Pattern Recognition 162 (2025) 111364

Contents lists available at ScienceDirect

Pattern Recognition

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

Interpretable categorical data clustering via hypothesis testing

Lianyu Hu [a], Mudi Jiang [a], Junjie Dong [a], Xinying Liu [a], Zengyou He [a,b,*]

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

ARTICLE INFO

Keywords:
Interpretable clustering
Categorical data
Binary decision trees
Statistical hypothesis test
Splitting criteria

ABSTRACT

Categorical data clustering algorithms are extensively investigated but it is still challenging to explain or understand their output clusters. Hence, it is highly demanded to develop interpretable clustering algorithms that are capable of explaining categorical clusters in terms of decision trees or rules. However, most existing interpretable clustering algorithms focus on numeric data and the development of corresponding algorithms for categorical data is still in the infant stage. In this paper, we tackle the problem of interpretable categorical data clustering by growing a binary decision tree in an unsupervised manner. We formulate the candidate split evaluation issue as a multiple hypothesis testing problem, where the null hypothesis posits that there is no association between each attribute and the candidate split. Subsequently, the p-value for each candidate split is calculated by aggregating individual test statistics from all attributes. Thereafter, a significance-based splitting criteria is established. This involves choosing an optimal split with the smallest p-value for tree growth and using a significance level to stop the non-significant split. Extensive experimental results on real-world data sets demonstrate that our algorithm achieves comparable performance in terms of cluster quality and explainability relative to those of state-of-the-art counterparts.

1. Introduction

In the realm of machine learning, cluster analysis is an essential technique for data exploration. Its main objective is to divide heterogeneous data sets into multiple groups, each with similar characteristics. This methodology is extensively employed across various disciplines, including biology, medicine, astronomy, and social sciences. The challenge escalates when dealing with categorical data [1], where heterogeneity is prevalent both between and within attributes due to their discrete nature, making categorical data clustering a particularly complex task.

Numerous clustering algorithms have been developed for processing categorical data [2], with their results often utilized in various subsequent applications [3]. Mirroring their numerical data counterparts, these categorical data clustering algorithms typically focus on optimizing a specific target function [4]. However, there is a noticeable lack in the interpretability and explainability of these clustering outcomes, posing decision-making challenges for users when employing these algorithms.

Recently, the development of interpretable clustering algorithms has received much attention. In general, existing research efforts towards this direction can be categorized into two classes, as outlined in [5]: (1) Pre-modeling (interpretability), emphasizing that the clustering procedure should be controllable and comprehensible for end-users. (2) Post-modeling (explainability), which entails providing rational explanations for clustering outcomes, regardless of the algorithm's inherent opacity. In terms of specific interpretable models, decision trees are widely adopted [6–8], with a particular focus on binary trees [9–11]. However, most of these methods are designed for numerical data, and to our knowledge, only the clustering method in [10] is specifically tailored for categorical data.

The algorithm in [10] follows a three-step process: growing, pruning, and joining, extending the CUBT framework [9] to derive the final decision tree. It employs splitting criteria and dissimilarity measures suitable for categorical variables, with each leaf node corresponding to a specific cluster. Despite of its success on providing an interpretable clustering result in terms of a decision tree, the tree construction procedure in [10] involves multiple parameters, and the decision to further split a node lacks a clear statistical interpretation.

This paper introduces an interpretable clustering algorithm for categorical data based on hypothesis testing. The primary motivations for developing this method are as follows. First, the interpretability of existing categorical data clustering algorithms is inadequate, potentially leading to clustering results that do not fulfill the requisite trustworthiness in high-stakes applications. Second, while the algorithm in [10] employs the decision tree as an interpretable model, it still lacks the

* Corresponding author at: School of Software, Dalian University of Technology, Dalian, 116024, China.
E-mail addresses: hly4ml@gmail.com (L. Hu), zyhe@dlut.edu.cn (Z. He).

https://doi.org/10.1016/j.patcog.2025.111364
Received 13 March 2024; Received in revised form 5 August 2024; Accepted 12 January 2025
Available online 19 January 2025

interpretability on the tree growing process, i.e., why samples in a leaf node should be further divided into two clusters. Finally, all current decision tree-based interpretable clustering algorithms, whether for numeric or categorical data, do not assess each cluster and guide the tree construction from a significance testing perspective.

To our knowledge, there is still a lack of research efforts that tackle the interpretable categorical data clustering issue from a significance testing aspect. Meanwhile, in response to the challenges posed by existing methods, we propose SigTree, a clustering approach for categorical data that offers stronger interpretability with respect to both the tree construction process and output clusters. This method conducts the clustering process by directly building a binary decision tree. For each node of the tree, the candidate splits are treated as binary variables, where the outcome of each binary split is defined based on whether samples at this node contain a specific category of a target attribute. Concurrently, excluding the aforementioned target attribute, other remaining attributes are considered as pseudo-class variables, leading to multiple pseudo-class variables corresponding to different attributes.

Under the null hypothesis of no association between a candidate split and each pseudo-class variable, chi-squared tests are used for association testing. If the samples at the current node do not exhibit a significant clustering structure, then all candidate splits will show no association with most pseudo-class variables. Hence, during the tree growing process, the optimal split for the current node is selected, and its p-value is calculated via the combination of multiple chi-squared test statistics. If this p-value is less than a predetermined significance level, the samples at the current node are divided into two subsequent child nodes (clusters). The growth of the decision tree stops when all leaf nodes cannot produce any further statistically significant divisions.

In summary, the main contributions of this work are as follows:

- This is the first study to incorporate hypothesis testing into the interpretable categorical data clustering issue.
- This method ensures that each node split in the binary decision tree is both controllable and statistically reliable, and the decision-making process is comprehensible based on the selected categories.
- Extensive experiments on real data sets demonstrate that this approach achieves better performance compared to the de facto standard k-modes method, without the need to specify the number of clusters, and also maintains competitive performance relative to other interpretable clustering methods.

The structure of this paper is outlined as follows: Section 2 reviews methods most relevant to our study. Section 3 provides a detailed explanation of our proposed method. Section 4 displays our experimental results. Finally, Section 5 offers a summary of the paper.

2. Related work

To set the context for our work on developing an interpretable clustering method for categorical data, this section introduces relevant algorithms in categorical data clustering and concentrates on the most commonly used interpretable clustering methods.

2.1. Categorical data clustering

In categorical data clustering, algorithms typically consist of two essential components: an objective function and the corresponding search process to optimize this function. With a specific objective function in place, search processes generally proceed either in an iterative manner, as seen in methods like k-modes [12], or in a hierarchical way, exemplified by algorithms like the agglomerative approach ROCK [13] and the divisive approach DHCC [14]. The objective functions used in categorical data clustering can be broadly divided into two types:

(1) Objective function based on pairwise comparison: It requires computing the pairwise distance between two samples. The Hamming distance is extensively used for categorical data [15], and other distance measures are thoroughly reviewed in the literature [16]. Additionally, pairwise similarity-based representation learning methods have recently gained considerable attention [17–19].

(2) Objective function based on a set-based measure: This type of objective function evaluates the heterogeneity of one or more sets of samples without using the pairwise similarity. Entropy, a classical concept for measuring the uncertainty of random variables, is widely utilized in categorical data clustering [20,21]. It is naturally suited for assessing the compactness of a single cluster [22] or for measuring the similarity between clusters [23].

However, current categorical data clustering methods seldom integrate hypothesis testing or lack a focus on enhancing algorithm's interpretability. From the viewpoint of hypothesis testing, the DV method [24] sequentially extracts statistically significant clusters. In terms of interpretability, only the algorithm in [10] is available in the literature, which constructs a binary decision tree to yield an interpretable clustering result.

2.2. Interpretable clustering

Many works have concentrated on clarifying the process of clustering, emphasizing the use of simple and explicit rules for easy explanation. Accompanying this trend [25], explainable clustering methods have attracted substantial interest (particularly explainable k-means clustering [8,11,26]), due to the requirement on an interpretable clustering model in various real life applications [7]. In addition to the commonly used binary decision trees for describing how each cluster is formed, other initiatives in this direction have included, but are not confined to, the use of logical formulas [27], hyper-rectangles [28], hypercubes [29] and polytopes [30] for illustrating the generation of clusters.

Our work is focused on constructing unsupervised decision trees [6,9,10], where pseudo-class labels used for tree construction are directly derived from the attribute values. This method markedly contrasts with those algorithms in [5,7,8,11,26], where decision trees are built based on cluster assignments generated by other clustering algorithms. Moreover, in contrast to those methods based on a joint optimization of both tree construction and cluster formation [7,31], our method is simple and quick since the clustering procedure is quite similar to standard decision tree construction method.

In comparison to the interpretable categorical data clustering method in [10], our approach employs a hypothesis testing procedure to select optimal splits. As a result, the tree construction process in our method is also interpretable since a node will be further divided only the split is statistically significant in terms of p-values.

3. Methods

3.1. Preliminaries

Given a categorical data set $\mathcal{X} = \{x_1, \ldots, x_N\}$ consisting of N samples and M attributes, the feature value x_{im} of the ith sample on the mth attribute is one of the Q_m categories in the set $\{A_1^{(m)}, \ldots, A_{Q_m}^{(m)}\}$. The goal of interpretable categorical data clustering is to divide $\mathcal{X}$ into clusters (subsets) $\{\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(K)}\}$, where each cluster $\mathcal{X}^{(k)}$ can be described via simple rules.

In the context of binary decision trees, each non-leaf node D is divided into two child nodes: a left child node D_{Left} and a right child node D_{Right}. Each category $A_q^{(m)}$ can be employed as the candidate splitting point, where samples in the node D are allocated to D_{Left} or D_{Right} according to whether their corresponding feature values equal to $A_q^{(m)}$ or not. Hence, we can define a binary random variable $S_q^{(m)}$ for each candidate split: $S_q^{(m)} = 1$ if $x_{im} = A_q^{(m)}$ and $S_q^{(m)} = 0$ otherwise.

Similarly, for the mth attribute, we can define a random variable C_m that takes values in the set $\{A_1^{(m)}, \ldots, A_{Q_m}^{(m)}\}$.

To construct the interpretable clustering tree, we employ a category-based decision rule to split each non-leaf node. This requires computing the p-value for each candidate split variable $S_q^{(m')}$ against the other $M-1$ attribute variables, denoted as $pval(S_q^{(m')}, C')$, where $C' = \{C_m | 1 \le m \le M, m! = m'\}$. The candidate split $S_{q^*}^{(m^*)}$ that can yield the smallest p-value will be chosen to divide the node into two child nodes.

Algorithm 1: SigTree

Input : $\mathcal{X}$ – categorical data set

α^* – significance level

Output: An unsupervised decision tree D

```
1  D ← BinaryTree(X, α*)
2  Function BinaryTree(X, α*):
3      D ← create node (samples = X, pval = Null, cat = Null,
         Left = Null, Right = Null)
4      if |X| ≤ 5 then
5          return D
6      pval*, S_{q*}^{(m*)} ← OptimalSplit(X)
7      if pval* > α* then
8          return D
9      X_Left, X_Right ← X(S_{q*}^{(m*)})
10     if |X_Left| ≥ 3 and |X_Right| ≥ 3 then
11         D.pval ← pval*
           // Save the category indexed by the keys
12         D.cat ← A_{q*}^{(m*)}
           // Apply BinaryTree to the child nodes
13         D.Left ← BinaryTree(X_Left, α*)
14         D.Right ← BinaryTree(X_Right, α*)
15     return D
16 Function OptimalSplit(X):
       // Splitting criteria
17     foreach m' ∈ [1, ..., M], q ∈ [1, ..., Q_m] do
18         S_{q*}^{(m*)} ← argmin_{m'!=m,q} pval(S_q^{(m')}, C')
19     return pval*, S_{q*}^{(m*)}
```

3.2. The SigTree algorithm

Based on the concepts and symbols introduced in Section 3.1, we will present the SigTree algorithm. Given any categorical data set as input, SigTree($\mathcal{X}$) produces an unsupervised decision tree D. This tree delineates the clustering process: its leaf nodes represent the final clusters, and each non-leaf node specifies the decision rule for splitting. The comprehensive procedure of the SigTree algorithm is presented in Algorithm 1.

To store the necessary information for tree construction (`BinaryTree`), each non-leaf node in the SigTree algorithm includes five essential components (as outlined in Line 3): 'samples', 'pval', 'cat', 'Left', and 'Right'. These components respectively store the samples contained in the current node, the p-value of the optimal split, the corresponding optimal decision category, and the samples in the subsequent left and right child nodes that either contain or do not contain the optimal category. As leaf nodes cannot be further divided, they only retain the 'samples' component.

The tree growth procedure in SigTree is mainly determined by two criteria: the splitting criteria (`OptimalSplit` outlined in Lines 17∼19), and the stopping criteria. The detailed explanation of splitting criteria will be provided in Section 3.3. As for the stopping criteria, we will clarify them here. We employ two types of stopping criteria: one is the significance-based stopping criteria (Line 7), which we have proposed and detailed in Section 3.4, and the other is the trivial stopping criteria, akin to those used in conventional decision trees.

Our trivial stopping criteria requires that each leaf node in the SigTree algorithm forms a cluster with at least three samples. This is achieved through two operations: First, we avoid splitting nodes with no more than five samples (Line 4). Second, we preclude the split that would lead to the generation of a child node with less than three samples (Line 10).

3.3. Splitting criteria

3.3.1. Split testing issue

To determine the statistical significance of each candidate split, we tackle this issue from the viewpoint of multiple hypothesis testing. For a candidate split variable based on the qth category of the m'-th attribute, we have $M-1$ pseudo-class variables corresponding to the remaining attributes, i.e., C_m in which $m! = m'$. Hence, we need to handle $M-1$ association testing issues. The null hypothesis H_{0m} posits that $S_q^{(m')}$ is independent of the mth attribute C_m, while the alternative hypothesis H_{1m} claims the presence of an association. The collective null hypothesis, which encompasses all $M-1$ individual hypotheses, is expressed as:

$$H_0 : \bigcap_{m!=m'} H_{0m},$$

and this is contrasted against the global alternative hypothesis:

$$H_1 : \bigcup_{m!=m'} H_{1m}, \tag{1}$$

where the global alternative hypothesis implies that at least one H_{0m} is false. In our context, the global alternative hypothesis H_1 suggests that the candidate split is associated with at least one attribute. Associations across multiple attributes can cumulatively strengthen the conclusion that the candidate split is significant.

Under each null hypothesis H_{0m}, we utilize the chi-squared test for association testing. Then, we combine the test statistics from all individual chi-squared tests to derive the overall p-value, i.e., $pval(S_q^{(m')}, C')$. This p-value assesses the statistical significance against the global null hypothesis H_0, with a smaller value indicating a more significant split. Thus, the optimal split for each node is identified by the smallest p-value.

3.3.2. Chi-squared test

Note that the classic CHAID method [32] has employed the chi-squared test as the split criteria for growing a decision tree, our approach exhibits several distinct aspects: (1) We utilize binary splits in tree construction, more apt for interpretable clustering, in contrast to the multi-way splits in CHAID. (2) The chi-squared test in the CHAID algorithm is applied in a supervised setting. That is, the true class variable is given in the training stage. However, in our problem, the true class label information is missing and the chi-squared test is conducted in an unsupervised manner.

In our application of the chi-squared test as the splitting criteria, we begin by calculating a chi-squared statistic for each pair $\langle S_q^{(m')}, C_m \rangle$. Here, the categorical variable C_m consists of Q_m categories $\{A_1^{(m)}, \ldots,$

$A^{(m)}_{Q_m}\}$. We then construct a $2 \times Q_m$ contingency table to represent the observed co-occurrence frequencies as shown below:

	$A^{(m)}_1$	…	$A^{(m)}_{Q_m}$	Total
$S^{(m')}_q = 1$	$N^{(m',q,m)}_{11}$	$N^{(m',q,m)}_{1j}$	$N^{(m',q,m)}_{1Q_m}$	$N^{(m',q,m)}_{1\cdot}$
$S^{(m')}_q = 0$	$N^{(m',q,m)}_{01}$	$N^{(m',q,m)}_{0j}$	$N^{(m',q,m)}_{0Q_m}$	$N^{(m',q,m)}_{0\cdot}$
Total	$N^{(m',q,m)}_{\cdot 1}$	$N^{(m',q,m)}_{\cdot j}$	$N^{(m',q,m)}_{\cdot Q_m}$	$N^{(m',q,m)}$

(2)

Following the above contingency table, the expected frequencies are calculated as:

$$E^{(m',q,m)}_{sj} = \frac{N^{(m',q,m)}_{s\cdot} \cdot N^{(m',q,m)}_{\cdot j}}{N^{(m',q,m)}}. \tag{3}$$

For each pair $\langle S^{(m')}_q, C_m \rangle$, we compute the chi-squared statistic:

$$\chi^2(S^{(m')}_q, C_m) = \sum_{s=0}^{1} \sum_{j=1}^{Q_m} \frac{(N^{(m',q,m)}_{sj} - E^{(m',q,m)}_{sj})^2}{E^{(m',q,m)}_{sj}}. \tag{4}$$

These individual chi-squared statistics are summed to yield a collective test statistic for the global null hypothesis H_0:

$$\chi^2(S^{(m')}_q) = \sum_{m!=m'} \chi^2(S^{(m')}_q, C_m). \tag{5}$$

The corresponding degrees of freedom are calculated as:

$$df(S^{(m')}_q) = \sum_{m!=m'} Q_m. \tag{6}$$

Finally, the analytical p-value, i.e., $pval(S^{(m')}_q, C')$, is derived from the collective chi-squared statistic, utilizing the chi-squared cumulative distribution function with the calculated degrees of freedom under the assumption of independence [1] across all H_{0m} studies.

In addition, when assessing splits involving attributes that have only two categories ($Q_m = 2$), i.e., in cases of 2×2 contingency tables, special attention is required to avoid overestimating the chi-squared value. In such scenarios, the Yates's correction is employed if any expected frequency $E^{(m',q,m)}_{sj}$ in the 2×2 contingency table falls below 5. The chi-squared statistic in Eq. (4) is adjusted accordingly:

$$\chi^2(S^{(m')}_q, C_m) = \sum_{s=0}^{1} \sum_{j=1}^{2} \frac{(|N^{(m',q,m)}_{sj} - E^{(m',q,m)}_{sj}| - 0.5)^2}{E^{(m',q,m)}_{sj}}. \tag{7}$$

3.4. Significance-based stopping criteria

Developing effective stopping criteria is crucial to ensure that the tree growth procedure neither underfits nor overfits the data. Our algorithm tackles this challenge through hypothesis testing by comparing the optimal split p-value against the significance level threshold (α^* in Line 7). This stopping criterion notably differs from trivial or conventional criteria, which typically rely on heuristic rules. The treatment of node splitting issue as a significance testing problem offers a statistically rigorous way to govern the splitting process.

Specifically, the selection of an optimal split involves comparisons among all candidate splits, and α^* is determined using a multiple-comparison correction method. We employ the Bonferroni correction [33] to control the Family-Wise Error Rate (FWER) for the optimal split. Nodes that produce an optimal split with $pval^*$ less than or equal to α^* are considered for subdivision. Since the number of comparisons for splits may vary for each node, we apply the constant $|\mathcal{Q}|$ uniformly across all nodes to ensure stringent control. $|\mathcal{Q}|$ represents the total number of categories across all attributes in the categorical data set. The adjusted significance level α^* is thus calculated as:

$$\alpha^* = \frac{\alpha}{|\mathcal{Q}|}, \tag{8}$$

where α is the standard significance level parameter (typically 0.01 or 0.05) in single hypothesis testing.

3.5. Time complexity analysis

Here we provide an analysis on the average-case time complexity of the SigTree algorithm. Let T_N denotes the average number of basic operations required by SigTree to finish the clustering process over N categorical samples. When the sample size is 5 or fewer, only one basic operation (Line 4) is required:

$$T_N = 1, \text{if } N \le 5. \tag{9}$$

The calculation of T_N involves two parts. The first part, $\texttt{OptimalSplit}_N$, corresponds to the cost required for assessing all candidate splits to find the optimal one. The second part, accounts for the average cost of recursively calling SigTree on two child nodes. Under the assumption that the number of samples in the left child node is equally likely to be any number h ($1 \le h \le N-1$), we have:

$$T_N = \texttt{OptimalSplit}_N + \frac{2}{N-1} \sum_{h=1}^{N-1} T_h, \tag{10}$$

where the summation term accounts for the average number of basic operations across all $N-1$ potential split scenarios.

$\texttt{OptimalSplit}_N$ is mainly the cost of scanning the data set to collect the cell frequencies of contingency tables. Each candidate split $S^{(m')}_q$ (where q ranges from 1 to $Q_{m'}$) constructs a $2 \times Q_m$ contingency table with the mth attribute, and the $Q_{m'}$ possible splits lead to $2Q_{m'}Q_m$ cells. Therefore, when it is aggregated across all attributes, $\texttt{OptimalSplit}_N$ can be calculated as follows:

$$\begin{aligned} \texttt{OptimalSplit}_N &= N \cdot 2 \sum_{m'=1}^{M} \sum_{m!=m'} Q_{m'} Q_m \\ &= N \cdot Y, \end{aligned} \tag{11}$$

where $Y = 2\sum_{m'=1}^{M} \sum_{m!=m'} Q_{m'} Q_m$ is a fixed constant for a given categorical data set.

Then, we can simplify the recursive relation in Eq. (10) as follows:

$$T_N = NY + \frac{2}{N-1} \sum_{h=1}^{N-1} T_h, \text{ for } N > 1.$$

To multiply both sides by $N-1$, we obtain

$$(N-1)T_N = (N-1)NY + 2 \sum_{h=1}^{N-1} T_h, \text{ for } N > 1. \tag{12}$$

If we replace N by $N-1$, then we have

$$(N-2)T_{N-1} = (N-2)(N-1)Y + 2 \sum_{h=1}^{N-2} T_h, \text{ for } N > 2. \tag{13}$$

Now, we subtract Eq. (13) from Eq. (12) to derive the following recurrence relation:

$$NT_N = (N+1)T_{N-1} + 2NY, \text{ for } N > 1. \tag{14}$$

Theorem 1. *$T_N = 2Y(N+1)\mathcal{H}_N - 2YN$ where $\mathcal{H}_N = 1 + \frac{1}{2} + \cdots + \frac{1}{N}$ is a Harmonic number.*

Proof. To multiply both sides of Eq. (14) by $\frac{1}{N(N+1)}$, we obtain

$$\frac{T_N}{N+1} = \frac{T_{N-1}}{N} + \frac{2Y}{N+1}.$$

If we let $S_N = \frac{T_N}{N+1}$, then we have

$$S_N = S_{N-1} + \frac{2Y}{N+1}.$$

By unfolding the above recurrence, we can obtain the following equation:

$$S_N = 2Y \sum_{h=1}^{N} \frac{1}{h+1}. \tag{15}$$

Now, we express and rewrite $\sum_{h=1}^{N} \frac{1}{h+1}$ as follows:

$$\begin{aligned}\sum_{h=1}^{N} \frac{1}{h+1} &= \sum_{1\le h\le N} \frac{1}{h+1} = \sum_{1\le h-1\le N} \frac{1}{h} \\ &= \sum_{2\le h\le N+1} \frac{1}{h} \\ &= \Big(\sum_{1\le h\le N} \frac{1}{h}\Big) - \frac{1}{1} + \frac{1}{N+1} \\ &= \mathcal{H}_N - \frac{N}{N+1}\,.\end{aligned} \tag{16}$$

Therefore, we have

$$\begin{aligned}T_N = (N+1)S_N &= 2Y(N+1)(\mathcal{H}_N - \frac{N}{N+1}) \\ &= 2Y(N+1)\mathcal{H}_N - 2YN\,.\end{aligned} \quad \square \tag{17}$$

Finally, the average-case time complexity of SigTree with N samples as input is $\mathcal{O}(T_N) = \mathcal{O}(2Y(N+1)\mathcal{H}_N - 2YN) = \mathcal{O}(2Y(N+1)\mathcal{H}_N) = \mathcal{O}(2Y(N+1)\mathcal{H}_N) = \mathcal{O}(YN\mathcal{H}_N)$. Since $\mathcal{H}_N < 1 + \ln N$, the final time complexity of SigTree in the average-case is $\mathcal{O}(YN\ln N)$.

4. Results

In this section, we conduct a thorough evaluation of our SigTree algorithm on eighteen real-world categorical data sets, focusing on both clustering quality (Section 4.3) and explainability (Section 4.4).

4.1. Comparison methods and parameter settings

The primary goal of our experiments is to empirically demonstrate the superiority of SigTree over two counterpart clustering methods that respectively utilize hypothesis testing and unsupervised decision trees, namely DV [24] and the extended CUBT algorithms [10].

In a more broader context, SigTree is compared with both classic algorithms [12,20] and state-of-the-art (SOTA) methods [17,19] in categorical data clustering. Recognizing that most current interpretable clustering algorithms, especially explainable k-means methods, are tailored for numerical data, we apply one-hot encoding to convert each categorical sample into a numerical representation. On the transformed numeric data, the clustering results of leading explainable k-means methods [8,11,26] can be obtained to make a comparison with SigTree.

The brief descriptions and parameter settings of competing algorithms in the performance comparison are summarized as follows:

- DV [24]: This algorithm produces consistent clustering result in each run. It automatically determines the number of clusters through the iterative extraction of statistically significant clusters one by one.
- CUBT [10]: It offers two variants for categorical data: CUBT^{Ham} and CUBT^{MI}. Both variants follow the same joining process, building upon the same maximal growth tree, denoted as CUBT_{max}. They differ in the measure used in the pruning stage: the former uses Hamming distance and the latter employs mutual information. We use the default parameters as provided in the original code by the authors.[1] These parameters are set as follows: the maximum tree depth `lp=7`, the minimal size of each leaf node `minsize=10` and the minimal size of each non-leaf node `minsplit=20`.
- Classic categorical data clustering methods: the k-modes method [12] and the entropy-based method [20]. In k-modes, K samples are randomly chosen from the data set to serve as the initial modes. The entropy-based method employs a Monte Carlo search procedure, to derive locally optimal K clusters, starting with all samples initially placed in a single cluster. Hence, we may obtain different clustering results in different runs of these two methods.

[1] https://www.i2~m.univ-amu.fr/perso/badih.ghattas/cubt.php

Table 1
The properties of 18 UCI categorical data sets.

Data set	*Abbr.*	N	M	$\lvert\mathcal{Q}\rvert$	K
Lenses	Ls	24	4	9	3
Lung Cancer	Lc	32	56	159	3
Soybean (Small)	So	47	21	58	4
Zoo	Zo	101	16	36	7
Promoter Sequences	Ps	106	57	228	2
Hayes-Roth	Hr	132	4	15	3
Lymphography	Ly	148	18	59	4
Heart Disease	Hd	303	13	57	5
Solar Flare	Sf	323	9	25	6
Primary Tumor	Pt	339	17	42	21
Dermatology	De	366	33	129	6
House Votes	Hv	435	16	48	2
Balance Scale	Bs	625	4	20	3
Credit Approval	Ca	690	9	45	2
Breast Cancer	Bc	699	9	90	2
Mammographic Mass	Mm	824	4	18	2
Tic-Tac-Toe	Tt	958	9	27	2
Car Evaluation	Ce	1728	6	21	4

- SOTA categorical data clustering methods: We include two recently proposed methods in the comparison: CDE [19] and CDCDR [17]. For CDE, the default parameters in [19] were used. For CDCDR, we selected non-default options that demonstrated superior performance in the experiment of [17]: Spectral Embedding is used as the 'Graph Embedding Method' and the joint operation is chosen as the 'Integration Operation'.
- SOTA explainable k-means methods: These methods approximate k-means clustering by constructing top-down binary decision trees with K leaves. They utilize axis-aligned cuts of reference centers, each employing a unique strategy: Iterative Mistake Minimization (IMM)[2] [11], Random Threshold (RDM) [26], and ExShallow (SHA)[3] [8].

The SigTree algorithm, along with the code for generating all experimental results, is available at: https://github.com/hulianyu/SigTree. In a specific scenario of the SigTree algorithm, if the $pval^*$ of the best candidate split at the root node exceeds α^* (as outlined in Line 7), the algorithm is compelled to retain this initial optimal split, thereby finalizing two clusters. In our performance comparison, for each algorithm that cannot output consistent clustering result in each run (except for SigTree, DV, and CUBT), 50 independent runs are conducted on each data set to report the average results. The number of clusters K for these algorithms is specified to be the ground-truth cluster number of each data set. All experiments are conducted on an Intel i7-10700F@2.90 GHz personal computer with 16G RAM.

4.2. Data sets and performance metrics

Table 1 presents the properties of 18 real-world categorical data sets. The notations used here are consistent with those in the previous sections. These data sets, characterized by categorical features,[4] have been downloaded from the UCI Machine Learning Repository [34].

For evaluating clustering quality, we employ two widely-used external validation metrics [35]: Purity and F-score. These metrics evaluate the clustering results by comparing the set of predicted clusters $\pi = \{\pi_1, \ldots, \pi_{\hat{K}}\}$ with the set of ground-truth clusters $\pi^* = \{\pi_1^*, \ldots, \pi_K^*\}$. Higher values of these metrics indicate better clustering quality.

The Purity is defined as follows:

$$\text{Purity} = \frac{1}{N}\sum_{k=1}^{\hat{K}} \max_{k'\in\{1,\ldots,K\}} |\pi_k \cap \pi_{k'}^*|, \tag{18}$$

[2] https://github.com/navefr/ExKMC
[3] https://github.com/lmurtinho/ShallowTree
[4] https://archive.ics.uci.edu/datasets?FeatureTypes=Categorical

Table 2
The clustering quality comparison in terms of Purity and F-score for all competing algorithms. When DV fails to identify any statistically significant clusters on one data set, its clustering outcome is represented as '—'.

Metric	Data set	SigTree	DV	CUBT^{Ham}	CUBT^{MI}	k-modes	Entropy	CDE	CDCDR	IMM	RDM	SHA
Purity	Ls	0.625	—	0.625	0.625	0.682	0.686	0.648	0.625	0.643	**0.703**	0.666
	Lc	0.531	—	0.500	0.500	0.533	**0.611**	0.549	0.561	0.523	0.499	0.591
	So	**1**	**1**	0.574	0.787	0.882	0.914	0.935	0.940	0.948	0.863	**1**
	Zo	0.871	**0.950**	0.733	0.832	0.831	0.876	0.872	0.871	0.888	0.810	0.917
	Ps	**0.811**	—	0.802	0.802	0.554	0.750	0.737	0.751	0.682	0.647	0.786
	Hr	0.485	—	**0.500**	**0.500**	0.412	0.443	0.481	0.423	0.475	0.498	0.485
	Ly	0.554	**0.791**	0.662	0.757	0.623	0.756	0.742	0.730	0.706	0.656	0.733
	Hd	0.568	**0.657**	0.571	0.541	0.588	0.597	0.600	0.587	0.564	0.559	0.562
	Sf	**0.728**	—	0.533	0.687	0.544	0.550	0.536	0.577	0.518	0.492	0.526
	Pt	0.395	—	0.327	0.295	0.430	**0.471**	0.463	0.444	0.419	0.387	0.439
	De	0.842	0.497	0.790	0.650	0.715	0.792	0.841	0.816	0.841	0.673	**0.887**
	Hv	0.917	0.892	**0.926**	0.894	0.866	0.870	0.855	0.857	0.848	0.822	0.848
	Bs	0.590	—	0.560	0.547	0.513	0.550	0.579	**0.622**	0.570	0.548	0.564
	Ca	0.558	—	0.726	0.726	**0.748**	0.644	0.650	0.632	0.721	0.571	0.561
	Bc	0.931	0.951	0.924	0.924	0.919	0.964	0.966	**0.973**	0.883	0.838	0.883
	Mm	**0.826**	—	0.816	0.814	0.818	0.761	0.780	0.817	0.788	0.753	0.780
	Tt	**0.793**	—	0.678	0.678	0.653	0.653	0.653	0.653	0.653	0.653	0.653
	Ce	0.700	—	0.700	0.700	0.701	0.704	0.702	0.700	0.708	**0.749**	0.711
Mean		**0.707**	—	0.664	0.681	0.667	0.700	0.700	0.699	0.688	0.651	0.700
Average Rank		**4.5**	—	5.86	5.86	6.17	**4.5**	4.67	4.97	5.83	7.67	4.97
F-score	Ls	**0.498**	—	0.443	0.443	0.478	0.460	0.400	0.332	0.391	0.482	0.461
	Lc	**0.497**	—	0.471	0.471	0.436	0.486	0.436	0.470	0.411	0.396	0.467
	So	**1**	**1**	0.667	0.844	0.826	0.877	0.913	0.928	0.919	0.787	**1**
	Zo	0.887	**0.964**	0.793	0.916	0.705	0.705	0.765	0.790	0.772	0.630	0.796
	Ps	0.530	—	0.593	0.593	0.506	**0.669**	0.634	0.641	0.609	0.593	0.665
	Hr	0.375	—	0.330	0.350	0.353	0.369	**0.422**	0.348	0.368	0.386	0.375
	Ly	**0.651**	0.268	0.379	0.537	0.397	0.471	0.456	0.505	0.430	0.441	0.432
	Hd	0.400	0.132	**0.507**	0.297	0.393	0.368	0.390	0.412	0.363	0.384	0.348
	Sf	**0.637**	—	0.380	0.453	0.386	0.381	0.397	0.420	0.367	0.350	0.358
	Pt	**0.319**	—	0.167	0.173	0.186	0.182	0.195	0.179	0.204	0.170	0.169
	De	**0.844**	0.443	0.790	0.526	0.606	0.677	0.751	0.730	0.733	0.569	0.819
	Hv	0.512	0.685	0.749	0.658	0.773	**0.779**	0.774	0.778	0.748	0.730	0.748
	Bs	**0.557**	—	0.245	0.266	0.410	0.425	0.466	0.450	0.458	0.451	0.455
	Ca	0.668	—	0.396	0.315	0.642	0.596	0.606	**0.678**	0.664	0.572	0.565
	Bc	0.747	0.817	0.798	0.786	0.871	0.936	0.940	**0.952**	0.803	0.767	0.803
	Mm	0.609	—	0.323	0.479	**0.703**	0.666	0.696	0.702	0.669	0.655	0.665
	Tt	0.155	—	0.297	0.382	0.538	**0.562**	0.544	0.535	0.538	0.546	0.543
	Ce	**0.552**	—	0.475	0.220	0.408	0.387	0.393	0.379	0.398	0.498	0.400
Mean		**0.580**	—	0.489	0.484	0.534	0.555	0.565	0.568	0.547	0.523	0.559
Average Rank		**3.75**	—	6.92	7.03	5.83	4.83	4.39	4.61	5.72	6.67	5.25

where $|\pi_k \cap \pi^*_{k'}|$ denotes the number of samples in both π_k and $\pi^*_{k'}$.

The F-score is defined as follows:

$$\text{F-score} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \tag{19}$$

where TP (True Positive) is the number of sample pairs with the same label in π^* that are assigned to the same cluster in π, FP (False Positive) is the number of sample pairs with different labels in π^* that are assigned to the same cluster in π, and FN (False Negative) is the number of sample pairs with the same label in π^* that are assigned to different clusters in π.

For evaluating the explainability of clustering results in terms of decision trees, we use three classic structural metrics [8]: the number of leaf nodes (nLeaf) [25], the maximal depth of the tree (maxDepth) [36], and the average depth of the leaf nodes (avgDepth) [37]. These metrics quantify the complexity of the rules used to form the clusters, with each cluster corresponding to a leaf node. Lower values of these metrics indicate better explainability, as they correspond to more intuitive and concise tree structures.

4.3. Performance comparison on clustering quality

The results of clustering quality comparison, based on two metrics, are displayed in Table 2. To evaluate the overall clustering quality of each algorithm, we calculate the Mean and Average Rank for each metric across all data sets. The best clustering results are highlighted in boldface. The execution times for producing these results, recorded in seconds, are listed in Table 3. An analysis of Tables 2 and 3 reveals several key observations, which are detailed in the subsections below:

4.3.1. Overall performance

SigTree achieves the best overall performance in terms of two external metrics, while maintaining acceptable execution times compared to a range of algorithms. This indicates that our testing-based splitting criteria is effective in the unsupervised tree growth process so as to form meaningful categorical clusters. Specifically, SigTree outperforms other algorithms on 5 data sets in terms of Purity and 9 data sets with respect to F-score (Ls, Lc, So, Ly, Sf, Pt, De, Bs, Ce). Compared to the widely-used k-modes algorithm, SigTree shows an overall improvement of approximately 6% in Purity and 8.6% in F-score, while maintaining comparable runtime. Furthermore, SigTree demonstrates significantly superior performance in terms of F-score compared to three competing algorithms, while the second-best and worst-performing methods with respect to this metric do not exhibit significant difference, as evidenced in the Critical Difference (CD) plot shown in Fig. 1(b).

Table 3
Running time comparison of different clustering algorithms. Through a one-sided Wilcoxon signed-rank test across all data sets at the 95% confidence interval, algorithms significantly slower than SigTree are marked in red, and those significantly faster than SigTree are marked in blue. The last row shows total running time of each algorithm.

Data set	SigTree	DV	CUBT^{Ham}	CUBT^{MI}	*k*-modes	Entropy	CDE	CDCDR	IMM	RDM	SHA
Ls	0.003	0.038	0.640	0.680	0.004	0.012	0.019	0.006	0.002	0.015	0.017
Lc	0.502	1.944	1.580	1.600	0.058	0.112	0.439	0.015	0.004	0.032	0.033
So	0.110	1.719	1.110	1.110	0.035	0.104	0.077	0.003	0.003	0.026	0.025
Zo	0.079	2.270	1.110	1.060	0.056	0.483	0.043	0.004	0.004	0.031	0.031
Ps	2.641	38.451	26.400	26.210	0.301	0.331	0.903	0.026	0.004	0.076	0.032
Hr	0.007	0.106	1.290	1.130	0.021	0.086	0.016	0.002	0.002	0.017	0.016
Ly	0.090	25.449	8.150	7.820	0.111	0.598	0.084	0.008	0.004	0.046	0.029
Hd	0.165	149.244	14.030	14.780	0.168	1.771	0.077	0.011	0.006	0.091	0.047
Sf	0.048	0.653	3.610	2.710	0.112	1.261	0.026	0.006	0.005	0.037	0.032
Pt	0.172	15.402	5.850	5.760	0.306	10.887	0.057	0.020	0.013	0.166	0.106
De	0.972	112.967	41.060	41.530	0.571	4.523	0.344	0.028	0.010	0.219	0.079
Hv	0.343	16.208	7.250	8.250	0.270	0.483	0.082	0.012	0.004	0.060	0.025
Bs	0.011	0.588	11.750	8.130	0.123	0.481	0.018	0.005	0.005	0.060	0.030
Ca	0.049	8.608	708.100	708.180	0.331	0.400	0.062	0.011	0.005	0.063	0.028
Bc	0.380	113.818	266.440	267.960	0.338	0.487	0.154	0.013	0.006	0.127	0.035
Mm	0.015	0.122	6.220	3.950	0.175	0.324	0.016	0.006	0.003	0.026	0.024
Tt	0.174	19.597	47.090	21.080	0.570	0.573	0.050	0.015	0.006	0.092	0.033
Ce	0.025	4.504	61.470	25.510	0.660	2.923	0.033	0.024	0.008	0.156	0.049
Total	5.784	511.687	1213.150	1147.450	4.209	25.840	2.501	0.217	**0.093**	1.341	0.672

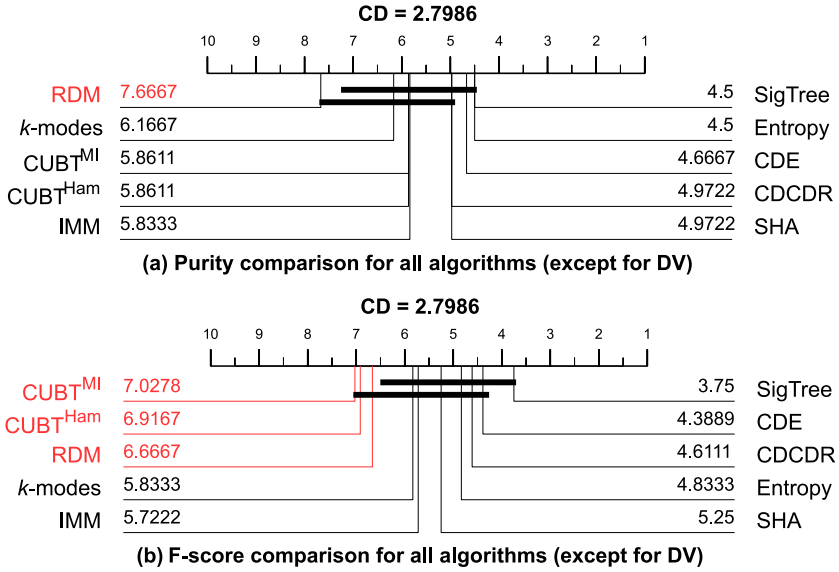

Fig. 1. Clustering quality comparison of SigTree and other methods was conducted using a two-tailed Bonferroni–Dunn test [38] at the 95% confidence interval. The Critical Difference (CD) was then determined based on the number of comparisons, the corresponding critical value, and the number of data sets in the performance comparison. Algorithms demonstrating significantly inferior performance to SigTree are marked in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.3.2. Comparison with DV

SigTree runs significantly faster than DV, requiring only about 1% of the total time needed by DV across all data sets. As a categorical data clustering algorithm that also employs hypothesis testing procedure, DV may refuse to report clustering results if no statistically significant clusters can be identified. As shown in Table 2, DV only outputs clustering results on 7 data sets. Contrarily, our algorithm finds significant splits at the root node to form at least two clusters in almost all data sets. In cases like the data sets Ls, Bs, and Ce, where the *p*-value of the best candidate split at the root node is larger than the significance level, SigTree still produces fairly good clustering results when it is forced to divide the data set with the initial best split. Among the 7 data sets (So, Zo, Ly, Hd, De, Hv, Bc) where DV can yield clustering results, DV outperforms SigTree in terms of both Purity and F-score on only 2 data sets (Zo, Bc). With respect to the automatic determination of cluster number, DV aligns with the ground-truth cluster number on 2 data sets (So, Zo), which is not superior to SigTree, which also correctly predicts the cluster number on 2 data sets (So, De) from the 7 data sets, as shown in Table 4.

4.3.3. Comparison with CUBT

The CUBT variants, CUBT^{Ham} and CUBT^{MI}, are more time-consuming than other competitors and generally produce clusters of significantly lower quality than SigTree, as illustrated in Fig. 1. This is further underscored by the fact that SigTree requires only about 0.5% of the total time taken by CUBT^{Ham} to achieve an improvement of over 3.8% in Purity and 18.6% in F-score. Similarly, in comparison to CUBT^{MI}, SigTree gains more than 19.8% improvement in F-score. Concerning the accuracy of cluster number prediction, only the CUBT^{MI} variant aligns with the ground-truth cluster number on one data set (Ly), while SigTree correctly predicts the cluster number on 6 data sets (So, Hr, Hd, Sf, De, Ca). Additionally, SigTree exhibits the smallest average deviation from the ground-truth cluster number, as indicated in the penultimate row of Table 4.

4.3.4. Comparison with non-interpretable clustering algorithms

In terms of both Purity and F-score, SigTree outperforms *k*-modes, entropy-based method, CDE, and CDCDR on 7, 5, 4, and 4 data sets, respectively. In contrast, the two SOTA algorithms CDE and CDCDR

Table 4
The cluster number prediction of various algorithms. ΔK represents the average prediction deviation from K. The predicted cluster number for SigTree and CUBT methods corresponds to nLeaf, while IMM, RDM, and SHA have nLeaf = K. The last row displays the average nLeaf. CUBT_{Max} produces a significantly larger nLeaf than other methods (using the aforementioned Wilcoxon test).

Data set	K	SigTree	DV	CUBT_{Max}	CUBT^{Ham}	CUBT^{MI}
Ls	3	2	—	2	2	2
Lc	3	2	—	2	2	2
So	4	**4**	**4**	3	2	3
Zo	7	6	**7**	4	3	4
Ps	2	4	—	5	3	3
Hr	3	**3**	—	9	5	4
Ly	4	2	13	8	5	**4**
Hd	5	**5**	32	19	6	6
Sf	6	**6**	—	18	7	7
Pt	21	6	—	22	8	6
De	6	**6**	2	19	4	7
Hv	2	11	5	20	6	7
Bs	3	2	—	33	13	8
Ca	2	**2**	—	42	6	7
Bc	2	8	8	37	6	9
Mm	2	5	—	23	12	7
Tt	2	17	—	62	14	6
Ce	4	2	—	64	8	9
$\Delta\ K$		**3.222**	—	17.944	4.278	3.444
Mean (nLeaf)	**4.500**	5.167	—	21.778	6.222	5.611

outperform SigTree only on 1 and 3 data sets (Hd, Ca, Bc), respectively. However, our algorithm is more time-consuming than these algorithms. This is mainly because the time complexity of SigTree is proportional to $N \ln N$ while the time complexities of these non-interpretable clustering algorithms are only proportional to N.

4.3.5. Comparison with explainable k-means algorithms

SigTree generally requires more time to construct the decision tree than these algorithms, primarily because they select optimal cuts based on the k-means objective function and reference means, whereas SigTree evaluates candidate splits from all possible categories. This contrast is most evident with IMM, the fastest one among these algorithms, which has a time complexity of $\mathcal{O}(K|Q|N \ln N)$, with $|Q|$ being the number of features after one-hot encoding. However, SigTree exhibits superior performance compared to these algorithms. In particular, it significantly outperforms RDM among the 10 compared algorithms, as shown in Fig. 1. Specifically, in terms of both external metrics, SigTree outperforms IMM, RDM, and SHA on 7, 8, and 3 data sets, respectively. Meanwhile, the best performer among them, SHA, does not surpass SigTree on any data set.

4.4. Performance comparison on explainability

In this subsection, we assess the explainability of the SigTree algorithm by comparing it with the interpretable clustering algorithms (CUBT^{Ham}, CUBT^{MI}, IMM, RDM and SHA). The results of explainability comparison, based on three metrics, are presented as follows: 'nLeaf' is shown in Table 4, while 'maxDepth' and 'avgDepth' are given in Table 5. To facilitate a fair comparison with IMM, RDM, and SHA, where the number of leaf nodes is predetermined by the ground-truth cluster number K, we introduce Table 6 based on Table 5. This table showcases the overall performance on 10 data sets where the cluster number predicted by SigTree deviates by no more than 1 from K, ensuring a comparison in terms of maxDepth and avgDepth based on approximately the same nLeaf number.

In the comparison of nLeaf, we particularly examine the distinction between SigTree and CUBT algorithms, observing that they can determine the number of leaf nodes adaptively without using the ground-truth cluster number as input. SigTree generates an nLeaf number not exceeding K on 13 data sets. In contrast, CUBT^{Ham} and CUBT^{MI} can only achieve this goal on 6 data sets. Notably, CUBT_{Max} produces significantly more leaf nodes than all other algorithms. This suggests that two CUBT variants rely heavily on post-processing CUBT_{Max} to enhance their explainability in terms of nLeaf. Despite the use of pruning and joining processes, CUBT^{Ham} and CUBT^{MI} still produce more leaf nodes than the purely tree growth algorithm, SigTree.

In the comparison of both maxDepth and avgDepth, SigTree demonstrates closely matched performance with SHA, its top competitor. This comparison shows that SHA excels in terms of Mean, while SigTree stands out with respect to #Best. Notably, SHA aims to form shallow trees by utilizing measures like Weighted Average Depth in its objective function, which are directly relevant to our explainability performance metrics. SigTree, in contrast, achieves comparable results using its testing-based splitting criteria, without explicitly targeting tree depth during the tree growing process. Moreover, SigTree can achieve superior performance over IMM in terms of these two depth-based metrics. The worst-performing competitors are those CUBT variants. Specifically, CUBT_{Max} and CUBT^{Ham} are significantly inferior to SigTree.

Additionally, considering that the nLeaf number can impact the maxDepth and avgDepth metrics, we focus on a subset of data sets where SigTree's nLeaf aligns with other explainable k-means algorithms. The Table 6 shows that SigTree can be the top performer among all interpretable clustering algorithms in terms of both maxDepth and avgDepth.

4.5. The comparison via the visualization of decision trees

In this section, we provide a practical example to illustrate the tree structures created by various interpretable clustering algorithms, as shown in Fig. 2. Each algorithm employs unique splitting strategies at non-leaf nodes, resulting in distinct hierarchical structures. Achieving perfect clustering results (both Purity and F-score equal to 1) on the So data set depends on choosing appropriate splitting points at all non-leaf nodes. We found that only SigTree, IMM and SHA can achieve this objective on this data set. If we take the clustering quality into consideration, only SHA can beat SigTree in terms of both clustering quality and explainability on this data set.

Empirically, this example suggests that attributes frequently used in splitting points are critical to the construction of an accurate and concise decision tree. Commonly utilized attributes in all trees include 'Canker Lesion', 'Stem Canker', and 'Fruit Pods'. SHA, achieving perfect clustering with the smallest maxDepth and avgDepth, incorporates all these commonly used attributes. Other algorithms that achieve perfect

Table 5
The explainability comparison in terms of maxDepth and avgDepth for all interpretable clustering algorithms. #Best indicates the number of times that one algorithm is the best performer across all data sets. $CUBT_{Max}$ and $CUBT^{MI}$ produce a decision tree whose maxDepth and avgDepth are both significantly larger than those of SigTree (using the aforementioned Wilcoxon test).

Data set	SigTree	$CUBT_{Max}$	$CUBT^{Ham}$	$CUBT^{MI}$	IMM	RDM	SHA
			Metric: maxDepth				
Ls	**1**	**1**	**1**	**1**	2	2	2
Lc	**1**	**1**	**1**	**1**	2	2	2
So	3	2	**1**	2	2.98	2.66	2
Zo	4	3	**2**	3	4.1	4.14	3.98
Ps	2	3	2	2	**1**	**1**	**1**
Hr	**2**	4	3	3	**2**	**2**	**2**
Ly	**1**	4	4	3	2.88	2.64	2.9
Hd	**3**	6	4	5	3.46	3.22	3.2
Sf	4	6	4	4	4.24	3.9	**3.42**
Pt	**4**	7	5	**4**	16.1	8.48	7.68
De	5	7	**3**	4	4.32	4.48	3.6
Hv	5	6	4	6	**1**	**1**	**1**
Bs	**1**	6	5	4	2	2	2
Ca	**1**	7	4	4	**1**	**1**	**1**
Bc	6	7	5	6	**1**	**1**	**1**
Mm	3	6	5	5	**1**	**1**	**1**
Tt	5	7	6	5	**1**	**1**	**1**
Ce	**1**	6	6	4	2.94	2.92	2.82
Mean	2.889	4.944	3.611	3.667	3.057	2.580	**2.422**
#Best	**9**	2	5	3	7	7	8
			Metric: avgDepth				
Ls	**1**	**1**	**1**	**1**	1.667	1.667	1.667
Lc	**1**	**1**	**1**	**1**	1.667	1.667	1.667
So	2.25	1.667	**1**	1.667	2.245	2.165	2
Zo	2.833	2.250	**1.667**	2.250	3.157	3.129	2.997
Ps	2	2.4	1.667	1.667	**1**	**1**	**1**
Hr	**1.667**	3.222	2.600	2.250	**1.667**	**1.667**	**1.667**
Ly	**1**	3.250	2.800	2.250	2.226	2.160	2.225
Hd	**2.400**	4.421	3	3.333	2.620	2.564	2.552
Sf	3.167	4.444	3	3.286	3.110	2.993	**2.800**
Pt	**2.833**	4.818	3.625	3.167	9.590	5.575	4.931
De	3.333	4.737	**2.250**	3	3.110	3.170	2.767
Hv	3.909	4.650	3.167	3.857	**1**	**1**	**1**
Bs	**1**	5.061	4.077	3.125	1.667	1.667	1.667
Ca	**1**	5.929	2.833	3	**1**	**1**	**1**
Bc	3.875	5.432	3.333	4.222	**1**	**1**	**1**
Mm	2.400	4.739	3.917	3.571	**1**	**1**	**1**
Tt	4.235	6.065	4.786	3.333	**1**	**1**	**1**
Ce	**1**	6	4.250	3.333	2.235	2.230	2.205
Mean	2.272	3.949	2.776	2.740	2.276	2.036	**1.952**
#Best	**9**	2	5	2	7	7	8

Table 6
The comparison of maxDepth and avgDepth on the ten data sets: Ls, Lc, So, Zo, Hr, Hd, Sf, De, Bs, Ca, where SigTree produces a nLeaf number approaching K with a maximum deviation of 1. The notations used here are consistent with those in Table 5.

Data set	SigTree	$CUBT_{Max}$	$CUBT^{Ham}$	$CUBT^{MI}$	IMM	RDM	SHA
			Metric: maxDepth				
Mean	**2.5**	4.3	2.8	3.1	2.81	2.74	2.52
#Best	**6**	2	5	2	2	2	3
			Metric: avgDepth				
Mean	**1.965**	3.373	2.243	2.391	2.191	2.169	2.078
#Best	**6**	2	5	2	2	2	3

clustering, SigTree and IMM, each uses two of these frequently used attributes. Interestingly, their initial splits are based on unique attributes, showcasing the diversity of these approaches at finding effective splits. Additionally, $CUBT_{Max}$ and $CUBT^{Ham}$, conforming to such cases, did not achieve perfect clustering, possibly because their nLeaf numbers do not align with K, but they avoided dividing samples of a single disease into different leaf nodes (clusters). In contrast, RDM, exhibiting worse clustering quality, incorrectly splits samples of the same disease into different leaf nodes. It utilizes two unique attributes and applies a commonly used attribute only at its last non-leaf node. This is also in stark contrast to other trees where this attribute 'Canker Lesion' is exclusively associated with the category `brown`.

5. Conclusions

In this paper, we introduce SigTree, an interpretable clustering method for categorical data based on hypothesis testing. This method is characterized by its distinct splitting point selection criteria based on statistical association testing for constructing an unsupervised decision tree. Overall, it has the following salient features: (1) It operates without the need for manually specifying different parameters (e.g., the number of clusters) for different data sets. (2) As its only parameter, the significance level is easy to be specified with a clear statistical interpretation. (3) It does not need to post-process the initial decision tree since whether each leaf node should be further divided is determined

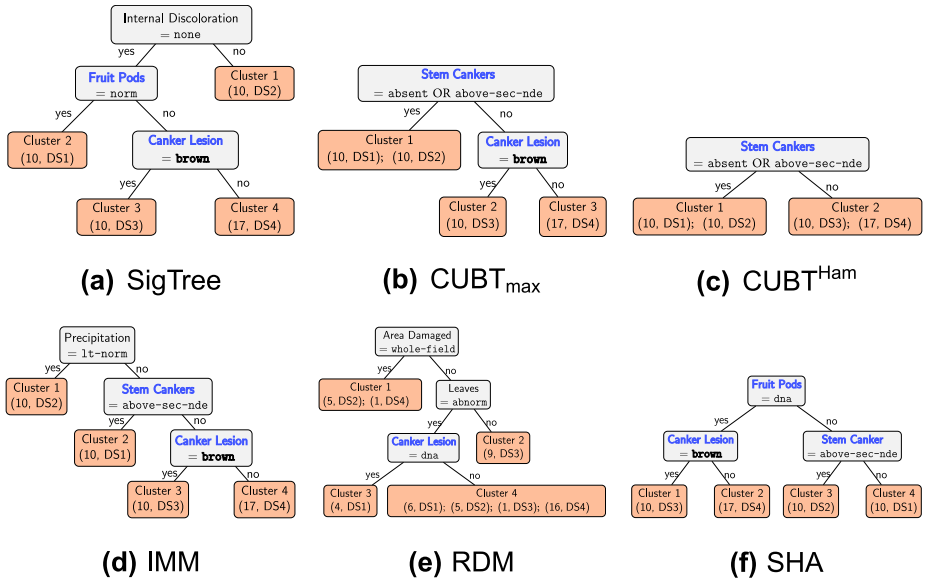

Fig. 2. Decision trees for interpretable clustering algorithms on the So data set. Each splitting point in the non-leaf nodes is presented using the actual attribute and category names, denoted as 'Attribute = `Category`'. The `dna` category stands for 'does not apply'. The leaf nodes, indicative of predicted clusters, are labeled with the actual cluster names corresponding to the samples they contain, depicted as '(number of samples, cluster name)'. The original data set comprises four clusters, where each cluster is associated with a specific soybean disease: DS1 for Diaporthe stem canker, DS2 for Charcoal rot, DS3 for Rhizoctonia root rot, and DS4 for Phytophthora rot. Attributes used in the decision trees of six algorithms, appearing two or more times, are highlighted in blue. Notably, 'Canker Lesion = `brown`' emerges as the most frequently used splitting point.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

according to a rigorous significance testing procedure. (4) It does not need the clustering result of a third-party non-interpretable clustering algorithm as the input to facilitate the decision tree construction. Empirical results on real data sets show that our method is comparable to those SOTA categorical data clustering algorithms with respect to the clustering quality. More importantly, it can beat existing interpretable clustering algorithms for categorical data in terms of both clustering quality and explainability.

There are two main limitations in our method that need to be further addressed. Firstly, finding the optimal split in SigTree is time-consuming as it requires calculating p-values for all candidate splits. To address this, future research could focus on designing branch-and-bound methods to reduce the search space. Secondly, it is challenging to provide a theoretical guarantee that our method can obtain an optimal decision tree, primarily due to the absence of an explicit clustering objective function. We will explore the possibility of assessing the statistical significance of each candidate unsupervised decision tree. Accordingly, the p-value of an entire decision tree can serve as the objective function to guide us to find more accurate and concise decision trees.

In addition to those two limitations, our algorithm encounters dilemmas when applied to larger data sets, particularly those with a high number of attributes. This is fundamentally due to inherent flaws in the chi-squared test, specifically its lack of an interpretable rejection of the null hypothesis [39]. In other words, larger sample sizes are more likely to inflate the chi-squared test statistic as calculated according to Eq. (4). On one hand, larger absolute differences between observed and expected frequencies in each cell of contingency tables would be reported for data sets with more samples. On the other hand, the increase on the number of attribute will yield a larger test statistic as well. This inflated chi-squared test statistic skews p-values extremely close to zero, making it difficult to control the false discovery rate using thresholds like those in Eq. (8). In future research, along with the testing-based framework for developing interpretable categorical data clustering algorithms, we will explore more valid alternative statistical tests to replace the currently used chi-squared test.

CRediT authorship contribution statement

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

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.

Acknowledgment

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

Data availability

Data will be made available on request.

References

[1] A. Agresti, Categorical Data Analysis, John Wiley & Sons, Hoboken, 2012.
[2] S. Naouali, S. Ben Salem, Z. Chtourou, Clustering categorical data: A survey, Int. J. Inf. Technol. Decis. Mak. 19 (01) (2020) 49–96.
[3] M. Ackerman, S. Ben-David, S. Brânzei, D. Loker, Weighted clustering: Towards solving the user's dilemma, Pattern Recognit. 120 (2021) 108152.
[4] L. Bai, J. Liang, Cluster validity functions for categorical data: a solution-space perspective, Data Min. Knowl. Discov. 29 (6) (2015) 1560–1597.
[5] S. Bandyapadhyay, F.V. Fomin, P.A. Golovach, W. Lochet, N. Purohit, K. Simonov, How to find a good explanation for clustering? Artificial Intelligence (2023) 103948.
[6] J. Basak, R. Krishnapuram, Interpretable hierarchical clustering by constructing an unsupervised decision tree, IEEE Trans. Knowl. Data Eng. 17 (1) (2005) 121–132.

[7] 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.
[8] E. Laber, L. Murtinho, F. Oliveira, Shallow decision trees for explainable k-means clustering, Pattern Recognit. 137 (2023) 109239.
[9] R. Fraiman, B. Ghattas, M. Svarc, Interpretable clustering using unsupervised binary trees, Adv. Data Anal. Classif. 7 (2013) 125–145.
[10] 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.
[11] M. Moshkovitz, S. Dasgupta, C. Rashtchian, N. Frost, Explainable k-means and k-medians clustering, in: International Conference on Machine Learning, 2020, pp. 7055–7065.
[12] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Min. Knowl. Discov. 2 (3) (1998) 283–304.
[13] S. Guha, R. Rastogi, K. Shim, ROCK: A robust clustering algorithm for categorical attributes, Inf. Syst. 25 (5) (2000) 345–366.
[14] T. Xiong, S. Wang, A. Mayers, E. Monga, DHCC: Divisive hierarchical clustering of categorical data, Data Min. Knowl. Discov. 24 (2012) 103–135.
[15] M.K. Ng, M.J. Li, J.Z. Huang, Z. He, On the impact of dissimilarity measure in k-modes clustering algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 29 (3) (2007) 503–507.
[16] S. Boriah, V. Chandola, V. Kumar, Similarity measures for categorical data: A comparative evaluation, in: Proceedings of the 2008 SIAM International Conference on Data Mining, SIAM, 2008, pp. 243–254.
[17] L. Bai, J. Liang, A categorical data clustering framework on graph representation, Pattern Recognit. 128 (2022) 108694.
[18] C. Zhu, L. Cao, J. Yin, Unsupervised heterogeneous coupling learning for categorical representation, IEEE Trans. Pattern Anal. Mach. Intell. 44 (1) (2022) 533–549.
[19] S. Jian, G. Pang, L. Cao, K. Lu, H. Gao, CURE: Flexible categorical data representation by hierarchical coupling learning, IEEE Trans. Knowl. Data Eng. 31 (5) (2019) 853–866.
[20] T. Li, S. Ma, M. Ogihara, Entropy-based criterion in categorical clustering, in: International Conference on Machine Learning, 2004, p. 68.
[21] H.-L. Chen, K.-T. Chuang, M.-S. Chen, On data labeling for clustering categorical data, IEEE Trans. Knowl. Data Eng. 20 (11) (2008) 1458–1472.
[22] D. Barbará, Y. Li, J. Couto, COOLCAT: An entropy-based algorithm for categorical clustering, in: Proceedings of the Eleventh International Conference on Information and Knowledge Management, 2002, pp. 582–589.
[23] L. Bai, J. Liang, H. Du, Y. Guo, An information-theoretical framework for cluster ensemble, IEEE Trans. Knowl. Data Eng. 31 (8) (2018) 1464–1477.
[24] P. Zhang, X. Wang, P.X.-K. Song, Clustering categorical data based on distance vectors, J. Amer. Statist. Assoc. 101 (473) (2006) 355–367.
[25] H. Hwang, S.E. Whang, Xclusters: explainability-first clustering, in: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence, vol. 37, (no. 7) 2023, pp. 7962–7970.
[26] 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.
[27] B. Kim, J.A. Shah, F. Doshi-Velez, Mind the gap: A generative approach to interpretable feature selection and extraction, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, vol. 28, 2015, pp. 2260–2268.
[28] J. Chen, Y. Chang, B. Hobbs, P. Castaldi, M. Cho, E. Silverman, J. Dy, Interpretable clustering via discriminative rectangle mixture model, in: IEEE International Conference on Data Mining, 2016, pp. 823–828.
[29] L. Chen, C. Zhong, Z. Zhang, Explanation of clustering result based on multi-objective optimization, PLoS One 18 (10) (2023) 1–30.
[30] 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.
[31] D. Bertsimas, A. Orfanoudaki, H. Wiberg, Interpretable clustering: an optimization approach, Mach. Learn. 110 (2021) 89–138.
[32] G.V. Kass, An exploratory technique for investigating large quantities of categorical data, J. R. Stat. Soc. Ser. C. Appl. Stat. 29 (2) (1980) 119–127.
[33] X. Cui, T. Dickhaus, Y. Ding, J.C. Hsu, Handbook of Multiple Comparisons, CRC Press, 2021.
[34] D. Dua, C. Graff, UCI machine learning repository, 2019, University of California, Irvine, School of Information and Computer Sciences.
[35] M. Rezaei, P. Fränti, Set matching measures for external cluster validity, IEEE Trans. Knowl. Data Eng. 28 (8) (2016) 2173–2186.
[36] R. Piltaver, M. Luštrek, M. Gams, S. Martinčić-Ipšić, What makes classification trees comprehensible? Expert Syst. Appl. 62 (2016) 333–346.
[37] V.F. Souza, F. Cicalese, E. Laber, M. Molinaro, Decision trees with short explainable rules, in: Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 12365–12379.
[38] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[39] T.Z. Baharav, D. Tse, J. Salzman, OASIS: An interpretable, finite-sample valid alternative to Pearson's χ^2 for scientific discovery, Proc. Natl. Acad. Sci. 121 (15) (2024) e2304671121.

Lianyu Hu received the MS degree in computer science from Ningbo University, China, in 2019. He is currently working toward the Ph.D. degree in the School of Software at Dalian University of Technology. His current research interests include machine learning, cluster analysis and data mining.

Mudi Jiang received the MS degree in software engineering from Dalian University of Technology, China, in 2023. He is currently working toward the Ph.D. degree in the School of Software at the same university. His current research interests include data mining and its applications.

Junjie Dong received the BS degrees in chemistry from Dalian University of Technology and University of Leicester in 2022, respectively. He is currently working toward the MS degree in the School of Software at Dalian University of Technology. His research interests include bioinformatics and data mining.

Xinying Liu received the MS degree in applied statistics from China University of Geosciences, in 2023. She is currently working toward the Ph.D. degree in the School of Software at Dalian University of Technology. Her current research interests include machine learning and data mining.

Zengyou He received the BS, MS, and Ph.D. degrees in computer science from Harbin Institute of Technology, China, in 2000, 2002, and 2006, respectively. He is currently a professor in the School of Software, Dalian University of Technology. His research interest include data mining and bioinformatics.