具有高可理解性的二分决策树生成算法研究*

蒋艳凰[1+], 杨学军[1], 赵强利[2]

[1](国防科学技术大学 计算机学院,湖南 长沙 410073)

[2](清华得实科技股份有限公司,北京 100085)

Constructing Binary Classification Trees with High Intelligibility

JIANG Yan-Huang[1+], YANG Xue-Jun[1], ZHAO Qiang-Li[2]

[1](School of Computer, National University of Defence Technology, Changsha 410073, China)

[2](Tsinghua Dascom Scie-Tech Co., Ltd, Beijing 100085, China)

+ Corresponding author: Phn: 86-731-4573681, E-mail: jiang-yh@163.com

http://www.nudt.edu.cn

Received 2002-10-29; Accepted 2002-12-31

Jiang YH, Yang XJ, Zhao QL. Constructing binary classification trees with high intelligibility. *Journal of Software*, 2003,14(12):1996~2005.
http://www.jos.org.cn/1000-9825/14/1996.htm

Abstract: Binarization is the most popular discretization method in decision tree generation, while for the domain with many continuous attributes, it always gets a big incomprehensible tree which can't be described as knowledge. In order to get a more intelligible decision tree, this paper presents a new discretization algorithm, RCAT, for continuous attributes in the generation of binary classification tree. It uses simple binarization to solve the multisplitting problem through mapping a continuous attribute into another probability attribute based on statistic information. Two pruning methods are introduced to simplify the constructed tree. Empirical results of several domains show that, for the two-class problem with a preponderance of continuous attributes, RCAT algorithm can generate a much smaller decision tree efficiently with higher intelligibility than binarization while retaining predictive accuracy.

Key words: machine learning; binary classification tree; information gain; pruning; range-splitting based on continuous attributes transform (RCAT) algorithm

摘 要: 二分离散化是决策树生成中处理连续属性最常用的方法,对于连续属性较多的问题,生成的决策树庞大,知识表示难以理解.针对两类分类问题,提出一种基于属性变换的多区间离散化方法——RCAT,该方法首先将连续属性转化为某类别的概率属性,此概率属性的二分法结果对应于原连续属性的多区间划分,然后对这些

* Supported by the National Natural Science Foundation of China under Grant No.69825104 (国家自然科学基金)

JIANG Yan-Huang was born in 1976. She is a Ph.D. candidate at the School of Computer, National University of Defence Technology. Her research interests are machine learning, artificial intelligence and image processing. **YANG Xue-Jun** was born in 1963. He is a professor and doctoral supervisor of the School of Computer, National University of Defence Technology. His current research areas include computer architecture, parallel compilation. **ZHAO Qiang-Li** was born in 1973. He is a senior engineer of Tsinghua Dascom Scie-Tech Co., LTD. His interested areas include artificial intelligence and information security.

区间的边缘进行优化,获得原连续属性的信息熵增益,最后采用悲观剪枝与无损合并剪枝技术对 RCAT 决策树进行简化.对多个领域的数据集进行实验,结果表明:对比二分离散化,RCAT 算法的执行效率高,生成的决策树在保持分类精度的同时,树的规模小,可理解性强.

关键词: 机器学习;二分决策树;信息熵增益;剪枝;RCAT 算法

中图法分类号: TP18 **文献标识码:** A

1 Introduction

Nowadays, many fields need to get useful and understandable knowledge from amount of data, so under the satisfaction of predictive accuracy, intelligibility becomes more and more important for a learning algorithm[1,2]. There are three strategies used to get comprehensible knowledge from pre-classified cases. The first strategy is converting the classifier that has blackbox problem into another understandable one; Setiono and Liu[3] extract rules from neural networks by pruning and hidden-unit splitting. Boz[4] provides a DecText system to extract a decision tree from a trained neural network. The second strategy is the simplification of big decision trees or rule sets into simple ones; Windeatt and Ardeshir[5] analyze the relationship between recognition accuracy and intelligibility, and use several pruning methods to simplify classifiers. These two strategies will reduce the classifying accuracy when improving on intelligibility, and therefore how to get a good trade-off between these two facets is very important[6,7]. The third one is the development of new algorithms to construct understandable classifiers; Auer[8] uses a searching method with high complexity to find an optimal two layer decision tree. Langley[9] uses a tabular format as the representation of knowledge, which is easy to be understood.

Decision tree is an empirical learning method which can get knowledge from a set of pre-classified cases, where continuous attributes need to be discretized at first. Binarization[10] is the most popular discretization method in decision tree generation, in which the value range is discretized into only two intervals according to certain threshold. If a continuous attribute is selected as an expanding attribute for an interior node, it will split this node into only two branches. This leads to the fact that one continuous attribute may be selected many times in one path from the root to a leaf, and results in a big incomprehensible tree with many nodes in the end[8].

In this paper, we explore a discretization method, RCAT, for continuous attributes and apply it to construct understandable decision trees for two-class problems, which belongs to the third strategies mentioned above. Unlike other multi-interval discretization[11~13], RCAT uses a simple binarization to solve the multisplitting problem through mapping a continuous attribute into a probability attribute. The binarization of the probability attribute corresponds to a multisplitting of the original one. Experimental results show that, compared with binarization, our method can generate much smaller decision trees with higher intelligibility while retaining predictive accuracy.

In Section 2, we give a description of the concepts used in this paper. In Section 3, we describe the RCAT algorithm in detail. Two pruning algorithms are introduced in Section 4. Experimental results and their analysis are shown in Section 5. Finally we give a summary of our work in Section 6.

2 Concept Description

For a decision tree, each node of the tree is associated with a set of cases. Each interior node has an expanding attribute to split the node into some branches, and each leaf node gives a class name as the classified result. In this paper, we let *Nclass* denote the number of classes, *TS* the total training set, T a set of cases, and $|T|$ the cardinality of T. Analogously, for any set, let s denote a case, A a continuous attribute, $val_A(s)$ the value of attribute A in case s, and D a node of a decision tree.

2.1 Discretization and partition

We will get a partition for a case set according to a discretization of a continuous attribute. Suppose that $\{a_1,a_2,\ldots,a_{k-1}\}$ is the set of cutting points of attribute A in an ascending order, which discretizes the value range of attribute A into k intervals. This discretization defines a partition $\coprod_{i=1}^{k} T_i$ for the case set T as follows:

$$T_i=\begin{cases}\{s\in T \mid val_A(s)\le a_1\} & \text{if } i=1\\ \{s\in T \mid a_{i-1}<val_A(s)\le a_i\} & \text{if } 1<i<k\\ \{s\in T \mid a_{k-1}<val_A(s)\ & \text{if } i=k\end{cases}$$

and the partition has following properties:

① non-empty subsets: $|T_i|\ge 1$, for all $i\in\{1,2,\ldots,k\}$,

② covers the whole domain: $\bigcup_{i=1}^{k} T_i = T$,

③ subsets are disjoint: $T_i \bigcap T_j=\varnothing$, for all $i\ne j$, $1\le i,j\le k$ and

④ subsets are sorted: if $s_i\in T_i$, $s_j\in T_j$, and $i<j$, then $val_A(s_i)<val_A(s_j)$.

From ④ we know that the cases which have identical value of attribute A will be in the same subset, and both the case sequence in each subset and the subset sequence are sorted into an ascending order by the values of attribute A.

2.2 Evaluation functions: information gain

There are many evaluation functions used to evaluate candidate partitions, such as Gini Index, Information Gain, Gain Ratio, Minimum Description Length Principle, and Minimum Training Set Error etc. In this paper, we use information gain as our evaluation function. Given that some attribute divides the case set T into k subsets, the information gain of this partition is

$$gain = Info(T)-\sum_{i=1}^{k}\frac{|T_i|}{|T|}\times Info(T_i),$$

thereinto $Info(T)$ is the entropy of set T, calculated as:

$$Info(T)=-\sum_{j=1}^{NClass}\frac{freq(C_j,T)}{|T|}\times\log_2\left(\frac{freq(C_j,T)}{|T|}\right),$$

where $freq(C_j,T)$ denotes the number of cases belonging to class C_j in T.

3 RCAT Algorithm

In order to construct decision trees with high intelligibility, we explore a new method-RCAT to discretize a continuous attribute into several intervals. Unlike other multisplitting methods, RCAT uses a simple binarization to solve the multisplitting problem through mapping a continuous attribute into another probability attribute, and the binarization of the probability attribute has a corresponding multisplitting of the original one. In the following sections, we will focus on the two-class problems to discuss RCAT algorithm in detail.

3.1 Interval initialization

Interval initialization is a simple discretization. In order to guarantee the reliability of statistic information, we let the number of cases in each interval be larger than a given limit. Thus the first property of its partition should be changed to: ① $|T_i|\ge LIMIT_S$, for all $i\in\{1,2,\ldots,k\}$. Another constraint for the initialization is the setting of an upper limit for the number of intervals, namely $k\le LIMIT_R$. In initialization, we first sort the cases in T by their values of attribute A, provided that the sorted case sequence is $s_1,s_2,\ldots,s_{|T|}$, where $val_A(s_i)\le val_A(s_{i+1})$, then segment the value range of A into $LIMIT_R$ intervals. The length of each interval is

$$L=(val_A(s_{|T|})-val_A(s_1))/LIMIT_R.$$

We then enumerate the number of cases in each interval. If the number in an interval is less than $LIMIT_S$, combine this interval with its next adjacent interval. For the last interval with too short of cases, we will combine it with its previous interval.

3.2 Binarization of probability attribute

Suppose that we get k_1 intervals after initialization, now let us compute the probability estimation of each class for every interval. Because we just deal with the two-class problems, given that interval R_i is associated with case set T_i, the frequency of the first class in T_i is $freq(C_1,T_i)$, and the probability estimations for either class in T_i is

$$p(C_1,T_i)=freq(C_1,T_i)/|T_i|,\qquad p(C_2,T_i)=1-p(C_1,T_i).$$

Provided that the probability estimations of each interval for the first class are $p(C_1,T_1),p(C_1,T_2),\ldots,p(C_1,T_{k1})$ respectively, we can map the value space of A into a probability space of C_1 by using another continuous attribute P whose values lie in [0,1] to replace the attribute A. The mapping rule is

$$\text{if}\quad val_A(s)\in R_i \quad\text{then}\quad val_P(s)=p(C_1,T_i).$$

Now we can use the binarization to find the best threshold for attribute P. Let an ascending ordered value sequence of P be $p_1,p_2,\ldots,p_{k1}$, then every distinct value $t=p_i, 1\le i\le k_1$ in this sequence is a possible threshold, which splits the whole set T into two sets: T_i with $val_p(s)\le t$ and T_2 with $val_p(s)>t$. We can get the information gain:

$$gain(P,t)=Info(T)-\sum_{i=1}^{2}\frac{|T_i|}{|T|}\times Info(T_i).$$

Let $t=p^*$ be the threshold with the maximum information gain, then p^* is just the threshold we need.

3.3 Interval combination

We give a description of the relations between attribute A and P in Fig.1, where horizontal and vertical axes represent the value space of attribute A and P respectively. The intervals on horizontal axis in Fig.1 are the initial intervals of attribute A, and the fatter ones (the value range of the interval is larger than that of others in the value space of A) indicate that they have been combined with their adjacent intervals whose number of cases are less than $LIMIT_S$. The taller intervals indicate that the cases in their intervals belong to the first class with high frequency. $p^*$is the best threshold of attribute P, which separates the initial intervals into two sets; one contains the intervals whose probability is below or equal to p^* and the other set contains the remaining intervals whose probability is over p^*.

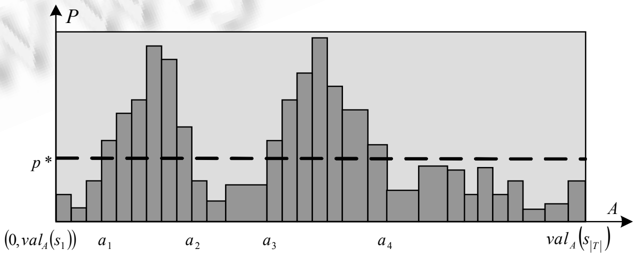


Fig. 1 Relations between attribute A and P

Fig.1 Relation between attributes A and P

Both sets divided by p^* may have several initial intervals. We combine the adjacent intervals in the same set since they have a similar property for attribute P. Suppose that after combination, the total number of intervals in either of sets is k_2, obviously $k_2\le k_1$. In Fig.1 we have k_1=29 and k_2=5. The binarization of attribute P corresponds to the k_2-splitting of attribute A, the case sets after combination can be described as follows:

$$T_1=\{s\mid val_A(s)\le a_1 \text{ or } a_2<val_A(s)\le a_3 \text{ or } a_4<val_A(s), s\in T\}$$

$$T_2=\{s\mid a_1<val_A(s)\le a_2 \text{ or } a_3<val_A(s)\le a_4, s\in T\}$$

where T_1 has three combined intervals, and T_2 has two.

3.4 Boundary optimization

From Fig.1, we can see that the combined intervals belong to T_1 or T_2 alternately. The cases whose values of the original attribute are at the boundary of some interval should be in T_1, but we put them to T_2 or reverse will reduce the information gain of the attribute. Let the k_2 combined intervals associate with the case sets $T_1,T_2,\ldots,T_{k2}$ respectively, and the cases in each subset are sorted by their value of attribute A, then the information gain of this partition is

$$gain(A)=Info(T)-\sum_{i=1}^{k_2}\frac{|T_i|}{|T|}\times Info(T_i).$$

Elomaa and Rousu[11] use boundary points to get an optimal k splitting of the continuous attributes, but we will use boundary points to do optimization for RCAT. Here is the definition of the boundary points:

Definition 1. Let a sequence T of cases be sorted by the value of a continuous attribute A. The augmented set of the boundary points is defined as follows:

1. The highest value of attribute A in the sequence T is an additional boundary point.
2. A value $b\in Dom(A)$ is a boundary point if and only if there exists a pair of cases $s_1,s_2\in T$, having different classes, such that $val_A(s_1)=b<val_A(s_2)$; and there does not exist another case $s\in T$ such that $val_A(s_1)<val_A(s)<val_A(s_2)$.

From Definition 1, we can know that every case set has at least one boundary point with respect to a continuous attribute. The boundary points of attribute A separate the case sequence T into some case blocks. Now we use boundary points to optimize the border of the combined interval and the definition of optimizing operator is given as:

Definition 2. Suppose that an arbitrary discretization of the continuous attribute A gives the case set T a partition $\coprod_{i=1}^{k}T_i$, whose information gain is $gain(A)$. Let the sorted case sequence of T_i be $s_{1,i},s_{2,i},\ldots,s_{|Ti|,i}$, and that of T_{i+1} be $s_{1,i+1},s_{2,i+1},\ldots,s_{|Ti|,i+1}$, do one of the following operators:

① $T_i'=T_i-\{s\mid val_A(s)=val_A(s_{|T_i|,i}), s\in T_i\}$, $T_{i+1}'=T_{i+1}\cup\{s\mid val_A(s)=val_A(s_{|T_i|,i}), s\in T_i\}$

② $T_i'=T_i\cup\{s\mid val_A(s)=val_A(s_{1,i+1}), s\in T_{i+1}\}$, $T_{i+1}'=T_{i+1}-\{s\mid val_A(s)=val_A(s_{1,i+1}), s\in T_{i+1}\}$

the other subsets are the same as before. Let the information gain after the operator be $gain$, if we have $gain>gain(A)$, then the operator is an optimizing operator for the partition $\coprod_{i=1}^{k}T_i$, and $\coprod_{i=1}^{k}T_i$ is a partition which can be optimized.

Definition 3. Let OP denote an optimizing operator, executing the optimizing operator $r(r\ge 1)$ times on a partition $\coprod_{i=1}^{k}T_i$ can be described as $OP_r\left(\cdots OP_2\left(OP_1\left(\coprod_{i=1}^{k}T_i\right)\right)\right)$, which is called one optimizing process. If we cannot do any optimizing operator on the result of some optimizing process, then the whole process is an optimal optimizing process for the original partition.

Theorem 1. The cutting points of an optimal optimizing process of a partition are defined on boundary points.

Proof. Let $\coprod_{i=1}^{k}T_i$ be the partition of case set T according to an arbitrary discretization of continuous attribute A. If k=1, there are none optimizing process for the partition, the conclusion is true. Otherwise provided that after an optimal optimizing process $OP_r\left(\cdots OP_2\left(OP_1\left(\coprod_{i=1}^{k} T_i\right)\right)\right)$, the cutting points are $\{v_1, v_2, \ldots, v_{h-1}\}$, and the new partition after this optimizing process is $\coprod_{i=1}^{h} T_i'$. Let v_t be the cutting point between T_t' and T_{t+1}', and v_t is not a boundary point, see Fig.2.

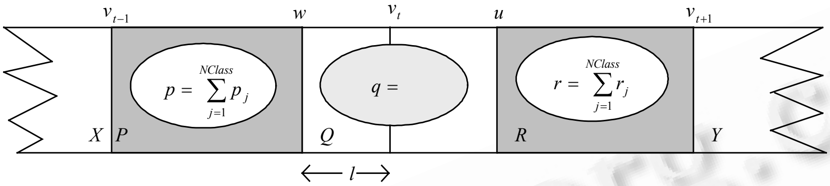


Fig.2 Boundary optimization

Let us define two further points from the sorted case sequence T. Point w is the lower border (a boundary point) of the uniform block (a block contains the cases whose values of the attribute between two adjacent boundary points) which contains v_t if the border is positioned between the thresholds v_{t-1} and v_t. Otherwise, if cut point v_{t-1} is also within the same block as v_t, we define w to be v_{t-1}. By respective logic, we define u to be either the boundary point which is the upper border of the uniform block that contains $v_t$or the threshold v_{t+1}. In any case, there are only instances of one class between the point w and u, given that this class is c. We abbreviate $freq(C_j,T)$ to be $freq(j,T)$, and define that

$$E(l) = \frac{1}{|T|}\left(|T_t'|\log_2|T_t'| + |T_{t+1}'|\log_2|T_{t+1}'|\right) - \frac{1}{|T|}\left(\sum_{j=1}^{NClass}\left(freq(j,T_t')\log_2 freq(j,T_t') + freq(j,T_{t+1}')\log_2 freq(j,T_{t+1}')\right)\right)$$

$$= \frac{1}{|T|}\left((p+l)\log_2(p+l) + (r+q-l)\log_2(r+q-l)\right) -$$

$$\frac{1}{|T|}\left((p_c+l)\log_2(p_c+l) + (r_c+q-l)\log_2(r_c+q-l) + \sum_{j\neq c}\left(p_j\log_2 p_j + r_j\log_2 r_j\right)\right).$$

The information gain of the partition $\coprod_{i=1}^{h} T_i'$ can be written as

$$gain(l) = Info(T) - \left(\sum_{i=1}^{t-1}\frac{|T_i'|}{|T|}Info(T_i') + E(l) + \sum_{i=t+2}^{h}\frac{|T_i'|}{|T|}Info(T_i')\right).$$

It has $gain'(l) = -E'(l)$. Let $gain'(l^*) = -E'(l^*) = 0$, we have

$$l^* = \frac{p\cdot(r_c+q) - p_c\cdot(r+q)}{r+q-p_c-r_c}.$$

The second derivative of $gain(l^*)$ is expressed by

$$gain''(l^*) = -E''(l^*) = -\frac{1}{|T|}\left(\frac{1}{p+l^*} - \frac{1}{p_c+l^*} + \frac{1}{r+q-l^*} - \frac{1}{r_c+q-l^*}\right),$$

which is larger than zero because $0\le l^*\le q$, in other words, $gain(l^*)$ is not a local maximum. Since l^* is chosen arbitrarily, we have shown that $gain(l)$ can only obtain its maximum value when the threshold v_t is placed at either of the points w and u, where l=0 and l=q respectively. That is, we can do an optimizing operator further on the partition $\coprod_{i=1}^{h} T_i'$, so the optimizing process $OP_r\left(\cdots OP_2\left(OP_1\left(\coprod_{i=1}^{k} T_i\right)\right)\right)$ is not an optimal optimizing process. This conflicts with the previous hypothesis and the theorem has been proved.

From the proof of theorem 1, we can know that any partitions with some cutting points not defined on the boundary points can be optimized. In our system, we use an easier optimizing process, but maybe not an optimal one, which makes all the cutting points to be their better adjacent boundary points. The boundary optimizing strategy is: checking each adjacent subsets T_i and T_{i+1}, if $val_A(s_{|Ti|,i})$ is a boundary point itself, do nothing for this boundary, and check the boundary of T_{i+1} and T_{i+2} in turn; otherwise, find the lower border w and the upper border u of the uniform block which contains $s_{|Ti|,i}$, and do boundary optimizing operator as follows:

$$COP_1: T_i' = T_i - \{s \mid val_A(s) > w, s \in T_i\}, T_{i+1}' = T_{i+1} \cup \{s \mid val_A(s) > w, s \in T_i\},$$
$$COP_2: T_i' = T_i \cup \{s \mid val_A(s) \le u, s \in T_{i+1}\}, T_{i+1}' = T_{i+1} - \{val_A(s) \le u, s \in T_{i+1}\}.$$

Suppose that the information gain of the partition after either of the above operators is $gain(COP_1)$ or $gain(COP_2)$ respectively, if $gain(COP_1) \ge gain(COP_2)$, COP_1 will be chosen as the boundary optimizing operator for the adjacent subsets T_i and T_{i+1}, otherwise COP_2 will be chosen. If the operator makes one subsets empty, the number of the subsets will reduce one. Provided that there are k intervals after the whole optimizing process, and attribute A has been selected as the expanding attribute of the current interior node D, it will split D into k branches.

4 Pruning

RCAT is a dynamic discretization method for continuous attributes. In our decision tree generation, every interior node uses the RCAT algorithm to evaluate each continuous attribute. After decision tree construction, we may get a big tree with many redundant nodes. There are two pruning methods in our system to simplify the decision tree constructed by RCAT.

4.1 Pessimistic pruning

Firstly we use pessimistic pruning[14] method to get a more reliable tree. When the original tree DT is used to classify the N cases in the training set from which it is generated, let some leaf account for K of these cases with J of them misclassified. J/K does not provide a reliable estimate of the error rate of that leaf when unseen cases are classified, since the tree is tailored to the training set. A more realistic error rate is $(J+1/2)/K$. Let ST be a subtree of DT containing $L(ST)$ leaves and let ΣJ and ΣK be the corresponding sums over the leaves of ST, then the standard error of this number of misclassification is:

$$se = \sqrt{\frac{\left(\sum J + L(ST)/2\right) \times \left(\sum K - \left(\sum J + L(ST)/2\right)\right)}{\sum K}}.$$

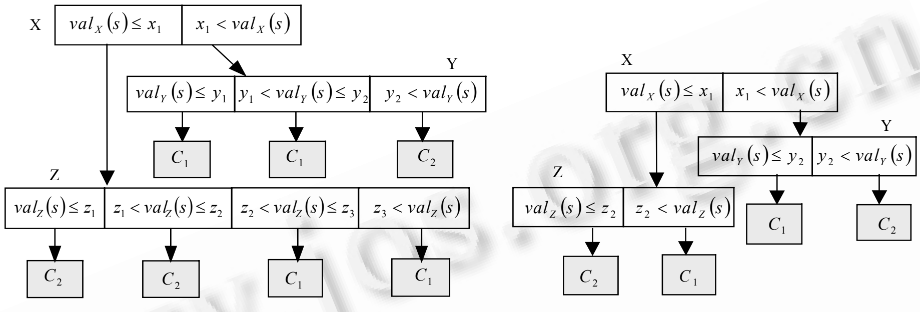
If ST is replaced by the best leaf, let E be the number of cases from the training set that it misclassifies, the pessimistic pruning method will replace ST by the best leaf whenever $E+1/2 \le (\Sigma J + L(ST)/2) + se$. All non-leaf subtrees are examined just once to see whether they should be pruned out.

4.2 Combination pruning

After pessimistic pruning, some redundant nodes remain in the decision tree if the tree has adjacent leaves signed as the same class, so a combination pruning method which doesn't influence classifying accuracy is explored to simplify the decision tree further. We give the definition of adjacent leaves as follows:

Definition 4. In the decision tree constructed by RCAT, if two leaf nodes have the same parent associated with a continuous attribute, and the intervals of the attribute for these two leaves are adjacent, we call these two nodes adjacent leaves.

Combination pruning uses depth-first method to access the nodes of decision tree from left to right. If two adjacent leaves have the same results (signed as the same class), then these two leaves should be combined into one leaf with the same class as before. The interval of this combined leaf is enlarged, and the son number of their parent is reduced by one. In Fig.3 (a), the expanding attributes of interior nodes *X*, *Y*, *Z* are *X*, *Y*, *Z* accordingly. The first two sons of the node *Y* are adjacent leaves, and should be combined into one leaf with the interval of $val_Y(s) \le y_2$. Node *Z* has two adjacent sons with the same class C_2, and two adjacent sons are signed as C_1. After combination, *Z* has only two sons. Fig.3 (b) is the pruned decision tree of Fig.3 (a).



(a) Before pruning (b) After pruning

Fig. 3 Combination Pruning

Fig.3 Combination pruning

5 Empirical Evaluation

5.1 Experiment set-up

Six data sets are selected from the UCI Irvine machine learning repository[15] with all continuous attributes and two classes. A description of data sets can be found in Table 1. For the unknown values in Breast-w and Heart data sets, we set their values to the average of all known values of the corresponding attributes.

Table 1 Description of learning tasks

Abbrev	Domain	Cases	Attributes	Classes	Default Acc (%)
Australian	Credit card application	690	14	2	55.5
Breast-w	Breast cancer (Wisc)	699	9	2	65.5
Bupa	Bupa liver-disorders	345	6	2	58.0
Heart	Heart disease	270	13	2	55.6
Ionosphere	Radar returns recognition	351	34	2	64.1
Pima	Pima Indian diabetes	768	8	2	65.1

We implement binarization and RCAT in decision tree generation. For both of the algorithms, we use information gain as the evaluation function, and adopt pessimistic pruning and combination pruning to simplify the constructed tree. Experiments are conducted by 10-fold cross validation, namely, for each data set, we use 9/10 of data as training data to build the decision tree and the rest 1/10 as test data to evaluate performance, and report the average results of these ten times.

5.2 Performance comparison

As introduced earlier, predictive accuracy and intelligibility are the most two important characteristics for a learning algorithm. We compare RCAT with binarization from the following facets: predictive error rate, number of nodes in decision tree (including interior nodes and leaf nodes), and building time. The parameters of RCAT are set to $LIMIT_R$=10 and $LIMIT_S$=10. The experiment is conducted on Pentium 4-2GHz with 256MB memory, Linux OS. Table 2 reports the empirical results. For predictive error rate and number of nodes, the results consist of mean and deviation values; for building time, we just report the average time of 10-fold cross validation.

Table 2 Results of binarization and RCAT

Task	Error rate (%)		Nodes		Time (ms)	
	Binarization	RCAT	Binarization	RCAT	Binarization	RCAT
Australian	15.80±4.61	14.06±3.28	37.4±6.5	12.5±2.8	39.1	21.9
Breast-w	5.36±2.56	4.93±1.96	13.6±3.86	9.6±1.9	14.3	13.6
Bupa	31.47±6.51	35.29±8.43	40.2±6.8	13.1±2.5	7.8	5.3
Heart	19.26±6.49	16.67±6.11	15.4±4.2	11.2±0.6	6.1	4.7
Ionosphere	10.00±3.09	11.14±5.63	19.4±4.5	8.8±4.4	62.5	36.0
Pima	26.84±3.93	25.79±3.52	70.6±13.5	25.8±5.3	42.2	29.7
Average	18.12	17.98	33	14	28.7	18.5

From Table 2, we can see that for all of the data sets, the number of nodes in the binary classification trees constructed by RCAT are much less than those built by binarization. For generalization, RCAT algorithm gets lower error rates for four data sets than those of binarization, and the average error rate is a little lower too. These results mean that RCAT algorithm can generate decision trees with higher intelligibility than binarization while retaining predictive accuracy. For building time, all of the trees constructed by RCAT need less time than those built by binarization. The reason is that RCAT needs no calculation of the information gain for every distinct values when it processes continuous attributes.

Decision tree for Breast-w dataset

```
Tree constructed by Binarization:
Cell_size <=2.0 :
  |Nuclei <=3.0 : 0 (404|2)
  |Nuclei >3.0 :
  |  |clump <=3.0 : 0 (11|0)
  |  |clump >3.0 : 1 (10|2)
Cell_size >2.0 :
  |Nuclei <=3.0 :
  |  |Cell_size <=6.0 :
  |  |  |Cell_shape <=2.0 : 0 (17|1)
  |  |  |Cell_shape >2.0 :
  |  |  |  |Mitoses <=1.0 :
  |  |  |  |  |Adhesion <=3.0 : 0 (13|3)
  |  |  |  |  |Adhesion >3.0 : 1 (7|2)
  |  |  |  |Mitoses >1.0 : 1 (4|0)
  |  |Cell_size >6.0 : 1 (21|0)
  |Nuclei >3.0 : 1 (193|9)
```

```
Tree constructed by RCAT(10/10):
Cell_size <=2.5 :
   | Nuclei <=3.5 : 0 (404|2)
   | 3.5000 <Nuclei :
   |    | clump <=3.5 : 0 (11|0)
   |    | 3.5 <clump : 1 (10|2)
 2.5 <Cell_size :
   | Nuclei <=3.5 :
   |    | Cell_size <=3.5 : 0 (25|5)
   |    | 3.5 <Cell_size : 1 (31|7)
   | 3.5 <Nuclei : 1 (193|9)
```

Decision tree for Ionosphere dataset

```
Tree constructed by Binarization:
Att5 <=0.0409 : b (67|0)
Att5 >0.0409 :
  |Att27 <=0.9999 :
  |  |Att3 <=0.0870 : b (5|0)
  |  |Att3 >0.0870 :
  |  |  |Att8 <=-0.6727 : b (4|3)
  |  |  |Att8 >-0.6727 :
  |  |  |  |Att3 <=0.7273 :
  |  |  |  |  |Att24 <=0.2263 :
  |  |  |  |  |  |Att7 <=0.8512 : g (47|1)
  |  |  |  |  |  |Att7 >0.8512 : b (2|0)
  |  |  |  |  |Att24 >0.2263 : b (5|3)
  |  |  |  |Att3 >0.7273 : g (159|2)
  |Att27 >0.9999 :
  |  |Att1 <=0.0000 : b (19|0)
  |  |Att1 >0.0000 :
  |  |  |Att3 <=0.6616 : b (9|0)
  |  |  |Att3 >0.6616 :
  |  |  |  |Att8 <=-1.0000 : b (6|0)
  |  |  |  |Att8 >-1.0000 :
  |  |  |  |  |Att16 <=0.7253 : g (13|2)
  |  |  |  |  |Att16 >0.7253 : b (4|0)
```

```
Tree constructed by RCAT(10/10):
Att5 <=0.0414 : b (67|0)
 0.0414 <Att5 :
   | Att6 <=-0.8925 : b (17|0)
   | -0.8925 <Att6 <=0.7955 : g (219|25)
   | 0.7955 <Att6 :
   |    | Att11 <=-0.2194 : g (6|1)
   |    | -0.2194 <Att11 : b (16|0)
```

Fig.4 Tree constructed by binarization and RCAT(10/10)

Decision tree can be used to describe knowledge directly. Figure 4 gives the decision trees learned from all of the cases in Breast-w and Ionosphere datasets respectively. We can see that the trees built by RCAT are much smaller than those generated by binarization, and the knowledge represented by them can be understood more easily.

6 Conclusions

Aiming at two-class problems, we explore a discretization method named RCAT for continuous attributes. It uses binarization to get the multisplitting result through mapping a continuous attribute into another probability attribute based on statistic information. Applying RCAT algorithm to a decision tree generation, we can get a much smaller tree with higher intelligibility using less building time than binarization while retaining predictive accuracy.

References:

[1] Domingos P. The role of occam's razor in knowledge discovery. Data Mining and Knowledge Discovery, 1999,3(4):409~425.

[2] Zhou ZH, Chen SF. Rule extraction from neural networks. Journal of Computer Research and Development, 2002,39(4):398~405 (in Chinese with English abstract).

[3] Setiono R, Liu H. Extracting rules from neural networks by pruning and hidden-unit splitting. Neural Computation, 1997,9(1): 205~225.

[4] Boz O. Converting a trained neural network to a decision tree: DECTEXT-decision tree extractor [Ph.D. Thesis]. Pennsylvania: University of Lehigh, 2000.

[5] Windeatt T, Ardeshir G. An empirical comparison of pruning methods for ensemble classifiers. In: Hoffmann F, Hand DJ, eds. Proceedings of the 4th International Conference on Intelligent Data Analysis. Lisbon: Springer-Verlag, 2001. 208~217.

[6] Fournier D, Crémilleux B. A quality index for decision tree pruning. Knowledge-Based System, 2002,15(1):37~43.

[7] Subramani M, William RS, Malcolm BD, Michael JP. Two-Stage machine learning model for guideline development. Artificial Intelligence in Medicine, 1999,16(1):51~72.

[8] Auer P, Holte RC, Maass W. Theory and application of agnostic PAC-learning with small decision trees. In: Prieditis A, Russell S, eds. Proceedings of the 12th International Conference on Machine Learning. San Francisco: Morgan Kaufmann Publishers, Inc., 1995. 21~29.

[9] Langley P. Induction of condensed determinations. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining. Portland: AAAI Press, 1996. 327~330.

[10] Quinlan JR. C4.5: Programs for Machine Learning. San Mateo: Morgan Kaufmann Publishers, Inc., 1993.

[11] Elomaa T, Rousu J. General and efficient multisplitting of numerical attributes. Machine Learning, 1999,36(3):201~244.

[12] Dougherty J, Kohavi R, Sahami M. Supervised and unsupervised discretization of continuous feature. In: Prieditis A, Russell S, eds. Proceedings of the 12th International Conference on Machine Learning. San Francisco: Morgan Kaufmann Publishers, Inc., 1995. 194~202.

[13] Liu H, Hussain F, Tan CL, Dash M. Discretization: An enabling technique. Data Mining and Knowledge Discovery, 2002,6(4): 393~423.

[14] Quinlan JR. Simplifying decision tree. International Journal of Man-Machine Studies, 1987,27:221~234.

[15] Bay SD. UCI KDD Archive. 1999. http://kdd.ics.uci.edu.

附中文参考文献:

[2] 周志华,陈世福.神经网络规则抽取.计算机研究与发展,2002,39(4):398~405.