at each node, allow us to automatically determine whether a node should be further divided during the tree growth process. Therefore, our algorithm is able to determine the number of clusters adaptively without the need to use the ground-truth cluster number as input.

In summary, the main contributions of this paper can be summarized as follows:

- We present the first significance-based interpretable clustering algorithm for sequences, which provides a statistically explainable split decision and automatically determines the number of clusters.
- We propose a new cluster-critical pattern mining algorithm that identifies sequential patterns closely related to the clustering task. Moreover, this algorithm can be used as a general feature extraction method in the context of sequence clustering.
- Extensive experiments on real sequential data sets show that our method can achieve comparable performance to non-interpretable sequence clustering methods. More importantly, it offers several advantages over existing interpretable clustering methods.

The remainder of this paper is organized as follows. Section 2 summarizes and discusses related methods. Section 3 presents the details of our proposed clustering algorithm. Section 4 presents the empirical results on real sequential data sets. Section 5 concludes the paper.

2. Related work

Since there is only a piece of work that focuses on interpretable sequence clustering, we will mainly discuss related methods from two domains: discrete sequence clustering and interpretable clustering. Moreover, some discussions on significance-based cluster analysis would be provided as well.

2.1. Sequence clustering

As briefly mentioned in the introduction, existing sequence clustering methods can be broadly categorized into four types, as elaborated below.

The feature/pattern-based method is composed of two steps. In the first step, we first transform sequences into fixed-length vectors via a feature extraction method. The most widely used feature extraction methods include sequential pattern mining algorithms [9,10] and sequence embedding approaches [11,12]. It is easy to see that the interpretability of sequential patterns (i.e., subsequences) is better than that of embedded vectors, since the latter do not have explicitly understandable features that correspond to the original samples. After the transformation, we can employ a standard vectorial data clustering algorithm such as k-means to obtain the final

clustering result. Hence, to produce interpretable clustering outcomes, we may apply existing interpretable clustering algorithms to the transformed vector data. In the experiment section, we will utilize such a strategy as the baseline to demonstrate the effectiveness of both our pattern mining algorithm and the significance-based tree construction method.

The hierarchical method follows the popular hierarchical clustering framework, employing either divisive or agglomerative heuristics to generate a dendrogram. To apply such a general framework to the sequence clustering issue, we need to choose a proper distance function to measure the dissimilarity between two sequences [13,14]. Note that the hierarchical method will also generate a tree structure, however, its interpretability is insufficient. This is primarily because (1) the tree is large, consisting of n leaf nodes (where n is the number of sequences in the data set), and (2) providing concise explanations for each cluster is challenging.

The partitional method takes cluster assignment identifiers as variables and tries to minimize an objective function so as to directly output the cluster result [15,5]. Note that although decision trees are used in [4], the decision tree is primarily employed for generating initial partitions, rather than for producing final interpretable clustering results. In other words, each decision tree can be very large in size and less accurate. Accordingly, the final random forest cannot be employed as an interpretable clustering model. The MinDL method [16] can be viewed as an interpretable clustering method for sequences in the sense that it provides a single pattern representation for each cluster. However, such a pattern is the one with the minimum sum of edit distances to all sequences in the cluster. In other words, although it can be considered a "prototype-based" interpretable clustering method, its interpretability is less intuitive than ours, where each pattern is either contained within sequences in a cluster as a subsequence or not. In addition, the MinDL algorithm usually reports a large number of clusters even after tuning its hyperparameters manually.

The model-based method typically assumes a finite mixture of probability distributions for the sequential data. The most popular models in the literature are Markov models [17,18] and Hidden Markov Models (HMM) [19]. We can determine the model structure (e.g., the number of hidden states in an HMM) using model selection techniques and estimate parameters with the maximum likelihood estimation algorithms. Such model-based approaches do provide a probabilistic explanation on why certain sequences are allocated to their designated cluster. However, it is generally not intuitive or easy for end users to understand.

2.2. Interpretable clustering

Recently, interpretable cluster analysis has gained much attention in the field of machine learning. In addition to clustering accuracy, interpretable clustering methods aim to provide a concise, human-understandable explanation of the meaning of each cluster. To date, many interpretable clustering algorithms have been proposed for handling standard vectorial data. Essentially, research efforts towards this direction can be categorized into different classes according to the interpretable model employed. Generally, the most widely used models include rules [20], decision trees [21], and geometric boundaries such as hyper-rectangles [22], hypercubes [23], polyhedra [24], and polyhedron [25]. A detailed comparison and systematic categorization of many existing interpretable clustering methods can be found in a recent survey [26].

The decision tree [27,8] is probably the most popular model in the literature because the path from the root to each leaf node concisely describes why samples are allocated to the corresponding cluster. Moreover, the decision tree can provide a set of non-overlapping clusters and each sample will be assigned to only one cluster. Since this paper also adopts the decision tree as the interpretable clustering model, the following paragraphs will further focus on the discussion of tree-based interpretable clustering methods.

To construct a decision tree for forming and explaining clusters, there are typically two strategies. One strategy is to first utilize an existing non-interpretable clustering algorithm to obtain an initial clustering result. Then, the cluster identifier for each sample can be used to supervise the decision tree construction procedure (e.g. [27,28]). Another strategy is to build the decision tree in an unsupervised manner without relying on a third-party clustering algorithm. The basic idea is to formulate the tree construction problem as an optimization problem (e.g. [8,29]). To obtain the solution, both heuristic algorithms and methods that can approximate the global optimum have been developed.

Note that all existing tree-based interpretable clustering algorithms focus on tackling standard vectorial data [7,30]. One exception is the proposed method in [6], which aims at constructing an unsupervised decision tree for the purpose of interpretable cluster analysis on sequential data. However, just as we mentioned in the introduction, the algorithm in [6] still has several drawbacks. More precisely, it needs to know the number of ground-truth clusters and the split/non-split decision at each node is not statistically interpretable. To alleviate these issues, a new interpretable clustering algorithm will be developed in this paper.

2.3. Significance-based clustering

In the literature on cluster analysis, researchers typically adopt an optimization-based approach to solve the clustering problem, where the division of samples into clusters is formulated as an optimization problem. However, optimization-based clustering algorithms can always report a set of clusters, failing to guarantee that the corresponding clustering result is statistically meaningful. Building on this observation, researchers have started developing algorithms that address cluster analysis from a significance testing perspective (e.g., [31,32]).

It is important to note that these significance-based clustering algorithms primarily focus on vector data. To date, no significance-based clustering algorithms have been developed for discrete sequences. Therefore, from this perspective, the method proposed in this paper is also novel.

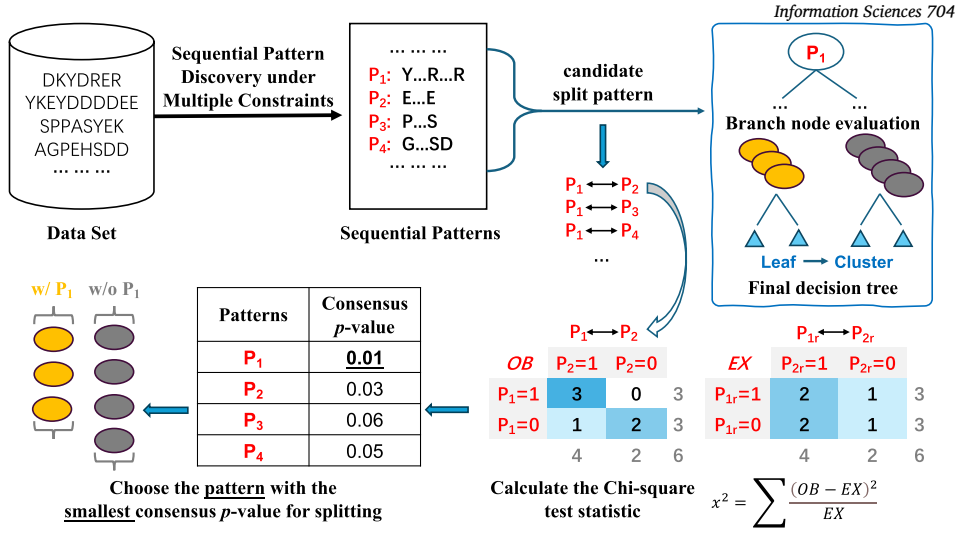

Fig. 1. The workflow of SigISC. It is mainly composed of two key steps: (1) Extracting cluster-critical sequential patterns under multiple constraints and (2) Constructing clustering decision trees via hypothesis testing.

3. Method

3.1. An overview of the algorithm

In order to effectively conduct interpretable cluster analysis via decision tree on a given sequential dataset, we first need to solve two issues: (1) what is employed to serve as the splitting point and (2) how to evaluate the goodness of each candidate splitting point? For the first issue, we choose the sequential pattern as the splitting point during decision tree construction since such patterns are easy to understand by end-users. Since cluster analysis is an unsupervised machine learning issue, the cluster identifier of each sequence is unknown, we have to identify some substitutes to characterize clusters in order to assess candidate splitting points. That is, to tackle the second issue, we try to identify a set of sequential patterns that are potentially correlated with different clusters. Once we have such a cluster-critical pattern set, one candidate splitting point can be evaluated based on its correlation with all cluster-critical patterns.

Hence, our algorithm consists of two steps: extracting cluster-critical sequential patterns and constructing clustering decision trees. In the pattern extraction procedure, we extract a series of sequential patterns under multiple constraints. These constraints are imposed in order to obtain a concise and non-redundant pattern set with the hope that each pattern is positively correlated with at least one underlying cluster. In the tree construction procedure, we take the pattern set in the previous step as the candidate splitting point set. During the recursive decision tree construction process, if each pattern is regarded as a binary variable, then the problem is to build an unsupervised decision tree from a categorical data set in which each feature only takes two values. To evaluate each pattern-based splitting point, we employ the chi-squared test to assess its correlation with all remaining patterns in terms of p-values. To obtain a consensus p-value, we can employ the p-value combination method in meta analysis or use the sum of chi-squared statistics under the independence assumption. The pattern with the smallest p-value will be employed as the splitting point for dividing the sequential data set into two subsets. The workflow of SigISC is summarized in Fig. 1.

3.2. Extracting cluster-critical sequential patterns

3.2.1. What are cluster-critical sequential patterns?

Since the underlying clusters are unknown, it is a quite challenging task to evaluate if one pattern is associated with the cluster identifier variable. Under the assumption that the sequential data set is composed of at least two clusters, it is reasonable to claim that such patterns should meet at least the following constraints:

- Firstly, such a pattern should be frequent. If the data set can be divided into several distinct clusters, then sequential patterns that occur frequently in each cluster can be observed. This is because sequences in each cluster should be similar to each other, leading to the generation of multiple frequent sequential patterns with respect to each cluster. Therefore, a cluster-critical sequential pattern should be a frequent one whose frequency is no less than a minimum support threshold.
- Secondly, such a pattern must not be too frequent. Since sequences from different clusters are quite dissimilar, one cluster-critical sequential pattern should only appear in one cluster in an ideal case. That is, if a sequential pattern occurs in most of the sequences, then it is unable to distinguish different clusters. Therefore, the target sequential pattern should be no larger than a maximum support threshold at the same time.

Algorithm 1 Extracting Cluster-Critical Sequential Patterns.

Input: A sequential dataset S, the pattern length k and a percentage parameter f.
Output: A sequential pattern set P
1: P = all patterns of length k with non-zero support
2: C = distinct support values of patterns in P are sorted in a non-decreasing order
3: Let u be the position of support value that is nearest to $n/2$ in C
4: Set maximal support threshold $maxS = C[u]$
5: Set minimal support threshold $minS = 2$
6: **if** $u - f * |C| > 0$ **then**
7: $minS = C[u - f * |C|]$
8: **end if**
9: P = retain only patterns in P whose supports are within $[minS, maxS]$
10: Sort patterns in P in a non-increasing order of supports
11: **for** $i = 1$ to $|P|$ **do**
12: **for** $j = i + 1$ to $|P|$ **do**
13: **if** $EditDistance(i, j) < 2$ **then** delete the i-th pattern from P
14: **end if**
15: **end for**
16: **end for**
17: **if** NOT all edit distances among patterns are k **then**
18: Remove patterns from P whose edit distances to all other patterns are k
19: **end if**
20: **return** P

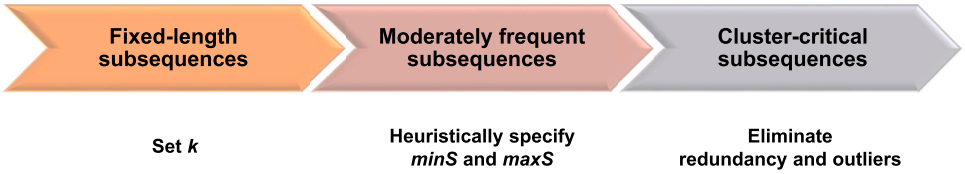

Fig. 2. The flowchart of mining cluster-critical sequential patterns.

- Thirdly, such a pattern cannot be too short. Only when the pattern is longer enough, then it can be expected this pattern is capable of explaining and characterizing its associated cluster. Hence, we need to impose some constraints on the length of candidate cluster-critical patterns as well.
- Finally, such patterns should be non-redundant with respect to each other. If too many redundant patterns are extracted from some clusters, then some bias would be introduced to favor the corresponding clusters during the subsequent clustering decision tree construction procedure. Hence, we also need to impose some constraints on the redundancy of identified pattern set.

Overall, cluster-critical sequential patterns are defined as patterns that can strongly characterize a specific cluster while remaining sufficiently distinct from patterns associated with other clusters. To ensure that this pattern set provides both meaningful and distinct explanations for their respective clusters, we impose several quantifiable (or actionable) constraints in two main aspects: (1) Characterizing a specific cluster: To ensure clear interpretability without being overly simplistic, the length of the pattern should be moderately small and fixed. Additionally, patterns should be frequent enough to be meaningful, so a minimum support threshold will be established. (2) Distinguishing from other clusters: The frequency of patterns cannot be too large, as this would reduce their discriminative power across clusters. Therefore, a maximum support threshold will be determined. Furthermore, patterns need to be diverse to avoid redundancy in the explanations and outlying patterns should be removed to avoid bias in the interpretation. To address this, we will employ a simple refinement process for redundancy and outlier removal.

3.2.2. Notations

Let $S = \{S_1, S_2, ..., S_n\}$ denote a sequential data set of n discrete sequences, where each S_i is an ordered list of items. Each item that composes the sequence is drawn from an alphabet. For a sequence S_i, we use $S_i[x, y]$ to denote a substring of S_i that starts from the x-th position and ends at the y-th position, where $y > x$ and $y - x + 1$ is the length of substring. In the context of frequent sequential pattern mining, one pattern typically refers to one substring or its generalization (i.e., subsequence).

3.2.3. Pattern mining algorithm

The cluster-critical sequential pattern mining algorithm under multiple constraints is given in Algorithm 1, and its concise flowchart is shown in Fig. 2. To avoid repetition among patterns generated by the inclusion relationship, we only consider patterns of a fixed length k (line 1). Since it is not an easy task to directly specify a proper support threshold across different data sets, here we choose the minimum and maximal support threshold in an adaptive manner.

We first sort distinct support values of patterns in the initial pattern set P in a non-decreasing order and the sorted support list is denoted by C (line 2). Note that support value of a pattern here is defined as the number of sequences in S that contains the pattern

Table 1
The contingency table for variables P_j and P_i. Each cell n_{st} (s and t are either 0 or 1) represents the number of cases that both $P_j = s$ and $P_i = t$ are true in the sequential data set.

	$P_i = 1$	$P_i = 0$	Total
$P_j = 1$	n_{11}	n_{10}	$n_{1.}$
$P_j = 0$	n_{01}	n_{00}	$n_{0.}$
Total	$n_{.1}$	$n_{.0}$	n

as a subsequence without the gap constraint. We set the maximal support threshold $maxS$ to be the value that is nearest to $n/2$ in C (lines 3∼4). The rationale for setting this threshold parameter in such a manner is as follows. First, there is at least two clusters in the data set if the target data set is clusterable. Second, it is reasonable to assume that the cluster size is balanced, i.e., the number of sequences in each cluster is approximately the same.

To specify the minimal support threshold, we employ the following heuristic method. We first set the initial value of $minS$ to be 2 (line 5). Since u is the position of $maxS$ in C, if $u - f * |C| > 0$, then it means that there are still at least f percentage of support values in C that are less than $maxS$, where $0 < f < 1$ is a user-specified parameter. To maintain a concise pattern set, we will raise the minimal support threshold from the default value of 2 to $C[u - f * |C|]$ (lines 6∼8). After specifying two support threshold parameters, we can further reduce the pattern set (line 9).

To reduce the redundancy among patterns, we calculate the edit distance between each pattern and remaining patterns. If their edit distance is less than 2, then we will delete one pattern from the pattern set, as shown in lines 10∼16. Note that in line 10, we sort patterns in P in a non-increasing order of supports so as to we will delete the pattern with a larger support in lines 11∼16. This is because patterns with smaller supports are more likely to be cluster-critical patterns after filtering patterns based on two support constraints.

We also include one step for removing those potentially outlying patterns. That is, if the edit distance between one pattern and all remaining patterns is k, then this pattern is probably an outlier such that it should not be included in the final pattern set. Note that if all patterns in P are such "outliers", then it means that we don't need to remove any patterns since they are all good representatives for different clusters in this case (lines 17∼19).

3.3. Constructing clustering decision trees

3.3.1. Split point evaluation

Based on the set of m sequential patterns $P = \{P_1, P_2, ..., P_m\}$ returned by Algorithm 1, we can transform the sequential dataset $S = \{S_1, S_2, ..., S_n\}$ into a new data set $B = \{B_1, B_2, ..., B_n\}$, where each B_i is composed of m feature values and each P_j is regarded as a feature. The j-th feature value of B_i is either 1 or 0, according to whether the i-th sample contains the j-th pattern as a subsequence without the gap constraint.

To build an unsupervised binary decision tree, we choose the presence status of each pattern as the candidate split point, i.e., $P_j = 1$. That is, each non-leaf node is divided into a left child node and a right child node, where the left/right child node is composed of sequences with $P_j = 1/0$. To assess the goodness of each possible split point, we regard each pattern/feature as a binary variable. If one split point can yield a good partition, then it is reasonable to expect that the corresponding pattern variable is positively correlated with most of other $m - 1$ pattern variables. Based on this observation, we assess split points in terms of p-values based on the chi-squared test, as elaborated below.

If $P_j = 1$ is the candidate split point at the current node, we will first calculate the chi-squared statistic for measuring the correlation between P_j and P_i, where $i \neq j$. In detail, we can construct the contingency table as shown in Table 1 (where we assume it is constructed at the root node).

Based on above notations, we can calculate the chi-squared statistic as follows:

$$\chi^2(P_j, P_i) = \sum_{s=0}^{1} \sum_{t=0}^{1} \frac{(n_{st} - e_{st})^2}{e_{st}}, \tag{1}$$

where e_{st} represents the expected frequency that is calculated as:

$$e_{st} = \frac{n_{s.} \cdot n_{.t}}{n}. \tag{2}$$

Note that when there is a $e_{st} < 5$, we will employ the Yates's correction in Equation (1) by replacing $(n_{st} - e_{st})^2$ with $(|n_{st} - e_{st}| - 0.5)^2$.

Note that each $\chi^2(P_j, P_i)$ is chi-squared random variable, we can obtain a p-value based on $\chi^2(P_j, P_i)$ and the degree of freedom $df = (2-1)(2-1) = 1$. To obtain a final p-value for assessing each candidate split point $P_j = 1$, we use the p-value combination method [33] to combine $m - 1$ p-values derived from $m - 1$ test statistics $\chi^2(P_j, P_1), ..., \chi^2(P_j, P_{j-1}), \chi^2(P_j, P_{j+1}), ..., \chi^2(P_j, P_m)$. Note that many p-value combination methods are available in the literature, e.g., Stouffer's Z-score method, Fisher's method, Pearson's method, Mudholkar's and George's method, and Tippett's method. All these methods can be employed to fulfill our task at hand. Here

Algorithm 2 Tree Construction.

Input: The sequential data set S, the pattern set P, the significance level α.
Output: An unsupervised clustering tree T

```
1: B = S(P)
2: h = 0
3: return T = CTNode(B, α)
4: function CTNODE(B, α)
5:     T = new node (B)
6:     if |B| ≤ 5 then return T
7:     end if
8:     Find the split point P_j = 1 with minimum p-value pval(P_j)
9:     h = h + 1
10:    if pval(P_j) > α/|P|^h then return T
11:    end if
12:    B_l, B_r = B(P_j = 1)
13:    if |B_l| ≥ 3 and |B_r| ≥ 3 then
14:        T.leftChild = CTNode(B_l, α)
15:        T.rightChild = CTNode(B_r, α)
16:    end if
17:    return T
18: end function
```

we use Stouffer's Z-score method in our algorithm and the effect of using different p-value combination methods will be empirically investigated in Section 4.6.

3.3.2. Tree construction

The unsupervised decision tree construction algorithm is given in Algorithm 2. Taking both the sequential data S and the pattern set P as the input, we will first transform S into a categorical data set B as described in the first paragraph of Section 3.3.1 (line 1).

Then, we call the function $CTNode(B, \alpha)$ to create a root node for the clustering tree (line 3). In the function of $CTNode(B, \alpha)$, if the number of samples in B is no larger than 5, we will not further divide it (lines 5∼ 7). Otherwise, we try to find the best split point with the minimum p-value (line 8).

Suppose $P_j = 1$ is the best split point, we will compare its p-value with the adjusted significance level to determine whether the corresponding partition is statistically significant. We will discuss why the adjusted significance level is $\alpha/|P|^h$ in Section 3.3.3. If the p-value cannot exceed the adjusted significance level, then we will not further divide the current node (lines 10∼11). Otherwise, we will utilize the split point $P_j = 1$ to divide samples in B into two subsets: B_l and B_r (line 12).

Note that in line 12, we will check if the size of B_l and B_r is large than 2. If one of two subsets is too small, i.e., the size is less than 3, then it is not meaningful to further divide the current node since it will produce clusters with only two samples. If both subsets are large enough, we will recursively call $CTNode(B_l, \alpha)$ and $CTNode(B_r, \alpha)$ to create two child nodes (lines 14∼15).

3.3.3. The adjusted significance level

Why do we need to adjust the significance level in Algorithm 2? This is mainly because we are tackling a multiple hypothesis testing issue. That is, each candidate split point evaluation issue corresponds to a single hypothesis testing problem. At each node, we need to consider $|P|$ such significance testing problems since the number of candidate split points equals to $|P|$. Hence, we need to conduct a multiple testing correction to control the Type-I error.

In the literature of multiple hypothesis testing, people typically tries to control the Family-Wise Error Rate (FWER), which is defined as the probability of making at least one Type-I error. The Bonferroni correction method [34] is generally employed to ensure that the FWER is less than the significance level α. The basic idea is to utilize an adjusted significance level, which is defined as ratio between α and the number of tested hypotheses. If we only reject those null hypotheses whose p-values are less than adjusted significance level, then it can be proved that the FWER is no larger than α.

In our context, at the root node, we have $|P|$ candidate split points (accordingly, $|P|$ significance tests), hence the adjusted significance level should be $\alpha/|P|$ (line 10). To achieve this, initially we set $h = 0$ (line 2) and h is increased by 1 (line 9). When we proceed further to grow the tree, at each node, we have to check $|P|$ split points as well. If we assume the tree structure is fixed, then the combination of h split points at all visited nodes so far corresponds to a composite null hypothesis. The number of possible composite null hypotheses is $|P|^h$. Therefore, α should be divided by $|P|^h$, where h will be increased by one once we try to evaluate all patterns in P to find the best split point for the current node.

3.3.4. An illustration of decision tree construction

Based on the set of cluster-critical sequential patterns mined in Section 3.2.1, we can leverage these patterns to construct a final decision tree in the manner described above for any given sequential data. This is achieved by selecting the optimal split pattern and applying a significance-based threshold for controlling the tree growth. As shown in Fig. 3, we demonstrate how the final decision tree is formed using a small but real sequential dataset (Activity). Fig. 3(a) presents the dataset, which is composed of 35 sequences (S1∼S35). In Fig. 3(d), the final decision tree partitions the dataset into two leaf nodes (representing two clusters): {S1∼S14} and {S15∼S35}.

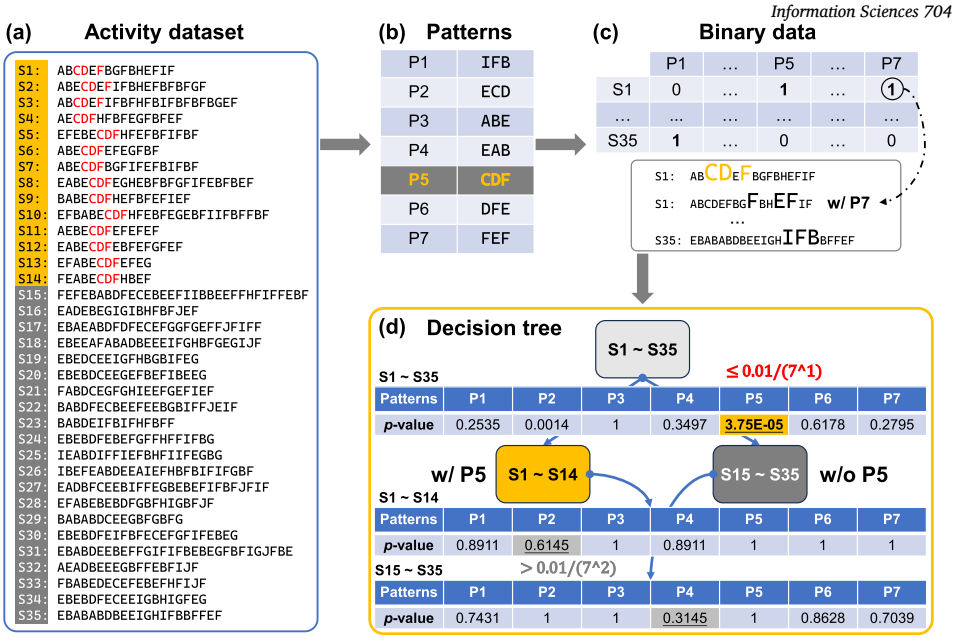

Fig. 3. Illustration of how to obtain the final decision tree from the Activity dataset.

The process is detailed as follows: Fig. 3(b) shows the seven mined cluster critical sequential patterns (P1~P7). In Fig. 3(c), for the convenience of referencing patterns used to build the decision tree, the data is represented in binary format, where each sequence is assigned a value of 0 or 1 based on the presence or absence of a pattern. For example, since S1 contains pattern P7, it is assigned a value of 1 in the corresponding column. In Fig. 3(d), we assess the *p*-value for each of these cluster-critical patterns as a potential split point. At the root node, P5 generates the smallest *p*-value and simultaneously meets our significance threshold, thus it is retained as the split point. This divides the dataset into a left child node (S1~S14), which contains P5, and a right child node (S15~S35), which does not. Recursively, at the node corresponding to S1~S14, the smallest *p*-value yielded by P2 is 0.6145, and at the node corresponding to S15~S35, the smallest *p*-value achieved by P4 is 0.3145. Since the *p*-values of P2 and P4 do not meet the significance level (set to 0.01/49), so the splits are halted at these two leaf nodes. The construction of the decision tree is completed in which only an individual pattern at the root node determines and explains how two clusters are formed.

3.3.5. Time complexity

The proposed SigISC method is composed of two subsequent algorithmic procedures: the extraction cluster-critical sequential patterns and the construction of clustering decision tree. In the next two paragraphs, we will provide the time complexity analysis on these two procedures respectively.

As presented in Algorithm 1, we need to first collect the support values of all patterns of length k. To generate all candidate patterns whose support values are non-zeros, we can scan each sequence from the left to right with a sliding window of length k. Let l denote the number of average length of sequences in S, then this step takes $O(nl)$ time. To count the support values of all candidate patterns in P, we can employ a dynamic programming procedure to fulfill this task, whose time complexity is $O(kl|P|)$. Sorting all distinct support values requires at most $O(|P|log|P|)$ time and the selection of two support thresholds needs $O(|P|)$ time. Finally, we need to check all pattern pairs to remove redundant ones, which requires at most $O(|P|^2k^2)$. Overall, the time complexity of the pattern mining step would be $O(nl + kl|P| + k|P| + |P|log|P| + |P| + |P|^2k^2)$, which can be approximately simplified to be $O(nl + klm + m^2k^2)$ under the assumption that $|P|$ will not be reduced too much after filtering patterns based on multiple constraints, where m is the number of final reported patterns.

The time complexity analysis of the clustering decision tree construction method is similar to that of supervised decision tree such as CART [35]. If the decision tree is assumed to be approximately balanced, then the average time complexity of the tree construction procedure would be $O(m^2nlogn)$, where the term $O(m^2)$ is the number of chi-squared tests during split point selection at each node.

4. Experiments

4.1. Data sets

The same 14 sequential data sets that have been used in [5,6] are employed for comparing different algorithms. Some important characteristics of these data sets are recorded in Table 2, where n represents the number of sequences, $|I|$ denotes the alphabet size, $minl$ ($maxl$) denotes the minimal (maximal) sequence length, and #Classes represents the number of classes/clusters.

Table 2
The main characteristics of 14 sequential data sets.

Dataset	Domain	n	$\|I\|$	$minl$	$maxl$	#Classes
Activity	Human Actions	35	10	12	43	2
Aslbu	Sign Language	424	250	2	54	7
Auslan2	Sign Language	200	16	2	18	10
Context	Context Data	240	94	22	246	5
Epitope	Bioinformatics	2392	20	9	21	2
Gene	Bioinformatics	2942	5	41	216	2
News	Natural Language	4976	27884	1	6779	5
Pioneer	Robotics	160	178	4	100	3
Question	Natural Language	1731	3612	4	29	2
Reuters	Natural Language	1010	6380	4	533	4
Robot	Robotics	4302	95	24	24	2
Skating	Human Actions	530	82	18	240	7
Unix	Command Logs	5472	1697	1	1400	4
Webkb	Natural Language	3667	7736	1	20628	3

4.2. Evaluation measures

For evaluating clustering quality, we employ three external evaluation measures: Purity, NMI (Normalized Mutual Information) and F1-score. Higher values of these metrics indicate better clustering quality. These metrics compare the predicted clusters $\pi = \{\pi_1, \pi_2, \cdots, \pi_L\}$ and the ground-truth clusters π^{GT}, and are defined as follows:

$$\text{Purity} = \frac{1}{n} \cdot \sum_{i}^{L} \max_{j} |\pi_i \cap \pi_j^{GT}| . \tag{3}$$

$$\text{NMI} = \frac{1}{2} \cdot \frac{\mathbf{H}(\pi) + \mathbf{H}(\pi^{GT}) - \mathbf{H}(\pi, \pi^{GT})}{\mathbf{H}(\pi) + \mathbf{H}(\pi^{GT})}, \tag{4}$$

where $\mathbf{H}(\pi)$ and $\mathbf{H}(\pi^{GT})$ denote the entropy of partition variable, while $\mathbf{H}(\pi, \pi^{GT})$ represents the joint entropy of these two partition variables.

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{5}$$

where

$$\text{Precision} = \frac{TP}{TP + FP}, \ \text{Recall} = \frac{TP}{TP + FN} . \tag{6}$$

In Equation (6), TP (True Positive) counts the number of pairs of sequences that are in the same cluster in both π^{GT} and π. FP (False Positive) counts the number of pairs of sequences that are in the same cluster in π but belong to different clusters in π^{GT}. FN (False Negative) counts the number of pairs of sequences that are in the same cluster in π^{GT} but are assigned to different clusters in π.

For evaluating the interpretability of tree-based clustering model, we use three commonly used structural metrics [27]: the number of leaf nodes in the tree ($\#Leaf$), the maximal depth of the tree ($Depth_{max}$) and the average depth of all leaf nodes ($Depth_{avg}$). Since one decision tree would possess better explainability if it is small and shallow. Hence, above three metrics can characterize the interpretability since $\#Leaf$ is associated with the tree size and $Depth_{max}$ and $Depth_{avg}$ quantify whether the tree is shallow or not. A decision tree that have smaller values in terms of these measures is expected to be more interpretable.

4.3. Baselines

To show the advantages of our algorithm, the following algorithms are included in the performance comparison. These algorithms can be further divided into three categories:

(1) Non-interpretable sequence clustering algorithms: HC [36], FB-LL [9], MCSC [17], k-Median [15], MinDL [16], kkmeans [37], RFSC [4], RSC [5]. We compare with these algorithms with respect to the clustering quality in terms of three external measures. The parameter settings for these methods are same to that used in [4,5].

(2) Interpretable sequence clustering algorithm [6]. It is the only tree-based interpretable sequence clustering algorithm in the literature. In the experiment, we use its default parameters.

(3) Interpretable clustering algorithms for vectorial data. Just as we have pointed out, we can obtain interpretable clustering result by first transforming sequences into vectors and then apply existing interpretable clustering methods for vectorial data. Hence, the following interpretable clustering algorithms with their default settings are employed: CUBT [38], IMM [28], SHA [27].

For our algorithm, its parameters are specified as follows: we set k, f and α to be 3, 0.3 and 0.01, respectively. The Stouffer's Z-score method is used as the p-value combination method and each individual p-value has a default weight of 1. All experiments

Table 3
Clustering quality comparison of SigISC, HC, FB-LL, MCSC, *k*-Median, MinDL, kkmeans, RFSC and RSC. '#HIT' denotes the number of datasets where the method performs best in at least two evaluation metrics.

Datasets	Evaluation	SigISC	HC	FB-LL	MCSC	*k*-Median	MinDL	kkmeans	RFSC	RSC
	Purity	**1**	0.600	0.643	0.686	0.695	0.600	0.720	0.700	0.797
Activity	NMI	**1**	0.038	0.129	0.118	0.231	0.078	0.225	0.248	0.324
	F1-score	**1**	0.652	0.613	0.537	0.620	0.515	0.612	0.664	0.669
	Purity	0.382	0.380	0.501	0.373	0.441	0.507	0.469	0.507	0.475
Aslbu	NMI	0.010	0.028	0.220	0.046	0.139	0.177	0.157	0.230	0.192
	F1-score	0.371	0.366	0.286	0.183	0.268	0.126	0.257	0.360	0.259
	Purity	0.260	0.195	0.309	0.275	0.316	0.295	0.321	0.347	**0.389**
Auslan2	NMI	0.277	0.159	0.336	0.221	0.315	0.245	0.316	0.312	**0.356**
	F1-score	0.254	0.173	0.246	0.153	0.251	0.193	**0.256**	0.209	0.236
	Purity	0.550	0.425	0.518	0.354	0.572	0.550	**0.610**	0.377	0.505
Context	NMI	0.370	0.466	0.407	0.071	0.515	0.300	**0.593**	0.201	0.418
	F1-score	0.389	0.487	0.444	0.235	0.523	0.211	**0.573**	0.352	0.417
	Purity	0.597	0.559	0.559	0.683	0.594	0.670	0.656	0.674	**0.685**
Epitope	NMI	0.081	0.044	0.096	0.093	0.054	0.060	**0.126**	0.105	0.082
	F1-score	0.495	0.671	0.635	0.585	0.550	0.277	0.582	0.568	**0.697**
	Purity	0.999	0.511	**1**	0.519	0.935	0.965	0.999	0.977	0.992
Gene	NMI	0.993	0.060	**0.994**	0.012	0.782	0.158	0.989	0.875	0.940
	F1-score	**0.999**	0.665	**0.999**	0.500	0.914	0.059	0.998	0.956	0.984
	Purity	0.267	0.210	0.269	0.241	0.284	**0.535**	0.462	0.266	0.271
News	NMI	0.112	0.002	0.106	0.007	0.036	**0.249**	0.226	0.023	0.023
	F1-score	0.329	0.253	0.329	0.218	0.263	0.256	**0.344**	0.282	0.214
	Purity	**0.888**	0.656	0.652	0.644	0.662	0.713	0.807	0.790	0.641
Pioneer	NMI	**0.614**	0.064	0.175	0.090	0.097	0.181	0.459	0.465	0.194
	F1-score	**0.720**	0.657	0.476	0.406	0.515	0.338	0.652	0.686	0.459
	Purity	0.578	0.518	0.518	**0.745**	0.604	0.570	0.560	0.656	0.653
Question	NMI	0.074	0.022	0.019	**0.295**	0.081	0.047	0.015	0.086	0.080
	F1-score	0.639	**0.666**	0.619	0.665	0.591	0.546	0.514	0.561	0.554
	Purity	0.461	0.255	0.321	0.347	0.448	0.287	**0.699**	0.528	0.468
Reuters	NMI	0.302	0.097	0.101	0.048	0.154	0.062	**0.482**	0.293	0.153
	F1-score	0.410	0.298	0.350	0.292	0.388	0.391	**0.582**	0.479	0.351
	Purity	**0.723**	0.515	0.544	0.637	0.557	0.535	0.618	0.595	0.635
Robot	NMI	**0.095**	0.046	0.035	0.056	0.017	0.019	0.068	0.032	0.065
	F1-score	0.309	**0.655**	0.592	0.538	0.521	0.522	0.564	0.522	0.544
	Purity	0.223	0.183	0.222	0.221	0.220	**0.232**	0.228	0.221	**0.232**
Skating	NMI	0.037	0.029	0.041	0.023	0.041	**0.112**	0.032	0.039	0.038
	F1-score	0.134	**0.252**	0.193	0.150	0.181	0.246	0.164	0.169	0.187
	Purity	0.566	0.444	0.491	0.464	0.479	**0.756**	0.451	0.610	0.515
Unix	NMI	0.168	0.004	0.094	0.031	0.055	0.224	0.018	**0.235**	0.103
	F1-score	0.398	**0.484**	0.422	0.308	0.371	0.227	0.438	0.430	0.352
	Purity	0.515	0.444	0.443	0.448	0.479	**0.723**	0.473	0.491	0.551
Webkb	NMI	0.092	0.020	0.055	0.019	0.072	**0.199**	0.126	0.078	0.128
	F1-score	0.401	0.421	0.455	0.378	0.426	0.341	0.403	**0.473**	0.264
	#HIT	**3**	0	1	1	0	**3**	2	1	2
	Purity	0.572 (3.929)	0.421 (8.393)	0.499 (5.929)	0.474 (6.179)	0.520 (5.214)	0.567 (4.286)	**0.577** (3.750)	0.553 (3.929)	0.558 **(3.393)**
Mean (Average Rank)	NMI	**0.302** (3.857)	0.077 (7.714)	0.201 (4.536)	0.081 (7.071)	0.185 (5.179)	0.151 (5.143)	0.274 **(3.786)**	0.230 (3.893)	0.221 (3.821)
	F1-score	0.489 (4.143)	0.479 **(3.714)**	0.476 (3.786)	0.368 (7.143)	0.456 (5.000)	0.303 (7.764)	**0.496** (4.000)	0.479 (4.321)	0.442 (5.429)

were conducted on a PC with Intel(R) Core(TM) i7-10700F 2.90 GHz and 16 GB Memory. The implementation codes for SigISC and other interpretable comparison methods are available at: https://github.com/hulianyu/SigISC.

4.4. Clustering quality comparison

In Table 3, the performance comparison results among SigISC and conventional sequence clustering algorithms in terms of Purity, NMI and F1-score are recorded. From this table, it can be observed that our method can beat other algorithms with respect to the

Table 4
Clustering quality comparison of SigISC and some interpretable clustering algorithms. Except for ISCT, other algorithms need a feature/pattern set as input. Note that the CUBT algorithm has two variants: $CUBT^{Ham}$ and $CUBT^{MI}$.

Datasets	Evaluation	SigISC		$CUBT^{Ham}$		$CUBT^{MI}$		IMM		SHA		ISCT
		FSet 1	FSet 2	FSet 1	FSet 2	FSet 1	FSet 2	FSet 1	FSet 2	FSet 1	FSet 2	
Activity	Purity	0.686	**1**	0.600	**1**	0.600	**1**	0.718	0.906	0.683	0.970	0.994
	NMI	0.082	**1**	0	**1**	0	**1**	0.163	0.778	0.111	0.925	0.973
	F1-score	0.601	**1**	0.672	**1**	0.672	**1**	0.640	0.895	0.619	0.964	0.989
Aslbu	Purity	0.422	0.382	0.469	0.382	0.469	0.382	0.446	0.382	0.447	0.373	0.491
	NMI	0.109	0.010	0.141	0.006	0.141	0.006	0.125	0.006	0.125	0	0.218
	F1-score	0.412	0.371	0.337	0.371	0.337	0.371	0.348	0.371	0.345	0.373	0.317
Auslan2	Purity	0.340	0.260	0.235	0.295	0.270	0.285	0.352	0.376	0.351	**0.377**	0.348
	NMI	0.357	0.277	0.214	0.258	0.247	0.232	0.344	**0.362**	0.343	**0.362**	0.331
	F1-score	0.266	0.254	0.265	0.261	0.269	0.258	0.259	**0.290**	0.259	**0.290**	0.265
Context	Purity	**0.608**	0.550	0.554	0.525	0.479	0.467	0.540	0.520	0.550	0.516	0.569
	NMI	0.446	0.370	0.439	0.319	0.260	0.274	0.417	0.310	0.430	0.308	**0.557**
	F1-score	0.378	0.389	0.459	0.404	0.378	0.335	0.461	0.391	0.463	0.388	**0.551**
Epitope	Purity	**0.710**	0.597	0.632	0.617	0.632	0.620	0.559	0.562	0.559	0.563	0.627
	NMI	0.084	0.081	0.104	0.080	0.098	0.074	0.073	0.068	0.073	0.067	**0.114**
	F1-score	0.171	0.495	0.544	0.527	0.544	0.523	0.610	**0.612**	0.610	**0.612**	0.587
Gene	Purity	**1**	0.999	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	NMI	0.988	0.993	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	F1-score	0.998	0.999	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
News	Purity	**0.293**	0.267	0.264	0.264	0.266	0.251	0.266	0.259	0.260	0.260	0.254
	NMI	0.082	**0.112**	0.033	0.065	0.039	0.051	0.037	0.053	0.028	0.054	0.020
	F1-score	0.286	0.329	0.291	**0.330**	0.290	0.329	0.291	0.324	0.291	0.324	0.262
Pioneer	Purity	0.869	**0.888**	0.869	0.869	0.719	0.719	0.766	0.837	0.755	0.829	0.798
	NMI	0.512	**0.614**	0.541	0.541	0.241	0.321	0.389	0.492	0.366	0.482	0.549
	F1-score	0.581	**0.720**	0.709	0.709	0.403	0.512	0.626	0.671	0.612	0.663	0.688
Question	Purity	0.525	0.578	0.568	0.604	0.582	0.578	0.527	0.542	0.530	0.541	**0.656**
	NMI	0.001	0.074	0.076	0.095	0.076	0.055	0.011	0.021	0.015	0.019	**0.122**
	F1-score	0.608	0.639	0.565	0.610	0.561	0.638	0.645	**0.655**	0.645	**0.655**	0.572
Reuters	Purity	0.368	0.461	0.348	0.417	0.348	0.463	0.414	0.407	0.411	0.407	**0.579**
	NMI	0.124	0.302	0.109	0.179	0.109	0.240	0.143	0.166	0.144	0.163	**0.392**
	F1-score	0.365	0.410	0.398	0.392	0.398	0.408	0.360	0.399	0.364	0.397	**0.501**
Robot	Purity	0.653	**0.723**	0.652	0.705	0.670	0.707	0.564	0.580	0.564	0.570	0.639
	NMI	0.049	**0.095**	0.041	0.056	0.055	0.064	0.013	0.037	0.013	0.031	0.071
	F1-score	0.139	0.309	0.401	0.337	0.293	0.434	0.561	0.608	0.561	**0.610**	0.551
Skating	Purity	**0.262**	0.223	0.208	0.219	0.208	0.213	0.224	0.218	0.220	0.223	0.208
	NMI	**0.066**	0.037	0.021	0.044	0.026	0.028	0.035	0.030	0.033	0.033	0.041
	F1-score	0.094	0.134	0.174	0.164	0.187	0.153	0.157	0.158	0.157	0.159	**0.224**
Unix	Purity	0.546	**0.566**	**0.566**	0.528	0.537	0.528	0.532	0.523	0.530	0.520	0.506
	NMI	0.168	0.168	**0.191**	0.142	0.150	0.140	0.140	0.124	0.143	0.124	0.110
	F1-score	0.398	0.398	0.397	0.404	0.401	0.382	**0.420**	0.409	0.416	0.408	0.385
Webkb	Purity	0.505	0.515	0.502	**0.545**	0.489	0.527	0.443	0.460	0.444	0.460	0.502
	NMI	0.088	0.092	0.079	**0.097**	0.080	**0.097**	0.042	0.034	0.042	0.036	0.094
	F1-score	0.425	0.401	0.354	0.345	0.395	0.343	0.386	0.433	0.388	0.430	**0.452**
Mean (Average Rank)	Purity	0.556 (4.607)	0.572 **(4.464)**	0.533 (5.929)	0.569 (4.857)	0.519 (6.607)	0.553 (6.250)	0.525 (6.571)	0.541 (7.143)	0.522 (7.179)	0.543 (7.107)	**0.584** (5.286)
	NMI	0.226 (5.464)	0.302 (4.107)	0.213 (6.179)	0.277 (4.500)	0.180 (7.000)	0.256 (6.357)	0.209 (7.107)	0.249 (7.071)	0.205 (7.179)	0.257 (7.321)	**0.328 (3.714)**
	F1-score	0.409 (8.357)	0.489 (6.179)	0.469 (6.500)	0.490 (5.500)	0.438 (7.250)	0.478 (7.143)	0.483 (6.036)	0.515 **(3.750)**	0.481 (6.107)	0.520 (3.929)	**0.525** (5.250)

overall NMI. Although our method cannot outperform other algorithms in terms of Purity and F1-score, its performance is at least comparable to these existing methods. Notably, in terms of achieving complete dominance in certain datasets (indicated by #HIT), SigISC slightly outperforms SOTA methods, such as kkmeans and RSC. Hence, we can conclude that our method is competitive to current non-interpretable sequence clustering algorithms with respect to the clustering quality.

In Table 4, we compare our method with some interpretable clustering algorithms. Among these algorithms, ISCT is specially developed for handling sequential data. Other remaining algorithms require to transform sequences into vectors in order to conduct

Table 5
The selection of $minS$ and $maxS$ for each dataset, utilized for mining the frequent patterns set (Fset). FSet 1 is composed of top 70% frequent subsequences of length k, and FSet 2 is reported by our pattern mining algorithm (lines 1∼8 in Algorithm 1).

		Activity	Aslbu	Auslan2	Context	Epitope	Gene	News	Pioneer	Question	Reuters	Robot	Skating	Unix	Webkb
$minS$	Fset 1	5	4	7	30	16	1549	23	8	11	14	52	30	81	34
	Fset 2	10	9	24	44	36	2	57	20	29	32	137	69	240	89
$maxS$	Fset 1	20	14	71	755	274	6201	330	61	181	111	1719	467	5126	1302
	Fset 2	16	14	71	114	274	1462	330	61	181	111	1719	252	2487	1302

Table 6
The performance comparison of interpretable clustering algorithms in terms of $\#Leaf$.

Dataset	SigISC	CUBT^{Ham}	CUBT^{MI}	IMM	SHA	ISCT
Activity	**2**	**2**	**2**	**2**	**2**	**2**
Aslbu	**2**	**2**	**2**	**2**	1	7
Auslan2	5	6	**4**	10	10	9.7
Context	6	**5**	8	**5**	**5**	**5**
Epitope	13	7	11	**2**	**2**	**2**
Gene	**2**	**2**	**2**	**2**	**2**	**2**
News	7	6	8	**5**	**5**	**5**
Pioneer	4	**3**	**3**	**3**	**3**	**3**
Question	3	7	5	**2**	**2**	**2**
Reuters	5	6	6	**4**	**4**	**4**
Robot	12	12	11	**2**	**2**	**2**
Skating	11	9	**7**	**7**	**7**	**7**
Unix	15	15	11	**4**	**4**	**4**
Webkb	9	18	7	**3**	**3**	**3**
Mean ($\#Leaf$)	6.857	7.143	6.214	3.786	**3.714**	4.121

the cluster analysis. To show the advantage of our pattern mining algorithm on extracting cluster-critical features, in addition to the feature set derived from our algorithm, we also employ the widely used feature extraction method based on frequent patterns (FSet 1 in the table). The corresponding $minS$ and $maxS$ support thresholds for these two pattern sets on each data set are summarized in Table 5. From Table 4, we have the following important observations.

Firstly, our method and ISCT generally have better performance than other algorithms. This is easy to understand since these two algorithms are designed for clustering discrete sequences. Secondly, ISCT is better than our method because ISCT takes the number of ground-truth clusters as input, while our method can determine the number of clusters adaptively. Finally, it is clearly visible that FSet 2 is better than FSet 1, indicating that our pattern mining algorithm is able to extract cluster-critical patterns.

4.5. Interpretability comparison

We also compare our algorithm with those interpretable clustering algorithms with respect to three interpretability metrics: $\#Leaf$, $Depth_{max}$ and $Depth_{avg}$. We use the FSet 2 in Table 4 derived from our pattern mining algorithm (Algorithm 1) as the input for two CUBT variants, IMM and SHA. The detailed experimental results are recorded in Table 6 and Table 7, respectively. The comparison on runtime is provided in Table 8 (with all runtimes including the time spent on pattern mining). We can observe that our SigISC is nearly the most time-efficient method, similar to IMM and SHA, and requires only 0.4% of the total runtime of the ISCT method.

From Table 6, we can observe that both our algorithm and CUBT variants construct a tree with more leaf nodes than other interpretable clustering algorithms. This is easily explained, as IMM, SHA, and ISCT use the number of ground-truth clusters as input, which strictly controls the number of leaf nodes. Consequently, the depth-based metrics of IMM, SHA, and ISCT shown in Table 7 are also constrained under this setting, as the number of branch nodes in the binary tree is determined by the number of leaf nodes. IMM, SHA, and ISCT are not applicable in the following scenarios: (1) When the number of clusters in the data is unknown, and we lack a priori knowledge to set the input number of clusters; (2) When the ground truth implies that the data contains only a single cluster, but we still wish to explore its potential heterogeneity. In this latter scenario, as shown in an interesting experiment in Table 9, only our SigISC and CUBT variants can be utilized.

From Table 7, it can be observed that the depth-based metrics in the tree constructed by our algorithm are comparable to those of the two CUBT variants, even though we did not use any additional measures to control the tree depth. In contrast, CUBT by default imposes a trivial depth parameter that constrains the depth of the tree (i.e., $Depth_{max} \leq 7$).

Overall, based on the experimental results with respect to interpretability, our method does not outperform existing solutions that requires the number of ground-truth clusters as input. However, our algorithm demonstrates better overall performance than two CUBT variants, which are similar to our method in that the number of clusters can be determined adaptively. Especially in the experiments on single-cluster data shown in Table 9, our method can potentially more accurately assess the tendency of homogeneity in the data, thus generating fewer splits and leaf nodes. This advantage is largely due to the significance-based criteria we employ,

Table 7
The performance comparison of interpretable clustering algorithms in terms of $Depth_{max}$ and $Depth_{avg}$.

Dataset	SigISC	CUBT^{Ham}	CUBT^{MI}	IMM	SHA	ISCT
$Depth_{max}$						
Activity	**1**	**1**	**1**	**1**	**1**	**1**
Aslbu	1	1	1	1	0	6
Auslan2	**3**	**3**	**3**	4	4	8.7
Context	**3**	**3**	6	3.3	3.2	4
Epitope	5	5	5	**1**	**1**	**1**
Gene	**1**	**1**	**1**	**1**	**1**	**1**
News	4	**3**	4	3.96	3.9	4
Pioneer	3	**2**	**2**	**2**	**2**	**2**
Question	2	5	3	**1**	**1**	**1**
Reuters	4	3	4	2.98	**2.96**	3
Robot	8	7	6	**1**	**1**	**1**
Skating	4	4	4	3.92	**3.84**	6
Unix	6	7	6	2.74	**2.42**	3
Webkb	5	7	6	**2**	**2**	**2**
Mean ($Depth_{max}$)	3.571	3.714	3.714	2.207	**2.094**	3.121
$Depth_{avg}$						
Activity	**1**	**1**	**1**	**1**	**1**	**1**
Aslbu	1	1	1	1	0	3.857
Auslan2	2.6	2.667	**2.25**	3.416	3.476	5.4
Context	2.667	**2.4**	4	2.576	2.548	2.8
Epitope	3.846	3.286	4	**1**	**1**	**1**
Gene	**1**	**1**	**1**	**1**	**1**	**1**
News	3	**2.667**	3.25	2.792	2.76	2.8
Pioneer	2.25	**1.667**	**1.667**	**1.667**	**1.667**	**1.667**
Question	1.667	3.429	2.4	**1**	**1**	**1**
Reuters	2.8	2.667	2.833	2.245	**2.24**	2.25
Robot	4.667	5.333	4.273	**1**	**1**	**1**
Skating	3.545	3.444	3	3.017	**2.98**	3.857
Unix	4.533	5.333	4.091	2.185	**2.105**	2.25
Webkb	3.444	5	3.857	**1.667**	**1.667**	**1.667**
Mean ($Depth_{avg}$)	2.716	2.921	2.759	1.826	**1.746**	2.253

Table 8
The running time comparison among interpretable clustering algorithms.

Datasets	SigISC	CUBT^{Ham}	CUBT^{MI}	IMM	SHA	ISCT
Activity	**0.07**	0.74	0.74	**0.07**	0.09	3.72
Aslbu	**0.08**	0.66	0.44	**0.08**	0.10	25.58
Auslan2	**0.07**	0.96	0.91	**0.07**	0.10	29.07
Context	0.32	4.20	4.35	**0.21**	0.24	37.50
Epitope	0.30	35.60	8.42	**0.25**	0.26	17.31
Gene	**0.56**	12.91	1.66	**0.56**	0.57	353.50
News	3.47	263.90	20.95	**3.38**	3.41	1671.32
Pioneer	**0.08**	0.63	0.61	**0.08**	0.10	9.46
Question	**0.15**	18.88	2.05	**0.15**	0.18	8.38
Reuters	0.66	12.01	4.65	**0.60**	0.63	106.22
Robot	1.35	250.74	83.00	**0.87**	0.89	63.49
Skating	0.29	7.38	9.82	**0.20**	0.23	47.13
Unix	1.54	318.49	52.39	**1.11**	1.15	362.73
Webkb	2.89	145.86	37.76	**2.69**	2.71	332.61
Total	11.83	1072.94	227.73	**10.31**	10.66	3068.01

where our method primarily relies on the significance level to control the formation of the final decision tree, rather than the trivial stopping criteria used in the CUBT method.

As shown in Fig. 4, which extends the $\#Leaf$ results of SigISC from Table 6, the leaf nodes are categorized based on the stopping criteria used. From the figure, we observe that, for all datasets (except Webkb), the leaf nodes are predominantly determined by the significance-based stopping criterion. Moreover, since each branch node follows the same criterion, this suggests that, in practice, the significance-based stopping criterion plays a key role in the growth of the decision tree, with fewer cases influenced by the trivial stopping criterion.

To intuitively compare interpretability, we display the decision tree structures of SigISC and CUBT^{MI} (the better-performing CUBT variant), as neither requires specifying the number of clusters. A representative example on the Webkb dataset is shown in Fig. 5, enabling a fair comparison of tree structures since they use the same number of unique splitting patterns. Although SigISC generates more clusters than CUBT^{MI} ($\#Leaf$: 9 vs. 7), it still produces shallower tree structures than CUBT^{MI} ($Depth$: 5 vs. 6). From

Table 9
The $\#Leaf$ comparison of single-cluster data between SigISC and CUBT methods. Here, n^{single} denotes the number of sequences in the single-cluster data. The specific single-cluster data is selected from one of the clusters in each dataset by sorting all clusters in a descending order of n^{single} and choosing the cluster in the second-largest position. Note that the cluster-critical sequential patterns used for the single-cluster data remain consistent with its corresponding dataset.

Datasets	n^{single}	SigISC	CUBT$^{\text{Ham}}$	CUBT$^{\text{MI}}$
Auslan2	20	2	**1**	**1**
Context	50	**2**	**2**	4
Epitope	1056	**3**	6	6
News	996	**1**	3	3
Pioneer	42	**1**	**1**	**1**
Question	835	**3**	6	5
Reuters	253	**1**	3	3
Robot	2097	8	14	**7**
Skating	88	5	4	**3**
Unix	1590	**3**	11	10
Webkb	1116	**6**	15	11
Mean ($\#Leaf$)		**3.182**	6	4.909

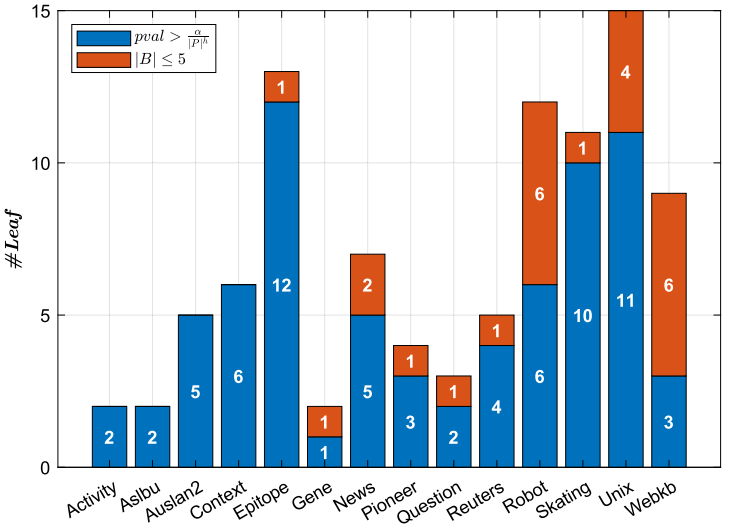

Fig. 4. Bar plot showing the number of leaf nodes determined by two different stopping criteria of SigISC across different datasets. The stopping criteria for each node are as follows: the trivial criterion of $|B| \leq 5$, and the significance-based criterion of $|B| > 5 \wedge p_{\text{val}} > \frac{\alpha}{|P|^h}$ (abbreviated in the legend).

a morphological perspective, CUBT$^{\text{MI}}$ generates the deepest tree structure for its corresponding number of leaf nodes, continuously splitting along the rightmost nodes. In contrast, SigISC introduces multiple branch nodes from both sides of their parent nodes at the first and second layers, allowing for further compression and resulting in shallower tree structures.

4.6. The effect of p-value combination method

To evaluate the impact of p-value combination method on the clustering results of our method, we conduct a series of experiments. We compare the Stouffer's Z-score method, Fisher's method, Pearson's method, Mudholkar's and George's method, and Tippett's method in terms of clustering quality and interpretability.

In Fig. 6 (orange sections), we compared five p-value combination methods in terms of Purity, NMI and F1-score. From this figure, it can be seen that in most datasets the Fisher's method performs the best with respect to both Purity and NMI and the Pearson's method is the best performer in terms of F1-score.

In Fig. 6 (blue sections), we compared five methods in terms of $\#Leaf$, $Depth_{max}$ and $Depth_{avg}$. As shown in this figure, $\#Leaf$, $Depth_{max}$ and $Depth_{avg}$ reported by the Fisher's method and Tippett's method are typically larger than other methods. Meanwhile, the Pearson's method can yield the minimal clustering tree with respect to these metrics.

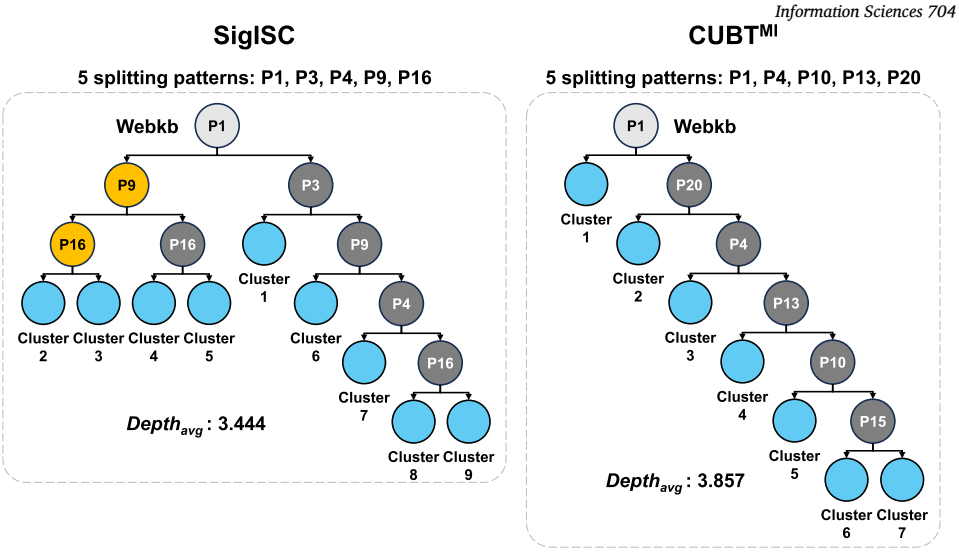

Fig. 5. Decision trees for SigISC and CUBT^{MI} on the Webkb dataset.

Overall, the use of different *p*-value combination methods can affect the performance of SigISC with respect to both clustering quality and interpretability. None of these *p*-value combination methods can achieve both the best clustering quality and interpretability. To make a trade-off, the Stouffer's Z-score method seems to be a good choice.

5. Conclusion

In this paper, we introduce SigISC, a tree-based interpretable clustering method for discrete sequences based on significance testing. This method is characterized by a novel approach to cluster-critical sequential pattern extraction and a significance-based procedure for evaluating and selecting split points during decision tree construction in an adaptive manner. It allows us to construct a clustering tree that is explainable in terms of both its output and the split decisions. The effectiveness of the proposed method is verified through a series of experiments on real sequential data sets.

However, our algorithm still has several limitations. (1) The pattern extraction method remains highly heuristic, and its underlying theoretical principles need further exploration. Inevitably, there exist better methods for extracting cluster-critical patterns, particularly the combination of different pattern lengths, which remains an open problem. In addition to selecting suitable pattern extraction parameters, incorporating additional constraints to search for cluster-critical patterns may require the use of a multi-objective optimization approach. (2) The *p*-value combination method is highly sensitive to the number of patterns, making it challenging to specify an appropriate significance level parameter that can adapt to varied datasets. Future work could explore alternative approaches that do not rely on *p*-value combination but instead assess multiple *p*-values from different perspectives or conditions, jointly optimizing them to provide more robust interpretations. (3) The *p*-values of splitting patterns are mainly used for assessing nodes in a top-down manner. Related topics, such as using *p*-values for pruning arbitrary decision trees, could present interesting and unexplored challenges.

CRediT authorship contribution statement

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

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.

Acknowledgement

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

Data availability

Data will be made available on request.

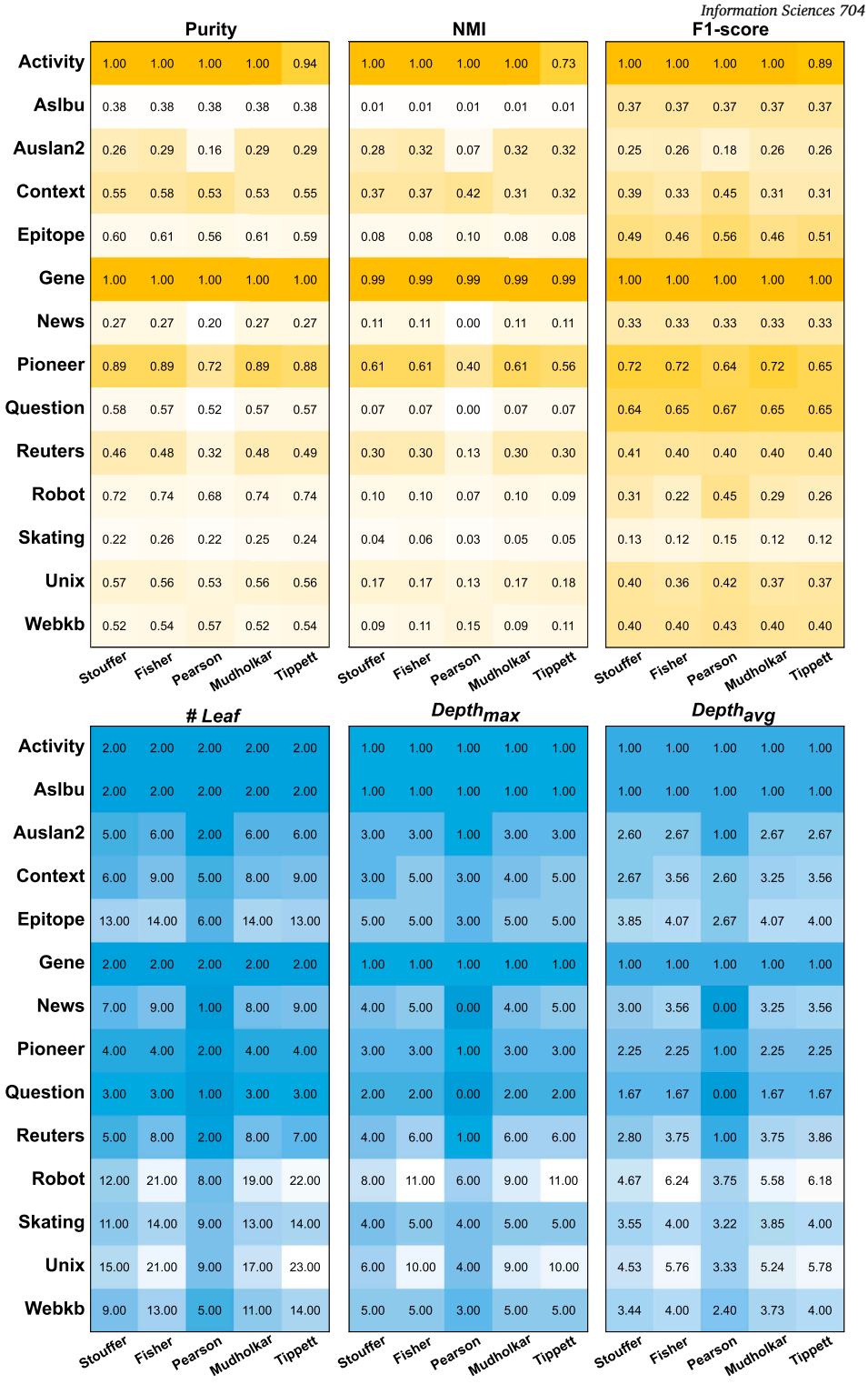

Fig. 6. The effect of p-value combination method on SigISC in terms of clustering quality and interpretability.

References

[1] G.J. Oyewole, G.A. Thopil, Data clustering: application and trends, Artif. Intell. Rev. 56 (7) (2023) 6439–6475.
[2] Q. Zou, G. Lin, X. Jiang, X. Liu, X. Zeng, Sequence clustering in bioinformatics: an empirical study, Brief. Bioinform. 21 (1) (2020) 1–10.

[3] J. De Weerdt, S. Vanden Broucke, J. Vanthienen, B. Baesens, Active trace clustering for improved process discovery, IEEE Trans. Knowl. Data Eng. 25 (12) (2013) 2708–2720.
[4] M. Jiang, J. Wang, L. Hu, Z. He, Random forest clustering for discrete sequences, Pattern Recognit. Lett. 174 (2023) 145–151.
[5] M. Jiang, L. Hu, X. Han, Y. Zhou, Z. He, A randomized algorithm for clustering discrete sequences, Pattern Recognit. (2024) 110388.
[6] J. Dong, X. Yang, M. Jiang, L. Hu, Z. He, Interpretable sequence clustering, Inf. Sci. 689 (2025) 121453.
[7] 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.
[8] M. Gabidolla, M.Á. Carreira-Perpiñán, Optimal interpretable clustering using oblique decision trees, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 400–410.
[9] V. Guralnik, G. Karypis, A scalable algorithm for clustering sequential data, in: Proceedings 2001 IEEE International Conference on Data Mining, IEEE, 2001, pp. 179–186.
[10] 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.
[11] 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.
[12] 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.
[13] M. Ramoni, P. Sebastiani, P. Cohen, Bayesian clustering by dynamics, Mach. Learn. 47 (2002) 91–121.
[14] S.-J. Oh, J.-Y. Kim, A hierarchical clustering algorithm for categorical sequence data, Inf. Process. Lett. 91 (3) (2004) 135–140.
[15] 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.
[16] Y. Chen, P. Xu, L. Ren, Sequence synopsis: optimize visual summary of temporal event data, IEEE Trans. Vis. Comput. Graph. 24 (1) (2018) 45–55.
[17] 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) (2014) 2339–2353.
[18] V. Melnykov, Clickclust: an r package for model-based clustering of categorical sequences, J. Stat. Softw. 74 (2016) 1–34.
[19] P. Smyth, Clustering sequences with hidden Markov models, in: Proceedings of the 10th International Conference on Neural Information Processing Systems, 1996, pp. 648–654.
[20] E. Carrizosa, K. Kurishchenko, A. Marín, D.R. Morales, On clustering and interpreting with rules by means of mathematical optimization, Comput. Oper. Res. 154 (2023) 106180.
[21] B. Liu, Y. Xia, P.S. Yu, Clustering through decision tree construction, in: Proceedings of the Ninth International Conference on Information and Knowledge Management, 2000, pp. 20–29.
[22] J. Chen, Y. Chang, B. Hobbs, P. Castaldi, M. Cho, E. Silverman, J. Dy, Interpretable clustering via discriminative rectangle mixture model, in: IEEE 16th International Conference on Data Mining, IEEE, 2016, pp. 823–828.
[23] L. Chen, C. Zhong, Z. Zhang, Explanation of clustering result based on multi-objective optimization, PLoS ONE 18 (10) (2023) e0292960.
[24] 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, vol. 36, 2022, pp. 7309–7316.
[25] C. Lawless, O. Gunluk, Cluster explanation via polyhedral descriptions, in: International Conference on Machine Learning, in: PMLR, 2023, pp. 18652–18666.
[26] L. Hu, M. Jiang, J. Dong, X. Liu, Z. He, Interpretable clustering: a survey, arXiv preprint arXiv:2409.00743, 2024.
[27] E. Laber, L. Murtinho, F. Oliveira, Shallow decision trees for explainable k-means clustering, Pattern Recognit. 137 (2023) 109239.
[28] M. Moshkovitz, S. Dasgupta, C. Rashtchian, N. Frost, Explainable k-means and k-medians clustering, in: International Conference on Machine Learning, in: PMLR, 2020, pp. 7055–7065.
[29] D. Bertsimas, A. Orfanoudaki, H. Wiberg, Interpretable clustering: an optimization approach, Mach. Learn. 110 (1) (2021) 89–138.
[30] L. Hu, M. Jiang, J. Dong, X. Liu, Z. He, Interpretable categorical data clustering via hypothesis testing, Pattern Recognit. 162 (2025) 111364.
[31] F.R. Guo, R.D. Shah, Rank-transformed subsampling: inference for multiple data splitting and exchangeable p-values, J. R. Stat. Soc., Ser. B, Stat. Methodol. 87 (1) (2025) 256–286.
[32] Y. Xie, X. Jia, S. Shekhar, H. Bao, X. Zhou, Significant dbscan+: statistically robust density-based clustering, ACM Trans. Intell. Syst. Technol. 12 (5) (2021) 1–26.
[33] L.-C. Chang, H.-M. Lin, E. Sibille, G.C. Tseng, Meta-analysis methods for combining multiple expression profiles: comparisons, statistical characterization and an application guideline, BMC Bioinform. 14 (2013) 1–15.
[34] X. Cui, T. Dickhaus, Y. Ding, J. Hsu, Handbook of Multiple Comparisons, CRC Press, 2022.
[35] H.M. Sani, C. Lei, D. Neagu, Computational complexity analysis of decision tree algorithms, in: Artificial Intelligence XXXV: 38th SGAI International Conference on Artificial Intelligence, Springer, 2018, pp. 191–197.
[36] 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, SIAM, 2009, pp. 401–412.
[37] K. Xu, L. Chen, S. Wang, A multi-view kernel clustering framework for categorical sequences, Expert Syst. Appl. 197 (2022) 116637.
[38] B. Ghattas, P. Michel, L. Boyer, Clustering nominal data using unsupervised binary decision trees: comparisons with the state of the art methods, Pattern Recognit. 67 (2017) 177–185.