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].

Manuscript received December 13, 2013; revised March 2, 2014 and July 30, 2014; accepted August 6, 2014. Date of publication September 29, 2014; date of current version June 12, 2015. This work was supported by the National Natural Science Foundation of China under Grant 61272289, Grant 61175123, and Grant 61170040. This paper was recommended by Associate Editor J. Basak.

R. Wang is with the Department of Computer Science, City University of Hong Kong, Hong Kong, and also with the Shenzhen Key Laboratory for High Performance Data Mining, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: ranwang3-c@my.cityu.edu.hk; wangran@siat.ac.cn).

S. Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: cssamk@cityu.edu.hk).

X.-Z. Wang is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: xizhaowang@ieee.org).

Q. Jiang is with the Shenzhen Key Laboratory for High Performance Data Mining, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: qs.jiang@siat.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2348012

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 deal 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 presorting 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 $\mathbb{S} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}$ be a node with N examples. Each example in $\mathbb{S}$ is represented by a group of conditional attributes $\mathbb{A} = \{A_1, A_2, \ldots, A_m\}$ 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 ith example in $\mathbb{S}$ is expressed as $\mathbf{e}_i = \{a_{i1}, a_{i2}, \ldots, a_{im}, c_i\}$ where a_{ij} is written as $A_j(\mathbf{e}_i)$ and denotes the value of the i-th example with respect to the j-th attribute; c_i is written as $C(\mathbf{e}_i)$ and denotes the class label of the i-th example. Suppose all the conditional attributes are continuous, then we have $A_j(\mathbf{e}_i) \in R$ 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 $\mathbb{S} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}$ be a node with N examples and A_j be a continuous valued attribute. Suppose all the examples in $\mathbb{S}$ are ranked by ascending values of A_j, i.e., $A_j(\mathbf{e}_1) < A_j(\mathbf{e}_2) < \ldots < A_j(\mathbf{e}_N)$. 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 $\mathbb{S}$.

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

$$CCP(\mathbb{S}, A_j) = \left\{x_{ji} | x_{ji} = \frac{A_j(\mathbf{e}_i) + A_j(\mathbf{e}_{i+1})}{2}, i = 1 \ldots, N-1\right\}. \quad (1)$$

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

Definition 3 (Partition): Let $\mathbb{S} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}$ 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 $\mathbb{S}$. If $\mathbb{S}_1 = \{\mathbf{e} \in \mathbb{S} | A_j(\mathbf{e}) \leq x_{ji}\}$ and $\mathbb{S}_2 = \{\mathbf{e} \in \mathbb{S} | A_j(\mathbf{e}) > x_{ji}\}$, then $\{\mathbb{S}_1, \mathbb{S}_2\}$ is called a partition of $\mathbb{S}$ induced by x_{ji}.

B. Frequency Based Heuristic Measures

Given that $\{\mathbb{S}_1, \mathbb{S}_2\}$ is a partition of node $\mathbb{S}$ induced by x_{ji}, the information gain of x_{ji} in node $\mathbb{S}$ is defined as

$$\text{Gain}(\mathbb{S}, x_{ji}) = f(\mathbb{S}) - \sum_{k=1}^{2} \frac{|\mathbb{S}_k|}{|\mathbb{S}|} f(\mathbb{S}_k) \quad (2)$$

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

There exists many forms for function $f(\mathbb{S}) = f(\mathbb{S}; p_1, p_2, \ldots, p_L)$. 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 $\mathbb{S}$ is defined as

$$f(\mathbb{S}) = -\sum_{l=1}^{L} p_l \log_2 p_l. \quad (3)$$

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., $p_l = 1$ 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., $p_l = 1/L$ 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 $\mathbb{S}$ is defined as

$$f(\mathbb{S}) = 1 - \sum_{l=1}^{L} p_l^2. \quad (4)$$

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

$$f(\mathbb{S}) = \sum_{l=1}^{L} \sin^2(\pi p_l) \quad (5)$$

sqrt measure defined as

$$f(\mathbb{S}) = \sqrt{\prod_{l=1}^{L} p_l} \quad (6)$$

and psin measure defined as

$$f(\mathbb{S}) = \sum_{l=1}^{L} p_l \sin(\pi p_l). \quad (7)$$

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

$$\begin{cases} f(p) = -p\log_2 p - (1-p)\log_2(1-p) \\ f(p) = -2p^2 + 2p \\ f(p) = \sin^2(\pi p) + \sin^2(\pi(1-p)) \\ f(p) = \sqrt{p(1-p)} \\ f(p) = p\sin(\pi p) + (1-p)\sin(\pi(1-p)). \end{cases} \quad (8)$$

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.

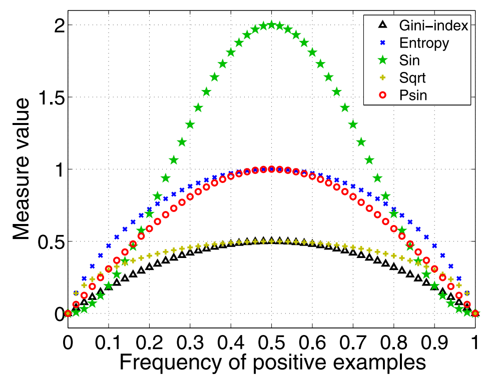


Fig. 1. Relationship between measure values and frequency of positive class.

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

$$\text{Split}\left(\mathbb{S}, x_{ji}\right) = -\frac{|\mathbb{S}_1|}{|\mathbb{S}|}\log_2\frac{|\mathbb{S}_1|}{|\mathbb{S}|} - \frac{|\mathbb{S}_2|}{|\mathbb{S}|}\log_2\frac{|\mathbb{S}_2|}{|\mathbb{S}|} \quad (9)$$

and the gain ratio is measured by

$$SP\left(\mathbb{S}, x_{ji}\right) = \frac{\text{Gain}\left(\mathbb{S}, x_{ji}\right)}{\text{Split}\left(\mathbb{S}, x_{ji}\right)} \quad (10)$$

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

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 $\hat{N}$, 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 $\mathbb{S}$ 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 $\{\mathbf{e}_i\}_{i=1}^N$ with m continuous valued attributes $A = \{A_j\}_{j=1}^m$ and one decision attribute $C \in \{C_l\}_{l=1}^L$; threshold number $\hat{N}$ to stop splitting a node.
Output: A binary DT.

Initialize Ω as an empty set;
Consider the set of all examples as the root-node, and add it to Ω;
while Ω *is not empty* **do**
  Select a node from Ω, denoted by $\mathbb{S}$;
  if $|\mathbb{S}| < \hat{N}$ **then**
    Treat $\mathbb{S}$ as a leaf node and assign it the class label $l^* = \arg\max_{l=1,\ldots,L} p_l$;
  else
    For each ASA A_j, $j = 1, \ldots, m$, sort the examples with ascending order;
    Get the CCPs of each ASA based on (1), i.e., x_{ji}, where $j = 1, \ldots, m$ and $i = 1, \ldots, N-1$;
    Calculate the splitting performance of each CCP based on (10), i.e., $SP(\mathbb{S}, x_{ji})$;
    Find the optimal splitting attribute A_{j^*} and its optimal CCP $x_{j^*i^*}$, where $\{j^*, i^*\} = \arg\max_{\{j,i\}} SP(\mathbb{S}, x_{ji})$;
    Split $\mathbb{S}$ into two child-nodes by $x_{j^*i^*}$, i.e., $\mathbb{S}_1 = \{\mathbf{e} \in \mathbb{S} | A_{j^*}(\mathbf{e}) \le x_{j^*i^*}\}$ and $\mathbb{S}_2 = \{\mathbf{e} \in \mathbb{S} | A_{j^*}(\mathbf{e}) > x_{j^*i^*}\}$;
    for $i = 1, 2$ **do**
      if *all the examples in* $\mathbb{S}_i$ *are from the same class* l^* **then**
        Treat $\mathbb{S}_i$ as a leaf node and assign it the class label l^*;
      else
        Add $\mathbb{S}_i$ to Ω;
      end
    end
  end
  Remove $\mathbb{S}$ from Ω;
end
return *the constructed tree.*

S_1: 1 1 1 1 1 1 2 2 2 2

S_2: 1 2 1 2 1 2 1 2 1 1

$\longrightarrow$

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., $\mathbb{S} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}$ 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 $\mathbb{S}$.

Definition 4 (Segment): Let $\mathcal{Q}$ be a permutation of examples in $\mathbb{S}$. Then a sub-queue of $\mathcal{Q}$, i.e., $\mathcal{SQ} = (\mathbf{e}_r, \mathbf{e}_{r+1}, \ldots, \mathbf{e}_t)$, $1 \le r \le t \le N$, is called a segment if and only if it satisfies the following requirements.

1) $C(\mathbf{e}_r) = C(\mathbf{e}_{r+1}) = \ldots = C(\mathbf{e}_t)$.
2) $C(\mathbf{e}_r) \ne C(\mathbf{e}_{r-1})$ if and only if $r \ne 1$.
3) $C(\mathbf{e}_t) \ne C(\mathbf{e}_{t+1})$ if and only if $t \ne N$.

The set of all segments in $\mathcal{Q}$ is denoted by $Seg(\mathcal{Q})$.

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

Specifically, if attribute A_j and its CCP x_{ji} split the node $\mathbb{S}$ into $\mathbb{S}_1 = \{\mathbf{e} \in \mathbb{S} | A_j(\mathbf{e}) \le x_{ji}\}$ and $\mathbb{S}_2 = \{\mathbf{e} \in \mathbb{S} | A_j(\mathbf{e}) > x_{ji}\}$, then we denote the sets of segments in $\mathbb{S}_1$ and $\mathbb{S}_2$ by $Seg(\mathbb{S}_1; x_{ji})$ and $Seg(\mathbb{S}_2; x_{ji})$ respectively.

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

$$Seg\#(\mathbb{S}) = \min_j \left| Seg\left(\mathbb{S}, A_j\right) \right|$$

where $|\ |$ denotes the number of elements in a set.

When the attribute values of examples in $\mathbb{S}$ 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 $\mathbb{S}$, and let this value be a, then the set of examples $\{\mathbf{e} \in \mathbb{S} | A_j(\mathbf{e}) = a\}$ is called a bar in $\mathbb{S}$ with respect to A_j, denoted by $Bar(\mathbb{S}, A_j = a)$. The value a is called a bar point.

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

$$bSeg\#(\mathcal{B}) = \left| Seg\left(\mathcal{Q}^*\right) \right|.$$

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 "1,2,1,2,1,2,1,2,1,1" and "1,2,1,2,1,1,2,1,2,1" could be treated

Algorithm 2: Computing the Number of Segments in a Bar

Input: All the examples in a bar $\mathcal{B}$.
Output: The number of segments in bar $\mathcal{B}$: $bSeg\#(\mathcal{B})$.

```
Get the numbers of examples belonging to each class: n_1, n_2, ..., n_L;
Sort n_1, n_2, ..., n_L in descending order. Suppose the sorted values are n'_1, n'_2, ..., n'_L;
Set initial value: bSeg#(B) = 0;
while n'_2 > 0 do
    bSeg#(B) = bSeg#(B) + 2n'_2;
    Set n_1 = n'_1 - n'_2, n_2 = 0 and n_i = n'_i for each i(3 <= i <= L), thus we get new values of n_1, n_2, ..., n_L;
    Get the sorted values n'_1, n'_2, ..., n'_L of n_1, n_2, ..., n_L in descending order;
end
if n'_1 != 0 then
    bSeg#(B) = bSeg#(B) + 1;
end
return bSeg#(B).
```

as the most chaotic permutations of "1,1,1,1,1,1,2,2,2,2." 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 $\mathbb{S}$ with respect to A_j. Then we can get t bar sub-queues $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_t$ and u nonbar sub-queues $\mathcal{SQ}_1, \mathcal{SQ}_2, \ldots, \mathcal{SQ}_u$, 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 $\mathbb{S}$ induced by A_j, denoted by $Seg(\mathbb{S}, A_j)$, is computed as

$$Seg\#\left(\mathbb{S}, A_j\right) = \sum_{i=1}^{t} bSeg\#\left(\mathcal{B}_i\right) + \sum_{i=1}^{u} \left|Seg\left(\mathcal{SQ}_i\right)\right|. \quad (11)$$

C. Illustrative Example

We now give an illustrative example to calculate the number of segments in a node induced by an attribute with duplicated

TABLE I
NODE $\mathbb{S}$ WITH 20 EXAMPLES

$\mathbf{e}$	$\mathbf{e}_1$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$
$A_j(\mathbf{e})$	3	3.2	3.6	3.9	4.5
$C(\mathbf{e})$	1	1	2	1	1
$\mathbf{e}$	$\mathbf{e}_6$	$\mathbf{e}_7$	$\mathbf{e}_8$	$\mathbf{e}_9$	$\mathbf{e}_{10}$
$A_j(\mathbf{e})$	5.0	5.0	5.0	5.6	6.1
$C(\mathbf{e})$	1	1	3	1	2
$\mathbf{e}$	$\mathbf{e}_{11}$	$\mathbf{e}_{12}$	$\mathbf{e}_{13}$	$\mathbf{e}_{14}$	$\mathbf{e}_{15}$
$A_j(\mathbf{e})$	6.8	7.2	7.7	8.0	8.0
$C(\mathbf{e})$	1	3	3	1	1
$\mathbf{e}$	$\mathbf{e}_{16}$	$\mathbf{e}_{17}$	$\mathbf{e}_{18}$	$\mathbf{e}_{19}$	$\mathbf{e}_{20}$
$A_j(\mathbf{e})$	8.0	8.0	8.0	8.6	9.0
$C(\mathbf{e})$	3	1	2	1	3

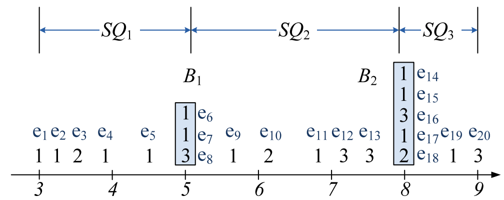


Fig. 3. Distribution of examples in Table I.

values. Let $\mathbb{S}$ 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 $\mathbb{S}$ 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 $\mathcal{B}_1 = Bar(\mathbb{S}, A_j = 0.5) = (\mathbf{e}_6, \mathbf{e}_7, \mathbf{e}_8)$ and $\mathcal{B}_2 = Bar(\mathbb{S}, A_j = 0.8) = (\mathbf{e}_{14}, \mathbf{e}_{15}, \mathbf{e}_{16}, \mathbf{e}_{17}, \mathbf{e}_{18})$. The three nonbar sub-queues are $\mathcal{SQ}_1 = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_5)$, $\mathcal{SQ}_2 = (\mathbf{e}_9, \mathbf{e}_{10}, \mathbf{e}_{11}, \mathbf{e}_{12}, \mathbf{e}_{13})$ and $\mathcal{SQ}_3 = (\mathbf{e}_{19}, \mathbf{e}_{20})$. Then the number of segments in $\mathbb{S}$ with respect to A_j is

$$\begin{aligned} Seg\#(\mathbb{S}, A_j) &= |Seg(\mathcal{SQ}_1)| + |Seg(\mathcal{SQ}_2)| + |Seg(\mathcal{SQ}_3)| \\ &\quad + bSeg\#(\mathcal{B}_1) + bSeg\#(\mathcal{B}_2) \\ &= 3 + 4 + 2 + 3 + 5 \\ &= 17. \end{aligned}$$

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 $\mathbb{S}$ 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

$$SP^* = \max_{\{j,i\}} SP\left(\mathbb{S}, x_{ji}\right) \quad (12)$$

where $SP(\mathbb{S}, x_{ji})$ 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 $\{\mathbf{e}_i\}_{i=1}^{N}$ with m continuous valued attributes $A = \{A_j\}_{j=1}^{m}$ and one decision attribute $C \in \{C_l\}_{l=1}^{L}$; threshold number $\hat{N}$ to stop splitting a node and parameter $\hat{K}$.

Output: A binary DT.

1 Initialize Ω as an empty set;
2 Consider the set of all examples as the root-node, and add it to Ω;
3 **while** Ω *is not empty* **do**
4 Select a node from Ω, denoted by $\mathbb{S}$;
5 **if** $|\mathbb{S}| < \hat{N}$ **then**
6 Treat $\mathbb{S}$ as a leaf node and assign it the class label $l^* = \arg\max_{l=1,\ldots,L} p_l$;
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., $SP(\mathbb{S}, x_{ji})$;
11 Get the best splitting performance SP^* by (12);
12 Get the set $\mathbb{X}$ 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 $\mathbb{X}$;
14 Get the optimal splitting attribute A_{j^*} and its optimal CCP $x_{j^*i^*}$ by (13);
15 Split $\mathbb{S}$ into two child-nodes by $x_{j^*i^*}$, i.e., $\mathbb{S}_1 = \{\mathbf{e} \in \mathbb{S} | A_{j^*}(\mathbf{e}) \leq x_{j^*i^*}\}$ and $\mathbb{S}_2 = \{\mathbf{e} \in \mathbb{S} | A_{j^*}(\mathbf{e}) > x_{j^*i^*}\}$;
16 **for** $i = 1, 2$ **do**
17 **if** *all the examples in* $\mathbb{S}_i$ *are from the same class* l^* **then**
18 Treat $\mathbb{S}_i$ as a leaf node and assign it the class label l^*;
19 **else**
20 Add $\mathbb{S}_i$ to Ω;
21 **end**
22 **end**
23 **end**
24 Remove $\mathbb{S}$ from Ω;
25 **end**
26 **return** *the constructed tree.*

to form the subset $\mathbb{X}$, where $j = 1, \ldots, m$ and $i = 1, \ldots, N-1$. Finally, the optimal CCP $x_{j^*i^*}$ is derived by

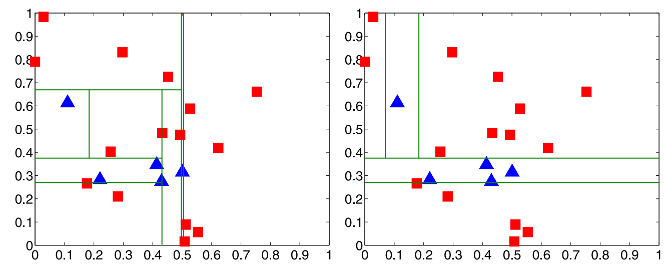
$$x_{j^*i^*} = \arg\min_{x_{ji} \in \mathbb{X}} \left[\frac{|\mathbb{S}_1|}{|\mathbb{S}|} \left| Seg\left(\mathbb{S}_1; x_{ji}\right) \right| + \frac{|\mathbb{S}_2|}{|\mathbb{S}|} \left| Seg\left(\mathbb{S}_2; x_{ji}\right) \right| \right]. \tag{13}$$

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.

TABLE II
2-D Binary Dataset

sample	feature1	feature2	label	sample	feature1	feature2	label
1	0.0000	0.7903	2	11	0.1106	0.6129	1
2	0.2207	0.2823	1	12	0.6232	0.4194	2
3	0.4136	0.3468	1	13	0.5009	0.3145	1
4	0.5274	0.5887	2	14	0.4302	0.2742	1
5	0.4327	0.4839	2	15	0.2815	0.2097	2
6	0.5120	0.0887	2	16	0.4524	0.7258	2
7	0.7535	0.6613	2	17	0.0289	0.9839	2
8	0.5083	0.0161	2	18	0.2569	0.4032	2
9	0.2969	0.8306	2	19	0.5538	0.0565	2
10	0.1764	0.2661	2	20	0.4935	0.4758	2



(a) (b)

Fig. 4. Difference between DTs induced by C4.5 and the proposed algorithm on the 2-D dataset. (a) DT of C4.5. (b) DT of Segment+C4.5.

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 $\hat{N} = 2$ 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 $\mathbb{S}$ be a node with N examples, and $p(0 < p < 1)$ be the frequency of positive class in $\mathbb{S}$. We suppose that the number of segments in $\mathbb{S}$ is ξ, where ξ is a positive integer that can take values of $2, 3, \ldots, M(p, N)$, thus

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

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

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

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

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

$$T(\xi) = \begin{cases} 2(pN)!\,(N-pN)!\,\mathcal{C}_{pN-1}^{k-1}\mathcal{C}_{(1-p)N-1}^{k-1} \\ \text{when } \xi = 2k \\ \\ (pN)!\,(N-pN)!\left(\mathcal{C}_{pN-1}^{k}\mathcal{C}_{(1-p)N-1}^{k-1} + \mathcal{C}_{pN-1}^{k-1}\mathcal{C}_{(1-p)N-1}^{k}\right) \\ \text{when } \xi = 2k+1, pN > k, (1-p)N > k \\ \\ (pN)!\,(N-pN)!\,\mathcal{C}_{pN-1}^{k}\mathcal{C}_{(1-p)N-1}^{k-1} \\ \text{when } \xi = 2k+1, (1-p)N = k \\ \\ (pN)!(N-pN)!\,\mathcal{C}_{pN-1}^{k-1}\mathcal{C}_{(1-p)N-1}^{k} \\ \text{when } \xi = 2k+1, pN = k. \end{cases} \tag{16}$$

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

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

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 ξ corresponding to distribution (15) is computed as

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

where r_ξ 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 $N = 10, 20, \ldots, 100$.

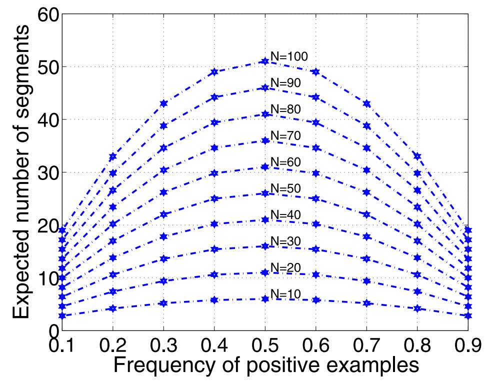


Fig. 5. Relationship between expected number of segments and frequency of positive class for different N.

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 $\mathbb{S}$ be a node with N examples. Suppose the number of ASAs in $\mathbb{S}$ is m and there are β_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 $\mathbb{S}$ in Algorithm 1 is $O\big(mN \log N + (N + 1) \sum_{j=1}^{m} \beta_j\big)$. 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

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

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 $O\big(mN \log N + (N + 1) \sum_{j=1}^{m} \beta_j\big)$. 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 $\hat{K}$

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

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

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 $(|\mathbb{S}_1|/|\mathbb{S}|)|Seg(\mathbb{S}_1; x_{ji})| + (|\mathbb{S}_2|/|\mathbb{S}|)|Seg(\mathbb{S}_2; x_{ji})|$ 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[1] 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 $1-((x_{\max}-x)/(x_{\max}-x_{\min}))$, where $x_{\max}$ and $x_{\min}$ 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 5×2-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 $((d_2-d_1)/d_1) * 100\%$. 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

[1]http://www.ics.uci.edu/~mlearn/MLRepository.html

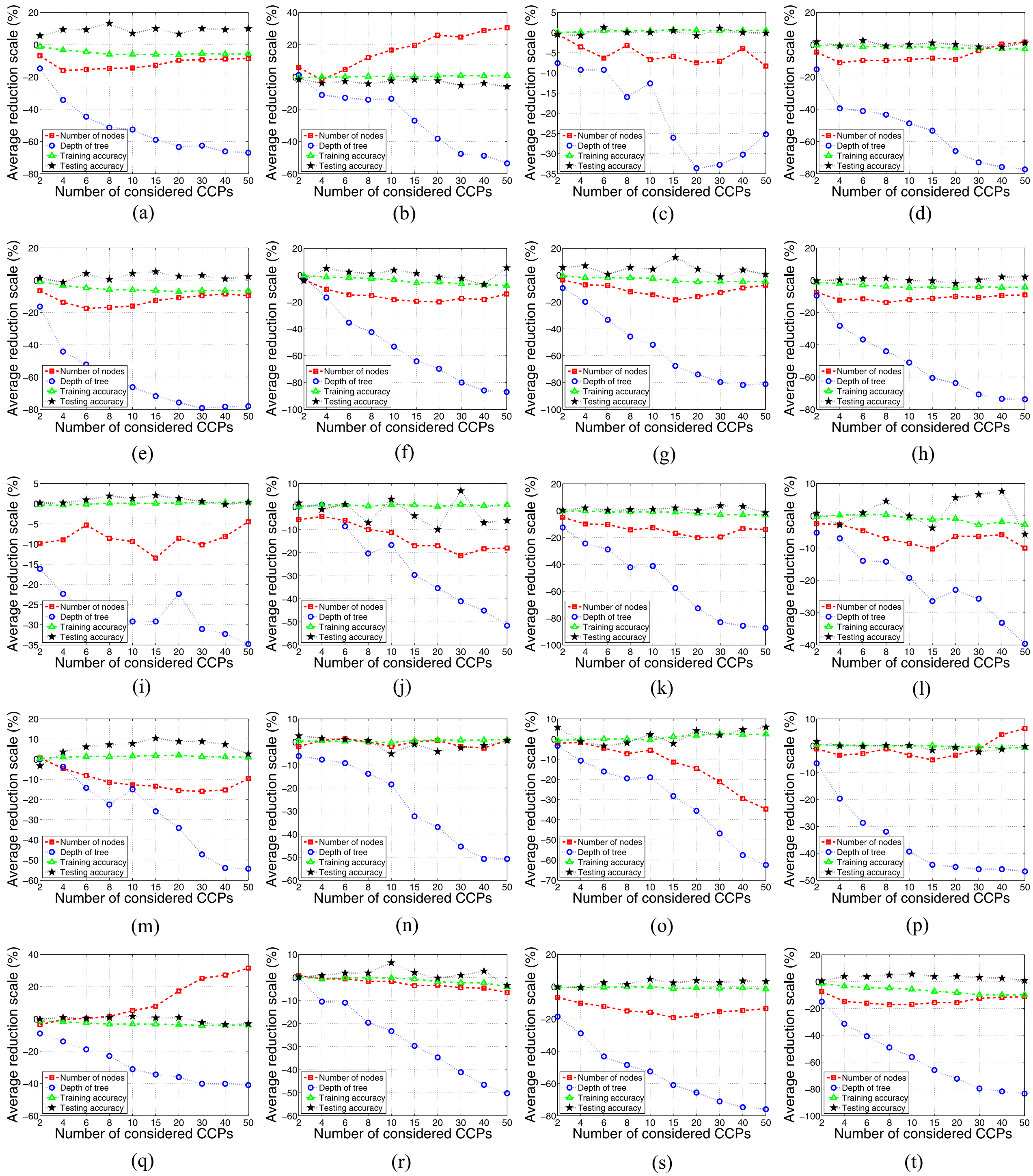


Fig. 6. Average reduction scale in accuracy and tree size of method *Segment+C4.5* compared with method *C4.5*. (a) Haberman. (b) Ionosphere. (c) Cancer. (d) Australian. (e) German. (f) Bupa. (g) Heart. (h) Transfusion. (i) Wdbc. (j) Wpbc. (k) Pima. (l) Plrx. (m) SPECTF. (n) CT. (o) Sonar. (p) Cotton. (q) Ecoli. (r) Libras. (s) Vowel. (t) Yeast.

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., $|\mathbb{S}_1|/|\mathbb{S}||Seg(\mathbb{S}_1; x_{ji})| + |\mathbb{S}_2|/|\mathbb{S}||Seg(\mathbb{S}_2; x_{ji})|$. 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., $|Seg(\mathbb{S}_1; x_1)| + |Seg(\mathbb{S}_2; x_1)| = |Seg(\mathbb{S}_1; x_2)| + |Seg(\mathbb{S}_2; x_2)|$. As demonstrated in Fig. 5, a larger number of examples corresponds to a larger expected number of segments, thus we further assume:

1) $|\mathbb{S}_1|/|\mathbb{S}| = 0.1$, $|\mathbb{S}_2|/|\mathbb{S}| = 0.9$, $|Seg(\mathbb{S}_1; x_1)| = 1$ and $|Seg(\mathbb{S}_2; x_1)| = 9$ for x_1;

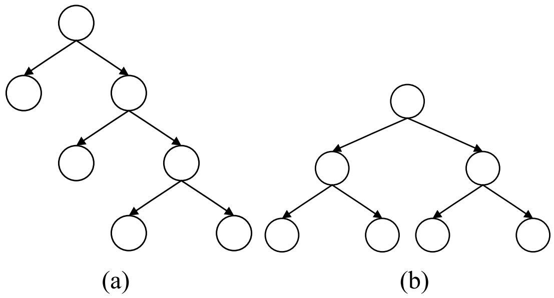


Fig. 7. Two DTs with different depths. (a) Tree with larger depth. (b) Tree with lower depth.

2) $|\mathbb{S}_1|/|\mathbb{S}| = 0.5$, $|\mathbb{S}_2|/|\mathbb{S}| = 0.5$, $|Seg(\mathbb{S}_1; x_2)| = 5$ and $|Seg(\mathbb{S}_2; x_2)| = 5$ for x_2.

Then, we have $|\mathbb{S}_1|/|\mathbb{S}||Seg(\mathbb{S}_1; x_1)| + |\mathbb{S}_2|/|\mathbb{S}||Seg(\mathbb{S}_2; x_1)| = 0.82$ and $|\mathbb{S}_1|/|\mathbb{S}||Seg(\mathbb{S}_1; x_2)| + |\mathbb{S}_2|/|\mathbb{S}||Seg(\mathbb{S}_2; x_2)| = 0.5$. 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., $|\mathbb{S}_1|/|\mathbb{S}||Seg(\mathbb{S}_1; x_{ji})| + |\mathbb{S}_2|/|\mathbb{S}||Seg(\mathbb{S}_2; x_{ji})|$, 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.

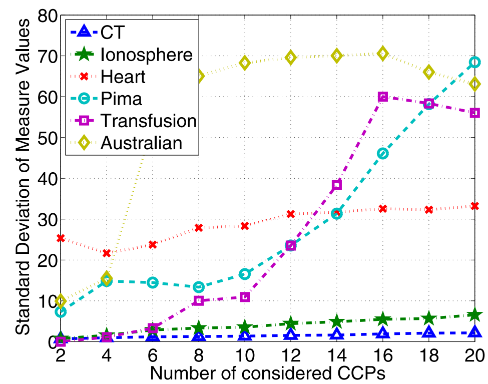


Fig. 8. Standard deviation of segment based measure values of the $\hat{K}$ considered CCPs when splitting the root-node.

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 $\surd$ 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, $\uparrow$ and $\downdownarrows$ 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

TABLE III
SELECTED DATASETS FOR PERFORMANCE COMPARISON

Dataset	Example#	Conditional Attribute#	Continuous Attribute#	Data Type	Class#	Class Distribution	Missing Values	$\hat{K}$
Haberman	306	3	3	Integer	2	225/81	None	8
Ionosphere	351	34	32	Real+Integer	2	225/126	None	2
Cancer	699(683)	9	8	Integer	2	458(444)/241(239)	16	6
Australian	690	14	7	Real+Integer	2	307/383	None	6
German	1000	24	3	Integer	2	700/300	None	15
Bupa	345	6	6	Real	2	145/200	None	50
Heart	270	13	5	Real+Integer	2	150/120	None	15
Transfusion	748	4	4	Integer	2	178/570	None	50
Wdbc	569	30	30	Real	2	212/357	None	15
Wpbc	198	33	33	Real	2	47/151	None	30
Pima	768	8	8	Real+Integer	2	268/500	None	30
Plrx	182	12	12	Real	2	130/52	None	40
SPECTF	267	44	44	Integer	2	212/55	None	15
CT	221	36	36	Real	2	101/120	None	2
Sonar	208	60	60	Real	2	97/111	None	50
Cotton	356	21	21	Real	6	100×6	None	2
Ecoli	336	5	5	Real	8	143/77/2/2/35/20/5/52	None	10
Libras	360	90	90	Real	15	24×15	None	10
Vowel	990	10	10	Real	11	90×11	None	10
Yeast	1484	6	6	Real	10	244/429/463/44/51/163/35/30/20/5	None	10

Note: The datum 699 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)" and "241(239)".

TABLE IV
COMPARISONS OF DIFFERENT DT INDUCTION HEURISTICS: TESTING ACCURACY

Datasets	IDE3	C4.5	CART	SIN	SQRT	PSIN	Segment	Segment+C4.5
Haberman	$67.09_{\pm 8.96}\surd$	$58.88_{\pm 7.46}$	$63.46_{\pm 6.78}$	$63.38_{\pm 6.78}$	$64.76_{\pm 8.64}$	$66.41_{\pm 7.60}$	$64.46_{\pm 6.66}$	$66.65_{\pm 6.66}\uparrow$
Ionosphere	$90.59_{\pm 5.73}$	$93.70_{\pm 3.37}\surd$	$89.13_{\pm 6.41}$	$88.59_{\pm 3.86}$	$90.87_{\pm 6.24}$	$89.15_{\pm 4.78}$	$76.63_{\pm 7.85}$	$92.28_{\pm 3.67}\downdownarrows$
Cancer	$94.58_{\pm 3.02}$	$94.00_{\pm 3.36}$	$94.29_{\pm 2.80}$	$93.41_{\pm 2.73}$	$94.88_{\pm 2.09}$	$94.58_{\pm 2.69}$	$81.12_{\pm 4.21}$	$95.17_{\pm 1.86}\uparrow\surd$
Australian	$70.87_{\pm 4.02}$	$73.04_{\pm 4.33}$	$72.89_{\pm 3.81}$	$73.48_{\pm 6.52}$	$71.03_{\pm 5.10}$	$72.01_{\pm 4.31}$	$58.39_{\pm 5.33}$	$74.93_{\pm 3.44}\uparrow\surd$
German	$62.70_{\pm 4.96}$	$60.70_{\pm 5.18}$	$63.90_{\pm 3.70}$	$64.80_{\pm 5.42}$	$62.10_{\pm 3.78}$	$62.70_{\pm 4.10}$	$70.00_{\pm 2.93}\surd$	$63.90_{\pm 2.70}\uparrow$
Bupa	$63.26_{\pm 7.54}$	$63.53_{\pm 7.11}$	$62.10_{\pm 6.72}$	$61.73_{\pm 6.47}$	$65.79_{\pm 6.65}$	$63.50_{\pm 6.84}$	$49.55_{\pm 6.54}$	$66.99_{\pm 5.77}\uparrow\surd$
Heart	$65.19_{\pm 7.80}$	$58.52_{\pm 8.57}$	$67.78_{\pm 6.43}\surd$	$62.96_{\pm 8.11}$	$67.41_{\pm 3.63}$	$63.33_{\pm 7.85}$	$53.70_{\pm 8.65}$	$66.30_{\pm 9.58}\uparrow$
Transfusion	$73.51_{\pm 4.87}$	$74.06_{\pm 3.90}$	$73.25_{\pm 5.23}$	$76.19_{\pm 4.30}\surd$	$72.72_{\pm 4.28}$	$72.59_{\pm 5.42}$	$75.80_{\pm 0.89}$	$75.53_{\pm 3.37}\uparrow$
Wdbc	$92.80_{\pm 2.54}$	$92.80_{\pm 1.62}$	$93.36_{\pm 3.84}$	$91.21_{\pm 2.60}$	$94.39_{\pm 3.40}$	$93.69_{\pm 3.10}$	$83.28_{\pm 6.55}$	$94.73_{\pm 2.20}\uparrow\surd$
Wpbc	$73.33_{\pm 9.81}$	$70.01_{\pm 9.34}$	$67.27_{\pm 9.31}$	$67.62_{\pm 6.02}$	$71.75_{\pm 9.42}$	$70.32_{\pm 9.11}$	$70.80_{\pm 4.80}$	$74.81_{\pm 8.22}\uparrow\surd$
Pima	$71.74_{\pm 4.64}$	$69.53_{\pm 2.79}$	$69.14_{\pm 4.93}$	$67.95_{\pm 5.17}$	$69.40_{\pm 3.79}$	$68.09_{\pm 3.55}$	$67.58_{\pm 3.37}$	$72.12_{\pm 6.88}\uparrow\surd$
Plrx	$60.53_{\pm 7.96}$	$57.75_{\pm 11.36}$	$64.82_{\pm 12.47}\surd$	$52.25_{\pm 9.13}$	$58.30_{\pm 11.82}$	$60.53_{\pm 11.07}$	$56.64_{\pm 7.24}$	$62.13_{\pm 13.30}\uparrow$
SPECTF	$76.11_{\pm 8.87}$	$71.24_{\pm 8.69}$	$74.66_{\pm 10.57}$	$71.09_{\pm 9.66}$	$75.94_{\pm 7.47}$	$71.94_{\pm 7.67}$	$78.34_{\pm 4.37}$	$78.63_{\pm 6.11}\uparrow\surd$
CT	$86.46_{\pm 7.21}$	$86.90_{\pm 7.68}$	$86.44_{\pm 7.56}$	$82.83_{\pm 6.29}$	$86.01_{\pm 9.76}$	$84.17_{\pm 9.79}$	$52.98_{\pm 10.14}$	$89.17_{\pm 6.08}\uparrow\surd$
Sonar	$76.47_{\pm 7.75}\surd$	$67.71_{\pm 8.75}$	$73.04_{\pm 7.86}$	$70.23_{\pm 7.20}$	$76.46_{\pm 5.67}$	$73.06_{\pm 7.54}$	$53.84_{\pm 10.59}$	$71.68_{\pm 5.20}\uparrow$
Cotton	$84.66_{\pm 1.83}$	$84.32_{\pm 2.61}$	$84.04_{\pm 3.00}$	$82.58_{\pm 1.86}$	$84.43_{\pm 1.84}$	$74.99_{\pm 5.52}$	$64.21_{\pm 3.01}$	$85.67_{\pm 3.27}\uparrow\surd$
Ecoli	$77.80_{\pm 1.80}$	$77.56_{\pm 1.79}$	$77.73_{\pm 2.70}$	$76.72_{\pm 4.22}$	$70.31_{\pm 3.09}$	$66.01_{\pm 3.22}$	$50.12_{\pm 2.17}$	$78.87_{\pm 2.54}\uparrow\surd$
Libras	$54.61_{\pm 5.61}$	$54.89_{\pm 4.31}$	$55.61_{\pm 4.79}$	$36.78_{\pm 2.11}$	$23.50_{\pm 6.31}$	$35.00_{\pm 3.48}$	$20.06_{\pm 1.98}$	$58.39_{\pm 4.74}\uparrow\surd$
Vowel	$72.85_{\pm 2.38}\surd$	$68.10_{\pm 2.40}$	$72.08_{\pm 2.30}$	$43.68_{\pm 2.38}$	$55.25_{\pm 4.49}$	$41.35_{\pm 3.42}$	$23.82_{\pm 1.03}$	$71.25_{\pm 1.47}\uparrow$
Yeast	$49.66_{\pm 1.30}$	$48.02_{\pm 1.69}$	$49.84_{\pm 1.79}$	$43.44_{\pm 1.95}$	$44.37_{\pm 2.12}$	$40.74_{\pm 1.47}$	$25.10_{\pm 1.27}$	$50.69_{\pm 1.69}\uparrow\surd$
Avg.	$73.24_{\pm 5.43}$	$71.26_{\pm 5.32}$	$72.74_{\pm 5.65}$	$68.54_{\pm 5.14}$	$69.98_{\pm 5.48}$	$68.21_{\pm 5.63}$	$58.82_{\pm 4.98}$	$74.49_{\pm 4.94}\uparrow\surd$

Note: For each dataset, the highest testing accuracy is marked with $\surd$. For method *Segment+C4.5*, $\uparrow$ and $\downdownarrows$ respectively represent that compared with methods *C4.5*, the performance is improved or degraded.

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, $\downarrow$ and $\upuparrows$ 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 that 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

Datasets	IDE3		C4.5		CART		SIN		SQRT		PSIN		Segment		Segment+C4.5	
	Depth	#Nodes	Depth	#Nodes	Depth	#Nodes	Depth	#Nodes	Depth	#Nodes	Depth	#Nodes	Depth	#Nodes	Depth	#Nodes
Haberman	16.50	129.80	25.10	150.80	16.90	130.80	43.10	157.60	17.10	139.20	15.60	127.20	45.30	**89.60**	**12.20↓**	128.60↓
Ionosphere	9.50	34.60	17.00	34.80	9.80	38.40	**7.90**	**31.60**	11.00	33.60	12.80	44.60	197.40	393.80	17.20⇈	36.80⇈
Cancer	**8.40**	39.20	11.90	50.60	8.50	45.00	76.20	198.60	10.10	**37.60**	10.00	49.00	366.60	732.20	10.80↓	47.40↓
Australian	16.80	185.60	48.40	237.80	**14.80**	**184.00**	69.90	324.60	19.30	191.60	15.00	188.80	328.80	656.60	28.50↓	214.80↓
German	24.20	471.00	61.10	544.00	21.40	452.40	29.60	331.00	24.90	494.40	22.00	442.40	32.50	**64.00**	**17.20↓**	475.20↓
Bupa	12.50	119.00	69.50	185.40	11.30	116.20	23.20	169.20	16.00	120.40	11.20	117.80	22.80	**44.60**	**9.00↓**	159.40↓
Heart	13.70	96.80	45.70	132.20	13.70	97.40	28.30	150.80	16.20	99.80	**13.10**	98.60	38.70	**76.40**	14.80↓	107.80↓
Transfusion	20.20	281.20	40.00	309.80	17.80	270.00	25.20	139.40	22.40	291.40	16.50	266.40	25.40	**49.80**	**10.50↓**	281.60↓
Wdbc	**7.50**	32.40	16.10	49.00	8.40	37.20	48.20	131.40	8.80	**30.60**	9.40	39.00	337.80	674.60	11.40↓	42.40↓
Wpbc	9.70	41.20	24.60	60.00	**8.80**	43.00	42.20	103.00	14.90	**40.20**	11.00	46.80	66.80	132.60	14.50↓	47.20↓
Pima	15.90	200.60	94.60	301.60	**15.00**	199.00	30.60	230.20	19.10	212.20	15.20	208.00	15.00	**29.00**	16.00↓	242.60↓
Plrx	16.30	**57.00**	40.10	81.80	15.60	59.00	27.40	108.20	24.60	62.40	**13.60**	60.00	148.20	295.40	26.80↓	77.00↓
SPECTF	11.60	**42.80**	26.70	64.20	**9.50**	45.40	35.40	101.60	15.80	45.80	10.90	51.00	36.80	72.60	19.80↓	55.60↓
CT	**7.30**	**22.20**	13.00	29.60	8.20	25.00	12.20	49.60	9.30	23.80	10.60	32.00	191.00	381.00	12.20↓	29.00↓
Sonar	8.30	31.40	20.50	57.60	**7.50**	36.40	16.10	61.00	9.20	**30.40**	8.20	36.00	156.20	311.40	7.70↓	37.60↓
Cotton	**6.80**	**30.40**	12.20	34.20	7.90	31.20	11.30	49.00	7.80	35.60	27.20	67.40	172.50	344.00	11.40↓	33.80↓
Ecoli	9.50	**43.80**	12.20	50.80	9.00	45.40	20.20	92.60	**8.20**	64.40	42.30	100.60	65.00	129.00	8.40↓	53.40⇈
Libras	**8.60**	75.80	21.90	95.00	10.90	**75.00**	91.00	190.00	14.50	137.40	94.80	198.20	134.40	267.80	16.80↓	93.40↓
Vowel	**11.20**	**163.40**	47.40	220.80	13.30	163.40	235.60	527.80	13.50	249.20	230.50	534.60	455.30	909.60	22.50↓	185.80↓
Yeast	19.10	382.20	74.20	483.00	20.00	374.40	90.60	586.00	**17.40**	440.20	133.50	639.00	72.30	**143.60**	32.50↓	400.80↓
Avg.	12.68	124.02	36.11	158.65	**12.42**	**123.43**	48.21	186.66	15.01	139.01	36.17	167.37	145.44	289.88	16.01↓	137.51↓

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*

Tree depth	Number of nodes	Testing accuracy
55.66%⇊	13.32%⇊	4.53%⇈

TABLE VII

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

Method	C4.5	CART	SIN	SQRT	PSIN	Segment	Segment +C4.5
IDE3	0.0251†	0.2959	0.0032†	0.0479†	0.0016†	0.0004†	0.0064†
C4.5	——	0.1084	0.1259	0.9405	0.2959	0.0028†	0.0002†
CART	——	——	0.0036†	0.4781	0.0228†	0.0013†	0.0043†
SIN	——	——	——	0.1259	1.0000	0.0028†	0.0001†
SQRT	——	——	——	——	0.0276†	0.0013†	0.0013†
PSIN	——	——	——	——	——	0.0012†	0.0002†
Segment	——	——	——	——	——	——	0.0003†

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 all each other, Friedman test is conducted, where the p value is $2.7166 \times e^{-9}$, 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 $\mathbb{S}$ 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 ξ, where ξ can take values of $2, 3, \ldots, M(p, N)$ and $M(p, N)$ is give in (14), how many distributions of examples in $\mathbb{S}$ will produce ξ 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 $\otimes$.

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: $\oplus, \otimes, \oplus, \otimes, \oplus, \otimes, \ldots$.
2) The first segment is a negative segment: $\otimes, \oplus, \otimes, \oplus, \otimes, \oplus, \ldots$.

Let ξ^+ denotes the number of positive segments and ξ^- denotes the number of negative segments. Then the following holds.

1) $\xi^+ = \xi^- = \xi/2$ when ξ is even.
2) $\xi^+ = (\xi+1)/2$ and $\xi^- = (\xi-1)/2$ when ξ is odd and the first segment is positive.
3) $\xi^+ = (\xi-1)/2$ and $\xi^- = (\xi+1)/2$ when ξ 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 $Group(M, k)$ denotes the number of ways to separate M examples into k groups. Then, the number of distributions for ξ segments is

$$T(\xi) = \begin{cases} 2Group\,(pN, k) \cdot Group\,((1-p)N, k) \\ \text{when } \xi = 2k \\ \\ Group\,(pN, k+1) \cdot Group\,((1-p)N, k) + \\ Group\,(pN, k) \cdot Group\,((1-p)N, k+1) \\ \text{when } \xi = 2k+1, pN > k, (1-p)N > k \\ \\ Group\,(pN, k+1) \cdot Group\,((1-p)N, k) \\ \text{when } \xi = 2k+1, (1-p)N = k \\ \\ Group\,(pN, k) \cdot Group\,((1-p)N, k+1) \\ \text{when } \xi = 2k+1, pN = k. \end{cases} \tag{19}$$

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

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

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

$$Group(M, k) = \mathcal{A}_M^M \mathcal{C}_{M-1}^{k-1} \tag{20}$$

where $\mathcal{A}$ represents the permutation.

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

References

[1] M. F. Amasyah and O. Ersoy, "Cline: A new decision-tree family," *IEEE Trans. Neural Netw.*, vol. 19, no. 2, pp. 356–363, Feb. 2008.

[2] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "A comparison of decision tree ensemble creation techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 173–180, Jan. 2007.

[3] F. Berzal, J. C. Cubero, N. Marın, and D. Sánchez, "Building multi-way decision trees with numerical attributes," *Inf. Sci.*, vol. 165, no. 1, pp. 73–90, 2004.

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Occam's razor," *Inf. Process. Lett.*, vol. 24, no. 6, pp. 377–380, 1987.

[5] M. Boullé, "Khiops: A discretization method of continuous attributes with guaranteed resistance to noise," in *Proc. 3rd Mach. Learn. Data Mining (MLDM)*, Leipzig, Germany, 2003, pp. 50–64.

[6] M. Boullé, "MODL: A Bayes optimal discretization method for continuous attributes," *Mach. Learn.*, vol. 65, no. 1, pp. 131–165, 2006.

[7] L. Breiman, L. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA, USA: Wadsworth International Group, 1984.

[8] W. Buntine and T. Niblett, "A further comparison of splitting rules for decision-tree induction," *Mach. Learn.*, vol. 8, no. 1, pp. 75–85, 1992.

[9] B. Chandra and P. Paul Varghese, "Fuzzy SLIQ decision tree algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1294–1301, Oct. 2008.

[10] B. Chandra and P. Paul Varghese, "Moving towards efficient decision tree construction," *Inf. Sci.*, vol. 179, no. 8, pp. 1059–1069, 2009.

[11] M. Dong and R. Kothari, "Look-ahead based fuzzy decision tree induction," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 3, pp. 461–468, Jun. 2001.

[12] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," in *Proc. 12th Int. Conf. Mach. Learn. (ICML)*, San Mateo, CA, USA: Morgan Kaufmann, 1995, pp. 194–202.

[13] T. Elomaa and J. Rousu, "General and efficient multisplitting of numerical attributes," *Mach. Learn.*, vol. 36, no. 3, pp. 201–244, 1999.

[14] U. M. Fayyad and K. B. Irani, "On the handling of continuous-valued attributes in decision tree generation," *Mach. Learn.*, vol. 8, no. 1, pp. 87–102, 1992.

[15] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proc. 13th Int. Joint Conf. Artif. Intell. (IJCAI)*, Chambery, France, 1993, pp. 1022–1027.

[16] S. R. Gaddam, V. V. Phoha, and K. S. Balagani, "K-means+ID3: A novel method for supervised anomaly detection by cascading K-means clustering and ID3 decision tree learning methods," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 345–354, Mar. 2007.

[17] H. W. Hu, Y. L. Chen, and K. Tang, "A dynamic discretization approach for constructing decision trees with a continuous label," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 11, pp. 1505–1514, Nov. 2009.

[18] Q. Hu *et al.*, "Rank entropy based decision trees for monotonic classification," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 11, pp. 2052–2064, Nov. 2012.

[19] Z. Huang, T. D. Gedeon, and M. Nikravesh, "Pattern trees induction: A new machine learning method," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 4, pp. 958–970, Aug. 2008.

[20] E. Hullermeier and S. Vanderlooy, "Why fuzzy decision trees are good rankers," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 6, pp. 1233–1244, Dec. 2009.

[21] E. B. Hunt, J. Marin, and P. J. Stone, *Experiments in Induction*. New York, NY, USA: Academic Press, 1966.

[22] L. A. Kurgan and K. J. Cios, "CAIM discretization algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 2, pp. 145–153, Feb. 2004.

[23] R. J. Lewis, "An introduction to classification and regression tree (CART) analysis," in *Proc. Annu. Meeting Soc. Acad. Emerg. Med.*, San Francisco, CA, USA, 2000, pp. 1–14.

[24] C. T. Lin *et al.*, "Genetic algorithm-based neural fuzzy decision tree for mixed scheduling in ATM networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 6, pp. 832–845, Dec. 2002.

[25] X. Liu, X. Feng, and W. Pedrycz, “Extraction of fuzzy rules from fuzzy decision trees: An axiomatic fuzzy sets (AFS) approach,” *Data Knowl. Eng.*, vol. 84, pp. 1–25, Mar. 2013.
[26] X. Liu and W. Pedrycz, “The development of fuzzy decision trees in the framework of axiomatic fuzzy set logic,” *Appl. Soft Comput.*, vol. 7, no. 1, pp. 325–342, 2007.
[27] S. Lomax and S. Vadera, “A survey of cost-sensitive decision tree induction algorithms,” *ACM Comput. Surv.*, vol. 45, no. 2, p. 16, 2013.
[28] N. Manwani and P. S. Sastry, “Geometric decision tree,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 181–192, Feb. 2012.
[29] J. K. Martin, “An exact probability metric for decision tree splitting and stopping,” *Mach. Learn.*, vol. 28, no. 2, pp. 257–291, 1997.
[30] M. Mehta, R. Agrawal, and J. Rissanen, “SLIQ: A fast scalable classifier for data mining,” in *Proc. 5th Int. Conf. Ext. Database Technol. (EDBT)*, Avignon, France, 1996, pp. 18–32.
[31] T. M. Mitchell, “The need for biases in learning generalizations,” Dep. Comput. Sci., Lab. Comput. Sci. Res., Rutgers Univ., Tech. Rep. CBM-TR-117, 1980.
[32] W. Pedrycz and Z. A. Sosnowski, “Genetically optimized fuzzy decision trees,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 3, pp. 633–641, Jun. 2005.
[33] J. R. Quinlan, “Induction of decision trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.
[34] J. R. Quinlan, “Improved use of continuous attributes in C4.5,” *J. Artif. Intell. Res.*, vol. 4, pp. 77–90, Mar. 1996.
[35] J. Shafer, R. Agrawal, and M. Mehta, “SPRINT: A scalable parallel classifier for data mining,” in *Proc. 22nd Int. Conf. Very Large Data Bases (VLDB)*, 1996, pp. 544–555.
[36] C. E. Shannon, “A mathematical theory of communication,” *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 5, no. 1, pp. 3–55, 2001.
[37] Y. Sheng, V. V. Phoha, and S. M. Rovnyak, “A parallel decision tree-based method for user authentication based on keystroke patterns,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 4, pp. 826–833, Aug. 2005.
[38] S. Tsang, B. Kao, K. Y. Yip, W. S. Ho, and S. D. Lee, “Decision trees for uncertain data,” *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 1, pp. 64–78, Jan. 2011.
[39] M. van Diepen and P. H. Franses, “Evaluating chi-squared automatic interaction detection,” *Inf. Syst.*, vol. 31, no. 8, pp. 814–831, 2006.
[40] X. Z. Wang, L. C. Dong, and J. H. Yan, “Maximum ambiguity-based sample selection in fuzzy decision tree induction,” *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 8, pp. 1491–1505, Aug. 2012.
[41] X. Z. Wang, D. S. Yeung, and E. C. C. Tsang, “A comparative study on heuristic algorithms for generating fuzzy decision trees,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 2, pp. 215–226, Apr. 2001.
[42] X. Z. Wang, H. Q. Zhao, and S. Wang, “The study of unstable cut-point decision tree generation based-on the partition impurity,” in *Proc. 8th Int. Conf. Mach. Learn. Cybern.*, Baoding, China, 2009, pp. 1891–1897.
[43] L. Wilkinson, “Tree structured data analysis: AID, CHAID and CART,” in *Proc. Sawtooth Softw. Conf.*, Sun Valley, ID, USA, 1992, pp. 431–444.

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.

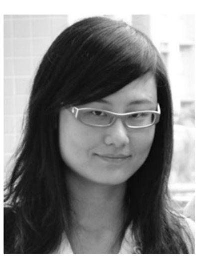
From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada, Mississauga, ON, Canada. He joined Bell Northern Research Canada as a member of Scientific Staff. In 1990, he became a Lecturer with the Department of Electronic Engineering, the City University of Hong Kong, Hong Kong, where he 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.

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 and unsupervised 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*.



Ran Wang (S’09–M’14) received the B.Sc. degree in computer science from the College of Information Science and Technology, Beijing Forestry University, Beijing, China, in 2009, and the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2014.

She is currently a Post-Doctoral Senior Research Associate with the Department of Computer Science, the City University of Hong Kong. She is also an Assistant Researcher with the Shenzhen Key Laboratory for High Performance Data Mining, Shenzhen, China, the Shenzhen Institutes of Advanced Technology, Shenzhen, and the Chinese Academy of Sciences, Beijing, in 2014. Her current research interests include pattern recognition, machine learning, fuzzy sets and fuzzy logic, and their related applications.

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.