Segment Based Decision Tree Induction With Continuous Valued Attributes

Ran Wang, Member, IEEE, Sam Kwong, Fellow, IEEE, Xi-Zhao Wang, Fellow, IEEE, and Qingshan Jiang

Abstract—A key issue in decision tree (DT) induction with continuous valued attributes is to design an effective strategy for splitting nodes. The traditional approach to solving this problem is adopting the candidate cut point (CCP) with the highest discriminative ability, which is evaluated by some frequency based heuristic measures. However, such methods ignore the class permutation of examples in the node, and they cannot distinguish the CCPs with the same or similar frequency information, thus may fail to induce a better and smaller tree. In this paper, a new concept, i.e., segment of examples, is proposed to differentiate the CCPs with same frequency information. Then, a new hybrid scheme that combines the two heuristic measures, i.e., frequency and segment, is developed for splitting DT nodes. The relationship between frequency and the expected number of segments, which is regarded as a random variable, is also given. Experimental comparisons demonstrate that the proposed scheme is not only effective to improve the generalization capability, but also valid to reduce the size of the tree.

Index Terms—Classification, continuous valued attributes, decision tree (DT) induction, segment.

I. INTRODUCTION

INDUCTION of decision trees (DTs) is a technique of supervised learning, which builds up a knowledge-based expert system by inductive inference from examples. Due to a good interpretability and simple implementation, DTs have been utilized in various application domains such as fuzzy rule extraction [9], [11], [19], [41], ensemble learning [2], user authentication [37], anomaly detection [16], sample selection [40], monotonic classification [18], object ranking [20], and uncertainty analysis [38], etc. Recent developments of DTs could be found from the literature, such as multivariate DT [1], cost-sensitive DT [27], fuzzy DT [24]–[26], [32], geometric DT [28], and DT for handling continuous label [17].

DTs can easily produce some well-organized classification rules and have relatively low computational loads, thus are treated as powerful classification tools. As it is mentioned in [31], any effective methodology of supervised learning must have its inductive bias. The inductive bias of DT proposed by Quinlan [33] is that we prefer a smaller tree to a bigger tree when both of them are acceptable. This bias is supported by an old philosophic idea, i.e., Occam’s razor [4], which clearly states that a model should be as simple as possible.

Traditional DT induction algorithms are typically designed for the data with symbolic/discrete valued attributes. For the one with continuous valued attributes, discretization must be conducted before or during the tree growth [5], [12]. Discretization before the tree growth is simple and easy to carry out, but the performance is poor since it neglects the relationship between the conditional attributes and the decision attribute. Discretization during the tree growth follows some guidelines given by the decision attribute, thus can achieve better performances. The main task in this kind of discretization is to split the currently chosen attribute into several intervals such that the discriminative ability on the training examples is high. Along this direction, one can further adopt binary splitting or multiple splitting [22], [29], which respectively divide the attribute into two or more intervals. The well-behavedness of multiple splitting have been demonstrated in many works [3], [6], [13], [15], but the inductive procedure is complex and the size of the induced tree is large. Thus, we only dealt with the typical binary splitting in this paper. Obviously, the trees generated are of two branches.

The induction of DT is a recursive process that follows a top-down approach by repeated splits of the training set. Generally, there are two key issues during the tree growth.

1) One is how to judge a leaf node.

2) The other is how to split a nonleaf node [8].

Usually, a leaf node is determined if its class purity is higher than a given coefficient, or the number of examples in it is smaller than a given threshold. As for splitting a nonleaf node, the typical solution is to sort the examples according to each attribute, evaluate all the possible splits by a certain heuristic measure, and select the one with the highest discriminative ability. The earliest method is known as IDE3 [21], which selects the split with the highest information gain. However, IDE3 is specially designed for discrete attributes, and tends...
to select the one with more values, which may lead to the over-fitting problem. Thus, C4.5 [14], [33], [34] is proposed to improve IDE3, which replace the criterion of information gain by gain ratio, and is extended to handle both discrete and continuous attributes. Both IDE3 and C4.5 are based on the heuristic of information entropy [36]. Later, classification and regression trees (CART) algorithm [7], [23] is proposed based on the heuristic of Gini-index by selecting the split that can maximally reduce the degree of sample disorder, and chi-squared automatic interaction detection (CHAID) algorithm [39], [43] is proposed based on the Chi square detection. The CART and CHAID algorithms have similar performances on many problems, however, CART is more effective with continuous attributes, and CHAID is designed for discrete attributes. Besides, in order to improve the learning efficiency, supervised learning in quest (SLIQ) algorithm [30] is proposed for classifying large-scale datasets with a pre-sorting stage, and scalable parallelizable induction (SPRINT) algorithm [35] is proposed by removing all the memory restrictions. Both SLIQ and SPRINT are based on the heuristic of Gini-index.

It is noteworthy that all the above introduced methods adopt frequency based heuristics, they consider the purity of a node [10] during the induction process, but ignore the class distribution/permutation of the sorted examples. In this case, when two or more possible splits have the same or similar discriminative abilities, they may fail to select a better one for the benefits of the further splitting on the branches. In other words, when the frequency information of one split is identical to another, it is possible that their class permutations are quite different. Obviously, these two splits cannot be differentiated by the frequency based measures, but differentiating them is helpful in generating a compact and high-performance tree, which is in accord with the aforementioned inductive bias.

Motivated by these facts, a new concept, i.e., segment of examples, is proposed in this paper. This concept takes the class permutation into consideration, thus can effectively differentiate the cases with similar or same frequency information. By jointly using the frequency and segment, a new heuristic measure for splitting nodes is proposed, and a hybrid scheme for DT induction is developed. Furthermore, the relationship between frequency and segment is discussed. This relationship demonstrates that the expected number of segments, which is regarded as a random variable, has some common features with the frequency based heuristic measures such as information entropy and Gini-index.

The rest of this paper is organized as follows: in Section II, some basic concepts in DT induction with continuous valued attributes are reviewed, and the common characteristics of frequency based heuristic measures are summarized; in Section III, the concept of segment is introduced, and the algorithm for evaluating the number of segments in a node is presented; in Section IV, the frequency and segment are combined to develop a new hybrid scheme for splitting nodes, then some related analyzes are described in detail; in Section V, experimental comparisons demonstrate the effectiveness of the scheme in reducing the tree size and improving the learning accuracy; and finally, conclusions are given in Section VI.

II. DT Induction With Continuous Valued Attributes

In this section, we introduce some basic concepts, as well as the framework of frequency based DT induction model with continuous valued attributes.

A. Basic Concepts

In a DT, each node represents a set of examples. A node is called a leaf node if it cannot be split, and a nonleaf node otherwise. Let \( S = \{e_1, e_2, \ldots, e_N\} \) be a node with \( N \) examples. Each example in \( S \) is represented by a group of conditional attributes \( A = \{A_1, A_2, \ldots, A_m\} \) and a decision attribute \( C \in \{C_1, C_2, \ldots, C_L\} \). Each conditional attribute is also called an available splitting attribute (ASA). The \( i \)-th example in \( S \) is expressed as \( e_i = \{a_{i1}, a_{i2}, \ldots, a_{im}, c_i\} \) where \( a_{ij} \) is written as \( A_j(e_i) \) and denotes the value of the \( i \)-th attribute with respect to the \( j \)-th attribute; \( c_i \) is written as \( C(e_i) \) and denotes the class label of the \( i \)-th example. Suppose all the conditional attributes are continuous, then we have \( A_j(e_i) \in R \) where \( j = 1, \ldots, m \).

Definition 1 (Cut Point): Let \( A_j \) be a continuous valued attribute whose values are restricted to \([\min(A_j), \max(A_j)]\). Each point \( x \in (\min(A_j), \max(A_j)) \) divides the interval \([\min(A_j), \max(A_j)]\) into two parts, i.e., \([\min(A_j), x]\) and \([x, \max(A_j)]\). We call \( x \) a cut point of attribute \( A_j \).

Obviously, the number of cut points for a continuous valued attribute is countless, usually, we only consider a potential subset as follows.

Definition 2 (Candidate Cut Point): Let \( S = \{e_1, e_2, \ldots, e_N\} \) be a node with \( N \) examples and \( A_j \) be a continuous valued attribute. Suppose all the examples in \( S \) are ranked by ascending values of \( A_j \), i.e., \( A_j(e_1) < A_j(e_2) < \ldots < A_j(e_N) \). The midpoint of any two adjacent values in this order is considered as a candidate cut point (CCP) of attribute \( A_j \) with respect to \( S \).

We denote \( \text{CCP}(S, A_j) \) as the set that contains all the CCPs of \( A_j \) with respect to \( S \), then

\[
\text{CCP}(S, A_j) = \left\{ x_j | x_j = \frac{A_j(e_i) + A_j(e_{i+1})}{2}, i = 1, \ldots, N - 1 \right\}
\]

(1)

If \( C(e_i) = C(e_{i+1}) \), i.e., the two examples \( e_i \) and \( e_{i+1} \) belong to the same class, we call the cut point \( x_{ji} \) a stable cut. Otherwise, if \( C(e_i) \neq C(e_{i+1}) \), i.e., the two examples belong to different classes, we call it an unstable cut [42].

Definition 3 (Partition): Let \( S = \{e_1, e_2, \ldots, e_N\} \) be a node with \( N \) examples, \( A_j \) be a continuous valued attribute, and \( x_{ji} \) be a CCP of \( A_j \) with respect to \( S \). If \( S_1 = \{e \in S | A_j(e) \leq x_{ji}\} \) and \( S_2 = \{e \in S | A_j(e) > x_{ji}\} \), then \( \{S_1, S_2\} \) is called a partition of \( S \) induced by \( x_{ji} \).

B. Frequency Based Heuristic Measures

Given that \( \{S_1, S_2\} \) is a partition of node \( S \) induced by \( x_{ji} \), the information gain of \( x_{ji} \) in node \( S \) is defined as

\[
\text{Gain}(S, x_{ji}) = f(S) - \sum_{k=1}^{2} \left| \frac{S_k}{|S|} \right| f(S_k)
\]

(2)

where \(|S|\) represents the number of examples in set \( S \), and \( f(S) \) is a function that measures the impurity of class labels.
in $S$. The general form of $f(S)$ is $f(S) = f(S; p_1, p_2, \ldots, p_L)$, where $p_i$ is the frequency of the $i$-th class, and $L$ is the number of classes in $S$. Clearly, $\sum_{i=1}^{L} p_i = 1$.

There exists many forms for function $f(S) = f(S; p_1, p_2, \ldots, p_L)$. Two commonly used ones are information entropy and Gini-index.

1) Information Entropy: Information entropy was first proposed by Shannon [36] in 1948 to measure the amount of information. It was used by Quinlan [33] to measure the impurity of a node in DT induction. The entropy of node $S$ is defined as

$$f(S) = -\sum_{i=1}^{L} p_i \log_2 p_i.$$  

Clearly, the more imbalance the frequency distribution is, the smaller the entropy will be. When all the examples are from the same class, i.e., $p_i = 1$ for a certain $l \in \{1, \ldots, L\}$, entropy arrives its minimum. When the numbers of examples from all the classes are equivalent, i.e., $p_i = 1/L$ for $l = 1, \ldots, L$, entropy arrives its maximum.

2) Gini-Index: Gini-index was first proposed by the Italian economist Corrado Gini in 1912 to measure the income divergence level. It was used by Breiman et al. [7] to measure the class impurity of a set. The Gini-index of set $S$ is defined as

$$f(S) = 1 - \sum_{i=1}^{L} p_i^2.$$  

Gini-index has similar characteristics to entropy, i.e., it arrives its minimum when all the examples belong to the same class, and maximum when examples from each class are with equal probability.

3) Other Heuristic Measures: Other than Gini-index and information entropy, Wang et al. [42] proposed three new frequency based measures, i.e., sin measure defined as

$$f(S) = \sum_{i=1}^{L} \sin^2(\pi p_i)$$  

sqrt measure defined as

$$f(S) = \sqrt{\prod_{i=1}^{L} p_i}$$  

and psin measure defined as

$$f(S) = \sum_{i=1}^{L} p_i \sin(\pi p_i).$$

For binary problem, $f(S)$ could be rewritten as $f(p)$, where $p$ is the frequency of positive class in $S$. Then, the measure functions (4)~(7) can respectively degenerate to the following:

$$f(p) = -p \log_2 p - (1 - p) \log_2 (1 - p)$$  

$$f(p) = -2p^2 + 2p$$  

$$f(p) = \sin^2(\pi p) + \sin^2(\pi (1 - p))$$  

$$f(p) = \sqrt{p(1 - p)}$$  

$$f(p) = p \sin(\pi p) + (1 - p) \sin(\pi (1 - p)).$$

It can be seen from (8) that the impurity measure function $f(p)$ has the following features, i.e., it is defined on $[0, 1]$, symmetric at $p = 0.5$, strictly increasing in $(0, 0.5)$, strictly decreasing in $(0.5, 1)$, and convex ($d^2f/dp^2(p) < 0$. The relationships between frequency $p$ and the five heuristic measures in (8) are shown in Fig. 1.

C. Framework of C4.5 Algorithm

Different DT algorithms have similar induction framework. Among them, C4.5 is treated as a powerful one. In C4.5, information entropy, i.e., (3), is adopted as the frequency measure, and the criterion of information gain, i.e., (2), is replaced by the gain ratio, which is defined as the ratio of the information gain to its split information. The split information is calculated as

$$\text{Split}(S, x_{ji}) = -\frac{|S_1|}{|S|} \log_2 \frac{|S_1|}{|S|} - \frac{|S_2|}{|S|} \log_2 \frac{|S_2|}{|S|}$$  

and the gain ratio is measured by

$$\text{SP}(S, x_{ji}) = \frac{\text{Gain}(S, x_{ji})}{\text{Split}(S, x_{ji})}$$

where the CCP with the highest $\text{SP}$ is selected to split $S$.

Besides, in order to tackle the over-partitioning problem, a node will be treated as a leaf node when the number of examples in it is smaller than a given threshold $\tilde{N}$, and the output is decided by the class label with the highest frequency in it. The basic framework of C4.5 with continuous valued attributes is then described in Algorithm 1.

In Algorithm 1, we assume that there are no duplicated values in an attribute. However, this assumption does not hold in most practical problems. In this case, we may give a constraint to Algorithm 1, i.e., when the examples in node $S$ are sorted with respect to attribute $A_j$, only the CCPs that are between two different attribute values are feasible for splitting.

III. SEGMENT IN EXAMPLE QUEUE AND BAR

In this section, we first present our motivation, then we propose some definitions regarding the segment based heuristic method, finally we give an illustrative example to compute the number of segments in a node.

A. Motivation

Consider a node with a set of examples, it is unnecessary to discuss the distribution or permutation when the examples cannot be sorted. However, if the examples can be sorted according to different attributes, there may have some useful information for classification. We first look at a binary example indicated in Fig. 2.

In Fig. 2, $S_1$ and $S_2$ represent two different class permutations of examples in a node with exactly the same frequency...
Algorithm 1: C4.5 DT With Continuous Valued Attributes

**Input:** Training examples \( \{ e_i \}_{i=1}^N \) with \( m \) continuous valued attributes \( A = \{ A_j \}_{j=1}^m \) and one decision attribute \( C \in \{ C_j \}_{j=1}^m \); threshold number \( \hat{N} \) to stop splitting a node.

**Output:** A binary DT.

1. Initialize \( \Omega \) as an empty set;
2. Consider the set of all examples as the root-node, and add it to \( \Omega \);
3. while \( \Omega \) is not empty do
   4. Select a node from \( \Omega \), denoted by \( S \);
   5. if \( |S| < \hat{N} \) then
      6. Treat \( S \) as a leaf node and assign it the class labels \( l^* = \arg \max_{i=1,...,L} p_i \);
   7. else
      8. For each ASA \( A_j, j = 1, \ldots, m \), sort the examples with ascending order;
      9. Get the CCPs of each ASA based on (1), i.e., \( x_{ji} \), where \( j = 1, \ldots, m \) and \( i = 1, \ldots, N-1 \);
     10. Calculate the splitting performance of each CCP based on (10), i.e., \( SP(S, x_{ji}) \);
     11. Find the optimal splitting attribute \( A_j^* \) and its optimal CCP \( x_{j*} \), where \( (j^*, r^*) = \arg \max_{j,i} SP(S, x_{ji}) \);
     12. Split \( S \) into two child-nodes by \( x_{j*} \), i.e., \( S_1 = \{ e \in S | A_j(e) \leq x_{j*} \} \) and \( S_2 = \{ e \in S | A_j(e) > x_{j*} \} \);
     13. for \( i = 1, 2 \) do
         14. if all the examples in \( S_i \) are from the same class \( l_i^* \) then
             15. Treat \( S_i \) as a leaf node and assign it the class label \( l_i^* \);
         16. else
             17. Add \( S_i \) to \( \Omega \);
         18. end
     19. end
   20. Remove \( S \) from \( \Omega \);
21. return the constructed tree.

\[
S_1: 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 2 \ 2 \ 2 \ 2 \\
S_2: 1 \ 2 \ 2 \ 1 \ 2 \ 2 \ 1 \ 2 \ 2 \ 1
\]

Fig. 2. Different permutations of examples with same class frequencies.

information, i.e., 0.6 for class 1 and 0.4 for class 2. That is to say, when splitting this node, any frequency based measure cannot differentiate \( S_1 \) and \( S_2 \). However, one obviously prefers \( S_1 \) over \( S_2 \), since \( S_1 \) can become two leaf nodes after one further splitting but \( S_2 \) cannot. Thus, it is necessary to find a new measure to differentiate such cases. It is noteworthy that the reason we can have this observation comes from the fact that the examples could be ordinal with regard to different attributes.

B. Proposed Definitions

Note that all of the following concepts are proposed under the same problem environment, i.e., \( S = \{ e_1, e_2, \ldots, e_N \} \) is a node or a set of examples to be split, \( A_j \) is a continuous valued attribute, and \( x_{ji} \) is a CCP of \( A_j \) with respect to \( S \).

**Definition 4 (Segment):** Let \( Q \) be a permutation of examples in \( S \). Then a sub-queue of \( Q \), i.e., \( SQ = \{ e, e_{i+1}, \ldots, e_j \} \), \( 1 \leq r \leq t \leq N \), is called a segment if and only if it satisfies the following requirements.

1. \( C(e_r) = C(e_{r+1}) = \ldots = C(e_j) \).
2. \( C(e_r) \neq C(e_{r-1}) \) if and only if \( r \neq 1 \).
3. \( C(e_i) \neq C(e_{i+1}) \) if and only if \( t \neq N \).

The set of all segments in \( Q \) is denoted by \( Seg(Q) \).

**Definition 5 (Segments Induced by an Attribute):** Let \( Q(A_j) \) be a permutation of examples in \( S \) with ascending values of \( A_j \). Then \( Seg(Q(A_j)) \), also denoted as \( Seg(S, A_j) \), is called a set of segments in \( S \) induced by \( A_j \).

Specifically, if attribute \( A_j \) and its CCP \( x_{ji} \) split the node \( S \) into \( S_1 = \{ e \in S | A_j(e) \leq x_{ji} \} \) and \( S_2 = \{ e \in S | A_j(e) > x_{ji} \} \), then we denote the sets of segments in \( S_1 \) and \( S_2 \) by \( Seg(S_1; x_{ji}) \) and \( Seg(S_2; x_{ji}) \) respectively.

**Definition 6 (Number of Segments in a Node):** The number of segments in node \( S \) is defined as

\[
Seg\#(S) = \min_j |Seg(S, A_j)|
\]

where \( |\cdot| \) denotes the number of elements in a set.

When the attribute values of examples in \( S \) are distinct with respect to \( A_j \), we can easily compute the number of segments with the above definitions. However, when there are duplicated values for some examples, these concepts are not sufficient to evaluate the discriminative ability of \( A_j \). To handle this issue, we further give the following definitions.

**Definition 7 (Bar):** Suppose there is a duplicated value of \( A_j \) at some examples in \( S \), and let this value be \( a \), then the set of examples \( \{ e \in S | A_j(e) = a \} \) is called a bar in \( S \) with respect to \( A_j \), denoted by \( Bar(S, A_j = a) \). The value \( a \) is called a bar point.

**Definition 8 (Number of Segments in a Bar):** Let \( B \) be a bar in \( S \) with respect to \( A_j \), and \( Q^* \) be a permutation of examples in \( B \) whose class labels are most chaotic. Then the number of segments in bar \( B \) is defined as

\[
bSeg\#(B) = |Seg(Q^*)|
\]

Generally, samples belonged to different classes are supposed to have different values with respect to a certain attribute. Otherwise the discriminative ability of the attribute is poor. To address this issue, the most chaotic case is selected for defining the number of segments in a bar, since we hope that the attribute could distinguish samples even with identical value. The most chaotic permutation could be treated as the one that can produce the maximum number of segments for the bar. For example, we suppose that the ten samples of \( S_1 \) and \( S_2 \) in Fig. 2 have identical value with respect to a certain attribute. Then, \( S_2 \) is referred to as the most chaotic permutation while \( S_1 \) is the most nonchaotic one. In fact, the most chaotic permutation of a bar may not be unique, e.g., both “1,2,1,2,1,1,2,1,1” and “1,2,1,2,1,1,2,1,1” could be treated
Algorithm 2: Computing the Number of Segments in a Bar

Input: All the examples in a bar $B$.
Output: The number of segments in bar $B$: $bSeg\#(B)$.
1 Get the numbers of examples belonging to each class: $n_1, n_2, \ldots, n_L$;
2 Sort $n_1, n_2, \ldots, n_L$ in descending order. Suppose the sorted values are $n'_1, n'_2, \ldots, n'_L$;
3 Set initial value: $bSeg\#(B) = 0$;
4 while $n'_2 > 0$ do
5 $bSeg\#(B) = bSeg\#(B) + 2n'_2$;
6 Set $n_1 = n'_1 - n'_2$, $n_2 = 0$ and $n_i = n'_i$ for each $i(3 \leq i \leq L)$, thus we get new values of $n_1, n_2, \ldots, n_L$;
7 Get the sorted values $n'_1, n'_2, \ldots, n'_L$ of $n_1, n_2, \ldots, n_L$ in descending order;
8 end
9 if $n'_1 \neq 0$ then
10 $bSeg\#(B) = bSeg\#(B) + 1$;
11 end
12 return $bSeg\#(B)$.

as the most chaotic permutations of “1,1,1,1,1,2,2,2,2.” However, the number of segments induced by them will be the same, which is also the maximum number of segments the bar can have.

Computing the number of segments in a nonbar sub-queue is straightforward. While computing the number of segments in a bar sub-queue is much more complicated. One possible solution is presented in Algorithm 2.

When computing the number of segments in a node induced by an attribute, we have to perform the following steps.
1) Sort the examples with ascending order according to the attribute, and find all the bar points.
2) Divide the order into several bar sub-queues and nonbar sub-queues based on the bar points.
3) Get the number of segments in each nonbar sub-queue directly, and compute the number of segments in each bar sub-queue based on Algorithm 2.
4) Sum up the numbers of segments in all the sub-queues, and return it as the final number of segments.

Let $t$ be the number of bar points in node $S$ with respect to $A_j$. Then we can get $t$ bar sub-queues $B_1, B_2, \ldots, B_t$ and $u$ nonbar sub-queues $S_{Q1}, S_{Q2}, \ldots, S_{Q_u}$, where each nonbar sub-queue is generated from the interval of two adjacent bar points, and $u$ only takes values of $t, t - 1$, and $t + 1$. More specifically, $u = t$, $u = t - 1$, and $u = t + 1$ respectively denote the cases that one, both, or neither of the two extreme values of $A_j$ are bar point(s). Therefore, the number of segments in $S$ induced by $A_j$, denoted by $Seg(S, A_j)$, is computed as

$$Seg\#(S, A_j) = \sum_{i=1}^{t} bSeg\#(B_i) + \sum_{i=1}^{u} |Seg(S_{Q_i})|. \quad (11)$$

C. Illustrative Example

We now give an illustrative example to calculate the number of segments in a node induced by an attribute with duplicated values. Let $S$ be a node with 20 examples from three classes, i.e., $\{1, 2, 3\}$, as shown in Table I. The class permutation of examples in $S$ with respect to $A_j$ could be described as shown in Fig. 3.

From Table I and Fig. 3, we can see that $A_j = 5.0$ and $A_j = 8.0$ are two bar points, which separate the permutation into two bar sub-queues and three nonbar sub-queues. The two bar sub-queues are $B_1 = Bar(S, A_j = 0.5) = (e_6, e_7, e_8)$ and $B_2 = Bar(S, A_j = 0.8) = (e_{14}, e_{15}, e_{16}, e_{17}, e_{18})$. The three nonbar sub-queues are $S_{Q1} = (e_1, e_2, e_3, e_4, e_5)$, $S_{Q2} = (e_9, e_{10}, e_{11}, e_{12}, e_{13})$ and $S_{Q3} = (e_{19}, e_{20})$. Then the number of segments in $S$ with respect to $A_j$ is

$$Seg\#(S, A_j) = |Seg(S_{Q1})| + |Seg(S_{Q2})| + |Seg(S_{Q3})| + bSeg\#(B_1) + bSeg\#(B_2)$$
$$= 3 + 4 + 2 + 3 + 5$$
$$= 17.$$

IV. SEGMENT BASED DT INDUCTION WITH CONTINUOUS VALUED ATTRIBUTES

In this section, we first develop the segment based DT induction algorithm, followed by a 2-D example to show its difference with C4.5. Then, we discuss the relationship between segment and frequency. Finally, we make an analysis on time complexity.

A. Segment Based DT Induction Model

Let $S$ be the given node, we now try to develop a hybrid scheme that utilizes both frequency and segment to split it.

First, the best splitting performance $SP^*$ is calculated as

$$SP^* = \max_{(i=1)} SP(S, x_{ji}) \quad (12)$$

where $SP(S, x_{ji})$ is defined in (10). Then, the $\hat{K}$ CCPs with splitting performances closest to $SP^*$ are selected from $\{x_{ji}\}$
Algorithm 3: Segment+C4.5 DT With Continuous Valued Attributes

**Input:** Training examples \( \{e_i\}_{i=1}^N \) with \( m \) continuous valued attributes \( A = \{A_j\}_{j=1}^m \) and one decision attribute \( C \in \{C_i\}_{i=1}^l \); threshold number \( \tilde{N} \) to stop splitting a node and parameter \( \hat{k} \).

**Output:** A binary DT.

1. Initialize \( \Omega \) as an empty set;
2. Consider the set of all examples as the root-node, and add it to \( \Omega \);
3. while \( \Omega \) is not empty do
   4. Select a node from \( \Omega \), denoted by \( S \);
      5. if \( |S| < \tilde{N} \) then
         6. Treat \( S \) as a leaf node and assign it the class label \( \ell^* = \arg \max_{i=1,\ldots,L} p_i \);
      7. else
         8. For each ASA \( A_j \), \( j = 1, \ldots, m \), sort the examples with ascending order;
         9. Get the CCPs of each ASA based on (1), i.e., \( x_{ji} \), where \( j = 1, \ldots, m \) and \( i = 1, \ldots, N-1 \);
         10. Calculate the splitting performance of each CCP based on (10), i.e., \( SP(S, x_{ji}) \);
         11. Get the best splitting performance \( SP^* \) by (12);
         12. Get the set \( X \) that contains the \( \hat{k} \) CCPs whose splitting performances are closest to \( SP^* \);
         13. Compute the number of segments in the two child-nodes induced by each CCP in \( X \);
         14. Get the optimal splitting attribute \( A_j^* \) and its optimal CCP \( x_{ji}^* \) by (13);
         15. Split \( S \) into two child-nodes by \( x_{ji}^* \), i.e.,
            \[ S_1 = \{ e \in S | A_j^*(e) \leq x_{ji}^* \} \]
            \[ S_2 = \{ e \in S | A_j^*(e) > x_{ji}^* \} \];
         16. for \( i = 1, 2 \) do
            17. if all the examples in \( S_i \) are from the same class \( \ell^* \) then
               18. Treat \( S_i \) as a leaf node and assign it the class label \( \ell^* \);
            19. else
               20. Add \( S_i \) to \( \Omega \);
            21. end
         22. end
      23. Remove \( S \) from \( \Omega \);
   24. end
25. return the constructed tree.

To form the subset \( \mathcal{X} \), where \( j = 1, \ldots, m \) and \( i = 1, \ldots, N-1 \). Finally, the optimal CCP \( x_{ji}^* \) is derived by

\[
x_{ji}^* = \arg \min_{x_{ji} \in \mathcal{X}} \left[ \frac{|S_1|}{|S|} \left| \text{Seg} \left( S_1; x_{ji} \right) \right| + \frac{|S_2|}{|S|} \left| \text{Seg} \left( S_2; x_{ji} \right) \right| \right].
\]

(13)

The segment based DT induction model with continuous valued attributes under the frequency strategy of C4.5 is then described in Algorithm 3. Generally speaking, when several CCPs have equivalent or approximately equivalent splitting performance, C4.5 selects one randomly while our approach selects the one with fewest segments.

It is noteworthy that by revising the splitting performance in line 10 of Algorithm 3, the proposed segment measure can be incorporated into any frequency based heuristic other than C4.5. Without losing generality, we only present C4.5 here.

**B. Example Demonstration**

In this section, we give a 2-D example to show the difference between C4.5 and the proposed scheme. As listed in Table II, a binary dataset containing 20 samples with two continuous features is give. Five samples are from class 1 and fifteen samples are from class 2. We conduct C4.5 algorithm and the proposed algorithm respectively on this dataset, and observe the difference of the two induced trees. Due to the small data size, we simply set \( \hat{k} = 2 \) in Algorithm 3, and \( \tilde{N} = 2 \) in both Algorithms 1 and 3.

When splitting the root node, there are 19 CCPs for each of the two ASAs, we denote them as \( \{x_{ji}\} \) where \( j = 1, 2 \) and \( i = 1, \ldots, 19 \). It is calculated that \( x_{1,14} \) gives the highest information gain ratio of 0.1737, and \( x_{2,5} \) gives the second highest information gain ratio of 0.1511. Accordingly, the segment measure values of these two CCPs are respectively 5.90 and 3.25. In this case, C4.5 selects \( x_{1,14} \) to split the node, while the proposed scheme selects \( x_{2,5} \) to split it. As a result, C4.5 induces a tree with eight leaf nodes and seven nonleaf nodes in eight depths as shown in Fig. 4(a), while the proposed scheme induces a tree with five leaf nodes and four nonleaf nodes in four depths as shown in Fig. 4(b). Obviously, the proposed tree is much simpler than the C4.5 tree, which effectively avoids the over-partitioning problem and the additional time complexity.
C. Relationship Between Segment and Frequency

In this section, we give an analysis on the relationship between frequency and the expected number of segments for binary cases. Let $S$ be a node with $N$ examples, and $p(0 < p < 1)$ be the frequency of positive class in $S$. We suppose that the number of segments in $S$ is $\xi$, where $\xi$ is a positive integer that can take values of $2, 3, \ldots, M(p, N)$, thus

$$M(p, N) = \begin{cases} 
2pN + 1 & \text{when } 0 < p < 0.5 \\
N & \text{when } p = 0.5 \\
2(1-p)N + 1 & \text{when } 0.5 < p < 1.
\end{cases}$$ (14)

The above statement implies a discrete probability distribution with the following form:

$$\begin{array}{c}
2
3
4
\ldots
M(p, N)
\end{array}
\begin{array}{c}
r_2
r_3
r_4
\ldots
r_{M(p, N)}
\end{array}$$ (15)

where the first row denotes the values that $\xi$ can take, and the second row denotes the corresponding probabilities under the conditions $r_2 \geq 0$ and $\sum_{k=2}^{M(p, N)} r_k = 1$.

In order to get the expectation of $\xi$, we first need to estimate the probability $r_\xi$, which equals to get the number of possible example permutations that can produce $\xi$ segments. Let $T(\xi)$ denotes this number, and $k = (\xi - \xi/2)/2$, thus for given $p$ and $N$, we have

$$T(\xi) = \begin{cases} 
2(pN)!((N-pN)^{\xi-k} (1-pN)^{k-1}) & \text{when } \xi = 2k \\
(pN)!((N-pN)^{\xi-k} (1-pN)^{k-1}) & \text{when } \xi = 2k + 1, pN > k, (1-pN) > k
\end{cases}$$ (16)

We place the derivation of (16) in the Appendix. Let $\sum_{\xi=2}^{M(p, N)} T(\xi) = T_{\text{sum}}$, then the probability listed in (15) can be represented as

$$r_\xi = T(\xi)/T_{\text{sum}}$$ (17)

where $\xi = 2, \ldots, M(p, N)$.

It is clear from (14) that $M(p, N)$ is symmetric about $p = 0.5$, i.e., $M(p, N) = M(1-p, N)$. Thus, for any fixed $N$, the distribution (15) induced by $p$ is identical to that induced by $1-p$. In other words, for $p$ and $1-p$, the second row listed in (15) has no change at all.

Based on the above discussion, the expectation of $\xi$ corresponding to distribution (15) is computed as

$$E[\xi] = \sum_{\xi=2}^{M(p, N)} \xi \cdot r_\xi$$ (18)

where $r_\xi$ is given in (17).

The analytic form of (18) is extremely complicated. Thus, it is difficult to analyze the change of (18) with $p$ for general $N$. However, for any fixed $N$, (18) is a function with respect to $p$, and we have some concrete expressions. Fig. 5 illustrates the cases of $N = 10, 20, \ldots, 100$.

It can be seen from Fig. 5 that $E[\xi]$, which could be regarded as a function of $p$ and rewritten as $E(p)$, has the common characteristics with frequency based heuristic measures, i.e., $E(p)$ is defined on $[0, 1]$, symmetric at $p = 0.5$, strictly increasing in $(0, 0.5)$, strictly decreasing in $(0.5, 1)$, and convex ($(d^2E/dp^2)(p) < 0$). In this way, the segment based scheme is considered as a general stochastic version of frequency based scheme. Specifically, when $p$ is given to a node, the frequency information can be determined, but the segment information cannot be determined. The number of segments for a given $p$ can vary from 2 to its maximum with the distribution listed in (15). It implies that we may use the expected number of segments to evaluate the quality of a tree.

D. Analysis of Time Complexity

We first analyze the time complexity of splitting a node in Algorithm 1. Let $S$ be a node with $N$ examples. Suppose the number of ASAs in $S$ is $m$ and there are $\beta_j$ CCPs for $A_j$. The main complexity in Algorithm 1 lies in steps 8 to 11. In step 8, sorting the examples for the $m$ ASAs leads to a minimum complexity of $O(mN \log N)$. In steps 9 to 10, calculating the splitting performances of the $\sum_{j=1}^{m} \beta_j$ CCPs leads to a complexity of $O(N \sum_{j=1}^{m} \beta_j)$. In step 11, finding the optimal CCP leads to a complexity of $O(\sum_{j=1}^{m} \beta_j)$. Thus, the complexity for splitting $S$ in Algorithm 1 is $O(mN \log N + (N + 1) \sum_{j=1}^{m} \beta_j)$. We further suppose that each ASA has the same number of CCPs, i.e., $\beta_j = N-1$, $j = 1, \ldots, m$. Thus, the time complexity of splitting a node in Algorithm 1 is

$$O\left( mN \log N + (N + 1) \sum_{j=1}^{m} \beta_j \right) = O\left( mN \log N + m(N-1)(N+1) \right) \approx O\left( mN(N + \log N) \right).$$

We then analyze the time complexity of splitting a node in Algorithm 3. The main complexity lies in steps 8 to 14. According to the above analysis, steps 8 to 11 lead to a complexity of $O(mN \log N + (N + 1) \sum_{j=1}^{m} \beta_j)$. We suppose that there are no example bars in the sorted queue, thus in steps 12 to 13, computing the number of segments for the $K$
CCPs leads to a complexity of $O(N \cdot \hat{K})$. Then, in step 14, finding the optimal CCP leads to a complexity of $O(\hat{K})$. Finally, the time complexity of splitting a node in Algorithm 3 is

$$O \left( mN \log N + (N + 1) \sum_{j=1}^{m} \hat{b}_j + (N + 1) \hat{K} \right)$$

$$\approx O \left( mN \log N + mN - 1 \right) (N + 1) + (N + 1) \hat{K}$$

$$\approx O \left( mN(N + \log N) + \hat{K} \hat{N} \right).$$

In most cases, the number of ASAs is much smaller than the number of examples, i.e., $m \ll N$. Thus, when $\hat{K}$ is set to be a very small number, the time complexities for splitting a node in both Algorithms 1 and 3 are $O(N^2)$.

V. EXPERIMENTS COMPARISONS

In this section, we conduct some experimental comparisons to demonstrate the effectiveness of the proposed scheme.

A. Comparative Methods

As aforementioned, the segment measure can be combined with any frequency based heuristic method. However, it is unnecessary to implement all of them, thus we only combine it with the most popular and powerful algorithm, i.e., C4.5. Finally, eight heuristic methods are listed as follows for performance comparison.

1) IDE3 [21]: For splitting a node, the CCP with the maximum information gain is selected, which adopts entropy, i.e., (3), as the frequency measure.

2) C4.5 [33], [34]: Algorithm 1 is realized.

3) CART [7], [23]: For splitting a node, the CCP with the maximum information gain is selected, which adopts Gini-index, i.e., (4), as the frequency measure.

4) SIN [42]: The sin measure, i.e., (5), is applied to evaluate the information gain, and the CCP with the maximum information gain is used to split a node.

5) SQRT [42]: This method has the same framework with SIN, but it applies the sqrt measure, i.e., (6), to evaluate the information gain.

6) PSIN [42]: This method also has the same framework with SIN, but it applies the psin measure, i.e., (7), to evaluate the information gain.

7) Segment+C4.5: Algorithm 3 is realized, which incorporates the proposed segment measure to C4.5.

8) Segment: This method just adopts the proposed segment measure to induce a DT, where the CCP with the minimum value of $(\sum_{S_1}/|S_1|)\text{Seg}(S_1; x_{j}) + (|S_2|/|S|)\text{Seg}(S_2; x_{j})$ is selected to split a node.

B. Experimental Design

Experimental comparisons are conducted on 15 binary University of California, Irvine (UCI) machine learning datasets and five multiclass UCI machine learning datasets as listed in Table III. For these datasets, not all the attributes are continuous. If the number of unique values in an attribute is less than ten, then we treat it as a discrete attribute and delete it from the dataset. Besides, for each attribute, the input values are normalized to $[0, 1]$ by $1 - (x_{\text{max}} - x)/(x_{\text{max}} - x_{\text{min}})$, where $x_{\text{max}}$ and $x_{\text{min}}$ are the maximum and minimum values among all the examples with regard to the attribute, and $x$ is the value to be normalized.

For all the methods, we stop splitting a node if the number of examples in it is less than five, i.e., $\hat{N} = 5$. Besides, for method Segment+C4.5, we tune $\hat{K}$ = [2, 4, 6, 8, 10, 15, 20, 30, 40, 50] and select the best one as the final parameter. For the 15 binary datasets, we conduct 10-fold cross-validation, and observe the average value of the 10 results. However, it is difficult to conduct 10-fold cross-validation on some multiclass datasets. Take dataset Ecoli as an instance, the class distribution 143/77/2/2/35/20/5/52 is highly unbalanced. In some classes, there are only two examples, which are not enough to be divided into ten folds. In this case, for the five multiclass datasets, we conduct $5 \times 2$-fold cross-validation, and observe average value of the $5 \times 2 = 10$ results. In each experiment, the eight heuristic methods listed in Section V-A are implemented on the same training and testing sets.

We evaluate the performance of a DT from two aspects, i.e., generalization capability and model complexity. The generalization capability is measured by the testing accuracy, while the model complexity is mainly reflected by the number of nodes and tree depth. Since the algorithms realized are all based on the typical binary splitting, the number of leaf nodes is always one more than the number of nonleaf nodes, thus only the total number of nodes is considered. We generally have the following expectations by incorporating the segment based measure.

1) Both the number of nodes and tree depth should be reduced compared with the single frequency based method.

2) With a proper setting of $\hat{K}$, the accuracy of the model should be improved.

The experiments are performed under MATLAB 7.9.0, which are executed on a computer with a 3.16-GHz Intel Core 2 Duo CPU, a maximum 4.00-GB memory, and 64-bit Windows 7 system.

C. Empirical Studies

Fig. 6 demonstrates the average reduction scales in the training accuracy, testing accuracy, number of nodes, and tree depth of Segment+C4.5 compared with C4.5 with different $\hat{K}$ values. As an example, if the depths of the trees induced by C4.5 and Segment+C4.5 are respectively $d_1$ and $d_2$, then the reduction scale in tree depth of Segment+C4.5 compared with C4.5 is calculated as $((d_2 - d_1)/d_1) \times 100\%$. Similar calculations are also applied to the number of nodes, training and testing accuracies. Thus, if the reduction scale is below zero, the measure value is reduced by the segment based method, otherwise increased. In this case, we generally hope that the reduction scales of accuracies are above zero, meanwhile the reduction scales of node number and tree depth are below zero. It can be seen from Fig. 6 that, the reduction scales of the four referred criteria are influenced by parameter $\hat{K}$. Basically we have the following observations:

1) Observation 1: The segment based methods can achieve reductions on tree depth for all the datasets. The absolute
reduction scale becomes larger with the increase of \( \hat{K} \). This is easy to explain, since with a larger \( \hat{K} \), the algorithm considers more CCPs with approximately equal splitting performances. In this case, there is a larger probability to select the CCP that can lead to a lower number of segments. Obviously, a lower number of segments is helpful in reducing the tree depth. Besides, we make a further investigation on the segment measure, i.e., \( |S_1|/|S|\cdot|\text{Seg}(S_1; x_j)| + |S_2|/|S|\cdot|\text{Seg}(S_2; x_j)| \). Given two different CCPs, i.e., \( x_1 \) and \( x_2 \), suppose they have the same information gain ratio, as well as the same number of segments after splitting, i.e., \( |\text{Seg}(S_1; x_1)| + |\text{Seg}(S_2; x_1)| = |\text{Seg}(S_1; x_2)| + |\text{Seg}(S_2; x_2)| \). As demonstrated in Fig. 5, a larger number of examples corresponds to a larger expected number of segments, thus we further assume:

1) \( |S_1|/|S| = 0.1, |S_2|/|S| = 0.9, |\text{Seg}(S_1; x_1)| = 1 \) and \( |\text{Seg}(S_2; x_1)| = 9 \) for \( x_1 \);
2) \( |S_1|/|S| = 0.5, |S_2|/|S| = 0.5, |\text{Seg}(S_1; x_2)| = 5 \) and \( |\text{Seg}(S_2; x_2)| = 5 \) for \( x_2 \).

Then, we have \( |S_1|/|S| |\text{Seg}(S_1; x_1)| + |S_2|/|S| |\text{Seg}(S_2; x_1)| = 0.82 \) and \( |S_1|/|S| |\text{Seg}(S_1; x_2)| + |S_2|/|S| |\text{Seg}(S_2; x_2)| = 0.5 \). Obviously, \( x_2 \) is preferred to \( x_1 \).

This observation demonstrates that the segment measure tends to select the CCP with two equally balanced sets, as a result, the tree depth is reduced. A simple illustration is shown in Fig. 7, where \( x_1 \) may induce a DT similar to Fig. 7(a) and \( x_2 \) may induce a DT similar to Fig. 7(b).

2) Observation 2: The segment based methods can also achieve reductions on the number of nodes, but the absolute reduction scales are much smaller than those on the tree depth. Five typical cases could be found. First, as shown in Fig. 6(g), the number of nodes for dataset Heart decreases gradually with \( \hat{K} \) increases in the front part of the figure, and attains the minimum when \( \hat{K} = 15 \), then becomes increasing in the latter part. Similar trends could also be found for many other datasets, e.g., Haberman, Australian, German, Bupa, Pima, SPECTF, Vowel, and Yeast. Second, as shown in Fig. 6(j) and (o), the numbers of nodes for datasets Wpbc and Sonar keep decreasing in the whole process, which indicate that the \( \hat{K} \) value that can achieve the maximum reduction on node number may be larger than 50. Third, as shown in Fig. 6(n) and (r), the numbers of nodes for datasets CT and Libras have no obvious change. Although the tree depth is largely reduced, the number of nodes cannot be reduced any more by the segment measure. Forth, as shown in Fig. 6(c) and (i), the numbers of nodes for datasets Cancer and Wdbc fluctuate in the whole process, with no clear and explicit trend. Finally, as shown in Fig. 6(b) and (q), the segment based method does not achieve any reduction, but leads to an increase on the number of nodes for datasets Ionosphere and Ecoli. We now try to find out the reason for the different cases. Fig. 8 demonstrates the standard deviation of the segment based measure values, i.e., \( |S_1|/|S| |\text{Seg}(S_1; x_\mu)| + |S_2|/|S| |\text{Seg}(S_2; x_\mu)| \), of the \( \hat{K} \) considered CCPs when splitting the root-node for datasets CT, Ionosphere, Heart, Pima, Transfusion, and Australian. Similar results could also be found when splitting other nodes. It is easy to observe that this standard deviation is large on datasets Heart, Pima, Transfusion, and Australian. In this case, the considered CCPs have quite different segment information, and choosing the one with the minimum value can obviously reduce the number of nodes. While this standard deviation is much smaller on datasets CT and Ionosphere, and has no obvious increase even when \( \hat{K} \) becomes large. That is to say, the considered CCPs have very close segment information. Distinguishing them is not effective enough to reduce the number of nodes, at the worst, it may lead to an increase on this number.

3) Observation 3: In general, the training and testing accuracies keep a stable trend. On most datasets, the training accuracy is slightly decreased, and the testing accuracy is slightly increased. This observation demonstrates that incorporating the segment measure may be helpful for tackling the over-fitting problem. However, this statement does not hold for dataset Ionosphere in Fig. 6. In this case, when the considered CCPs have very close segment information, the proposed method is not effective in both size reduction and accuracy improvement.

In conclusion, on most datasets, an appropriate value of \( \hat{K} \) can not only improve the generalization capability, but can also reduce the size of tree, which is crucial to improve the testing efficiency.

The best \( \hat{K} \) values among \{2, 4, 6, 8, 10, 15, 20, 30, 40, 50\} for method Segment+C4.5 are listed in the last column of Table III. With these \( \hat{K} \) values, Table IV summarizes the average testing accuracy and standard deviation of the cross-validation results of the eight heuristic methods, where the highest testing accuracy is marked with \( \sqrt{\ } \) for each dataset. It is observed that methods IDE3, C4.5, CART, SIN, Segment, and Segment+C4.5 can respectively perform best on 3, 1, 2, 1, 1, and 12 datasets out of 20. Among them, method Segment+C4.5 gives the best average result. Furthermore, ↑ and ▼ are used to demonstrate that whether incorporating the segment measure can improve the performance of single frequency based method. Clearly, Segment+C4.5 has higher accuracy than C4.5 on 19 datasets out of 20, it only fails to improve the performance on dataset Ionosphere. Besides, it is observed that method Segment gives the lowest average result, it performs very bad in most cases, and only gives satisfactory performance on dataset German. That is to say, the frequency based heuristic is crucial and important for inducing a high-performance tree. The proposed segment measure serves as an assistant to improve the capability of the frequency measure. It is only valid with certain constraint and threshold, which is reflected by the parameter \( \hat{K} \) in Algorithm 3.

Table V reports the average tree depth and number of nodes induced by the eight heuristic methods. For each dataset, the minimum values are highlighted in bold face. It can be seen

![Fig. 7](image-url) Two DTs with different depths. (a) Tree with larger depth. (b) Tree with lower depth.

![Fig. 8](image-url) Standard deviation of segment based measure values of the \( \hat{K} \) considered CCPs when splitting the root-node.
TABLE III
SELECTED DATASETS FOR PERFORMANCE COMPARISON

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Example#</th>
<th>Conditional Attribute#</th>
<th>Continuous Attribute#</th>
<th>Data Type</th>
<th>Class#</th>
<th>Class Distribution</th>
<th>Missing Values</th>
<th>( \bar{K} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haberman</td>
<td>306</td>
<td>3</td>
<td>3</td>
<td>Integer</td>
<td>2</td>
<td>225/81</td>
<td>None</td>
<td>8</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>34</td>
<td>32</td>
<td>Real+Integer</td>
<td>2</td>
<td>225/126</td>
<td>None</td>
<td>2</td>
</tr>
<tr>
<td>Cancer</td>
<td>699 (683)</td>
<td>9</td>
<td>8</td>
<td>Integer</td>
<td>2</td>
<td>458/(444/241)(239)</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>Australian</td>
<td>690</td>
<td>14</td>
<td>7</td>
<td>Real+Integer</td>
<td>2</td>
<td>307/383</td>
<td>None</td>
<td>6</td>
</tr>
<tr>
<td>German</td>
<td>1000</td>
<td>24</td>
<td>3</td>
<td>Integer</td>
<td>2</td>
<td>700/300</td>
<td>None</td>
<td>15</td>
</tr>
<tr>
<td>Bupa</td>
<td>345</td>
<td>6</td>
<td>6</td>
<td>Real</td>
<td>2</td>
<td>145/200</td>
<td>None</td>
<td>50</td>
</tr>
<tr>
<td>Heart</td>
<td>270</td>
<td>13</td>
<td>5</td>
<td>Real+Integer</td>
<td>2</td>
<td>150/120</td>
<td>None</td>
<td>15</td>
</tr>
<tr>
<td>Transfusion</td>
<td>748</td>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>2</td>
<td>178/570</td>
<td>None</td>
<td>50</td>
</tr>
<tr>
<td>Wdbc</td>
<td>569</td>
<td>30</td>
<td>30</td>
<td>Real</td>
<td>2</td>
<td>212/357</td>
<td>None</td>
<td>15</td>
</tr>
<tr>
<td>Wbpc</td>
<td>198</td>
<td>33</td>
<td>33</td>
<td>Real</td>
<td>2</td>
<td>47/151</td>
<td>None</td>
<td>30</td>
</tr>
<tr>
<td>Pima</td>
<td>768</td>
<td>8</td>
<td>8</td>
<td>Real+Integer</td>
<td>2</td>
<td>268/300</td>
<td>None</td>
<td>30</td>
</tr>
<tr>
<td>Pfrx</td>
<td>182</td>
<td>12</td>
<td>12</td>
<td>Real</td>
<td>2</td>
<td>130/52</td>
<td>None</td>
<td>40</td>
</tr>
<tr>
<td>SPECTF</td>
<td>267</td>
<td>44</td>
<td>44</td>
<td>Integer</td>
<td>2</td>
<td>212/55</td>
<td>None</td>
<td>15</td>
</tr>
<tr>
<td>CT</td>
<td>221</td>
<td>36</td>
<td>36</td>
<td>Real</td>
<td>2</td>
<td>101/120</td>
<td>None</td>
<td>2</td>
</tr>
<tr>
<td>Sonar</td>
<td>208</td>
<td>60</td>
<td>60</td>
<td>Real</td>
<td>2</td>
<td>97/111</td>
<td>None</td>
<td>50</td>
</tr>
<tr>
<td>Cotton</td>
<td>556</td>
<td>21</td>
<td>21</td>
<td>Real</td>
<td>2</td>
<td>100/59</td>
<td>None</td>
<td>2</td>
</tr>
<tr>
<td>Ecoli</td>
<td>336</td>
<td>5</td>
<td>5</td>
<td>Real</td>
<td>2</td>
<td>143/77/2/2/35/20/5/52</td>
<td>None</td>
<td>10</td>
</tr>
<tr>
<td>Libras</td>
<td>360</td>
<td>90</td>
<td>90</td>
<td>Real</td>
<td>15</td>
<td>24×15</td>
<td>None</td>
<td>10</td>
</tr>
<tr>
<td>Vowel</td>
<td>990</td>
<td>10</td>
<td>10</td>
<td>Real</td>
<td>11</td>
<td>90×11</td>
<td>None</td>
<td>10</td>
</tr>
<tr>
<td>Yeast</td>
<td>1484</td>
<td>6</td>
<td>6</td>
<td>Real</td>
<td>10</td>
<td>241×942/463/44/51/162/35/20/5/20</td>
<td>None</td>
<td>10</td>
</tr>
</tbody>
</table>

Note: The datum 689 in “699 (683)” about dataset Cancer denotes the number of records in the original database, while 683 denotes the number of records after removing the ones with missing values, and the same explanation for “458/(444/241)” and “212/357”.

That the methods that achieve the smallest tree depth and node number on different datasets are quite different. In fact, each method can induce the smallest tree in some cases. Among them, CART gives the lowest average value. However, the comparison between C4.5 and Segment+C4.5 is clear. In the last two columns of Table V, ↓ and ↑ are used to demonstrate that whether method Segment+C4.5 can reduce the tree size of its single frequency based method C4.5. Obviously, depth reduction is achieved on 19 datasets out of 20, and node number reduction is achieved on 18 datasets out of 20.

The average reduction scale of tree size and the average improvement of testing accuracy on the 20 datasets are listed in Table VI.

Finally, we make some statistical tests on the testing accuracies listed in Table IV. Paired Wilcoxon’s signed rank test is performed, which is a famous nonparametric statistical hypothesis test for assessing whether there exists significant difference between the elements of two sets, or whether one of two groups of independent observations tends to have larger values than the other. With an unknown distribution, the Wilcoxon’s signed rank test is safer and more rational than the t-test. The corresponding \( p \) values are reported in Table VII, and the significance level 0.05 is adopted. If the \( p \) value is smaller than 0.05, the two referred methods are considered as statistically different. It can be seen that in some cases, the six single frequency based methods, i.e.,
TABLE V

COMPARISONS OF DIFFERENT DT INDUCTION HEURISTICS: TREE SCALE

<table>
<thead>
<tr>
<th>Datasets</th>
<th>IDE3 Depth</th>
<th>C4.5 Depth</th>
<th>CART Depth</th>
<th>SIN Depth</th>
<th>SQRT Depth</th>
<th>PSIN Depth</th>
<th>Segment Depth</th>
<th>Segment+C4.5 Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haberman</td>
<td>16.50</td>
<td>129.80</td>
<td>25.10</td>
<td>150.80</td>
<td>16.90</td>
<td>130.80</td>
<td>43.10</td>
<td>157.60</td>
</tr>
<tr>
<td></td>
<td>11.00</td>
<td>33.60</td>
<td>12.80</td>
<td>44.60</td>
<td>197.40</td>
<td>393.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cancer</td>
<td>8.40</td>
<td>39.20</td>
<td>11.90</td>
<td>50.60</td>
<td>8.50</td>
<td>45.00</td>
<td>76.20</td>
<td>198.60</td>
</tr>
<tr>
<td></td>
<td>10.10</td>
<td>37.60</td>
<td>10.00</td>
<td>49.00</td>
<td>366.60</td>
<td>732.20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Australian</td>
<td>16.80</td>
<td>185.60</td>
<td>48.40</td>
<td>237.80</td>
<td>14.80</td>
<td>184.00</td>
<td>69.90</td>
<td>324.60</td>
</tr>
<tr>
<td>German</td>
<td>24.20</td>
<td>471.00</td>
<td>61.10</td>
<td>544.00</td>
<td>21.40</td>
<td>452.40</td>
<td>29.60</td>
<td>331.00</td>
</tr>
<tr>
<td>Bupa</td>
<td>12.50</td>
<td>119.00</td>
<td>69.50</td>
<td>185.40</td>
<td>11.30</td>
<td>116.20</td>
<td>23.20</td>
<td>169.20</td>
</tr>
<tr>
<td>Heart</td>
<td>13.70</td>
<td>96.80</td>
<td>45.70</td>
<td>132.20</td>
<td>13.70</td>
<td>97.40</td>
<td>28.30</td>
<td>150.80</td>
</tr>
<tr>
<td>Transfusion</td>
<td>20.20</td>
<td>231.20</td>
<td>34.00</td>
<td>309.80</td>
<td>17.30</td>
<td>270.00</td>
<td>25.20</td>
<td>139.40</td>
</tr>
<tr>
<td>Weibc</td>
<td>7.50</td>
<td>32.40</td>
<td>16.10</td>
<td>49.00</td>
<td>8.40</td>
<td>37.20</td>
<td>48.20</td>
<td>131.40</td>
</tr>
<tr>
<td>Wpbc</td>
<td>9.70</td>
<td>41.20</td>
<td>24.60</td>
<td>60.00</td>
<td>8.80</td>
<td>43.00</td>
<td>42.20</td>
<td>103.00</td>
</tr>
<tr>
<td>Pima</td>
<td>15.90</td>
<td>200.60</td>
<td>94.60</td>
<td>301.60</td>
<td>15.00</td>
<td>199.00</td>
<td>30.60</td>
<td>230.20</td>
</tr>
<tr>
<td>Pirx</td>
<td>16.30</td>
<td>57.00</td>
<td>40.10</td>
<td>81.80</td>
<td>15.60</td>
<td>59.00</td>
<td>27.40</td>
<td>108.20</td>
</tr>
<tr>
<td>SPECTF</td>
<td>11.60</td>
<td>42.80</td>
<td>26.70</td>
<td>64.20</td>
<td>9.50</td>
<td>45.40</td>
<td>35.40</td>
<td>101.60</td>
</tr>
<tr>
<td>CT</td>
<td>7.30</td>
<td>22.20</td>
<td>13.00</td>
<td>29.60</td>
<td>8.20</td>
<td>25.00</td>
<td>12.20</td>
<td>49.60</td>
</tr>
<tr>
<td>Sonar</td>
<td>8.30</td>
<td>31.40</td>
<td>20.50</td>
<td>57.60</td>
<td>7.50</td>
<td>36.40</td>
<td>16.10</td>
<td>61.00</td>
</tr>
<tr>
<td>Cotton</td>
<td>6.80</td>
<td>30.40</td>
<td>12.40</td>
<td>34.20</td>
<td>7.90</td>
<td>31.20</td>
<td>11.30</td>
<td>49.00</td>
</tr>
<tr>
<td>Ecoli</td>
<td>9.50</td>
<td>43.80</td>
<td>12.20</td>
<td>50.80</td>
<td>9.00</td>
<td>45.40</td>
<td>20.20</td>
<td>92.60</td>
</tr>
<tr>
<td>Libras</td>
<td>8.60</td>
<td>75.80</td>
<td>21.90</td>
<td>85.00</td>
<td>10.90</td>
<td>75.00</td>
<td>51.00</td>
<td>190.00</td>
</tr>
<tr>
<td>Vowel</td>
<td>11.20</td>
<td>163.40</td>
<td>47.40</td>
<td>220.80</td>
<td>13.30</td>
<td>163.40</td>
<td>23.60</td>
<td>527.80</td>
</tr>
<tr>
<td>Yeast</td>
<td>19.10</td>
<td>382.20</td>
<td>74.20</td>
<td>483.00</td>
<td>20.00</td>
<td>374.40</td>
<td>90.60</td>
<td>586.00</td>
</tr>
<tr>
<td>Avg.</td>
<td>12.68</td>
<td>124.02</td>
<td>36.11</td>
<td>158.65</td>
<td>12.42</td>
<td>123.43</td>
<td>48.21</td>
<td>186.66</td>
</tr>
</tbody>
</table>

Note: For each dataset, the minimum number of tree nodes and the smallest tree depth are in bold face. For method Segment+C4.5, ↓ and ↑ respectively represent that compared with method C4.5, the number of nodes or tree depth is reduced or not.

TABLE VI

AVERAGE REDUCTION SCALE OF TREE SIZE AND AVERAGE IMPROVEMENT OF TESTING ACCURACY OF Segment+C4.5 OVER C4.5

<table>
<thead>
<tr>
<th>Tree depth</th>
<th>Number of nodes</th>
<th>Testing accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>55.66%↓</td>
<td>13.32%↑</td>
<td>4.53%↑</td>
</tr>
</tbody>
</table>

TABLE VII

PAIRED WILCOXON'S SIGNED RANK TESTS OF TESTING ACCURACIES (p VALUES)

<table>
<thead>
<tr>
<th>Method</th>
<th>C4.5</th>
<th>CART</th>
<th>SIN</th>
<th>SQRT</th>
<th>PSIN</th>
<th>Segment</th>
<th>Segment+C4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDE3</td>
<td>0.0251↑</td>
<td>0.2959</td>
<td>0.0032↓</td>
<td>0.0479↑</td>
<td>0.0016↑</td>
<td>0.0047↑</td>
<td>0.0064↑</td>
</tr>
<tr>
<td>C4.5</td>
<td>0.1084</td>
<td>0.1259</td>
<td>0.9045</td>
<td>0.2959</td>
<td>0.0028↑</td>
<td>0.0002↑</td>
<td>0.0031↑</td>
</tr>
<tr>
<td>CART</td>
<td>0.0036↓</td>
<td>0.4781</td>
<td>0.0228↑</td>
<td>0.0013↑</td>
<td>0.0043↑</td>
<td>0.0001↑</td>
<td></td>
</tr>
<tr>
<td>SIN</td>
<td>0.1259</td>
<td>1.0000</td>
<td>0.0028↑</td>
<td>0.0001↑</td>
<td>0.0001↑</td>
<td>0.0013↑</td>
<td></td>
</tr>
<tr>
<td>SQRT</td>
<td>0.0276↑</td>
<td>0.0013↑</td>
<td>0.0013↑</td>
<td>0.0001↑</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSIN</td>
<td>0.0012↑</td>
<td>0.0007↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: For each test, ↑ represent that the two referred methods are significantly different with the significance level 0.05.

IDE3, C4.5, CART, SIN, SQRT, and PSIN, are of no significant difference. However, all of them are statistically different from Segment+C4.5 and Segment. Besides, Segment+C4.5 and Segment are also statistically different. Furthermore, in order to validate that whether the several methods are statistically different from each other, Friedman test is conducted, where the p value is 2.7166 × 10⁻⁹, which is much smaller than 0.05. These observations strongly attest to the effectiveness of the proposed scheme.

VI. CONCLUSION

Traditional DT induction models with continuous valued attributes only consider the frequencies of classes, which fail to differentiate the CCPs with the same or approximately equal splitting performance. In order to tackle this problem, the concept of segment is proposed in this paper. Theoretical analysis demonstrates that the expected number of segments, which is considered as the expectation of a random variable, has the common features of frequency based measures such as information entropy and Gini-index. The hybrid of frequency and segment is then used as a measure to split nodes. Empirical studies clearly demonstrate that the proposed method is effective in both improving the generalization capability and reducing the tree size.

Several possible research issues regarding this topic are listed as follows.

1) The performance of the proposed method is influenced by the parameter \( \hat{K} \). The optimal value of \( \hat{K} \) differs a lot on different datasets. Thus, it is necessary to discuss how to get the optimal \( \hat{K} \) value adaptively according to the characteristics of a given training set.

2) As analyzed in Section IV-C, the expected number of segment is regarded as a random variable. It is affected by the number of examples and the frequencies of classes in the node. It might be useful to find an analytic expression for this expectation in general case, which only relies on the frequencies of classes.

3) It might be interesting to extend the work to multi-splitting environment with mixed types of attributes. Consequently, the related analysis will be more complicated.

APPENDIX

A. Description

Consider a binary classification problem, let \( S \) be a set with \( N \) examples, and the frequency of positive class is \( p \). Clearly, different distributions of these examples will produce different numbers of segments.
B. Problem

Given a positive integer $\xi$, where $\xi$ can take values of $2, 3, \ldots, M(p, N)$ and $M(p, N)$ is given in (14), how many distributions of examples in $S$ will produce $\xi$ segments?

C. Solution

Note that all the examples of a segment must belong to the same class: $+$ or $-$. If all examples belong to the positive class, then we call the segment a positive segment denoted by $\oplus$. If all examples belong to the negative class, then we call it a negative segment denoted by $\ominus$.

No matter how to rank these examples, the distribution of segments must be one of the following cases:

1) The first segment is a positive segment: $\oplus, \ominus, \oplus, \ominus, \ldots$.
2) The first segment is a negative segment: $\ominus, \oplus, \ominus, \oplus, \ominus, \ldots$.

Let $\xi^+$ denotes the number of positive segments and $\xi^-$ denotes the number of negative segments. Then the following holds:

1) $\xi^+ = \xi^- = \xi/2$ when $\xi$ is even.
2) $\xi^+ = (\xi + 1)/2$ and $\xi^- = (\xi - 1)/2$ when $\xi$ is odd and the first segment is positive.
3) $\xi^+ = (\xi - 1)/2$ and $\xi^- = (\xi + 1)/2$ when $\xi$ is odd and the first segment is negative.

If $P$ and $Q$ are the numbers of ways to separate positive and negative examples into their corresponding segments, then the number of distributions is $P \cdot Q$.

Let $k = (\xi - \xi\%2)/2$, and $\text{Group}(M, k)$ denotes the number of ways to separate $M$ examples into $k$ groups. Then, the number of distributions for $\xi$ segments is

$$T(\xi) = \begin{cases} 
2\text{Group}(pN, k) \cdot \text{Group}((1-p)N, k) & \text{when } \xi = 2k \\
\text{Group}(pN, k+1) \cdot \text{Group}((1-p)N, k+1) + \\
\text{Group}(pN, k) \cdot \text{Group}((1-p)N, k+1) & \text{when } \xi = 2k+1, pN > k, (1-p)N > k \\
\text{Group}(pN, k+1) \cdot \text{Group}((1-p)N, k) & \text{when } \xi = 2k+1, (1-p)N = k \\
\text{Group}(pN, k) \cdot \text{Group}((1-p)N, k+1) & \text{when } \xi = 2k+1, pN = k. 
\end{cases}$$

Based on the above discussions, the problem is simplified to get the number of ways to separate $M$ examples into $k$ groups. We first suppose that the $M$ examples are in fixed order. In this case, the problem is further transferred to insert $k - 1$ separating marks into the $M$ examples. Obviously, this equals to select $k - 1$ inserting positions from the $M - 1$ positions that are between any two adjacent examples in this order. As a result, the number of ways to separate $M$ ordered examples into $k$ groups could be derived as

$$\text{Group}(M, k) = \mathcal{C}_{M-1}^{k-1}$$

where $C$ represents the combination. By further considering the different ranking orders of the $M$ examples, we get

$$\text{Group}(M, k) = \mathcal{A}_{M-1}^{k-1}$$

where $A$ represents the permutation.

Finally, by applying (19) and (20), we get (16).

REFERENCES

Xi-Zhao Wang (M'03–SM’04–F'12) received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1998. Since 2001, he has been a Full Professor and the Dean of the College of Mathematics and Computer Science, Hebei University, Hebei, China. From 1998 to 2001, he was a Research Fellow with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. Since 2014, he is also a Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His current research interests include supervised learning, active learning, reinforcement learning, manifold learning, transfer learning, unstructured learning, uncertainty, fuzzy sets and systems, fuzzy measures and integrals, rough set, and learning from big data.

Dr. Wang was the recipient of several awards from the IEEE SMC Society. He is a member of the Board of Governors of the IEEE International Conference on Systems, Man, and Cybernetics (SMC) in 2005, 2007–2009, and 2012–2014, the Chair of the Technical Committee on Computational Intelligence of the IEEE SMC, and a Distinguished Lecturer of the IEEE SMC. He was the Program Co-Chair of the IEEE SMC, in 2009 and 2010. He is an Editor-in-Chief of the International Journal of Machine Learning and Cybernetics. He is also an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, the Information Sciences journal, and the International Journal of Pattern Recognition and Artificial Intelligence.

Qingshan Jiang received the Ph.D. degrees in mathematics and computer science from Chiba University, Chiba, Japan, and from the University of Sherbrooke, Sherbrooke, QC, Canada, in 1996 and 2002, respectively.

He is a Professor with the Shenzhen Key Laboratory for High Performance Data Mining, Shenzhen, China, Shenzhen Institutes of Advanced Technology, Shenzhen, and the Chinese Academy of Sciences, Beijing, China. He has been a Professor with Xiamen University, Xiamen, China, since 2003. In 1999, he was a Post-Doctoral Fellow with the Fields Institute for Research in Mathematical Sciences, University of Toronto, Toronto, ON, Canada. His current research interests include pattern recognition, data mining, image processing, statistical analysis, and fuzzy logic.

Sam Kwong (M’93–SM’04–F’13) received the B.Sc. and M.S. degrees in electrical engineering from the State University of New York, Buffalo, NY, USA, and the University of Waterloo, Waterloo, ON, Canada, in 1983 and 1985, respectively, and the Ph.D. degree from the University of Hagen, Hagen, Germany, in 1996.

From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada, Mississauga, ON, Canada. He joined Bell Northern Research Canada as a member of Scientific Staff. In 1990, he became a Lecturer with the Department of Electronic Engineering, the City University of Hong Kong, Hong Kong, where he is currently a Professor and the Head of the Department of Computer Science. His current research interests include evolutionary computation, video coding, pattern recognition, and machine learning.

Dr. Kwong is an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and the Information Sciences journal.