A Dynamic Discretization Approach for Constructing Decision Trees with a Continuous Label

Hsiao-Wei Hu, Yen-Liang Chen, and Kwei Tang

Abstract—In traditional decision (classification) tree algorithms, the label is assumed to be a categorical (class) variable. When the label is a continuous variable in the data, two possible approaches based on existing decision tree algorithms can be used to handle the situations. The first uses a data discretization method in the preprocessing stage to convert the continuous label into a class label defined by a finite set of nonoverlapping intervals and then applies a decision tree algorithm. The second simply applies a regression tree algorithm, using the continuous label directly. These approaches have their own drawbacks. We propose an algorithm that dynamically discretizes the continuous label at each node during the tree induction process. Extensive experiments show that the proposed method outperforms the preprocessing approach, the regression tree approach, and several nontree-based algorithms.

Index Terms—Decision trees, data mining, classification.

1 Introduction

Data mining (DM) techniques have been used extensively by many businesses and organizations to retrieve valuable information from large databases and develop effective knowledge-based decision models. Classification is one of the most common tasks in data mining, which involves developing procedures for assigning objects to a predefined set of classes. Main classification methods existing today include decision trees, neural networks, logistic regression, and nearest neighbors.

Decision trees (DTs) have been well recognized as a very powerful and attractive classification tool, mainly because they produce easily interpretable and well-organized results and are, in general, computationally efficient and capable of dealing with noisy data [1], [2], [3]. DT techniques build classification or prediction models based on recursive partitioning of data, which begins with the entire training data; split the data into two or more subsets based on the values of one or more attributes; and then repeatedly split each subset into finer subsets until meeting the stopping criteria. Many successful applications have been developed in, for example, credit scoring [4], fraud detection [5], direct marketing, and customer relationship management [6].

In developing DT algorithms, it is commonly assumed that the label (target variable) is a categorical (class) variable or a Boolean variable, i.e., the label must be in a small, discrete set of known classes. However, in many practical situations, the label is continuous in the data, but the goal is to build a DT and simultaneously develop a class label for the tree. For example, in the insurance industry, customer segments based on the losses or claim amounts in the insured period predicted by relevant risk factors are often created for setting insurance premiums. Furthermore, in supply chain planning, customers are grouped based on their predicted future demands for mapping with products or supply channels [7].

Two possible approaches based on existing decision tree algorithms can be used to handle these situations. For convenience in our discussion, we call these Approach 1 and Approach 2, respectively. Approach 1 uses a data discretization method in the preprocessing stage to convert the continuous label into a class label defined by a finite set of disjoint intervals and then applies a decision tree algorithm. Popular data discretization methods include the equal width method [8], the equal depth method [9], the clustering method [10], the Monothetic Contrast Criterions (MCCs) method [11], and the 3-4-5 partition method [12]. Many researchers have discussed the use of a discretization method in the preprocessing stage of Approach 1 (see [13], [14], [15], [16], [17], [18], [19]). Approach 2 simply applies a regression tree algorithm, such as Classification and Regression Trees (CARTs) [20], using the continuous label directly.

However, these two approaches have their own drawbacks. In applying Approach 1, the discretization is based on the entire training data. It is very likely that the results cannot provide good fits for the data in leaf nodes because the general goal of a tree algorithm is to divide the training data set into more homogeneous subsets in the process of constructing a tree. We provide an illustration in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.24, to support this claim. Using the second approach, the size of a regression tree is usually large, since it takes many nodes to

- H.-W. Hu and Y.-L. Chen are with the Department of Information Management, National Central University, Chung-Li, Taiwan 320, Republic of China.
E-mail: 944403002@cc.ncu.edu.tw, ylchen@mgt.ncu.edu.tw.
- K. Tang is with the Krannert Graduate School of Management, Purdue University, West Lafayette, IN 47907. E-mail: ktang@purdue.edu.

*Manuscript received 12 Feb. 2008; revised 19 June 2008; accepted 18 Dec. 2008; published online 8 Jan. 2009.
Recommended for acceptance by S. Zhang.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2008-02-0092.
Digital Object Identifier no. 10.1109/TKDE.2009.24.*

TABLE 1
A Training Data Set with 15 Customers

Customer ID	Gender	Marital Status	Dependent Status	Expenditure (Label)
1	M	N	N	8
2	M	N	Y	8
3	M	N	Y	12
4	M	Y	Y	101
5	M	Y	Y	105
6	F	Y	Y	145
7	F	Y	Y	162
8	F	Y	Y	162
9	F	Y	Y	172
10	F	Y	Y	172
11	F	N	N	180
12	F	N	N	252
13	F	N	N	252
14	F	N	N	263
15	F	N	Y	263

∗ the range of expenditure is 0-300

achieve small variations in leaf nodes [21]. Furthermore, the prediction results are often not very accurate [22].

We aim to develop an innovative DT algorithm, namely *Continuous Label Classifier* (*CLC*), for performing classification on continuous labels. The proposed algorithm has the following two important features:

1. The algorithm dynamically performs discretization based on the data associated with the node in the process of constructing a tree. As a result, the final output is a DT that has a class assigned to each leaf node.
2. The algorithm can also produce the mean, median, and other statistics for each leaf node as part of its output. In other words, the proposed method is also capable of producing numerical predictions as a regression tree algorithm does.

We conduct extensive numerical experiments to evaluate the proposed algorithm, which includes mainly comparisons with Approaches 1 and 2 and several nontree-based methods. We implement Approach 1 as follows: apply C4.5 as the DT algorithm using the class label generated by discretization based on equal depth, equal width, k-means, or MCC in the preprocessing stage. We use CART as the regression tree algorithm for Approach 2. In addition, we compare the proposed algorithm with several nontree-based methods, including naïve Bayes, K-nearest neighbor (K-NN), support vector machine for classification (LibSVM) [23], simple linear regression (SLR), multiple linear regression (MLR), and support vector machine for regression (SVMreg) [24].

Seven real data sets are included in the experiment, and cross validation is used in our evaluation. We apply the 10-fold cross-validation (CV) scheme to large data sets, and the fourfold CV to small data sets. We use accuracy, precision, mean absolute deviation (MAD), running time, and memory requirements as the comparison criteria. Note that accuracy measures the degree to which we can correctly classify new cases into their corresponding classes, and precision reflects the tightness (widths) of the intervals that the class label is defined. These two criteria are potentially conflicting because when precision is low (wider intervals are used), it is easier to achieve higher precision, and vice versa. Therefore, these two criteria are used simultaneously

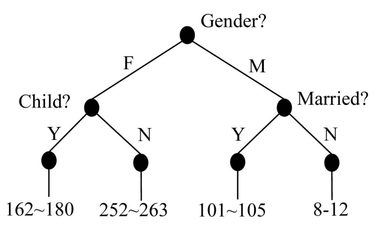


Fig. 1. A DT built from the training data set in Table 1.

in our evaluation. Finally, MAD is used to compare numerical predictions, in particular, by the proposed approach and Approach 2.

The experimental results show that the proposed algorithm can outperform Approach 1 and the nontree-based methods in both accuracy and precision. We also find that the proposed algorithm is capable of producing more accurate numerical predictions than CART.

The remainder of this paper is organized as follows: In Section 2, we formally state the problem. We introduce the proposed algorithm in Section 3. An extensive numerical evaluation is performed in Section 4. We review related work in Section 5 and give a conclusion and discuss several possible extensions in Section 6.

2 Problem Statement

Before giving a formal problem statement, we use a simple example to describe the problem and its requirements and expected results. Consider the training data given in Table 1, which contains 15 cases (records) with Expenditure as the (continuous) label and Gender, Marital Status, and Dependent Status as potential predictors.

Suppose that a DT algorithm produces the tree shown in Fig. 1. As in a typical DT, each internal node corresponds to a decision based an attribute and each branch corresponds to a possible value of the attribute. Using this tree, we can predict the range of Expenditure for a customer with a specific profile. Note that the range of Expenditure at a leaf node is not obtained by a discretization method in the preprocessing stage, but that is determined using the data distribution at the node after the tree has been developed. When a class label is a desired output of the algorithm, we may use the intervals in Expenditure determined by the data at the leaf nodes to define the classes for the label.

The problem is formally defined as follows. Suppose that the data for developing a DT have been randomly divided into the training and test sets. Let D denote the training set and $|D|$ the number of cases in D. The goal is to develop the *CLC* algorithm for constructing a decision tree $T = (V, E)$ using D, where E is the set of branches and V is the set of nodes in the tree. We can use the test set to evaluate the tree obtained from the training set. For each case in the test set, we traverse the DT to reach a leaf node. The case is correctly classified if its value of the continuous label is within the range of the interval at the leaf node. We use the percentage of correctly classified cases as the measure for the DT's accuracy. In addition, we apply the DT to the test set and use the average length of the intervals at the leaf nodes as the measure for the DT's precision. We aim to construct a DT and simultaneously develop a class label with the best performance in precision and accuracy from a data set containing a continuous label. For convenience in presentation, we use "label" as the continuous variable in the original data and "class label" as an output of the algorithm hereafter in the paper.

3 The Proposed Algorithm

The *CLC* algorithm generally follows the standard framework of classical DT methods, such as ID3 [25] and C4.5 [26]. The main steps of the algorithm are outlined as follows:

```
1.  CLC (Training Set $D$);
2.  initialize tree $T$ and put all records of $D$ in the root;
3.  while (some leaf $v_b$ in $T$ is a non-STOP node);
4.        test if no more splits should be done from node $v_b$;
5.        if yes, mark $v_b$ as a STOP node, determine its
          interval and exit;
6.        for each attribute $a_r$ of $v_b$, do
              evaluate the goodness of splitting node $v_b$ with
              attribute $a_r$;
7.        get the best split for it and let $a_{best}$ be the best split
          attribute;
8.        partition the node according to attribute $a_{best}$;
9.  endwhile; and
10. return $T$.
```

Using Steps 4 and 5 of the algorithm, we determine the nodes at which the tree should stop growing and the ranges of the intervals for defining (the classes of) the class label. Steps 6 and 7 are used to select the attribute for further splits from a node. Let $G(v_b, a_r)$ denote the goodness of splitting node $v_b \in V$ with attribute a_r, and *range*(v_b) the interval with the smallest and largest label values in v_b as the endpoints. Using these definitions, we rewrite steps 6 and 7 into the following more detailed steps:

6. For each splitting attribute a_r
   a. For each descendant node v_k of v_b
      Determine a set of nonoverlapping intervals $IV(v_k)$ according to the data at node v_k, where all these intervals are covered by *range*(v_k).
   b. From all $IV(v_k)$, determine $G(v_b, a_r)$, the goodness of splitting v_b with a_r.
7. The attribute with the maximum goodness value is the split attribute at v_b.

We use three sections to explain the following key steps in the algorithms:

1. (Step 6a) Determine a set of nonoverlapping intervals at node $v_k \in V$.
2. (Step 6b) Determine $G(v_b, a_r)$ from all $IV(v_k)$, where v_b is the parent node and v_k includes all descendant nodes generated after splitting.
3. (Steps 4 and 5) Stop tree growing at a node and determine an interval for defining the class at the leaf node for the class label.

3.1 Determining Nonoverlapping Intervals at v_k

We use an example to illustrate Step 6a because the procedure is quite complex. A formal and detailed description is given in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.24. In general, a distance-based partition algorithm is used to develop a set of nonoverlapping intervals $IV(v_k)$, following the distribution of the label at descendant node v_k.

For example, the label in Fig. 2a ranges from 10 to 90. Since there are only few numbers between 30 and 50, we may partition the range into three smaller, nonoverlapping intervals: 10-25, 60-65, and 75-90, as shown in Fig. 2b. These nonoverlapping intervals are found through the following three phases.

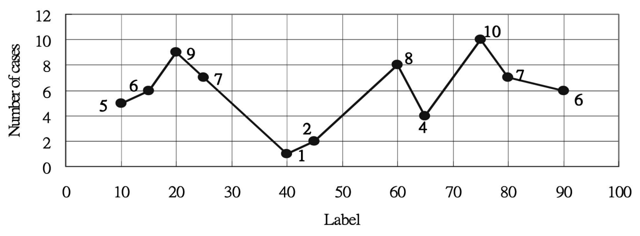
1. First, a statistical method is used to identify "noise instances," which may be outliers or results of errors. These instances can cause overfitting problems and are not likely to be seen in the test set or new data. Hence, in order to successfully find a good set of nonoverlapping intervals, we must first remove noise instances.

   To identify a noise instance, we count the number of cases in its neighboring ranges. If the number is small, we treat it as a noise instance and remove the cases associated with it from the process of forming the nonoverlapping intervals. For example, suppose that the neighboring range of C_i is determined by $C_i \pm 16$. From the data shown in Fig. 2a, the neighboring range for C_5 is from $24(40 - 16 = 24)$ to $56(40 + 16)$ and C_8 from $49(75 - 16)$ to $81(75 + 16)$, etc. The number of cases within the neighboring ranges for all the label values are {C_1:33, C_2:28, C_3:27, C_4:28, C_5:10, C_6:11, C_7:24, C_8:29, C_9:35, C_{10}:27, C_{11}:28}. We treat C_5 and C_6 as noise instances and exclude them and their corresponding data from further consideration because their numbers are small.
2. After removing the noise instances, we divide the remaining data into groups based on the distances between adjacent label values. If the distance is large, we assign them to different groups. Following the same example, we found a large distance between C_4 and C_7 as shown in Fig. 3a (after removing C_5 and C_6). Therefore, we divide the data into two groups, where Group 1 ranges from 10 to 25, as shown in Fig. 3b, and Group 2 ranges from 60 to 90, as shown in Fig. 3c.
3. We consider further partitions for the groups obtained in the last step by identifying possible splitting points in each group. For example, suppose that we identify "C_8" as a splitting point for Group 2, as shown in Fig. 4a. We may divide the group into two new groups: Group 2A ranging from 60 to 65 in Fig 4b and Group 2B ranging from 75 to 90 in Fig. 4c.

As mentioned, we give a brief and informal introduction to the development of nonoverlapping intervals at an internal node. A formal presentation is given in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2009.24.

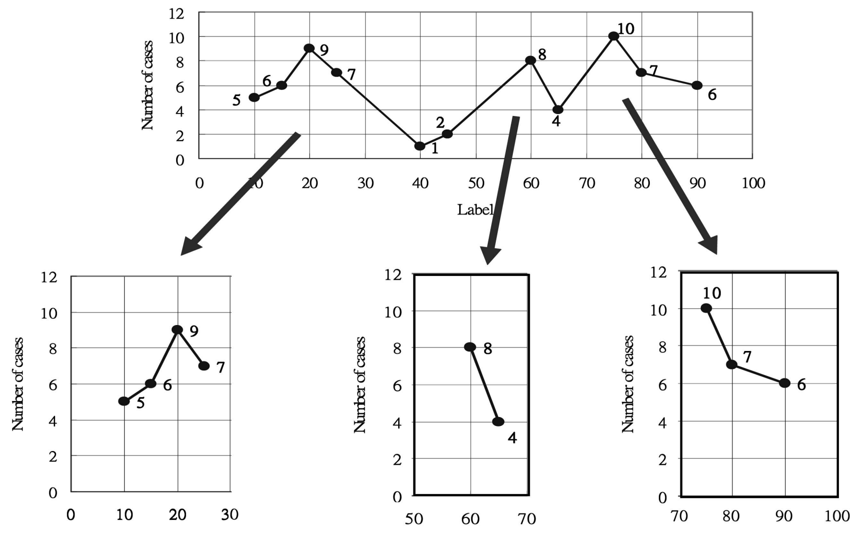
3.2 Computing the Goodness Value

Let $RL_{(i,j)}$ be the interval covering C_i to C_j and v_k be a descendant node of v_b. Suppose that after applying Step 6a



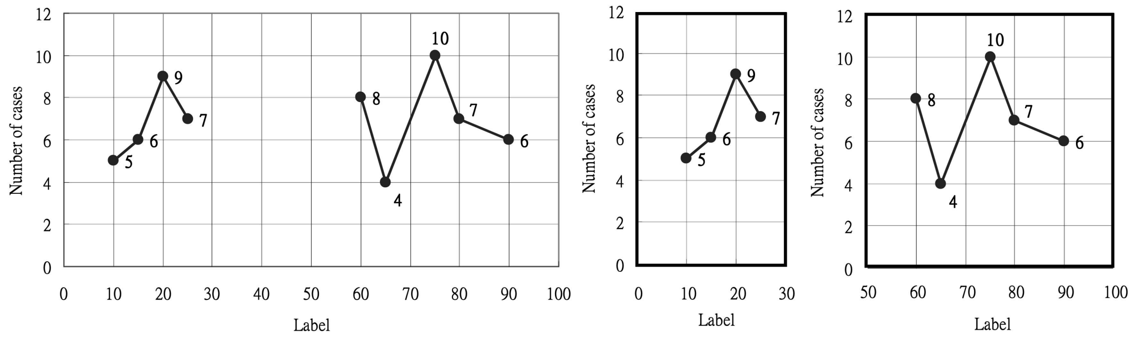
Label		Number of cases
C_1	10	5
C_2	15	6
C_3	20	9
C_4	25	7
C_5	40	1
C_6	45	2
C_7	60	8
C_8	65	4
C_9	75	10
C_{10}	80	7
C_{11}	90	6

(a)



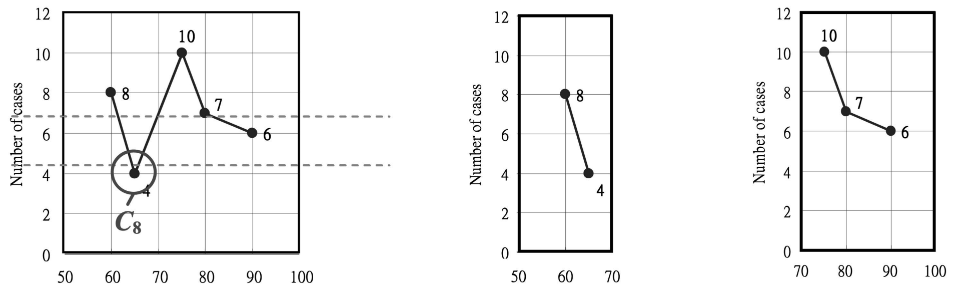
(b)

Fig. 2. Step 6a generates $IV(v_k)$ according to the data in v_k. (a) Data distribution at a node and (b) Three intervals.



(a) (b) (c)

Fig. 3. The second phase: Find nonoverlapping intervals based on the distances between adjacent label values. (a) Data after removing C_4 and C_5. (b) Group-1. (c) Group-2.



(a) (b) (c)

Fig. 4. The third phase: For each interval, further partition it into smaller intervals if necessary. (a) Group-2. (b) Group-2A. (c) Group-2B.

TABLE 2
Description of Data Sets

ID	Name	Attributes		Size
		Nominal	Continuous	
DS-1	Hepatitis Domain (Bil)	14	6	155
DS-2	Sick	23	7	3772
DS-3	Breast Tumor	8	2	286
DS-4	Low BWT	7	3	189
DS-5	Sensory	11	1	576
DS-6	Servo	4	1	167
DS-7	Mammographic	4	1	961

to partition the data in v_k, we have obtained a set of nonoverlapping intervals $IV(v_k) = \{RL_{u_1,u_2}, RL_{u_3,u_4}, \ldots, RL_{u_{2s-1},u_{2s}}\}$. Let $|IV(v_k)|$ denote the number of intervals in $IV(v_k)$ and $|RL_{(i,j)}| = C_j - C_i$. Let $D(v_k)$ be the data associated with node v_k and $|D(v_k)|$ be the number of cases at node v_k.

Our goal is to compute $G(v_b, a_r)$, the goodness of splitting v_b with a_r, from all $IV(v_k)$, which requires that we first obtain the goodness value for each descendant node v_k, denoted by $G(v_k)$ and then evaluate the following formula:

$$G(v_b, a_r) = \sum_{\text{for all } k} \frac{|D(v_k)|}{|D(v_b)|} G(v_k).$$

The goodness value of node $v_k, G(v_k)$ is computed from $IV(v_k)$. A larger $G(v_k)$ suggests that the intervals in $IV(v_k)$ form a better partition for the class label. In this paper, we define $G(v_k)$ based on three factors.

The first one is the standard deviation, which is used to measure the degree of concentration of the data at a node. A small standard deviation is desired because the data are more concentrated. We use $Dev(v_k)$ to denote the standard deviation at node v_k.

The second factor is the number of intervals ($|IV(v_k)|$). A smaller number of intervals suggest that the data at the node are more concentrated in a smaller number of ranges. Therefore, we have a higher probability of correctly classifying future data into the correct interval or a higher accuracy level.

The third factor is the length of intervals. The intervals should be as tight as possible, since shorter intervals imply higher precision. Therefore, we define $CR(v_k)$, the cover range of node v_k, as

$$CR(v_k) = 1 \Big/ \log_2\left(\frac{|range(v_b)|}{\sum_{\text{for all (i,j)}} |RL_{(i,j)}|} + 1\right),$$

where v_b is the parent node of v_k. We will use the example in Figs. 2a and 2b to show the calculation of $CR(v_k)$. From the information given by the figures, we obtain

$$IV(v_k) = \{RL_{(1,4)},\ RL_{(7,8)},\ RL_{(9,11)}\},$$
$$|RL_{(1,4)}| = 25 - 10 = 15,\ |RL_{(7,8)}| = 65 - 60 = 5, \text{ and}$$
$$|RL_{(9,11)}| = 90 - 75 = 15.$$

Assuming that $|range(v_b)| = 100 - 0 = 100, CR(v_k)$ is given as

$$CR(v_k) = 1 \Big/ \log_2\left(\frac{100}{15+5+15} + 1\right) = 1 \big/ 1.95 = 0.5128$$

Since the goodness of a node is better if it has lower standard deviation and fewer and tighter intervals, we define the goodness value of node v_k as

$$G(v_k) = \frac{1}{Dev(v_k)} \times \frac{1}{\log_2(|IV(v_k)| + 1)} \times \frac{1}{CR(v_k)}.$$

Finally, we select the attribute with the largest goodness value as the splitting attribute to complete Step 6b.

3.3 Stopping Tree Growing

Let R be the range of the label for the entire training set D, *majority* be the interval in $IV(v_b)$ containing the "majority" of the data at a node, *percent*(v_b, *majority*) be the percentage of the data at v_b whose label values are in *majority,* and *length*(*majority*) be the length of *majority*. If one of the following conditions is met, we stop splitting at node v_b. Otherwise, we attempt to produce further splits from the node.

1. All the attributes have been used in the path from the root to v_b.
2. *percent*(v_b, *majority*) $> \zeta_{\text{DR}}$ && *length*(*majority*)/$R < \zeta_{\text{LN}}$, where ζ_{DR} and ζ_{LN} are two given thresholds.
3. $|D(v_b)|/|D| < \zeta_{\text{D}}$, where ζ_{D} is a given threshold.
4. The goodness value cannot be improved by additional splits.

When we stop growing at v_b, we assign the interval covering most of the data (i.e., the majority interval) as the range for the class associated with the node. Before we finalize a majority interval, we try to merge it with adjacent intervals if it can cover more cases without losing too much precision. After we have determined that no more merges are possible, the majority intervals are used to define the classes associated with the leaf nodes, and thus, the class label for the tree.

4 Performance Evaluation

In Table 2, we list the data sets used in the experiments, which were downloaded from the UCI Machine Learning

TABLE 3
Parameters Set in Experiment

ID	Range of Label	ζ_{LN} (%)	ζ_{DR} (%)	ζ_{D} (%)	Precision (%)	Accuracy (%)
DS-1	0~5	8.00%	90%	1%	95.89%±0.60%	79.17%±13.85%
DS-2	0~11	4.55%	90%	1%	80.91%±0.24%	87.07%±2.07%
DS-3	0~50	12.00%	90%	2%	91.03%±1.97%	62.54%±10.71%
DS-4	10~50	12.50%	90%	2%	91.20%±0.99%	85.68%±5.37%
DS-5	12~18	16.67%	90%	1%	91.04%±2.59%	96.70%±2.14%
DS-6	0~8	2.50%	90%	3%	94.19%±4.98%	78.50%±9.38%
DS-7	17~97	6.25%	90%	1%	81.90%±1.61%	73.06%±3.37%

Repository [27]. Four data sets—DS-3, DS-5, DS-6, and DS-7—were adopted without changes because they have continuous labels. The remaining three data sets—DS-1, DS-2, and DS-4—have class labels. We use a continuous variable in the same data set as the label. We implemented the proposed algorithm in the Java language under the Microsoft XP operating system on a PC with a Pentium 4, 2 GHz processor and 1,024 MB main memory.

As discussed in Section 3.3, three thresholds, $\zeta_{\rm LN}$, $\zeta_{\rm DR}$, and $\zeta_{\rm D}$, are used in the *CLC* algorithm to control the tree growth. In the algorithm, one stopping condition is based on whether the length of the *majority* interval is tight enough (controlled by $\zeta_{\rm LN}$) and whether the amount of data in the majority interval is sufficient (controlled by $\zeta_{\rm DR}$). Another stopping condition is based on whether the amount of data in the node is small enough (controlled by $\zeta_{\rm D}$). In our pretest experiment, we observed the results of the *CLC* algorithm by varying the values of these three threshold values. After the pretest, we selected the best combination of the values for each data set and used them as the predetermined thresholds in constructing the DT. Table 3 contains these values and the precision and accuracy evaluated from cross validation of the results obtained by the *CLC* algorithm.

To evaluate the performance of the *CLC* algorithm, we performed the X-fold cross validation for each data set according to the following steps:

1. Divide the entire data set into X equal-sized subsets using stratified sampling.
2. For each subset, build a decision tree based on all data not belonging to the subset, and compute the accuracy and the precision using the data in the subset.
3. Obtain the average of the X accuracy rates as the accuracy of this cross-validation run.

For small-size data sets, the test results may be biased if we set X too large. Therefore, we set $X = 4$ for small data sets including DS-1, DS-3, DS-4, DS-5, and DS-6. For large data sets, DS-2 and DS-7, we set $X = 10$. In each of the X cross-validation test runs, the same parameter settings, including the seed values of random numbers for partitioning the data, were used for all the algorithms.

Three experiments were performed. The first is to compare *CLC* and C4.5 with four popular data discretization methods. The second is to compare *CLC* and CART. The third includes two comparisons: one in the running times of *CLC*, C4.5, and CART, and the other in the accuracies of *CLC* and several nontree-based methods.

4.1 First Experiment: Comparing CLC and Approach 1

In this experiment, we use accuracy and precision as our evaluation criteria, where precision is defined as

$$1 - \frac{\text{the averge interval length of all test data}}{\text{the total range of class labels in the data set}}.$$

In other words, a higher precision implies tighter prediction intervals.

We select four unsupervised discretization methods proposed by Dougherty et al. [12] to convert a label into a class label defined by k intervals, using the equal width, equal depth, k-means clustering, and MCC methods. These four discretization methods are discussed later in Section 5. After preprocessing, we use C4.5 with the class label produced by discretization to construct a decision tree. For convenience, we use EW-T to denote the approach of using C4.5 with the equal width discretization to build decision trees. Similarly, ED-T, KM-T, and MCC-T refer to the approaches of using C4.5 with the equal depth, k-means, and MCC preprocessing methods, respectively.

We first compare *CLC* with EW-T. For a fair comparison, we set the width of the bins in equal width discretization equal to the average interval length of the leaf nodes when we use a tree produced by *CLC* to classify the test data. This allows us to compare the accuracies of the two methods under the same precision level. The results of the comparison are given in Table 4, which clearly indicates that the average accuracy of the trees built by *CLC* is significantly

TABLE 4
A Comparison between *CLC* and EW-T

ID	***CLC***	**EW-T**
	Accuracy	Accuracy
DS-1	79.17%±13.85%	26.88%±24.07%
DS-2	87.07%±2.07%	50.92%±14.44%
DS-3	62.54%±10.71%	35.02%±37.19%
DS-4	85.68%±5.37%	37.57%±35.48%
DS-5	96.70%±2.14%	44.27%±36.93%
DS-6	78.50%±9.38%	48.53%±24.39%
DS-7	73.06%±3.37%	40.69%±36.60%

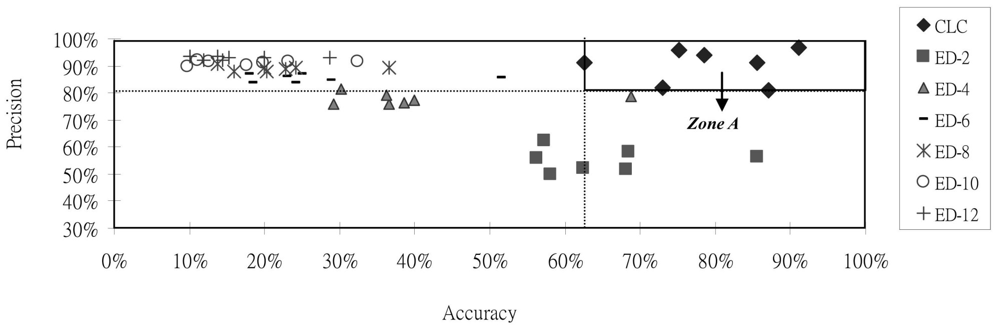


Fig. 5. A comparison between *CLC* and ED-T.

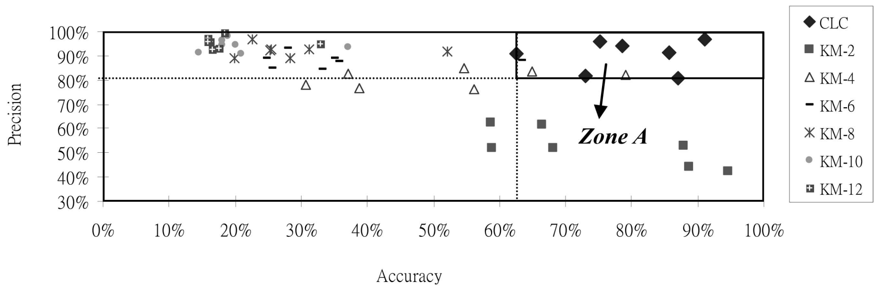


Fig. 6. The comparison between *CLC* and KM-T.

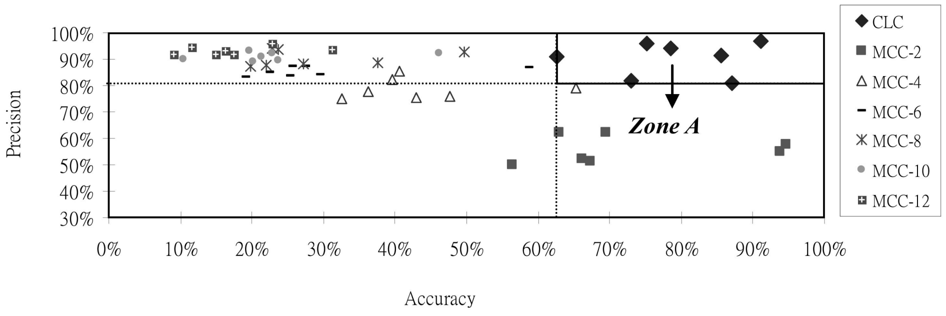


Fig. 7. A comparison between *CLC* and MCC-T.

higher than that of EW-T. Furthermore, the results also suggest that *CLC* produces more reliable trees because the variation in accuracy is consistently much smaller.

We further vary the number of bins k from 2 to 12 in comparing *CLC* with ED-T, KM-T, and MCC-T. Fig. 5 is a scatter plot used to compare *CLC* and ED-T in both accuracy and precision. Each point in the figure represents the accuracy and precision combination when a method is applied to a data set. All of the points associated with *CLC* are in the upper right part of the figure, implying that the *CLC* algorithm produces results with both higher accuracy and precision. We let Zone A be the minimum rectangular area that covers all *CLC*'s points. The figure clearly shows that all the points associated with ED-T fall in the left side, the lower side, or the lower left of Zone A, suggesting that they have either lower accuracy, lower precision, or both. We also find that using more bins in ED-T results in lower accuracy but higher precision.

Fig. 6 is a scatter plot for comparing *CLC* and KM-T. As defined before, Zone A is the minimum area covering the results associated with *CLC*. Although three points associated with KM-T fall into Zone A, the overall conclusion is still that *CLC* outperforms KM-T.

Fig. 7 is used to compare *CLC* and MCC-T. Since none of MCC-T-related points fall in Zone A, we conclude that *CLC* outperforms MCC-T.

4.2 Second Experiment: CLC and Regression Trees

In this section, we compare *CLC* with the popular regression tree algorithm, CART. To be consistent, we use the average as the numerical predicted value at each leaf node for both algorithms. Two criteria are used in the comparison: 1) MAD and 2) w-STDEV: weighted standard deviation.

We define MAD as follows:

$$MAD = \frac{1}{N}\left(\sum_{i=1}^{N}\frac{1}{R}|x_i - P(x_i)|\right),$$

where N is the total number of test cases, R is the total range of the test data, and x_i and $P(x_i)$ are the actual and predicted values of the ith test case, respectively. We further define w-STDEV as follows:

$$w - STDEV = \sum_{\text{for all leaf node } v_i} \frac{|D(v_i)|}{N} \times STDEV(v_i),$$

TABLE 5
A Comparison between *CLC* and *CART*

		DS-1	DS-2	DS-3	DS-4	DS-5	DS-6	DS-7
MAD (%)	***CLC***	2.81	4.68	14.92	8.05	6.53	3.69	6.19
	RT	3.93	4.83	16.83	10.47	9.74	7.93	12.89
w-STDEV	***CLC***	0.2098	0.4610	5.4916	3.1315	0.5043	0.2554	7.8314
	RT	0.5244	0.7517	10.4786	5.1026	0.7427	0.6826	12.6959

where $|D(v_i)|$ is the number of cases at leaf node v_i and $STDEV(v_i)$ is the standard deviation of the data at the leaf node v_i.

The results reported in Table 5 indicate that *CLC* produces more accurate and reliable numerical predictions than CART because its MAD and w-STDEV values are consistently smaller.

4.3 Third Experiment: Supplementary Comparisons

First, we compare the running times of *CLC* and two decision tree algorithms, C4.5 and CART. In the comparison, we vary the data size to observe the performances of these algorithms as the data size increases. Two data sets, DS-6 and DS-7, are selected for the comparison. We duplicate these two data sets repeatedly until reaching the intended sizes. The running times and memory requirements are reported in Figs. 8 and 9, respectively. As expected, the time increases as the data size increases, and *CLC* consumes more time than the other two algorithms, mainly because it performs discretization at each node. However, its running time still stays within a very reasonable range when the data size is fairly large. It is worth noting that, in constructing a decision tree or a classification algorithm, in general, accuracy is usually much more important than the running time for developing a tree. This is because a tree could be used repeatedly before it needs updates or revisions.

In Fig. 9, we show the memory requirements for the three algorithms. As expected, memory use increases as data size increases. The results also indicate that all three algorithms are efficient because the increase in memory use is not as fast as that in data size.

Next, we compare *CLC* with three nontree-based classification algorithms—Naïve Bayes, K-Nearest Neighbor (K-NN), and Support Vector Machine—for classification (LibSVM) [23], and another three nontree-based prediction algorithms—SLR, MLR, and Support Vector Machine—for regression (SVMreg) [24]. We follow the procedure used in Section 4.1 to convert the labels of the data sets into class labels. Consequently, the comparison is focused on accuracy while precision remains the same for all the algorithms of interest. The results of the comparison are listed in Tables 6 and 7, respectively, for classification and prediction algorithms. We find the results in Table 6 are very similar to those in Table 4, suggesting that *CLC* outperforms the three nontree-based algorithms. Furthermore, *CLC* has a smaller variation in accuracy, which implies that the performance of *CLC* is more consistent. Table 7 shows that *CLC* has the smallest MAD values in all the data sets except one situation, where SVM(SVNreg) has a slightly small MAD for DS-2. This confirms that *CLC* performs well against the three nontree-based prediction algorithms.

5 Related Work

Many classification methods have been developed in the literature, including DT, Bayesian, neural networks, k-nearest neighbors, case-based reasoning, genetic algorithm, rough sets, and fuzzy sets [28], [29]. DT is probably the most popular and widely used because it is computationally efficient and its output can easily be understood and interpreted.

There are two types of DTs according to the type of the label: regression trees and classification trees [20]. The goal of a regression tree is to predict the values of a continuous label. It is known that, compared to other techniques, a regression tree has the disadvantages of generally requiring more data, being more sensitive to the data, and giving less accurate predictions [22]. Main existing regression tree algorithms are CART [20], CHAID [30], and QUEST [31]. Classification trees, the second type of DTs, attempt to

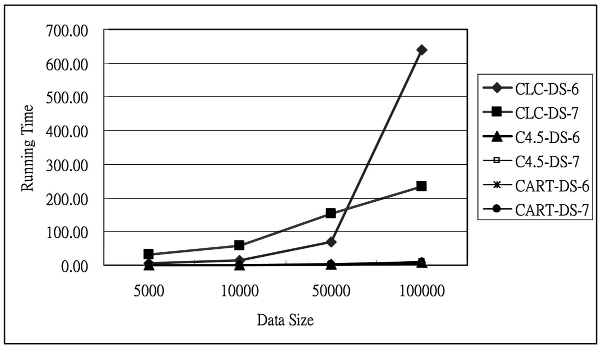


Fig. 8. The effects of the data size on running time.

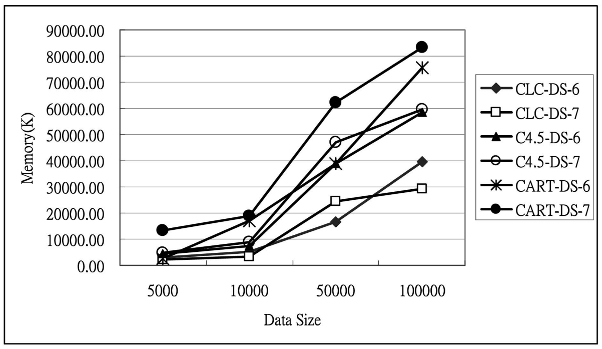


Fig. 9. The effects of the data size on memory requirement.

TABLE 6
A Comparison in Accuracy between *CLC* and Three Nontree-Based Classifiers

Accuracy(%)	DS-1	DS-2	DS-3	DS-4	DS-5	DS-6	DS-7
CLC	79.17±13.85	87.07±2.07	62.54±10.71	85.68±5.37	96.70±2.14	78.50±9.38	73.06±3.37
Naive Bayes	36.56±22.20	50.74±14.32	33.21±35.70	34.92±35.06	39.93±32.88	37.13±23.14	37.04±39.00
K-NN	33.33±23.63	50.27±14.62	30.32±43.28	35.45±39.89	39.06±43.52	29.94±30.53	38.19±38.33
SVM(LibSVM)	34.09±28.06	50.95±17.72	37.55±44.72	41.80±43.17	45.83±41.64	41.32±29.45	41.00±47.61

TABLE 7
A Comparison in MAD between *CLC* and Three Nontree-Based Prediction Methods

MAD(%)	DS-1	DS-2	DS-3	DS-4	DS-5	DS-6	DS-7
CLC	2.81	4.68	14.92	8.05	6.53	3.69	6.19
SLR	10.97	4.72	16.67	10.80	10.93	11.20	13.11
MLR	8.81	4.91	15.65	10.22	10.73	11.12	12.93
SVM(SVMreg)	7.97	4.62	15.24	9.92	10.62	9.06	12.91

develop rules for class labels. Quinlan developed the popular ID3 and C4.5 algorithms [26], [32].

Continuous data discretization, another area related to this paper, has recently received much research attention. The simplest discretization method, *equal width*, merely divides the range of observed values into a prespecified number of equal, nonoverlapped intervals. As Catlett [33] pointed out, this method is vulnerable to outliers that may drastically skew the results. Another simple method, *equal depth*, divides the range of the data into a prespecified number of intervals, which contains roughly the same number of cases. Another well-known method, *MCCs* [11], divides the range of the data into k intervals by finding the partition boundaries that produce the greatest contrast according to a given contrast function. The clustering method or the entropy method can be used to perform the same task [12]. These popular data discretization methods have been commonly used in the preprocessing phase when constructing DTs with continuous labels and have also been applied in various areas, such as data stream [17], [18], software engineering [19], Web application [14], detection [15], [16] and others [13].

As discussed in Section 1, the weakness of the preprocessing approach using a discretization method is that it is inherently a static approach, which essentially ignores the likelihood that the data distributions could be dramatically different at different nodes. This motivates our approach, which dynamically discretizes data at each node in the tree induction process. As shown in the last section, the proposed algorithm outperforms the preprocessing approach, the regression tree approach, and several non-tree-based algorithms.

6 Conclusion

Traditional decision tree induction algorithms were developed under the assumption that the label is a categorical variable. When the label is a continuous variable, two major approaches have been commonly used. The first uses a preprocessing stage to discretize the continuous label into a class label before applying a traditional decision tree algorithm. The second builds a regression tree from the data, directly using the continuous label. Basically, the algorithm proposed in this paper was motivated by observing the weakness of the first approach—its discretization is based on the entire training data rather than the local data in each node. Therefore, we propose a decision tree algorithm that allows the data in each node to be discretized dynamically during the tree induction process.

Extensive numerical experiments have been performed to evaluate the proposed algorithm. Seven real data sets are included in the experiments. In the first experiment, we compare our algorithm and C4.5 with the traditional preprocessing approach including the equal depth, equal width, k-means, and MCC methods. The results indicate that our algorithm performs well in both accuracy and precision. In the second experiment, we compare our algorithm with the popular regression tree algorithm, CART. The results of the experiment show that our algorithm outperforms CART. In the third experiment, we provide two supplementary experiments, including comparing the running times of our algorithm and C4.5 and CART, and comparing the performances of our algorithm and several nontree-based classifiers and prediction algorithms. The results also confirm the efficiency and accuracy of the proposed algorithm.

This work can be extended in several ways. We may consider ordinal data, which have mixed characteristics of categorical and numerical data. Therefore, it would be interesting to investigate how to build a DT with ordinal class labels or intervals of ordinal class labels. Furthermore, in practice, we may need to simultaneously predict multiple numerical labels, such as the stock price, profit, and revenue of a company. Accordingly, it is worth studying how to build DTs that can classify data with multiple, continuous labels. Finally, constraints may exist on the selection of intervals to be used to define a class label. For example, if the income is the label of the data, we may require that the boundaries of all intervals be rounded to the nearest thousand. We may add this constraint or even a set of user-specified intervals in the problem considered in this paper.

References

[1] J.R. Cano, F. Herrera, and M. Lozano, "Evolutionary Stratified Training Set Selection for Extracting Classification Rules with Trade Off Precision-Interpretability," *Data & Knowledge Eng.*, vol. 60, no. 1, pp. 90-108, 2007.

[2] G. Jagannathan and R.N. Wright, "Privacy-Preserving Imputation of Missing Data," *Data & Knowledge Eng.*, vol. 65, no. 1, pp. 40-56, 2008.

[3] X.B. Li, J. Sweigart, J. Teng, J. Donohue, and L. Thombs, "A Dynamic Programming Based Pruning Method for Decision Trees," *INFORMS J. Computing,* vol. 13, pp. 332-344, 2001.

[4] S. Piramuthu, "Feature Selection for Financial Credit-Risk Evaluation Decisions," *INFORMS J. Computing,* vol. 11, pp. 258-266, 1999.

[5] F. Bonchi, F. Giannotti, G. Mainetto, and D. Pedreschi, "A Classification-Based Methodology for Planning Audit Strategies in Fraud Detection," *Proc. Fifth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 175-184, 1999.

[6] M.J.A. Berry and G.S. Linoff, *Mastering Data Mining: The Art and Science of Customer Relationship Management.* Wiley, 1999.

[7] J.B. Ayers, *Handbook of Supply Chain Management,* second ed., pp. 426-427. Auerbach Publication, 2006.

[8] M.R. Chmielewski and J.W. Grzymala-Busse, "Global Discretization of Continuous Attributes as Preprocessing for Machine Learning," *Proc. Third Int'l Workshop Rough Sets and Soft Computing,* pp. 294-301, 1994.

[9] J. Cerquides and R.L. Mantaras, "Proposal and Empirical Comparison of a Parallelizable Distance-Based Discretization Method," *Proc. Third Int'l Conf. Knowledge Discovery and Data Mining,* pp. 139-142, 1997.

[10] M.R. Chmielewski and J.W. Grzymala-Busse, "Global Discretization of Continuous Attributes as Preprocessing for Machine Learning," *Int'l J. Approximate Reasoning,* vol. 15, pp. 319-331, 1996.

[11] T. Van de Merckt, "Decision Tree in Numerical Attribute Space," *Proc. 13th Int'l Joint Conf. Artificial Intelligence,* pp. 1016-1021, 1993.

[12] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and Unsupervised Discretization of Continuous Features," *Proc. Int'l Conf. Machine Learning,* pp. 194-202, 1995.

[13] J. Bogaert, R. Ceulemans, and E.D. Salvador-Van, "Decision Tree Algorithm for Detection of Spatial Processes in Landscape Transformation," *Environmental Management,* vol. 33, no. 1, pp. 62-73, 2004.

[14] L. Borzemski, "The Use of Data Mining to Predict Web Performance," *Cybernetics and Systems,* vol. 37, no. 6, pp. 587-608, 2006.

[15] W. Desheng, "Detecting Information Technology Impact on Firm Performance Using DEA and Decision Tree," *Int'l J. Information Technology and Management,* vol. 5, nos. 2/3, pp. 162-174, 2006.

[16] S.A. 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. Knowledge and Data Eng.,* vol. 19, no. 3, pp. 345-354, Mar. 2007.

[17] R. Jin and G. Agrawal, "Efficient Decision Tree Construction on Streaming Data," *Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 571-576, 2003.

[18] M. Last, "Online Classification of Nonstationary Data Streams," *Intelligent Data Analysis,* vol. 6, no. 2, pp. 129-147, 2002.

[19] M. Last, M. Friedman, and A. Kandel, "Using Data Mining for Automated Software Testing," *Int'l J. Software Eng. and Knowledge Eng.,* vol. 14, no. 4, pp. 369-393, 2004.

[20] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees.* Chapman & Hall, 1993.

[21] S. Kramer, "Structural Regression Trees," *Proc. 13th Nat'l Conf. Artificial Intelligence,* pp. 812-819, 1996.

[22] J. Yang and J. Stenzel, "Short-Term Load Forecasting with Increment Regression Tree," *Electric Power Systems Research,* vol. 76, nos. 9/10, pp. 880-888, June 2006.

[23] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, *A Practical Guide to Support Vector Classification,* http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, 2003.

[24] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, and K.R.K. Murthy, "Improvements to the SMO Algorithm for SVM Regression," *IEEE Trans. Neural Networks,* vol. 11, no. 5, pp. 1188-1193, Sept. 2000.

[25] J.R. Quinlan, "Introduction of Decision Trees," *Machine Learning,* vol. 1, pp. 81-106, 1986.

[26] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[27] P.M. Murphy and D.W. Aha, "UCI Repository of Machine Learning Database," http://archive.ics.uci.edu/ml/, 1994.

[28] Y.L. Chen, C.L. Hsu, and S.C. Chou, "Constructing a Multi-Valued and Multi-Labeled Decision Tree," *Expert Systems with Applications,* vol. 25, pp. 199-209, 2003.

[29] J. Han and M. Kamber, *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2001.

[30] G.V. Kass, "An Exploratory Technique for Investigating Large Quantities of Categorical Data," *Applied Statistics,* vol. 29, pp. 119-127, 1980.

[31] W.Y. Loh and Y.S. Shih, "Split Selection Methods for Classification Trees," *Statistica Sinica,* vol. 7, pp. 815-840, 1997.

[32] J.R. Quinlan, "Improved Use of Continuous Attributes in C4.5," *Artificial Intelligence,* vol. 4, pp. 77-90, 1996.

[33] J. Catlett, "Megainduction: Machine Learning on Very Large Databases," PhD thesis, Univ. of Sydney, 1991.

Hsiao-Wei Hu received the MS degree in information management from the National Central University of Taiwan. She is currently working toward the PhD degree in the Department of Information Management, National Central University, Taiwan. Her research interests include data mining, information retrieval, and EC technologies. Currently, she is focusing on classification techniques.

Yen-Liang Chen received the PhD degree in computer science from the National Tsing Hua University, Hsinchu, Taiwan. He is a professor of information management at the National Central University of Taiwan. His current research interests include data modeling, data mining, data warehousing, and operations research. He has published papers in Decision Support Systems, Operations Research, *IEEE Transactions on Software Engineering*, Computers & OR, *European Journal of Operational Research*, Information and Management, Information Processing Letters, Information Systems, *Journal of Operational Research Society*, and Transportation Research.

Kwei Tang received the BS degree from the National Chiao Tung University, Taiwan, the MS degree from Bowling Green State University, and the PhD degree in management science from Purdue University. He is the Allison and Nancy Schleicher chair of Management and associate dean for Programs and Student Services at the Krannert School of Management, Purdue University. He teaches data mining, quality management, applied statistics, and research methods. His current research interests are data mining, online process control, integration of quality functions in a manufacturing organization, e-commerce, and supply chain management.