Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 7870534, 12 pages
https://doi.org/10.1155/2020/7870534

Research Article

A Novel Multiway Splits Decision Tree for Multiple Types of Data

Zhenyu Liu,[1,2] **Tao Wen**,[1,2] **Wei Sun**,[2] **and Qilong Zhang**[1]

[1]*College of Computer Science and Engineering, Northeastern University, Shenyang 110819, China*
[2]*Department of Computer Science and Technology, Dalian Neusoft University of Information, Dalian, Liaoning, China*

Correspondence should be addressed to Zhenyu Liu; liuzhenyu@neusoft.edu.cn

Received 22 May 2020; Revised 26 September 2020; Accepted 30 October 2020; Published 16 November 2020

Academic Editor: Isabel S. Jesus

Classical decision trees such as C4.5 and CART partition the feature space using axis-parallel splits. Oblique decision trees use the oblique splits based on linear combinations of features to potentially simplify the boundary structure. Although oblique decision trees have higher generalization accuracy, most oblique split methods are not directly conducive to the categorical data and are computationally expensive. In this paper, we propose a multiway splits decision tree (MSDT) algorithm, which adopts feature weighting and clustering. This method can combine multiple numerical features, multiple categorical features, or multiple mixed features. Experimental results show that MSDT has excellent performance for multiple types of data.

1. Introduction

Despite the great success of deep neural network (DNN) model in image processing, speech recognition, and other fields in recent years, decision trees have competitive performance compared to DNN scheme, such as the advantage of interpretability, less parameters, and good robustness to noise, and can be applied to large-scale data sets with less computational cost. Therefore, the decision tree is still one of the hotspots in the field of machine learning today [1–3]. The research mainly focused on the construction method of decision trees, split criterion [4], decision trees ensemble [5, 6], mixing with other learners [7–9], decision trees for semisupervised learning [10], and so on.

Despite practical success, the optimal construction of decision trees has been theoretically proven to be NP-complete [11]. In order to avoid the local optimal solution, some researchers adopted evolutionary algorithms to build decision trees [12–14]. However, due to the time complexity, the most popular algorithms, such as ID3 [15], C4.5 [16], and CART [17], and their various modifications [18] are greedy by nature and construct the decision tree in a top-down, recursive manner. Besides, they only act on one dimension at a time and thus result in an axis-parallel split. In the induction of decision tree, if the candidate features are numerical, a suitable cut point needs to be searched. Instances in the training set are divided into the left node or the right node according to the following formula:

$$x_l \le \theta_l, \tag{1}$$

where x_l denotes the value of the instance on the feature A_l and θ_l is the cut point.

Axis-parallel trees have the advantages of fast induction and strong comprehensibility. However, in the case of highly correlated features, a very bad situation may arise. Figure 1 gives an illustration. The parallel splits will be carried out many times with a stair case-like structure, which leads to the complexity of the decision tree structure.

To solve the problem of parallel decision trees, some researchers introduced oblique decision trees. In such oblique decision trees, the nonleaf node tests the linear combination of features, i.e.,

$$\sum_{i=1}^{p} a_l x_l \le \theta, \tag{2}$$

where a_l represents the coefficient for the lth feature, θ is the threshold, and p is the number of features. In Figure 1, the instances of the two classes can be completely separated by one oblique split. Therefore, it is generally believed that the

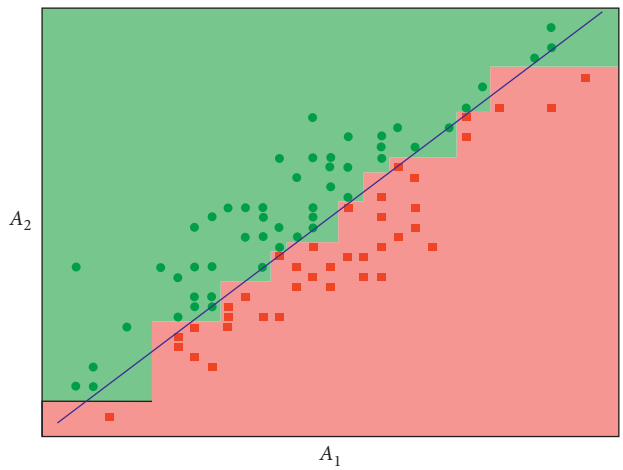

Figure 1: Axis-parallel splits and oblique split.

oblique splits can often produce smaller decision trees and better generalization performance for the same data.

It is much more difficult to search the optimal oblique hyperplanes than the optimal axis-parallel hyperplanes. To solve this problem, numerous techniques have been applied, for example, hill-climbing [17], simulated annealing [19], and genetic algorithm [20]. Among them, a large amount of research work has been done on reducing the risk of falling into local optimal solution, such as Simulated Annealing Decision Tree (SADT) [19], which used the simulated annealing algorithm; OC1 [21] method combined the ideas of CART-LC [17] and SADT.

In searching oblique hyperplanes, thousands of candidates have been tried in both simulated annealing algorithm and genetic algorithm, resulting in low time efficiency. So many researchers used linear discriminant analysis, linear regression, perceptron, SVM, and other methods to find suitable oblique hyperplanes. Fisher's decision tree (FDT) [22] takes advantage of dimensionality reduction of Fisher's linear discriminant and uses the decomposition strategy of decision trees to come up with an oblique decision tree. FDT is only applicable to binary classification problems. Based on ADTree [23], Hong et al. [24] proposed the multivariate ADTree. Paper [24] presented and discussed the different variations of ADTree (Fisher's ADTree, Sparse ADTree, and Regularized Logistic ADTree). Wickramarachchi et al. [25] explored a decision tree algorithm (HHCART). HHCART uses a series of Householder matrices to reflect the training data during tree construction. Shah and Sastry [26] defined separability of instances as the split criterion that optimized their evaluation function at each node and then presented the Alopex Perceptron Decision Tree algorithm for learning a decision tree. Menze et al. [27] presented an oblique tree forest method, which used LDA and ridge regression to conduct oblique splits.

In the above oblique methods, the trees with fewer nodes and better accuracy can be obtained. However, there are also some deficiencies, mainly including three aspects.

1.1. Inability to Directly Employ the Methods for Categorical Data. The oblique splits use the linear combination of features. Therefore, the categorical features need to be converted into one or more numerical features [28]. This transformation may bring new biases to the classification problems, thus reducing the generalization ability of the models.

1.2. High Time Cost. The oblique splits always require complex matrix calculation when using linear discriminant analysis, ridge regression, or other methods. Although these methods are more efficient than simulated annealing and genetic algorithm, they still pay more cost than the axis-parallel methods, such as C4.5.

1.3. Some Methods Cannot Be Suitable for Multiclassification Problems. Generally, the oblique split methods conduct the binary splits. Although the binary tree can also be directly used for multiclassification problems, some binary splits rely on class label, such as FDA, original SVM, etc., which makes some algorithms like FDT in [22] limited to binary classification problems. In addition, some models need to convert multiclassification problems into binary ones [7].

In order to overcome the above shortcomings, this paper proposes a multiway splits decision tree for multiple types of data (numerical, categorical, and mixed data). The specific characteristics of this method are as follows:

(i) Categorical features are handled directly.

(ii) The time complexity is similar to that of the axis-parallel split algorithms.

(iii) It is not necessary to convert multiclassification problems into binary ones by using the multiway splits directly.

The remainder of the paper is organized as follows. In Section 2, we review RELIEF-F and k-means algorithms briefly. Section 3 presents our algorithm and discusses its time complexity. Section 4 presents and analyzes the compared experimental results with other decision trees. The last section gives the conclusion of this paper.

2. Preliminaries

The proposed decision tree method needs to weight the features by RELIEF-F algorithm and split the nodes by the weighted k-means algorithm. Therefore, this section reviews the two algorithms and their variations.

2.1. RELIEF-F Algorithms. The RELIEF algorithm [29] is popular to feature selection. It estimates the weights of features according to the correlation between individual feature and class label. RELIEF randomly samples an instance R from the training set and then searches its two nearest neighbors H and M: H is from the same class (called near Hit) and M is from different class (called near Miss). If the distance between R and H on feature A is less than the distance between R and M, RELIEF will increase A's weight. On the contrary, RELIEF will decrease the weight.

In fact, RELIEF's estimate $W(A)$ of feature A is an approximation of the following difference of probabilities:

$$W(A) = P(\text{different value of } A|\text{nearest instance from different class}) - P(\text{different value of } A|\text{nearest instance from same class}), \tag{3}$$

where $P(\cdot|\cdot)$ represents the conditional probability.

RELIEF algorithm only deals with binary classification problems. Kononeill addressed an algorithm called RELIEF-F for multiclassification problems [30]. The algorithm picks m instances. For each instance R, its k_{nn} nearest neighbors are searched in each class.

The weight $W(A)$ is calculated as follows:

$$W(A) = W(A) - \sum_{j=1}^{k_{\text{nn}}} \text{diff}\frac{\left(A, R, H_j\right)}{\left(mk_{\text{nn}}\right)} + \sum_{T \notin \text{class}(R)} \frac{\left[p(T)/1 - p(\text{class}(R)) \sum_{j=1}^{k_{\text{nn}}} \text{diff}\left(A, R, M_j(T)\right)\right]}{\left(mk_{\text{nn}}\right)}, \tag{4}$$

where $p(T)$ represents the proportion of class T instances to the total instances and $M_j(T)$ represents the jth nearest neighbor to R in class T. $\text{diff}(A, R_1, R_2)$ calculates the difference between two instances R_1 and R_2 on the feature A as follows:

$$\text{diff}\left(A, R_1, R_2\right) = \begin{cases} \dfrac{\left|R_1[A] - R_2[A]\right|}{\max(A) - \min(A)}, & \text{if } A \text{ is numerical,} \\ 0, & \text{if } A \text{ is categorical And } R_1[A] = R_2[A], \\ 1, & \text{if } A \text{ is categorical And } R_1[A] \neq R_2[A]. \end{cases} \tag{5}$$

2.2. k-Means, k-Modes, and k-Prototypes. The k-means is widely used in real world applications due to its simplicity and efficiency.

Let D be a set of n instances. D is characterized by a set of p features and needs to be clustered into k clusters $C_1, C_2, \ldots, C_k$. First, randomly pick some instances as the centers of the initial k clusters $\mu_1, \mu_2, \ldots, \mu_k$, and then calculate the cluster label for each instance $\mathbf{x}_i$ as follows:

$$\text{label}_i = \arg\min_{1 \le j \le k} \left\|\mathbf{x}_i - \boldsymbol{\mu}_j\right\|. \tag{6}$$

After all the instances are partitioned, each cluster center will be updated by the following formula:

$$\boldsymbol{\mu}_\mathbf{j} = \frac{1}{n_j} \sum_{i=1}^{n_j} \mathbf{x}_\mathbf{i}, \quad \mathbf{x}_\mathbf{i} \in C_j. \tag{7}$$

Repeat formulas (6) and (7) until the variable E in formula (8) converges to the local optimal solution or the preset number of iterations is reached:

$$E = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \left\|\mathbf{x}_\mathbf{i} - \boldsymbol{\mu}_\mathbf{j}\right\|^2, \quad \mathbf{x}_\mathbf{i} \in C_j. \tag{8}$$

However, the classical k-means is only worked on the numerical data. The k-modes and k-prototypes are variants of k-means for categorical and mixed data, respectively [31]. When k-modes processes the categorical variables, the center of each cluster is represented by modes. When calculating the distance between instance and cluster center, the distance on each feature is calculated by formula (5) and then accumulated.

It is straightforward to integrate the k-means and k-modes into the k-prototypes. $\text{dis}(\mathbf{x_i}, \mu_\mathbf{j})$ is the distance between instance $\mathbf{x_i}$ and cluster center $\mu_\mathbf{j}$ as follows:

$$\text{dis}\left(\mathbf{x_i}, \boldsymbol{\mu_j}\right) = (1 - \gamma) \cdot \text{dis\_}n\left(\mathbf{x_i}, \boldsymbol{\mu_j}\right) + \gamma \cdot \text{dis\_}c\left(\mathbf{x_i}, \boldsymbol{\mu_j}\right), \tag{9}$$

where $\text{dis\_}n(\mathbf{x_i}, \mu_\mathbf{j})$ represents the distance on the numerical variables and $\text{dis\_}c(\mathbf{x_i}, \mu_\mathbf{j})$ represents the distance on the categorical variables, respectively. γ is used to adjust the proportion of $\text{dis\_}n(\mathbf{x_i}, \mu_\mathbf{j})$ and $\text{dis\_}c(\mathbf{x_i}, \mu_\mathbf{j})$, $\gamma \in [0, 1]$.

3. Our Proposed Algorithm

Our proposed MSDT has three differences with most oblique methods: (i) MSDT does not use greedy methods to pursue maximum impurity reduction, (ii) MSDT uses a combination of multiple variables to do multiway splits for nonleaf nodes, and (iii) MSDT treats categorical features in a similar way to numerical features.

3.1. Multiway Splits. Most oblique methods conduct binary splits, while the proposed algorithm performs multiway splits; that is, in one split, multiple hyperplanes are generated simultaneously, and the feature space is divided into several disjoint regions. Ho [32] categorized the linear split methods into three types, axis-parallel linear splits, oblique linear

splits, and piecewise linear splits, while our method falls into the third. Piecewise linear split methods find k anchors in feature space, and each instance is clustered according to the nearest neighbor anchor. Figure 2 shows the 5-way splits of the two-dimensional feature space.

3.2. Location of Anchor. Finding suitable split hyperplanes is the key problem in most decision tree induction algorithms. Under piecewise linear splits, the problem of finding appropriate hyperplanes is equivalent to that of finding appropriate anchors. Usually, anchor selection can use the class centroids, or cluster centers generated by some clustering algorithms. In MSDT, we first use RELIEF-F to weight features and then use k-means with weighted distance to cluster instances.

3.2.1. Why Do We Use k-Means? If the instances are linearly separable, it is obviously more efficient to use simply the class centroids than cluster centers as anchors. However, when the instances of some classes are distributed in different regions of the feature space, the class centroids may no longer be suitable for being anchors. For example, in Figure 3, the circular instances are distributed in two different areas. If the solid line that is perpendicular to the line between the two class centroids is used to separate the instances, the effect is obviously not satisfactory. The instances in Figure 3 are obviously distributed into two clusters. If the instances are divided by the dotted line that is a perpendicular bisector of the two cluster centers, at least the circular instances on the right side of the figure can be distinguished.

The split method proposed is based on clustering assumption. The clustering assumption states that the samples belonging to the same cluster belong to the same class. k-means methods partition instances according to some (dis) similarity measures; hence, the leaf nodes of MSDT can be regarded as some prototypes, and the class of a test instance depends on which prototype the instance is more similar to.

The univariate decision trees can produce a comprehensible classification mode, due to the knowledge representation method—a decision tree is a graphical representation and can be easily converted into a set of rules written in a natural language. Some researchers believe that multivariate decision trees are not able to convert into the comprehensible rules. The other researchers think that multivariate tree with fewer nodes is easy to understand. MSDT is easy to understand due to two reasons. One is that MSDT has fewer nodes compared to univariate decision trees. The other one is that the similarity with the prototype is easy to understand by the users and it can replace the rules generated by the univariate decision tree.

3.2.2. Why Do We Weight Features? The original k-means is an unsupervised clustering algorithm, which is suitable for unlabeled data. And the optimization goal is to minimize (8). The goal of split is to reduce the class impurity of current node as much as possible. Note that the two goals are not the same. Therefore, we estimate the correlations between features and

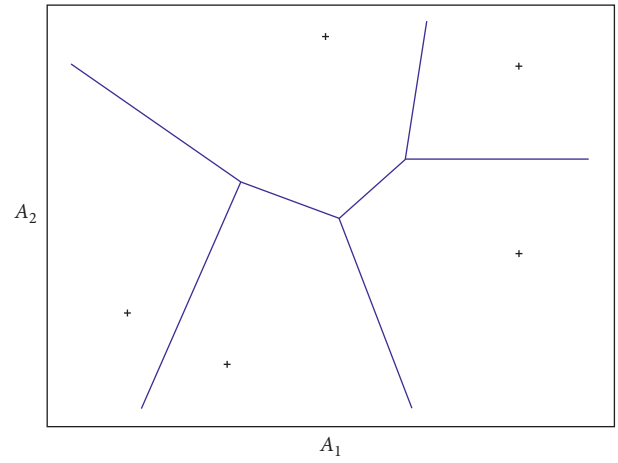

Figure 2: Piecewise linear splits.

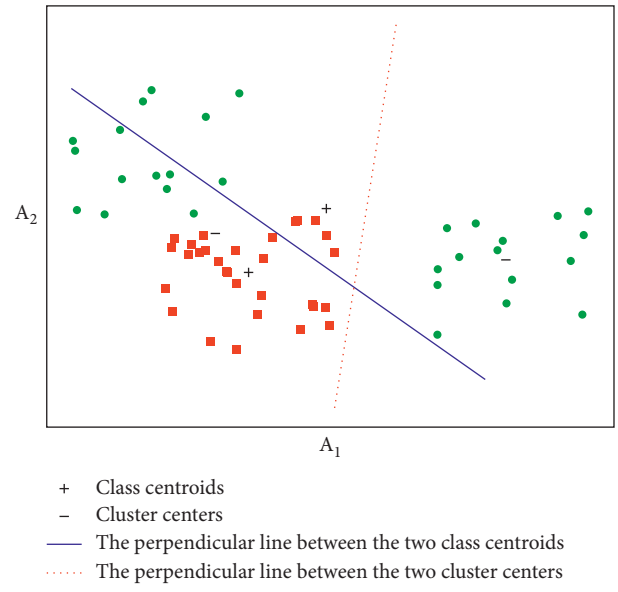

Figure 3: Split results of class centroids as anchors and cluster centers as anchors.

label to weight features. When calculating the distance from an instance to a cluster center, we give a larger weight to the feature strongly related to the label that enlarges the contribution of the feature to the distance. Otherwise, we give a smaller weight that reduces the contribution of the uncorrelated feature to the distance. In this way, the optimization goal of k-means algorithm is close to that of node split.

Figure 4 shows an example to illustrate the effectiveness of feature weighting. The solid line comes from unweighted features, and the dotted line comes from weighted features when the weight of A_1 is 0.05 and the weight of A_2 is 0.95. It is obvious that some instances have been corrected.

To further illustrate the role of feature weighting, we use dataset *iris* to carry out a simple experiment: 150 samples of dataset *iris* come from three classes, and each class has 50 samples. We directly use k-means algorithm to cluster and obtain 10 misclassified samples. The specific results are shown in Table 1.

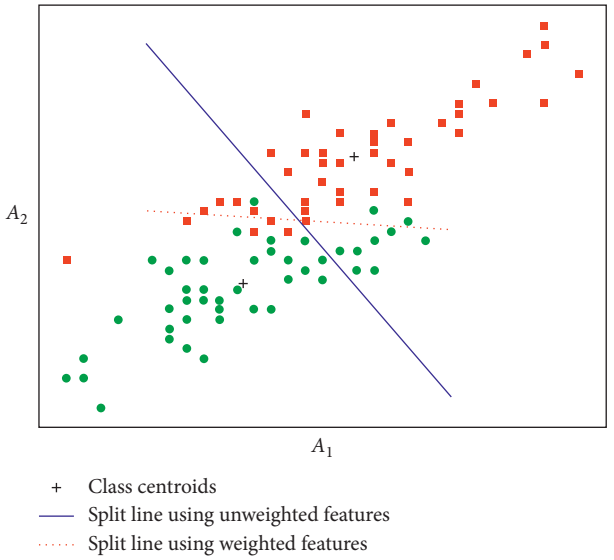

Figure 4: Split results of unweighted and weighted features.

Then, we use the RELIEF-F algorithm to calculate the weights of four features, which are 0.09, 0.14, 0.34, and 0.39, respectively. In the process of k-means clustering, the distances between instances and cluster centers are calculated by (10), where p indicates the number of features and w_l indicates the weight of the lth feature. We obtain 6 misclassified samples, and the specific results are shown in Table 2.

$$\left\|\mathbf{x_i} - \boldsymbol{\mu_j}\right\| = \sqrt{\sum_{l=1}^{p} w_l \cdot \left(x_{i,l} - \mu_{j,l}\right)^2}. \tag{10}$$

Our proposed split method is shown in Algorithm 1, which will be used to split nodes for numerical data.

In the fifth step of Algorithm 1, $l_{\max}$ represents the maximum number of iterations. In the experiments, we set it to 6 by default. The reason for setting such a small value is mainly to consider the time efficiency of the algorithm. In addition, the purpose of clustering is to split nodes. Even if the clustering algorithm does not converge, the partition results can still be accepted.

3.3. Categorical Feature. As mentioned in the previous subsection, the split method can be directly applied to numerical features. For categorical features, RELIEF-F algorithm can still be used to weight features. However, in the process of clustering, the representation of cluster center and the distance from instance to cluster center need to be redefined.

The k-modes extends the k-means by replacing the means of numerical variables with the modes of the categorical variables. Yet it is less precise to calculate the distance. What is more, choosing different modes may cause opposite conclusion while there are several modes for a feature.

Table 1: Split results of unweighted features for the *iris* dataset.

	Setosa	Versicolor	Virginica
Child node 1	50	0	0
Child node 2	0	44	4
Child node 3	0	6	46

Table 2: Split results of weighted features for the *iris* dataset.

	Setosa	Versicolor	Virginica
Child node 1	50	0	0
Child node 2	0	48	4
Child node 3	0	2	46

Here is an example. Suppose there are two clusters C_1 and C_2 described by two categorical features A_1 and A_2, and each cluster contains 10 instances as is shown in Table 3. The modes of C_1 and C_2 for A_1 are $a11$, which makes A_1 useless for distinguishing the distances between instances and the clusters. There are two modes for A_2 in C_1 and C_2, respectively. Suppose that there is an instance $\mathbf{q} = (a11, a21)$; if $\boldsymbol{\mu_1} = (a11, a21)$ is selected as the center of C_1 and $\boldsymbol{\mu_2} = (a11, a23)$ for C_2, distance between $\mathbf{q}$ and $\boldsymbol{\mu_1}$ is 0 and distance between $\mathbf{q}$ and $\boldsymbol{\mu_2}$ is 1; hence, $\mathbf{q}$ is nearer to C_1. If $\boldsymbol{\mu_1} = (a11, a22)$ is selected as the center of C_1 and $\boldsymbol{\mu_2} = (a11, a21)$ for C_2, distance between $\mathbf{q}$ and $\boldsymbol{\mu_1}$ is 1 and distance between $\mathbf{q}$ and $\boldsymbol{\mu_2}$ is 0; hence, $\mathbf{q}$ is nearer to C_2.

To avoid the less precision and the ambiguity of distance measure on the modes, we use the probability estimation of each categorical feature value to represent the cluster center and define a function to calculate the distance from instance to cluster center.

Let D be a set of categorical data described by p categorical features. Number of instances in D is n and instances are partitioned into k clusters. There are $d^{(i,j)}$ with different values $\omega_1, \omega_2, \ldots, \omega_{d\{i,j\}}$ for the lth feature A_l of the jth cluster C_j, $l \in \{1, 2, \ldots, p\}$, $j \in \{1, 2, \ldots, k\}$.

Definition 1. C_{j,x_l} represents the set of instances with value of x_l on the feature A_l in C_j, where $x_l \in \{\omega_1, \omega_2, \ldots, \omega_{d\{l,j\}}\} x_1 \in \{\omega_1, \omega_2, \ldots, \omega_{d\{l,j\}}\}$. The condition probability is estimated as follows:

$$P(x_l|j) = \frac{\left|C_{j,x_l}\right|}{\left|C_j\right|}, \tag{11}$$

$S_{j,l}$ is the summary of all values of A_l in C_j, defined as follows:

$$S_{j,l} = \{P(\omega_1|j), P(\omega_2|j), \ldots, P(\omega_d|j)\}. \tag{12}$$

Definition 2. The center of C_j is represented by the following vector:

$$\boldsymbol{\mu_j} = \left(S_{j,1}, S_{j,2}, \ldots, S_{j,p}\right). \tag{13}$$

Input: Current node training set D
Output: Divide D as $C_1, C_2, \ldots, C_k$, cluster centers $\mu_1, \mu_2, \ldots, \mu_k$, the weights $\mathbf{w}$
(1) Initialize the number of clusters k with the number of classes in D.
(2) Input D, call RELIEF-F to generate $\mathbf{w}$.
(3) Assign $w_{\max}$ with maximum in $\mathbf{w}$, excludes features whose weight is less than $\beta \cdot w_{\max}$, $\beta \in [0, 1]$, 0.2 by default.
(4) Initialize $\mu_1, \mu_2, \ldots, \mu_k$ by using the class centroids in D.
(5) **For** 1 to $l_{\max}$ **Do**
(6) Combining formulas (6) and (10), divide instances into $C_1, C_2, \ldots, C_k$
(7) Recalculate $\mu_1, \mu_2, \ldots, \mu_k$ according to formula (7).
(8) **If** $\mu_1, \mu_2, \ldots, \mu_k$ do not change significantly **Then break For**
(9) **End For**
(10) **Return** $C_1, C_2, \ldots, C_k$, $\mu_1, \mu_2, \ldots, \mu_k$ and $\mathbf{w}$

Algorithm 1: Multi_split.

Definition 3. $\text{diff}(A_l, \omega, S_{j,l})$ represents the distance between value ω and $S_{j,l}$ for A_l:

$$\text{diff}\left(A_l, \omega, S_{j,l}\right) = \begin{cases} 1 - P(\omega|j), & \text{if } \omega \in \{\omega_1, \omega_2, \ldots, \omega_d\}, \\ 1, & \text{others.} \end{cases} \tag{14}$$

Definition 4. $\text{dis\_}c(\mathbf{x_i}, \mu_\mathbf{j})$ represents the weighted distance between instance $\mathbf{x_i}$ and center $\mu_\mathbf{j}$:

$$\text{dis\_}c\left(\mathbf{x_i}, \boldsymbol{\mu_j}\right) = \sum_{l=1}^{p} \left(w_l \cdot \text{diff}\left(A_l, x_{i,l}, S_{j,l}\right)\right). \tag{15}$$

According to formula (15), in the above example, the weights of two features are 1. The distances between instance $\mathbf{q} = (a11, a21)$ and two cluster centers (μ_1 and μ_2) in Table 3 are $0.7 = 0.1 + 0.6$ and $1.2 = 0.6 + 0.6$, respectively. It means that $\mathbf{q}$ is closer to C_1, which is in accordance with the human's intuition.

To cluster categorical data, we use formula (13) to replace formula (7) in step 4 and step 7 of Algorithm 1 and formula (15) to formula (10) in step 6.

3.4. Mixed Features Data. For mixed data, the vector of cluster center consists of two parts: one is the means of numerical features and the other is the vector as shown in (13). In this case, we use (9) to calculate the distance from instance to cluster center, where dis_n and dis_c are obtained by (10) and (15), respectively. As the ratio of numerical and categorical features differs by the datasets, we choose γ in (9) that makes the most reduction of GINI index, where $\gamma \in \{0.1, 0.2, \ldots, 0.9\}$.

3.5. MSDT and Time Complexity Analysis. The multi_split function is prompted for node splits. Algorithm 2 describes the construction process of MSDT.

In step 2 of Algorithm 1, RELIEF-F is used to get the weights. Time complexity of RELIEF-F is $O(m \cdot p \cdot n \cdot \log_2 k_m)$, where p is the feature number, n is the instance number, m is the sampling number, and k_{nn} is the nearest neighbor number. In this paper, m is set $\log_2 n$, k_{nn} is set 1, and $\log_2 k_{\text{nn}}$ is negligible, so the time complexity of RELIEF-F in this paper is $O(p \cdot n \cdot \log_2 n)$.

Steps 4 to 9 of Algorithm 1 are the clustering process, and the time complexity is $O(I \cdot p \cdot n \cdot k)$, where k is cluster number and I is iteration number. When we use Algorithm 1 to split nodes, the max iterations Imax is 6; it means that time complexity may reach $O(6 \cdot p \cdot n \cdot k)$ in the worst case.

Considering the above two parts, the time complexity of Algorithm 1 is $O((6k + \log_2 n) \cdot p \cdot n)$. Compared with the time complexity of the classical axis-parallel splits, there is an extra k. When k is large, this algorithm is lower efficiency than the axis-parallel algorithms. Compared with binary splits, if the node numbers of the decision trees are the same, the operations in k-way splits are obviously less than in binary splits.

OC1 [21] is a classic oblique decision tree, whose time complexity is $O(p \cdot n^2 \cdot \log_2 n)$ in the worst case. In [25], the time complexities of HHCART(A) and HHCART(D) are $O((p + n \cdot \log_2 n) \cdot p^2 \cdot k)$ and $O((p + \log_2 n) \cdot p \cdot n \cdot k)$, respectively. In [22], the speed of FDT for splitting node is close to or even better than that of axis-parallel split method. The time complexity of this method is $O(p^2 \cdot n)$. Unfortunately, it can only be applied in binary classification problems.

In summary, when k is small, the efficiency of the proposed split method is close to classical axis-parallel split methods, and it is better than most oblique split methods.

4. Experiments

In this section, we use experimental results to demonstrate the effectiveness and performance of our proposed algorithm. In the first part, the experiments are used to illustrate the effectiveness of clustering, feature weighting, and the novel distance calculation method for categorical feature. The second parts compare MSDT with classical decision trees and another two oblique trees. Finally, we use a larger dataset covertype to compare with two axis-parallel trees.

4.1. Datasets. As shown in Table 4, the 20 UCI datasets [33] are used to evaluate the proposed algorithm, where the number of instances, the number of classes, and feature types

Table 3: The distribution of values.

	A_1 (feature 1)	A_2 (feature 2)
C_1 (cluster 1)	$a11$: 9, $a12$: 1	$a21$: 4, $a22$: 4, $a23$: 2
C_2 (cluster 2)	$a11$: 4, $a12$: 3. $a13$: 3	$a21$: 4, $a23$: 4, $a24$: 2

(numerical data 1–10, categorical data 11–15, and mixed data 16–20) are varied and are sufficiently representative to demonstrate the performance of MSDT. In column Features with $p_{(n)}$ & $p_{(c)}$, $p_{(n)}$ and $p_{(c)}$ is number of numerical features and categorical features, respectively. Abalone is treated as a 3-category classification problem (grouping classes 1–8, 9 and 10, and 11 on).

4.2. Comparison of Different Piecewise Linear Split Methods. The piecewise linear split methods can be summarized as two steps. First, find appropriate anchors. Then, divide instances according to the nearest anchor. On the basis of this approach, our proposed algorithm is improved in three aspects: feature weighting, clustering, and special categorical feature processing. This section combines these three changes into multiple functions and compares the performances in multiple types of data. These functions are shown in Table 5.

The pessimistic pruning algorithm is adopted after the decision trees are generated. In addition, the average results of all experiments are obtained by 10 repetitions of 10-fold cross-validation.

4.2.1. Numerical Data. In terms of numerical data, the proposed algorithm uses weighted k-means to optimize cluster center position. In order to demonstrate the role of clustering and feature weighting, we implement four different node split functions to generate decision trees. Fun0 directly uses the center of each class as the anchor. Instances are divided according to the nearest anchor. Euclidean distance is used for distance calculation. Fun1 also uses the center of each class as the anchor. However, in the process of selecting the nearest anchor for each instance, RELIEF-F is firstly used to calculate the weight of each feature. Then remove features whose weights are less than 1/5 of the maximum. Finally, the distance is calculated according to formula (10). Fun2 uses the center of each class as the initial cluster center of k-means and the outputs of k-means as the partition results. Fun3 combines Fun1 with Fun2 and is our proposed algorithm for numerical data.

Table 6 gives the classification accuracy of the 4 functions in 10 datasets, and the best entry in each row is bolded. As can be seen, Fun3 gets the best accuracy on 9 of 10 datasets and the average improvement is 4.16% higher than Fun0. In particular, the accuracy increases by more than 8% on Glass and Letter. The average accuracy of 10 datasets shows that Fun1 is about 1.07% higher than Fun0, and Fun3 is 1.39% higher than Fun2. The results show that feature weighting improves the classification performance. Fun2 is about 2.77% higher than Fun0, and Fun3 is 3.09% higher than Fun1. The reason for improvement is using clustering.

4.2.2. Categorical and Mixed Data. On the categorical and mixed data, we implement eight different split functions to generate decision trees. Fun0 directly uses the center of each class as the anchor. For categorical features, modes are used to replace the means of numerical features as the component of anchors. When calculating the distance between instance and anchor, the distance on each feature is calculated by formula (5) and then summed. The difference between Fun1 and Fun0 is that the weight of each feature is calculated by RELIEF-F. Then remove features whose weights are less than 1/5 of the maximum. The distance between the instance and the anchor is obtained by formula (15) and formula (9). Fun2 adds clustering process on Fun0. k-modes and k-prototypes are used for categorical data and mixed data, respectively. Fun3 combines Fun1 and Fun2. Fun4-7 corresponds to Fun0-3 respectively. On the categorical features, the calculation of cluster centers and distances adopts the method described in Section 3.3 (formulas (13) and (15), respectively). Fun7 is our proposed algorithm for categorical and mixed data.

Table 7 gives the classification accuracy of the 8 functions in 5 categorical datasets (*Balance*, *Car*, *Chess*, *Hayes*, and *MONK*) and 5 mixed datasets (*Abalone*, *CMC*, *Flags*, *TAE*, and *Zoo*), and the best entry in each row is bolded. Except for *CMC* and *Zoo*, Fun7 obtains the best accuracy, and the average is 11.77% higher than Fun0. As can be seen, Fun1 is better than Fun0, Fun3 is better than Fun2, Fun5 is better than Fun4, and Fun7 is better than Fun6. The average improvement is 5.37%. This is the contribution of feature weighting. It is shown that Fun2 is better than Fun0, Fun3 is better than Fun1, Fun6 is better than Fun4, and Fun7 is better than Fun5. The average improvement is 4.45%. The reason is the use of clustering. Meanwhile, we can see that Fun4-7 is averagely 1.48% better than Fun0-3. This improvement is the statistical distribution of feature values instead of modes.

4.3. Comparison with Other Decision Trees. In order to verify the performance of our proposed algorithm, we selected four decision trees: J48 (WEKA's implementation of C4.5), CART_{SL} (scikit-learn's implementation of optimal CART), OC1, and HHCART(A). Since CART_{SL} and OC1 do not support categorical features, we convert categorical features to numerical features using the One Hot method. The 10 repetitions of 10-fold cross-validation were used in our experiments to report the average accuracy and tree size of 5 classifiers on the test set. Friedman test and Nemenyi test will be used to analyze the algorithm difference.

The accuracy over the numerical datasets by each method is shown in Table 8. As can be seen, MSDT gets the best accuracy on 5 of 10 datasets and the average accuracy is 81.68%. It is 1.91%, 4.42%, 1.15%, and 1.81% higher than other four trees, respectively. In order to further demonstrate the differences of the classifiers, the Friedman test is used. We use the averages of the ranks of 5 classifiers on 10 datasets to calculate $F_F = 7.129032$. Here, with 5 algorithms and 10 datasets, F_F follows the $F-$distribution with 4 and 36 degrees of freedom, and the critical value is $F(4, 36) = 2.634$.

Input: training set D and the threshold value *minparent*
Output: the decision tree
(1) Create *node* according to the instances in D.
(2) **Procedure** *grow*(*node*)
(3) **If** $|D|$ is less than the *minparent* or the instances are not partitionable (All the instances are of the same class or have the same feature values) **Then**
(4) mark *node* as a leaf, and label it with the class of the majority of instances in D.
(5) **Return** *node*
(6) **End If**
(7) Call *multi_split* to get the cluster $C_1, C_2, \ldots, C_k$, $\mu_1, \mu_2, \ldots, \mu_k$, and $\mathbf{w}$.
(8) Save the values of $\mu_1, \mu_2, \ldots, \mu_k$, and $\mathbf{w}$ into the current *node* for the prediction.
(9) **For** $i = 1$ to k **Do**
(10) Create *node_i* according to instances in C_i, call *grow*(*node_i*).
(11) **End For**
(12) **End Procedure**
(13) Prune the *node*-rooted tree by pessimistic pruning algorithm.
(14) **Return** the *node*-rooted decision tree.

Algorithm 2: MSDT.

Table 4: Datasets.

No.	Name	Abb.	Instances	Features	Classes
1	Blood Transfusion Service Center	Blood	748	4	2
2	Glass identification	Glass	214	9	6
3	Image segmentation	Image	2130	19	7
4	Iris	Iris	150	4	3
5	Letter recognition	Letter	20000	16	26
6	Multiple features-mor	MF-mor	2000	6	10
7	Waveform database generator (version 1)	Wav1	5000	21	3
8	Waveform database generator (version 2)	Wav2	5000	40	3
9	Wine	Wine	178	13	3
10	Yeast	Yeast	1484	8	10
11	Balance scale	Balance	625	4	3
12	Car evaluation	Car	1728	6	4
13	Chess (King-Rook vs. King-Pawn)	Chess	3196	36	2
14	Hayes-Roth	Hayes	160	4	3
15	MONK's problems	MONK	432	6	2
16	Abalone	Abalone	4177	7&1	3
17	Contraceptive method choice	CMC	1473	2&7	3
18	Flags	Flags	194	10&19	8
19	Teaching assistant evaluation	TAE	151	1&4	3
20	Zoo	Zoo	101	15&1	7

Table 5: Split function description.

Function name	Function description		
	Feature weighting	Clustering	Cluster center and distance of categorical feature
Fun 0	✗	✗	k-modes
Fun 1	✓	✗	k-modes
Fun 2	✗	✓	k-modes
Fun 3	✓	✓	k-modes
Fun 4	✗	✗	Definitions 1–4
Fun 5	✓	✗	Definitions 1–4
Fun 6	✗	✓	Definitions 1–4
Fun 7	✓	✓	Definitions 1–4

Table 6: Comparison of the accuracy for different splitting functions on numerical data (%).

	Fun 0	Fun 1	Fun 2	Fun 3
Blood	76.44	76.78	77.02	**77.53**
Glass	61.68	63.46	66.92	**70.47**
Image	91.68	92.56	94.79	**96.01**
Iris	91.40	**96.00**	93.07	**96**
Letter	72.96	78.18	89.61	**91.77**
MF-mor	69.77	70.00	70.48	**71.13**
Wav1	**82.30**	81.76	80.92	**82.30**
Wav2	80.58	80.35	79.34	**80.65**
Wine	**95.73**	93.60	95.56	95.11
Yeast	52.63	53.19	55.14	**55.84**
Average	77.52	78.59	80.29	**81.68**

Table 7: Comparison of the accuracy for different splitting functions on categorical and mixed data.

	Fun 0	Fun 1	Fun 2	Fun 3	Fun 4	Fun 5	Fun6	Fun 7
Balance	72.51	70.75	73.42	73.47	68.94	71.04	69.23	**76.45**
Car	74.88	86.11	76.89	89.48	81.92	83.76	92.73	**96.45**
Chess	80.05	95.66	90.42	95.96	71.51	93.68	92.62	**99.17**
Hayes	71.00	76.88	72.00	77.88	59.25	71.56	66.69	**79.75**
MONK	73.98	85.23	83.94	88.03	88.80	97.41	88.17	**99.63**
Abalone	59.12	59.44	61.33	61.46	50.56	55.66	61.66	**62.84**
CMC	45.19	44.31	**49.04**	45.66	43.13	45.10	44.60	47.95
Flags	47.01	57.06	50.31	60.05	47.73	59.12	48.76	**62.42**
TAE	47.75	49.07	59.47	53.97	56.56	58.41	61.85	**63.77**
Zoo	95.64	**96.73**	95.74	96.53	95.54	96.04	95.74	96.34
Average	66.71	72.12	71.26	74.25	66.39	73.18	72.21	**78.48**

So, we reject the null hypothesis; namely, there are significant differences among the five classifiers. Nemenyi method is used for post hoc test. Critical interval (CD) is obtained by the following formula:.

$$\mathrm{CD} = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{16}$$

where k is the number of algorithms and N is the number of datasets. When $k = 5$, $N = 10$ and significance $\alpha = 0.05$, $q_\alpha = 2.728$, the calculated critical interval CD = 1.92899. In the case of these conclusions, the MSDT and OC1 have obvious performance advantages compared with CARTSL.

The accuracy over the categorical and mixed datasets by each method is shown in Table 9. MSDT gets the best accuracy on 4 of 10 datasets and the average accuracy is 78.48%. It is 3.62%, 1.1%, 1.56%, and 1.88% higher than four other trees, respectively. We use the averages of the ranks of 5 classifiers on 10 datasets to calculate $F_F = 1.48951$. Here, the critical value is $F(4, 36) = 2.634$. So, we cannot reject the null hypothesis; namely, there is no significant difference among the five classifiers. In the case of these conclusions, on categorical and mixed data, the advantages of three multivariate decision trees over two univariate decision trees are not so obvious. Especially in OC1, one categorical feature is transformed into multiple numerical features by One Hot method, which greatly increases the dimension of feature space. In the new feature space, the data becomes very sparse, and OC1 cannot find a suitable split hyperplanes.

The tree size over 20 datasets by each method is shown in Table 10. In terms of the complexity of model structure, the average number of nodes in three multivariate decision tree is lower than the other two univariate decision trees. We use the averages of the ranks of 5 classifiers on 20 datasets to calculate $F_F = 3.35294$. Here, with 5 algorithms and 20 datasets, F_F follows the *F-distribution* with 4 and 76 degrees of freedom, and the critical value is $F(4, 76) = 2.492$. So, we reject the null hypothesis. Nemenyi method is used for post hoc test. Critical interval is obtained; $CD = 1.364$. In the case of these conclusions, the MSDT has obvious performance advantages compared with J48.

4.4. Comparison on Big Data. The data set *covertype* comes from UCI [35], which is a 7-classification problem, which includes 581012 instances and 54 features. 10 of 54 features are numerical, and the remainders are Boolean. MSDT and J48 regard Boolean as categorical features, and $\mathrm{CART}_{\mathrm{SL}}$ is regarded as numerical features. The 10 repetitions of 10-fold cross-validation are used. Table 11 provides the accuracy of the three classifiers, the size of the tree, and the time to build the tree.

The three classifiers achieve similar accuracies on *covertype*. In terms of tree size, MSDT has the least number of nodes. The running time provided in Table 11 is the time to build a tree and does not include the time consumed by loading data and testing. J48 runs slower than $\mathrm{CART}_{\mathrm{SL}}$, which does not mean that there is a significant difference in time complexity between the two algorithms. The difference

Table 8: The accuracy of five classifiers on numerical data.

Dataset	Accuracy % (rank)				
	J48	$CART_{SL}$	OC1	HHCART(A)	MSDT
Blood	**78.09 ± 0.50(1)**	70.51 ± 0.70(5)	77.34 ± 0.56(3)	76.50 ± 0.45(4)	77.53 ± 0.68(2)
Glass	66.73 ± 2.84(3)	66.59 ± 2.06(4)	68.43 ± 2.80(2)	64.02 ± **3.14(1)**	
Image	96.55 ± 0.26(3)	96.32 ± 0.39(4)	96.78 ± 0.33(2)	**97.22 ± 0.25(1)**	96.01 ± 0.40(5)
Iris	95.13 ± 0.73(3)	95.07 ± 0.61(4)	94.67 ± 0.62(5)	95.33 ± 0.63(2)	**96.00 ± 0.00(1)**
Letter	88.09 ± 0.23(5)	88.19 ± 0.17(4)	89.72 ± 0.20(2)	88.23 ± 0.22(3)	**91.77 ± 0.16(1)**
MF-mor	**72.01 ± 0.62(1)**	65.11 ± 0.43(5)	70.15 ± 0.48(3)	69.83 ± 0.63(4)	71.13 ± 0.43(2)
Wav1	76.50 ± 0.32(4)	75.38 ± 0.52(5)	80.07 ± 0.33(2)	79.83 ± 0.33(3)	**82.30 ± 0.32(1)**
Wav2	75.29 ± 0.61(4)	74.27 ± 0.43(5)	79.68 ± 0.56(2)	79.01 ± 0.45(3)	**80.65 ± 0.42(1)**
Wine	93.54 ± 0.88(3)	90.00 ± 1.46(5)	92.54 ± 0.98(4)	94.47 ± 0.86(2)	**95.11 ± 1.23(1)**
Yeast	55.77 ± 0.93(3)	51.14 ± 1.01(5)	**55.90 ± 1.06(1)**	54.28 ± 1.11(4)	55.84 ± 0.93(2)
Average	79.77(3)	77.26(4.6)	80.53(2.6)	79.87(3.1)	**81.68(1.7)**

Table 9: The accuracy of five classifiers on categorical and mixed data.

Dataset	Accuracy % (rank)				
	J48	$CART_{SL}$	OC1	HHCART(A)	MSDT
Balance	64.00 ± 1.18(5)	76.91 ± 0.82(3)	80.36 ± 1.49(2)	**82.82 ± 1.12(1)**	76.45 ± 2.05(4)
Car	92.52 ± 0.63(5)	**97.31 ± 0.22(1)**	96.27 ± 0.76(4)	96.91 ± 0.53(2)	96.45 ± 0.23(3)
Chess	99.41 ± 0.10(2)	**99.54 ± 0.06(1)**	98.58 ± 0.15(5)	99.23 ± 0.17(3)	99.17 ± 0.21(4)
Hayes	75.13 ± 1.55(5)	**83.00 ± 1.52(1)**	79.26 ± 1.56(3)	76.37 ± 1.16(4)	79.75 ± 2.14(2)
MONK	96.11 ± 1.54(3)	91.94 ± 1.51(5)	93.37 ± 1.53(4)	98.57 ± 1.58(2)	**99.63 ± 0.85(1)**
Abalone	61.30 ± 0.45(2)	57.52 ± 0.54(5)	60.05 ± 0.34(3)	59.29 ± 0.43(4)	**62.84 ± 0.28(1)**
CMC	**51.61 ± 0.66(1)**	47.40 ± 0.47(4)	46.92 ± 1.46(5)	47.83 ± 0.73(3)	47.95 ± 1.38(2)
Flags	**64.34 ± 3.34(1)**	61.96 ± 1.03(3)	60.35 ± 3.62(5)	61.73 ± 1.49(4)	62.42 ± 2.04(2)
TAE	51.43 ± 4.67(5)	62.91 ± 1.87(3)	63.10 ± 3.97(2)	53.26 ± 4.57(4)	**63.77 ± 3.79(1)**
Zoo	92.77 ± 0.01(3)	95.35 ± 0.00(2)	90.98 ± 0.33(4)	89.99 ± 0.02(5)	**96.34 ± 0.45(1)**
Average	74.86(3.2)	77.38(2.8)	76.92(3.7)	76.60(3.2)	**78.48(2.1)**

Table 10: The tree size of five classifiers.

Dataset	Tree size (rank)				
	J48	$CART_{SL}$	OC1	HHCART(A)	MSDT
Blood	12.60 ± 0.87(2)	179.64 ± 1.00(5)	**7.88 ± 0.75(1)**	16.56 ± 0.88(3)	32.72 ± 2.50(4)
Glass	46.68 ± 1.66(5)	44.93 ± 0.56(3)	32.65 ± 0.78(2)	**27.73 ± 0.67(1)**	46.55 ± 2.15(4)
Image	80.28 ± 1.28(4)	69.22 ± 0.79(2)	**57.62 ± 0.93(1)**	70.43 ± 1.46(3)	161.59 ± 4.57(5)
Iris	8.40 ± 0.24(4)	8.56 ± 0.14(5)	5.10 ± 0.08(2)	5.60 ± 0.12(3)	**4.00 ± 0.00(1)**
Letter	2326.72 ± 9.04(5)	2098.23 ± 9.34(3)	2285.22 ± 9.81(4)	**1795.23 ± 8.91(1)**	1893.41 ± 11.64(2)
MF-mor	162.40 ± 5.54(2)	462.94 ± 1.89(5)	**153.32 ± 3.55(1)**	212.97 ± 4.74(4)	174.07 ± 1.29(3)
Wav1	546.74 ± 7.62(5)	484.45 ± 2.61(4)	470.26 ± 6.17(3)	465.34 ± 6.62(2)	**178.12 ± 5.12(1)**
Wav2	585.42 ± 9.45(5)	453.02 ± 1.73(2)	481.54 ± 8.05(4)	480.38 ± 9.17(3)	**126.18 ± 10.69(1)**
Wine	9.54 ± 0.31(4)	10.47 ± 0.62(5)	9.25 ± 0.30(3)	7.83 ± 0.27(2)	**4.00 ± 0.72(1)**
Yeast	328.70 ± 4.12(4)	463.38 ± 2.33(5)	260.47 ± 9.81(2)	303.25 ± 6.40(3)	**101.52 ± 9.00(1)**
Balance	42.05 ± 1.31(2)	140.83 ± 1.47(4)	159.93 ± 2.36(5)	57.82 ± 1.42(3)	**31.13 ± 5.48(1)**
Car	170.45 ± 1.27(4)	97.18 ± 1.11(2)	**91.46 ± 1.19(1)**	186.54 ± 2.28(5)	141.60 ± 1.32(3)
Chess	54.66 ± 1.07(4)	52.15 ± 0.64(2)	53.70 ± 0.96(3)	**49.62 ± 1.38(1)**	106.96 ± 2.90(5)
Hayes	27.13 ± 0.55(4)	22.09 ± 0.33(3)	20.73 ± 0.47(2)	29.75 ± 0.51(5)	**18.63 ± 0.94(1)**
MONK	39.14 ± 1.83(3)	59.09 ± 6.33(5)	56.28 ± 5.92(4)	20.93 ± 3.41(2)	**18.54 ± 1.08(1)**
Abalone	589.00 ± 18.45(3)	965.09 ± 2.66(5)	679.45 ± 13.25(4)	401.11 ± 9.52(2)	**270.67 ± 19.72(1)**
CMC	247.16 ± 6.21(3)	611.94 ± 3.87(5)	538.95 ± 8.36(4)	**108.92 ± 4.78(1)**	139.64 ± 6.33(2)
Flags	62.63 ± 2.07(5)	45.01 ± 0.52(3)	39.87 ± 1.28(2)	54.28 ± 2.11(4)	**26.86 ± 1.74(1)**
TAE	93.01 ± 3.75(4)	56.40 ± 0.53(2)	**50.57 ± 0.66(1)**	68.82 ± 0.72(3)	114.40 ± 1.76(5)
Zoo	11.00 ± 0.00(3)	**9.65 ± 0.00(1)**	12.07 ± 0.23(5)	10.53 ± 0.07(2)	11.08 ± 0.57(4)
Average	272.19(3.75)	316.71(3.55)	273.32(2.7)	218.68 (2.65)	**180.08 (2.35)**

may be caused by the different development language. There are two reasons why MSDT gets the most expensive time consumption. One is that the time complexity of our proposed method is higher than that of the axis-parallel methods when dividing a node. The other one is that the axis-parallel methods mainly perform relational operations,

Table 11: The performance of three classifiers on covertype

	J48	$CART_{SL}$	MSDT
Accuracy (%)	94.58 ± 0.09	94.30 ± 0.11	94.64 ± 0.12
Tree size	28348.11 ± 274.83	51728.89 ± 347.59	25736.90 ± 235.86
Time (s)	71.61 ± 1.52	45.58 ± 0.56	150.12 ± 2.66

for instance, "<." Our method needs to calculate a large number of distances, which requires arithmetic operations of real number. Although multiway splits can reduce the number of times to split nodes, the time consumed by our method is about 2 to 3 times that of the axis-parallel methods from the experimental results.

5. Conclusion

The decision trees generated by the oblique splits often have better generalization ability and fewer nodes. However, most oblique split methods are time-consuming and cannot be directly used for categorical data, and some of these methods can only be used for binary classification problems. Our proposed algorithm MSDT uses feature weighting and clustering to multiway splits of nonleaf nodes, which can be directly applied to multiclassification problems. Meanwhile, it has a time complexity similar to that of the axis-parallel algorithms. In addition, we give the representation of cluster center and the distance from instance to cluster center, which enables clustering to be used in categorical and mixed data. Experimental results show that MSDT has a good generalization accuracy on multiple types of data.

Data Availability

The data used to support the findings of this study are included in the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the National Nature Science Foundation of China under Grants 61772101, 61170169, and 61602075 and in part by the Ph.D. Scientific Research Starting Foundation of Liaoning Province under Grant 20180540084.

References

[1] A. C. Sick-Samuels, K. E. Goodman, G. Rapsinski et al., "A decision tree using patient characteristics to predict resistance to commonly used broad-spectrum antibiotics in children with gram-negative bloodstream infections," *Journal of the Pediatric Infectious Diseases Society*, vol. 9, no. 2, p. 142, 2019.

[2] S. Guney and A. Atasoy, "Freshness classification of horse mackerels with E-Nose system using hybrid binary decision tree structure," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 3, pp. 1–17, 2020.

[3] Z. Liu, L. Wang, X. Li, and X. Ji, "Optimize x265 rate control: an exploration of lookahead in frame bit allocation and slice type decision," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2558–2573, 2019.

[4] Y. Wang, S.-T. Xia, and J. Wu, "A less-greedy two-term Tsallis Entropy Information Metric approach for decision tree classification," *Knowledge-Based Systems*, vol. 120, pp. 34–42, 2017.

[5] Z.-H. Zhou and J. Feng, "Deep forest: towards an alternative to deep neural networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, pp. 3553–3559, Melbourne, Australia, August 2017.

[6] X. Guan, J. Liang, Y. Qian, and J. Pang, "A multi-view OVA model based on decision tree for multi-classification tasks," *Knowledge-Based Systems*, vol. 138, pp. 208–219, 2017.

[7] L. Zhang and P. N. Suganthan, "Oblique decision tree ensemble via multisurface proximal support vector machine," *IEEE Transactions on Cybernetics*, vol. 45, no. 10, p. 1, 2015.

[8] R. Katuwal, P. N. Suganthan, and L. Zhang, "An ensemble of decision trees with random vector functional link networks for multi-class classification," *Applied Soft Computing*, vol. 70, pp. 1146–1153, 2018.

[9] R. Katuwal and P. N. Suganthan, "Enhancing multi-class classification of random forest using random vector functional neural network and oblique decision surfaces," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Rio de Janeiro, Brazil, July 2018.

[10] Z. Liu, T. Wen, W. Sun, and Q. Zhang, "Semi-supervised self-training feature weighted clustering decision tree and random forest," *IEEE Access*, vol. 8, pp. 128337–128348, 2020.

[11] L. Hyafil and R. L. Rivest, "Constructing optimal binary decision trees is NP-complete," *Information Processing Letters*, vol. 5, no. 1, pp. 15–17, 1976.

[12] H. E. L. Cagnini, R. C. Barros, and M. P. Basgalupp, "Estimation of distribution algorithms for decision-tree induction," in *Proceedings of the IEEE Congress on Evolutionary Computation*, June 2017.

[13] R. Rivera-Lopez and J. Canul-Reich, "Construction of near-optimal axis-parallel decision trees using a differential-evolution-based approach," *IEEE Access*, vol. 6, pp. 5548–5563, 2018.

[14] M. P. Basgalupp, R. C. Barros, A. C. P. L. F. de Carvalho, A. A. Freitas, and D. D. Ruiz, "LEGAL-tree: a lexicographic multi-objective genetic algorithm for decision tree induction," in *Proceedings of the ACM Symposium on Applied Computing*, March 2009.

[15] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[16] J. R. Quinlan, "C4.5: programs for machine learning," *Machine Learning*, vol. 16, no. 3, pp. 235–240, 1994.

[17] W. Buntine, "Learning classification trees," *Statistics and Computing*, vol. 2, no. 2, pp. 63–73, 1992.

[18] A. Cherfi, A. K. Nouira, and A. Ferchichi, "Very fast C4.5 decision tree algorithm," *Applied Artificial Intelligence*, vol. 32, no. 2, pp. 119–137, 2018.

[19] R. S. Bucy and R. S. Diesposti, "Decision tree design by simulated annealing," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 27, no. 5, pp. 515–534, 1993.

[20] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho et al., "A survey of evolutionary algorithms for decision-tree induction," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 3, pp. 291–312, 2012.

[21] S. K. Freitas, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *Journal of Artificial Intelligence Research*, vol. 2, no. 1, pp. 1–32, 1996.

[22] A. López-Chau, J. Cervantes, L. López-García, and F. G. Lamont, "Fisher's decision tree," *Expert Systems with Applications*, vol. 40, no. 16, pp. 6283–6291, 2013.

[23] Y. Freund and L. Mason, "The alternating decision tree learning algorithm," in *Proceeding of the International Conference on Machine Learning*, Bled, Slovenia, June 1999.

[24] K. S. Hong, P. L. Ooi, C. K. Ye et al., "Multivariate alternating decision trees," *Pattern Recognition*, vol. 50, no. C, pp. 195–209, 2016.

[25] D. C. Wickramarachchi, B. L. Robertson, M. Reale, C. J. Price, and J. Brown, "HHCART: an oblique decision tree," *Computational Statistics & Data Analysis*, vol. 96, pp. 12–23, 2015.

[26] S. Shah and P. S. Sastry, "New algorithms for learning and pruning oblique decision trees," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 29, no. 4, pp. 494–505, 1999.

[27] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht, "On oblique random forests," in *Proceedings of the European Conference on Machine Learning & Knowledge Discovery in Databases*, Bled, Slovenia, September 2011.

[28] R. Gnanadesikan, *Methods For Satistical Data Analysis of Multivariate Observations*, Wiley, Hoboken, NJ, USA, 1977.

[29] K. Kira and L. A. Rendell, "A Practical approach to feature selection," in *Proceedings of the Ninth International Workshop on Machine Learning (ML 1992)*, pp. 249–256, Morgan Kaufmann Publishers Inc., Aberdeen, Scotland, July 1992.

[30] I. Kononenko, "Estimating attributes: analysis and extension of relief," in *Proceeding of the 1994 European Conference on Machine Learning*, pp. 171–182, Catania, Italy, April 1994.

[31] Z. Huang, "Extensions to the *k*-means algorithm for clustering large data sets with categorical values," *Data Mining and Knowledge Discovery*, vol. 2, no. 3, pp. 283–304, 1998.

[32] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[33] M. Lichman, "UCI machine learning repository," 2013, http://archive.ics.uci.edu/ml.