Volume 177, issue 5, 1 March 2007 ISSN 0020-0255

INFORMATION SCIENCES
Informatics and Computer Science
Intelligent Systems
Applications
AN INTERNATIONAL JOURNAL

Including special issue

The 3rd International Workshop
on Computational Intelligence in Economics and Finance (CIEF '2003)

*Guest Editor*

Shu-Heng Chen

Available online at
ScienceDirect
www.sciencedirect.com

Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 177 (2007) 1276–1289
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

# Relaxing instance boundaries for the search of splitting points of numerical attributes in classification trees

Ester Yen [a,*], I-Wen Mike Chu [b]

[a] *Imperial College, London, UK*
[b] *NASA Goddard Flight Space Center, Greenbelt, Maryland, USA*

Received 21 July 2004; received in revised form 9 August 2006; accepted 9 August 2006

**Abstract**

We propose a simple heuristic partition method (HPM) of classification tree to improve efficiency in the search for splitting points of numerical attributes. The proposal is motivated by the idea that the selection process of candidates in the splitting point selection can be made more flexible as to achieve a faster computation while retaining classification accuracy. We compare the performance of the HPM against Fayyad's method, as the latter is the improved version of the standard C4.5 algorithm on the search of splitting points. We demonstrate that HPM is more efficient, in some cases by as much as 50%, while producing essentially the same classification for six different data sets. Our result supports the relaxation of instance boundaries (RIB) as a valid approach that can be explored to achieve more efficient computations.


*Keywords:* Classification tree; Decision trees; Heuristic partition method; Numerical attributes; Splitting point; Relaxation of instance boundaries; Data mining

## 1. Introduction

Data mining is one of the most actively researched areas in information science with important real world applications (e.g. [15,17–19]). The task of data mining can be defined as the process of selection and exploration of large databases to discover models and patterns. For example, an enterprise may want to uncover useful information embedded in its available large databases of consumer information to facilitate its business decision-making. One of such tasks in data mining is *classification*, the aim of which is to build a model or a *classifier* for a given set of pre-classified examples, to classify categories of new events. Among the numerous approaches are regressions, Bayesian models, neural nets and classification trees just to name a few. In particular, a classification tree, or decision tree, is a useful choice when the tasks are to classify or predict outcomes and to generate easy-to-interpret rules.

* Corresponding author.
*E-mail address:* ester_yen@yahoo.com (E. Yen).

0020-0255/$ - see front matter 
doi:10.1016/j.ins.2006.08.014

A classification tree is typically constructed *top-down*, and a pre-classified training data set is split at each node by some criterion, utilizing an attribute (feature), into partitions. The building process is known as "recursive partitioning" as the splitting is iterated until all useful splits have been exhausted. The goodness of the available attributes is evaluated by an impurity measure to determine the most useful attribute for the split. Breiman et al. proposed the *Gini* criterion as the impurity function in *classification and regression trees* (CART) [3], and, alternatively, Quinlan proposed the ID3 classification tree algorithm in 1986 [28] and later the C4.5 algorithm in 1993 [29] using entropy (specifically as *Information Gain* and *Gain Ratio*).

Although C4.5 has been highly successful, it performs poorly on the search for splitting points of numerical attributes [13,14]. Numerous subsequent partition methods have been proposed (e.g. [30,35]) to improve the search efficiency, but many of them exhibit a limited success. Fayyad and Irani made the significant improvement over C4.5, in both efficiency and classification accuracy, by modifying the candidate selection for splitting points utilizing the important finding that only boundary points need to be considered as candidates [13]. The method, referred to as Fayyad's method, currently stands as the standard benchmark in the search for splitting points of numerical attributes. There are also numerous studies tackling different technical aspects, such as selection bias (e.g. [10,33,34]). In medical and financial applications, it is important that a classifier should give the estimate degrees of all potential classes. The classifiers should avoid classifying an instance into only one class.

In this work, we construct a HPM to further improve over both the C4.5 algorithm and Fayyad's method. The proposal is motivated by the idea that a more relaxed grouping scheme in the candidate selection of splitting points can reduce the number of unnecessary candidates further. The focus of this work, then, is to design an applicable grouping scheme that improves the search efficiency while retaining comparable classification result. We will refer to this approach as the relaxation of instance boundaries, or RIB.

This paper is organized as follows. In Section 2 we present an overview of the necessary materials underlying this work. In Section 3 we present our proposal for a HPM based on a relaxed grouping scheme in the splitting point selection. In Section 4, we present our results and compare them with Fayyad's method. In Section 5 the conclusion will be presented.

## 2. Classification tree overview

The top-down induction of decision trees (TDIDT) is a method for the task of concept learning (supervised learning) in the construction of classification trees. The aim of top-down induction of decision trees is to find a decision tree that is as small as possible and fits the data, because it is assumed that it is likely to find a good generalization of the training data that way. A tree is first built as a given set of pre-classified training examples is partitioned recursively according to a chosen attribute at each node, then a bottom-up tree pruning process follows to remove unreliable branches. The tree-building process is further divided into three steps: attributes evaluation, splitting point selection and training data partition. We will describe the selection of splitting point candidates in greater detail since different schemes in this step is the main motivation behind this work. Readers who are interested in greater details or other recent works are referred to the references listed [4–6,9,16,20,21], especially those of Fayyad and Irani [12–15]. The overview is intended to be minimal, and we do not discuss tree pruning.

### 2.1. Inductive learning

Inductive learning is the model building process in which an environment or database is analyzed to find patterns. Similar objects are grouped into classes and rules are formulated whereby it is possible to predict the class of new objects. This process of classification identifies classes that are characterized by a unique value or pattern that forms the class description.

In the supervised inductive learning, a supplied training dataset $x_i$ with defined classes is employed to fit the function $f(x_i) = y_i$ where $y_i$ are the output. The classifier $f$ can be tested with a test data set for accuracy, or be used to classify new data. The classification of new data is performed according to the probability calculation as described in the following. For the distribution of data $P(D = c)$ with the training data set $D$ and

$c = 1, 2, \ldots$, as the $c$th class, and the classification model $P(\mathbf{F}|D = c)$, where $\mathbf{F}$ is the feature vector of training dataset describing the instances, the output of the classification, $P(D = c|\mathbf{F})$, is given by

$$P(D = c|\mathbf{F}) \propto P(\mathbf{F}|D = c)P(D = c). \tag{1}$$

Eq. (1) shows the relationship between the classification model and the training data distribution. The feature vector $\mathbf{F}$ is composed of several features $F_i$ as

$$P(\mathbf{F}|D = c) = \prod_i P(F_i|D = c). \tag{2}$$

The form for classification differs between categorical and numerical attributes, which we explain below separately.

A categorical attribute $A$ has a split of the form $A \in V'$, where $V' \subset V$ and $V$ is the set of all possible values of attribute $A$. As in Fig. 1, the values of the "income" attributes are "High" and "Low" and are split into two branches: left branch "Income = High" and the right branch "Income = Low". The Time Complexity of the partition is $O(2^n)$ for categorical attributes, where $n$ is the number of possible partitions of an attribute. Clearly, the computational demand increases dramatically with increasing $n$ and this prompted some researchers to apply a threshold as a control. For example, Shih proposed a successful partition method for categorical attributes [32].

A numerical attribute $A$ has a binary split of the form $A \leqslant v$, where $v$ is a real number. The conditions for the left and right branches are $A \leqslant v$ and $A > v$, respectively. For example, in the partition of "Age" attribute in Fig. 1, the left branch corresponds to "Age $<=$ 20" and the right branch to "Age $>$ 20". The Time Complexity of the partition is $O(n)$ for numerical attributes.

Most of TDIDT algorithms such as ID3, C4.5, CART, SLIQ [25] and SPRINT [31] has built binary trees for numerical attributes. Berzal et al. proposed the multi-way decision trees with numerical attributes to reduce the tree complexity without decreasing classification accuracy [1].

The table in Fig. 1(a) exhibits the training data for constructing a classification tree that classifies potential customers for car dealers. In Fig. 1(b), the output variable has two classes – either "Yes" (Buy) or "No" (Not Buy). The tree model is a top-down decision mode where the root and the internal nodes are the decisions of the tree model. Hence, the classification or prediction is obtained from the sequence of decisions.

### 2.2. Top-down tree building

The training data are partitioned recursively, according to a chosen attribute at each node, until all training data are partitioned to the same class (see Fig. 2). For categorical attributes, a training dataset $T$ is split into two or more sub-training datasets $T_i, i = 1, 2, \ldots, j$. For numerical attributes, $T$ is split into two sub-training datasets $T_1$ and $T_2$.

The impurity function $\phi(p_1, p_2, \ldots, p_i)$ is the splitting index to evaluate the goodness of attributes to determine the optimal one for the split. Breiman et al. proposed *Gini* function [3], while Clark and Pregibon [8] and Quinlan [28] utilize *entropy* for the evaluation process. The two forms are defined as

$$Gini(T) = 1 - \sum_i p_i^2, \tag{3}$$

$$Entropy(T) = -\sum_i p_i \log_2 p_i, \tag{4}$$



**a**

| Age | Income | Class |
|---|---|---|
| 18 | Low | No |
| 23 | High | Yes |
| 40 | High | Yes |
| 19 | High | No |
| 25 | High | Yes |
| 45 | Low | No |

**b**

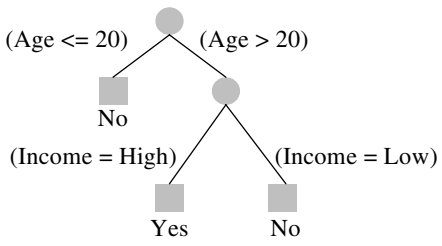


Fig. 1. A classification tree example.

```
Tree_Building(Training Data T)
    Partition(T);

Partition(T)
    If (all instances are the same class) then Return;
    Evaluate all attributes
      and Select the best attribute (A);
    Select the best partition point
      on the attribute (A);
    Split T into T_i;
    Partition(T_i);
```

Fig. 2. Tree building algorithm.

where $p_i$ is the relative frequency of class *of ith class* in the training dataset $T$. The *Gini*-based CART is a powerful algorithm that has been extensively studied (e.g. [20,23,26]) but is available as licensed software. The entropy-based C4.5, on the other hand, is a free distribution.

*2.2.1. Attributes evaluation*

At each node, available attributes are evaluated on the basis of separating the classes of the training examples. The choice of splitting attribute is made by selecting the attribute that will partition the original sample into sub-samples that are as homogenous as possible in relation to the class variable. The *Information Gain*, applied in ID3, is a popular impurity criterion that measures the average purity of the subsets produced by an attribute. *Gain Ratio* is a modification of *Information Gain* that reduces the bias on high-branch attributes, and is employed in C4.5, Fayyad's method and our HPM.

All three methods mentioned above uses entropy as a measure of available information associated with the nodes. Let $S$ be a set consisting of $s$ data samples with $m$ distinct classes, $C_i$, $I = 1,2,\ldots,m$, and let $s_i$ as the number of samples of $S$ in $C_i$, then the *Information* is conveyed via entropy as

$$I(T) = -\sum_{i=1}^{m} p_i \log_2 p_i, \tag{5}$$

where $p_i$ is the probability that an arbitrary sample belongs to class $C_i$, and is estimated by $s_i/s$.

Suppose a test attribute $A$ has $v$ distinct values $\{a_1,a_2,\ldots,a_v\}$ used to split $S$ into $v$ subsets, $\{S_1,S_2,\ldots,S_v\}$, where $S_j$ contains samples having value $a_j$, then the subsets correspond to the branches grown from the node containing $S$. Denoting $s_{ij}$ to be the number of samples of class $C_i$ in subset $S_j$, the *Expected Information* of attribute $A$ is evaluated as

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \cdots + s_{mj}}{s} I(s_{1j},\ldots,s_{mj}) \tag{6}$$

with the term $\frac{s_{1j}+\cdots+s_{mj}}{s}$ as the weight of the $j$th subset in $S$ having value $a_j$ of $A$, and the information of that subset is

$$I(s_{1j},\ldots,s_{mj}) = -\sum_{i=1}^{m} p_{ij} \log_2 p_{ij}, \tag{7}$$

where $p_{ij} = s_{ij}/S_j$ is the probability that a sample in $S_j$ belongs to class $C_i$. The *Information Gain* of attribute $A$, quantifying the change after the split, is given by

$$Gain(A) = I(T) - E(A). \tag{8}$$

Finally, the *Gain Ratio* (GR) defined as

$$\mathrm{GR}(A) = \frac{Gain(A)}{I(A)} \tag{9}$$

with

$$I(A) = \sum_{j=1}^{v} \frac{S_j}{S} \log_2 \frac{S_j}{S}. \tag{10}$$

The criterion of the test attribute selection is to choose the attribute with maximum GR($A$).

*2.2.2. Splitting point selection*

After a test attribute is selected, we search for the optimal splitting point among the instances regarding the test attribute. The instances are sorted according to the attribute value in ascending order (Fig. 3), and after which we compute the chosen *Information* function, according to the method specified, for the selected candidates. The one with maximal *Information* is then chosen as the splitting point. C4.5, Fayyad's method and the HPM utilize *Information Gain*, but differ in the selection process of the splitting point candidates. We explain briefly the selection process of C4.5 and Fayyad's method, but we defer presenting our modification according to RIB until the subsequent section.

*2.2.2.1. Splitting point selection – C4.5.* C4.5 measures the splitting point regarding all available instances. If the training dataset contains $n$ instances (assuming all instances have distinct values), then C4.5 algorithm evaluates $n - 1$ candidates and selects the one with optimal *Information Gain* as the splitting point. For example, as there are 10 instances in Fig. 3, the C4.5 algorithm performs 9 evaluations (candidate a, b, c, d, e, f, g, h, i) from the 10 instances.

The C4.5 algorithm is maximally inefficient in the search for splitting of numerical attributes since it evaluates all instances indiscriminately. For a large training data set, as in a typical machine learning application, this selection criterion is relatively expensive for handling numerical attributes.

*2.2.2.2. Splitting point selection – Fayyad's method.* Fayyad and Irani pointed out an inherent weakness in C4.5 which results in "bad" splitting points when more than two classes are present (Fig. 4) [13]. Their analysis of the binarization technique proved that optimal splits always fall on boundary points, independent of number of classes and their distribution (see Fig. 5). Hence, substantial reduction in time consumption can be obtained by considering only the boundary points need to be considered as potential candidates.

Fayyad's method overcomes the inefficiency by evaluating only the boundary points between classes (see Fig. 5). This method reduces the search time for the splitting points, but it has a drawback that it is inefficient

After Sorting

| Candidate | Attribute value | Class |
|---|---|---|
| a → | 4.3 | C1 |
| b → | 4.4 | C1 |
| c → | 4.5 | C1 |
| d → | 4.6 | C2 |
| e → | 4.7 | C1 |
| f → | 5.8 | C2 |
| g → | 6.6 | C2 |
| h → | 6.9 | C1 |
| i → | 7.3 | C2 |
| | 8.4 | C1 |

Fig. 3. Splitting point selection – C4.5.

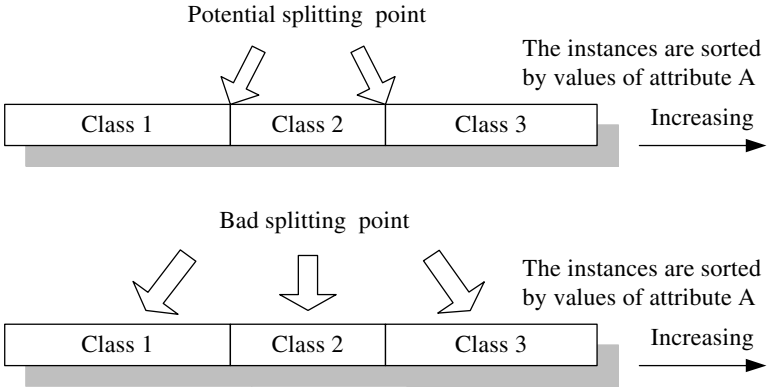


Fig. 4. Bad and potential splitting points.

After Sorting

| Candidate | Attribute value | Class |
|---|---|---|
| | 4.3 | C1 |
| | 4.4 | C1 |
| | 4.5 | C1 |
| a → | 4.6 | C2 |
| b → | 4.7 | C1 |
| c → | 5.8 | C2 |
| | 6.6 | C2 |
| d → | 6.9 | C1 |
| e → | 7.3 | C2 |
| f → | 8.4 | C1 |

Fig. 5. Splitting point selection – Fayyad's method.

if the attribute is uniformly distributed, and the number of candidates of splitting point is approximately $n - 1$. However, in this work we will not address this issue and the chosen data sets do not encounter this particular situation.

### 2.2.3. Training data partition

The splitting point, or threshold, is used as the decision condition of the node to split the training data $T$ into partitions $T_1$ and $T_2$ (see Fig. 6). The sub-training data is partitioned recursively, iterating the 3-step process, until all instances have been classified into partitions of the same class.

## 2.3. Other construction methods for numerical attributes

One difficulty in selecting the splitting point is that the values of numerical attributes are continuous. Some have suggested discretizing data to reduce the partition complexity via Equi-interval or Unequi-interval methods (see Fig. 7). Unfortunately, it has been proved that this discretization method can reduce accuracy. Nevertheless, the discretization process of numerical attributes is an important problem for classification tree learning. Dougherty et al. [11], Berzal et al. [1], Fayyad and Irani [14] and Liu et al. [24] have shown that discretization techniques can be successfully employed to build classification trees.

In some situations multi-interval discretization may perform better than binary discretization, and build decision trees that are more compact and more accurate. For example, Perner and Trautzsch [27] have examined four multi-interval discretization methods for induction of decision trees used in dynamic fashion. Kim and Loh have also worked on multiway methods considering bias [21].

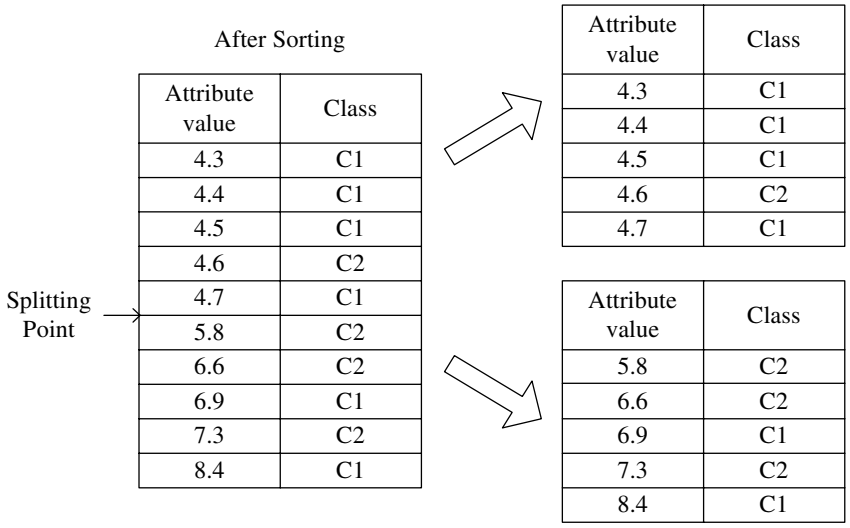


Fig. 6. Training data partition.

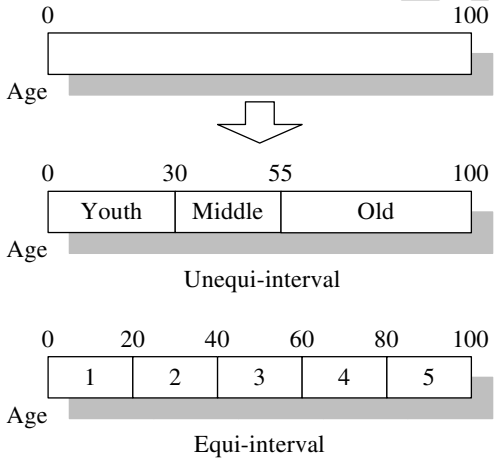


Fig. 7. Discretization.

Some approaches adopt an alternative view. For example, Chu regarded classification as a merging problem and proposed a bottom-up merging method to construct classification trees (see Fig. 8) [7]. However, although this method performs well for two classes, it has not been shown to work efficiently for more classes.

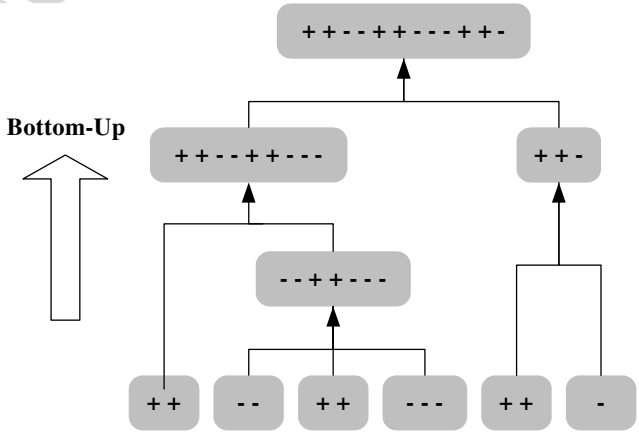


Fig. 8. Bottom-up construction.

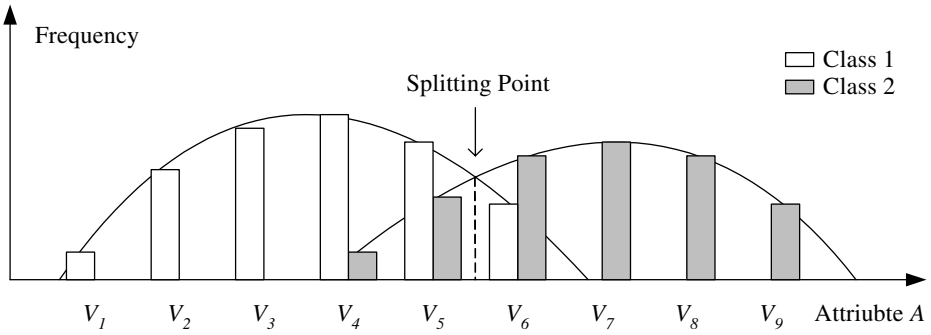


Fig. 9. Search of splitting point in Zhang and Wu's method.

Zhang and Wu used a binning approach to reduce the search time for splitting point [36]. They binned training data into histograms and then projected on a horizontal axis by the attribute. In Fig. 9 shows an example with $A = \{V_1, V_2, \ldots, V_9\}$ as the attribute with values The white rectangles represent the binned training data of Class 1, the gray rectangles represent that of Class 2, and the vertical axis shows the frequency of $V_i$. The splitting point is determined from the boundary of Class 1 and Class 2. Although this method reduces the searching time of splitting point, it also reduces the classification accuracy.

## 3. The RIB proposal for the HPM

Our idea is motivated by the realization that Fayyad's method can be relaxed further without compromising classification accuracy. The basic premise is that the information of instances and their collective pattern is not lost under some other different, but appropriately constructed, grouping schemes. We therefore seek to construct a minimal modification of Fayyad's splitting point selection to illustrate this point. We do not modify either attributes evaluation or training data partition.

Most obviously, we seek a modification which will reduce the number of splitting point candidates, provided that the algorithm does not inaccurately remove candidates that are legitimate. The minimal approach is to eliminate situations where a single isolated instance is located amidst instances of another class. For example, an instance of class 2 may be located between two instances of class 1.

Different from Fayyad's approach, we do not identify a splitting point candidate in this situation but accepts this isolated instance into the group – thus it is considered as a "noise". We carefully point out that this is not the unique grouping choice even in this minimal modification, but nevertheless a definitive result showing a faster search speed suffices to validate the working of RIB. In addition, we have only considered a very simple situation in the distribution of instances, and that other situations may exist.

Fig. 10 demonstrates graphically the different possible data distributions in the training data. One situation is where adjacent instances are of the same class (situation 1), and the other situation is the mixed case where adjacent instances are of different class (situation 2).

In this illustration, there are six splitting point candidates according to Fayyad's partition method (see Fig. 11) and they are labeled into five partition situations with respect to how they occur with respect to the data distribution. In partition situation 1, 2, 3 and 4, the corresponding splitting points are "a", "b", "c" and "d". In partition situation 5, the splitting points are "e" and "f".

Fayyad proved that the splitting point always occurs on the boundary points between classes, and used this fact to reduce the searching time over C4.5. However, the strict selection process in Fayyad's method can still result in inefficient partition situations where irrelevant splitting candidates are selected.

As shown in Fig. 12, if the splitting points occur in S2 and S4 (corresponding to points "b", "e" and "f" in Fig. 11) where adjacent instances are of different classes, the partition does not improve the resulting classification. The HPM is a more discriminatory design constructed to obviate this unnecessary identification by default.

We give a pedagogical explanation of the grouping scheme in HPM. First we consider grouping neighboring data that are of the same class (see Group A and B in Fig. 13, corresponding to situation 1 in Fig. 10), and

After Sorting

| Attribute value | Class | |
|---|---|---|
| 4.3 | C1 | Situation 1 |
| 4.4 | C1 | |
| 4.5 | C1 | |
| 4.6 | C2 | Situation 2 |
| 4.7 | C1 | |
| 5.8 | C2 | Situation 1 |
| 6.6 | C2 | |
| 6.9 | C1 | Situation 2 |
| 7.3 | C2 | |
| 8.4 | C1 | |

Fig. 10. Data distribution.

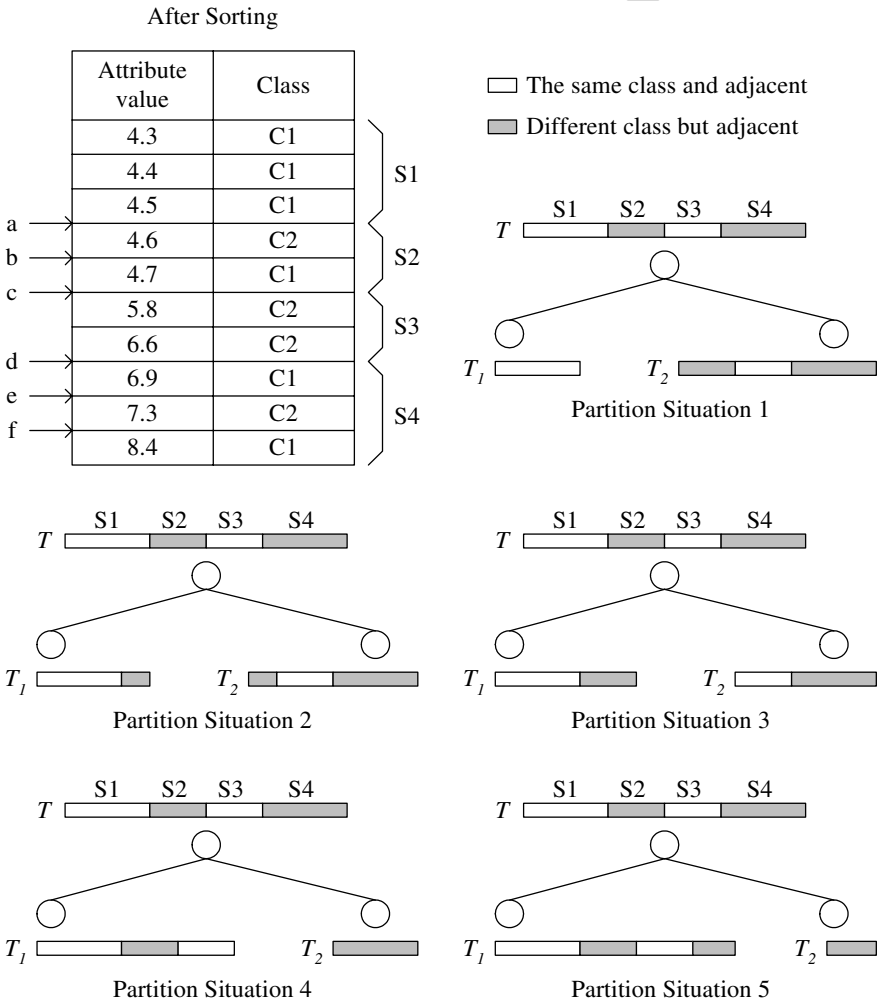


Fig. 11. Five partition situations according to Fayyad's method.

then in the second grouping, the ungrouped data are grouped with previously grouped data to form new groups (see Groups A′ and B′ in Fig. 13.). Clearly in this scheme, the selection of candidates is more relaxed than Fayyad's method.

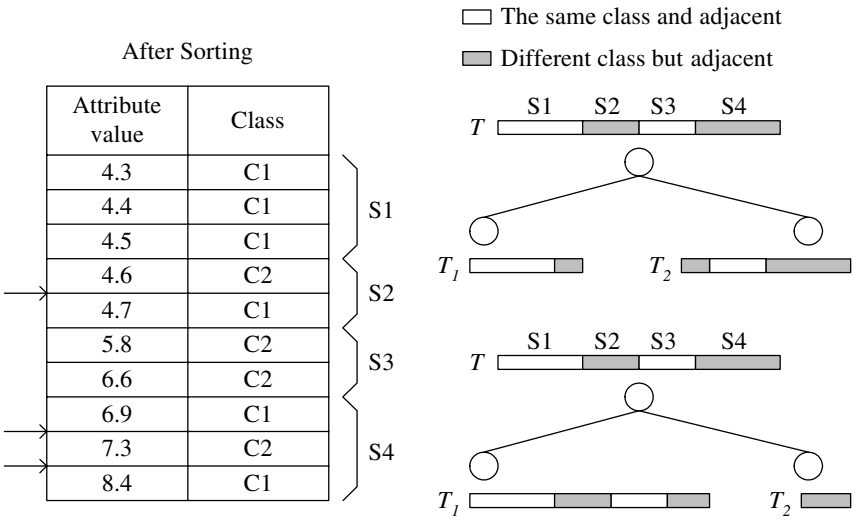


Fig. 12. Inefficient partition situation in Fayyad's method.

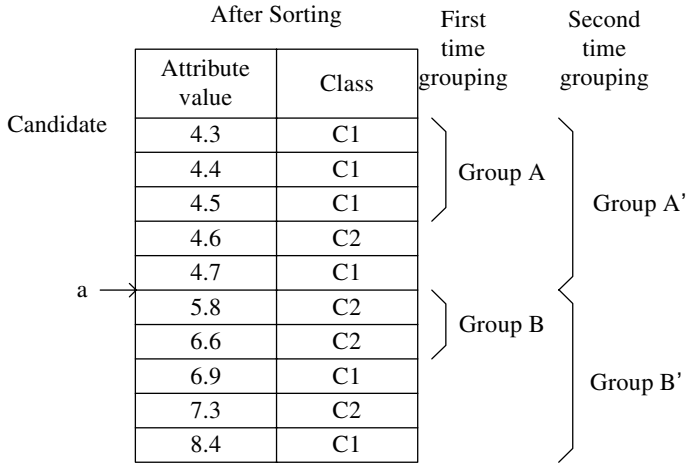


Fig. 13. Grouping in the heuristic partition method.

After the second time grouping, the candidates of splitting points then lie between the groups, such as point "a" between Groups A′ and B′. This simple procedure satisfies our goal in eliminating candidates that are the result of isolated instances as previously explained. Finally, we compute the *Information Gain* of the splitting point candidates and select the one with maximal information as the splitting point.

## 4. Experiment and results

In this section we compare the performance of the HPM with Fayyad's method. Experiments are carried out on six databases from the UC Irvine Machine Learning Repository [2]: They are Glass Identification data set, Pima Indians Diabetes date set, Thyroid Disease data set, Liver-disorders data set, Iris plant data set and Wine recognition data set. All sets contain no missing values. The properties of these databases have been listed in Table 1.

*Glass identification*: The set contains 6 classes and 9 attributes. The number of instances is 70, 17, 76, 13, 9 and 29 for the six classes, respectively. The attributes are the data of the percentage of Fe, Na, Mg, Si, Ca, etc., in the glass.

Table 1
The databases

| Data base | Field | Classes | Attributes | Instances | Missing |
|---|---|---|---|---|---|
| Glass identification | Science | 6 | 9 | 214 | No |
| Thyroid disease | Medical | 3 | 5 | 215 | No |
| Pima Indians diabetes | Medical | 2 | 8 | 768 | No |
| Liver-disorders | Medical | 2 | 6 | 345 | No |
| Iris plant | Plants | 3 | 4 | 150 | No |
| Wine recognition | Plants | 3 | 13 | 178 | No |

*Pima Indians diabetes*: This is the diabetes medical data set of two classes: positive and negative for diabetes. The positive class has 268 instances and the negative class has 500 instances. All attributes are the data of patients. In particular, all patients are females at least 21 years old of Pima Indian heritage.

*Thyroid disease*: This database has 3 classes, namely normal, hyper and hypo, and 5 attributes. There are 150 instances for the normal class, 35 instances for the hyper class and 30 instances for the hypo class. The 5 attributes are used to classify a patient's thyroid as euthyroidism, hypothyroidism or hyperthyroidism.

*Liver-disorders*: This set contains the liver disorders medical data of two classes. There are 6 numerical attributes and 345 instances: 145 for class 1 and 300 for class 2. The first 5 variables are blood tests that are believed to be sensitive to liver disorders and may arise from excessive alcohol consumption. The last variable is the number of half-pint equivalent of alcoholic beverages consumed per day.

*Iris plant*: This database is well known in multivariate analysis and also in pattern recognition. This database contains three types (classes) of iris and there are 50 data (instances) for each type. The four numerical attributes are sepal length, sepal width, petal length and petal width.

*Wine recognition*: This dataset contains three types: class 1 has 59 instances, class 2 has 71 instances and class 3 has 48 instances. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

The comparison with Fayyad's method is based on three criteria: search time for splitting points, tree size (number of nodes) and classification accuracy. For classification accuracy, we sample 1/3 of total instances from each database as the testing data. We run two modules in our study (Fig. 14). The first module is attributes evaluation and the input is the training data. The second module is the splitting point selection and the input is either the Fayyad's method or the HPM.

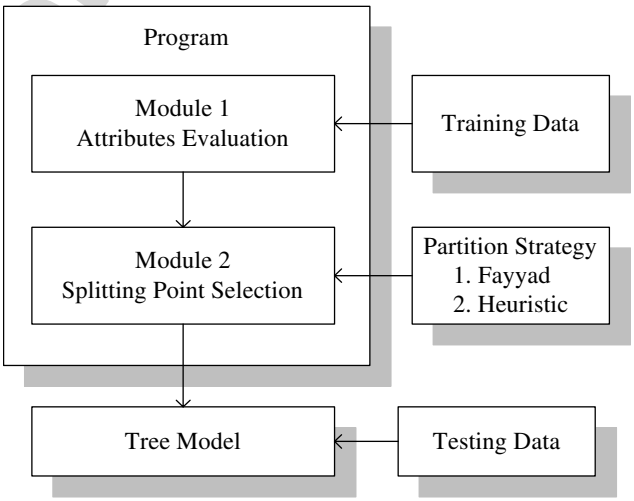


Fig. 14. The modules of program.

Table 2
Experiment results

| Data base | Partition method | Candidates (accumulated) | Time (s) | Nodes | Accuracy (%) |
|---|---|---|---|---|---|
| Glass identification | Fayyad | 241 | 12.74 | 107 | 46 |
| | Heuristic | 151 | 7.10 | 109 | 42 |
| Thyroid disease | Fayyad | 55 | 2.59 | 29 | 86 |
| | Heuristic | 37 | 1.77 | 29 | 85 |
| Pima Indians diabetes | Fayyad | 2772 | 141.07 | 347 | 67 |
| | Heuristic | 1662 | 71.37 | 319 | 66 |
| Liver-disorders | Fayyad | 832 | 40.58 | 147 | 59 |
| | Heuristic | 351 | 15.96 | 135 | 60 |
| Iris plant | Fayyad | 10 | 0.41 | 9 | 90 |
| | Heuristic | 7 | 0.37 | 11 | 94 |
| Wine recognition | Fayyad | 94 | 4.32 | 25 | 73 |
| | Heuristic | 85 | 4.04 | 39 | 90 |

We build a classification tree according to the HPM and the Fayyad's method for each of six data sets, and the result is shown in Table 2. In all data sets, the HPM has a faster search time as well as a smaller number of candidates. For tree size and classification accuracy, the differences between two methods are not significant.

As expected, the set with a greater number of candidates, hence a larger tree size, generally has a more significant reduction in both the number of candidates and the search time (Fig. 15). For all cases the two partition methods build essentially the same tress (with same number of nodes). Fig. 16 displays a clear relation between the number of candidates and the search time, and the relation is especially clear with larger sets. We easily infer that the evaluation of attributes is the dominant factor in computations, and that higher efficiency is achieved by reducing the number of evaluations.

For the classification accuracy, the HPM achieves similar performance as Fayyad's method for the Glass, Thyroid, Pima and Liver-disorders data sets. In Iris and Wine recognition data sets, the HPM performs better than Fayyad's method, especially in the Wine recognition data set.

We point out that this is consistent with Wu and Urpani's results [35] that trees with smaller size have worse classification accuracy. It may be an interesting study to examine how tree size can influence classification accuracy.

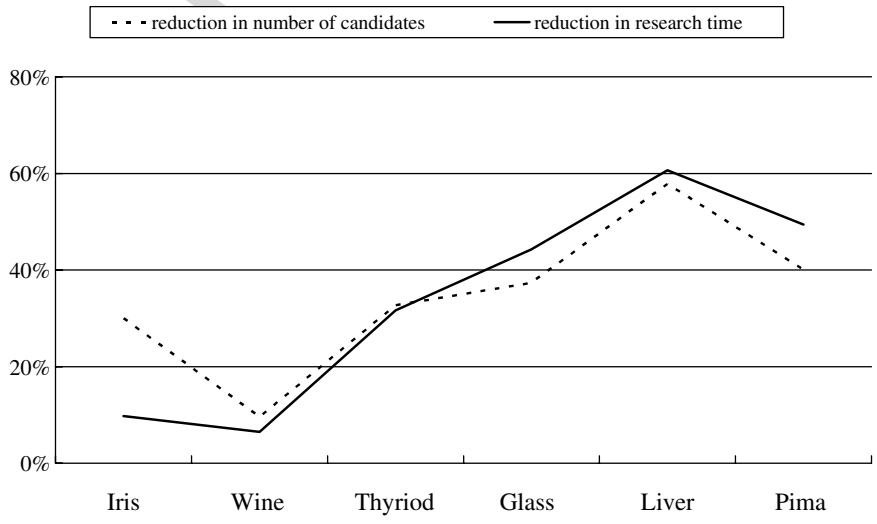


Fig. 15. The reduction of number of candidates and search time of HPM from Fayyad's method.

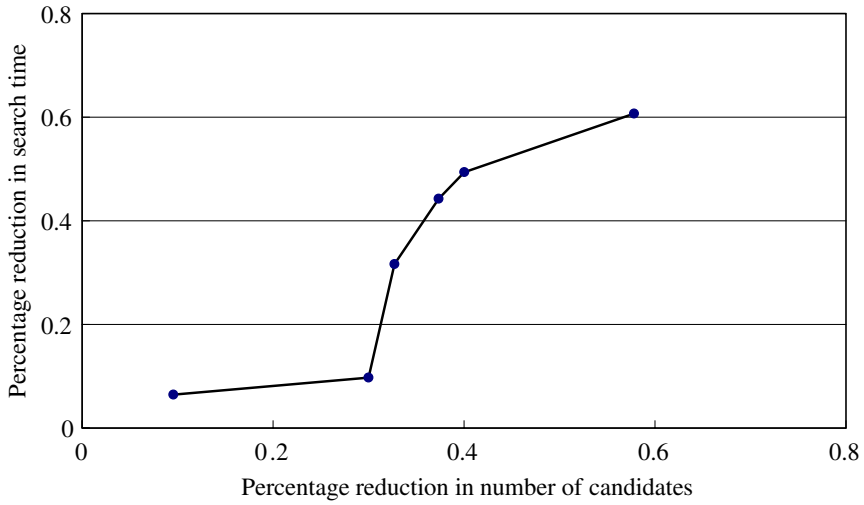


Fig. 16. Reduction in search time vs. reduction in number of candidates.

## 5. Conclusions

The direct connection between RIB and efficiency demonstrates an important flexibility in the selection of boundaries as splitting point candidates. This general idea is illustrated explicitly by the simple HPM, and it can be examined further or utilize in constructing more efficient methods. Future studies can address various technical details, but for this study, we point out that the result has a wider range of applicability than the narrow scope considered in this work. For example, it can be explored for machine learning algorithms.

At the minimal level, our simple HPM already provides a working and an efficient procedure, and essentially progress Fayyad's method one step farther. The smaller number of splitting point candidates contributes directly to the reduction of search time while giving essentially the same classification result. A logical extension of this work, then, is a systematic investigation of different grouping schemes to identify different situations of boundary patterns where candidate selection process can be made even simpler.

Our result is also consistent with other related studies that there is no general improvement in classification accuracy. This is expected, since HPM relaxes the candidate selection process and hence there is no reason to expect improvement in accuracy. We are also careful not to extend our conclusion as being applicable to all other existing partition methods, nor can we infer definitive information on tree size.

Interestingly, Kurgan and Cios have explored a discretization algorithm called class-attribute interdependence maximization (CAIM) algorithm and show that it achieves the highest classification accuracy over six other competing discretization algorithms [22]. As CAIM already offers a well-built machinery to boost accuracy, it offers a tantalizing prospect to incorporate RIB to build an overall superior algorithm.

This work can also be extended in other ways. We can study HPM with data sets that possess high dimensional attributes or great number of the instances, we can explore its application to other areas, or we can revisit and study other successful algorithms. Most importantly, as already mentioned, RIB should be studied further.



## References

[1] F. Berzal, J.C. Cubero, N. Marín, D. Sánchez, Building multi-way decision trees with numerical attributes, Information Sciences 165 (2004) 73–90.
[2] C.L. Blake, C.J. Merz, UCI Repository of Machine Learning Databases. Available from: <http://www.ics.uci.edu/~mlearn/MLRepository.html>, UC Irvine, Department of Information and Computer Science, 1998.
[3] L. Breiman et al., Classification and Regression Trees, Wadsworth, Belmont, 1984.
[4] T.Y. Chen, P.L. Poon, Classification-hierarchy table: a methodology for construction of the classification tree, in: Proceedings of Australian Software Engineering Conference, 1996, pp. 93–104.
[5] T.Y. Chen, P.L. Poon, T.H. Tse, A new restructuring algorithm for the classification-tree method, in: Proceedings of 9th International Workshop on Software Technology and Engineering Practice, IEEE Computer Society, Los Alamitos, CA, 1999, pp. 105–114.
[6] T.Y. Chen, P.L. Poon, T.H. Tse, Classification-tree restructuring methodologies: a new perspective, IEEE Proceedings: Software 149 (2) (2002) 65–74.

[7] M.J. Chu, The Classification of Continuous Attributes in Decision Tree Learning, Master thesis, Information and Computer Engineering, Chung Yuan Christian University, Taiwan, 1996.
[8] L.A. Clark, D. Pregibon, Tree-based models, in: J.M. Chambers, T.J. Hastie (Eds.), Statistical Models in S, Wadsworth Brooks/Cole, Pacific Grove, CA, 1992.
[9] D.G.T. Denison, B.K. Malick, Classification trees, in: Dey, Ghosh, Mallick (Eds.), Generalized Linear Models: A Bayesian perspective, 2000, pp. 365–372.
[10] A. Dobra, J. Gehrke, Bias correction in classification tree construction, in: Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufman, 2001, pp. 90–97.
[11] J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features, in: Proceedings of the 12th International Conference on Machine Learning, Morgan Kaufman, Los Altos, CA, 1995, pp. 194–202.
[12] U.M. Fayyad, K.B. Irani, What should be minimized in a decision tree? in: Proceedings of the Eighth National Conference on Artificial Intelligence AAAI-90, MIT Press, Cambridge, MA, 1990, pp. 749–754.
[13] U.M. Fayyad, K.B. Irani, On the handling of continuous-valued attributes in decision tree generation, Machine Learning 8 (1992) 87–102.
[14] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proceedings of the 13th International Joint Conference on Artificial Intelligence, 1993, pp. 1022–1027.
[15] U.M. Fayyad, From Data Mining to Knowledge Discovery: An Overview Advances in Knowledge Discovery and Data Mining, AAAI Press, 1993.
[16] J. Gehrke, R. Ramakrishnan, V. Ganti, RainForest – a framework for fast decision tree construction of large datasets, in: Proceedings of the 24th VLDB Conference, New York, USA, 1998, pp. 416–427.
[17] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufman, 2001.
[18] M.A. Hearst, Support vector machines, IEEE Intelligent Systems (1998) 18–28.
[19] D.M. Hawkins, Firm: Formal inference based recursive modeling, Technical Report, University of Minnesota, 1997.
[20] T. Hothorn, B. Lausen, Bundling classifiers by bagging trees, Computational Statistics and Data Analysis 49 (2005) 1068–1078.
[21] H. Kim, W.-Y. Loh, Classification trees with unbiased multiway splits, Journal of the American Statistical Association 96 (2001) 589–604.
[22] L. Kurgan, K. Cios, CAIM discretization algorithm, IEEE Transactions on Knowledge and Data Engineering 16 (2) (2004) 145–153.
[23] R.J. Lewis, An introduction to classification and regression tree (CART) analysis, The Annual Meeting of the Society for Academic Emergency Medicine, Francisco, California, 2000.
[24] H. Liu, F. Hussain, C.L. Tan, M. Dash, Discretization: an enabling technique, Journal of Data Mining and Knowledge Discovery 6 (4) (2002) 393–423.
[25] M. Mehta, R. Agrawal, J. Rissanen, SLIQ: a fast scalable classifier for data mining, in: Proceedings of the 5th International Conference on Extending Database Technology, Avignon, France, 1996, pp. 18–32.
[26] D. Pavlov, J. Mao, B. Dom, Scaling-up support vector machines using boosting algorithm, in: International Conference on Pattern Recognition (ICPR'00), vol. 2, 2000.
[27] P. Perner, S. Trautzsch, Multi-interval discretization methods for decision tree learning, in: A. Amin, D. Dori, P. Pudil, H. Freeman (Eds.), Advances in Pattern Recognition, LNCS 1451, Springer-Verlag, 1998, pp. 475–482.
[28] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.
[29] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, 1993.
[30] J.R. Quinlan, Improved use of continuous attributes in C4.5, Journal of Artificial Intelligence Research 4 (1996) 77–90.
[31] J.C. Shafer, R. Agrawal, M. Mehta, SPRINT: a scalable parallel classifier for data mining, in: Proceedings of 22nd International Conference on VLDB, Mumbai (Bombay), India, 1996, pp. 544–555.
[32] Y.S. Shih, Selecting the best splits for classification trees with categorical variables, Statistics and Probability Letters 54 (2001) 341–345.
[33] Y.S. Shih, Regression trees with unbiased variable selection, Statistica Sinica 12 (2002) 361–386.
[34] Y.S. Shih, A note on split selection bias in classification trees, Computational Statistics and Data Analysis 45 (2004) 457–466.
[35] X. Wu, D. Urpani, Induction by attribute elimination, IEEE Transactions on Knowledge and Data Engineering 11 (5) (1999) 805–812.
[36] S. Zhang, X. Wu, Large scale data mining based on data partitioning, Applied Artificial Intelligence 15 (2) (2001) 129–139.