Available online at www.sciencedirect.com

ScienceDirect

Information Sciences 178 (2008) 714–731

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

A discretization algorithm based on Class-Attribute Contingency Coefficient

Cheng-Jung Tsai [a,*], Chien-I. Lee [b], Wei-Pang Yang [c]

[a] *Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC*
[b] *Department of Information and Learning Technology, National University of Tainan, Tainan, Taiwan, ROC*
[c] *Department of Information Management, National DongHwa University, Hualien, Taiwan, ROC*

Received 27 September 2006; received in revised form 24 August 2007; accepted 2 September 2007

Abstract

Discretization algorithms have played an important role in data mining and knowledge discovery. They not only produce a concise summarization of continuous attributes to help the experts understand the data more easily, but also make learning more accurate and faster. In this paper, we propose a static, global, incremental, supervised and top-down discretization algorithm based on Class-Attribute Contingency Coefficient. Empirical evaluation of seven discretization algorithms on 13 real datasets and four artificial datasets showed that the proposed algorithm could generate a better discretization scheme that improved the accuracy of classification. As to the execution time of discretization, the number of generated rules, and the training time of C5.0, our approach also achieved promising results.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Data mining; Classification; Decision tree; Discretization; Contingency coefficient

1. Introduction

With the rapid development of information technology, electronic storage devices are widely used to record transactions. Since people are often unable to extract useful knowledge from such huge datasets, data mining [16] has become a research focus in recent years. Among the several functions of data mining, classification is crucially important and has been applied successfully to several areas such as automatic text summarization and categorization [17,38], image classification [15], and virus detection of new malicious emails [31]. Although real-word data mining tasks often involve continuous attributes, some classification algorithms such as AQ [18,26], CLIP [6,7] and CN2 [8] can only handle categorical attribute, while others can handle continuous attributes but would perform better on categorical attributes [36]. To deal with this problem, a lot of discretization algorithms have been proposed [11,12,22,28].

* Corresponding author.
E-mail addresses: tsaicj@cis.nctu.edu.tw (C.-J. Tsai), leeci@mail.nutn.edu.tw (C.-I. Lee), wpyang@mail.ndhu.edu.tw (W.-P. Yang).

0020-0255/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2007.09.004

Discretization is a technique to partition continuous attributes into a finite set of adjacent intervals in order to generate attributes with a small number of distinct values. Assuming that a dataset consisting of M examples and S target classes, a discretization algorithm would discretize the continuous attribute A in this dataset into n discrete intervals $\{[d_0, d_1], (d_1, d_2], \ldots, (d_{n-1}, d_n]\}$, where d_0 is the minimal value and d_n is the maximal value of attribute A. Such a discrete result $\{[d_0, d_1], (d_1, d_2], \ldots, (d_{n-1}, d_n]\}$ is called a *discretization scheme D* on attribute A. This discretization scheme should keep the high interdependency between the discrete attribute and the target class to carefully avoid changing the distribution of the original data [2,25,33].

Discretization is usually performed prior to the learning process and has played an important role in data mining and knowledge discovery. The modern classification systems such as CLIP4 [7] had also implemented some discretization algorithms as built-in functions. A good discretization algorithm not only can produce a concise summarization of continuous attributes to help the experts and users understand the data more easily, but also make learning more accurate and faster [24]. There are five different axes by which the proposed discretization algorithms can be classified [24]: *supervised* versus *unsupervised*, *static* versus *dynamic, global* versus *local, top-down* (splitting) versus *bottom-up* (merging), and *direct* versus *incremental.*

1. Supervised methods discretize attributes with the consideration of class information, while unsupervised methods do not.
2. Dynamic methods consider the interdependence among the features attributes and discretize continuous attributes when a classifier is being built. On the contrary, the static methods consider attributes in an isolated way and the discretization is completed prior to the learning task.
3. Global methods, which use total instances to generate the discretization scheme, are usually associated with static methods. On the contrary, local methods are usually associated with dynamic approaches in which only parts of instances are used for discretization.
4. Bottom-up methods start with the complete list of all continuous values of the attribute as cut-points and remove some of them by merging intervals in each step. Top-down methods start with an empty list of cut-points and add new ones in each step.
5. Direct methods, such as Equal Width and Equal Frequency [5], require users to decide on the number of intervals k and then discretize the continuous attributes into k intervals simultaneously. On the other hand, incremental methods begin with a simple discretization scheme and pass through a refinement process although some of them may require a stopping criterion to terminate the discretization.

In recent years, many researchers put their attentions on developing the dynamic discretization algorithms for some particular learning algorithms. For example, Berzal et al. [14] built multi-way decision trees by using a dynamic discretization method in each internal node to reduce the size of the resulting decision trees. Their experiments showed that the accuracy of these compact decision trees was also preserved. Wu et al. [36] defined a distributional index and then proposed a dynamic discretization algorithm to enhance the decision accuracy of naïve Bayes classifiers. However, the advantage of static approaches as opposed to dynamic approaches is the independence from the learning algorithms [24]. In other words, a dataset discretized by a static discretization algorithm can be used in any classification algorithms that deal with discrete attributes. Besides, since the bottom-up method starts with the complete list of all continuous values of the attribute as cut-points, and then remove some of them by merging intervals in each step, its computational complexity is usually worse than the top-down method. For example, the time complexity for discretizing a single attribute in Extended Chi2, which is the newest bottom-up method, is $O(km\log m)$ [33], while that of the newest top-down method CAIM is $O(m\log m)$ [21], where m is the number of distinct values of the discretized attribute and k is the number of incremental steps. This condition will get worse when the difference between the number of values in a continuous attribute and the number of produced intervals is large. Supposing that a continuous attribute contains 1000 different values and this attribute is discretized into 50 intervals, in general, a top-down approach requires only 50 steps, but a bottom-up approach would need 950 steps. Finally, supervised discretization algorithms are expected to lead to better performance as compared to unsupervised ones since they take the class information into account. Based on the above-mentioned reasons, we aimed at developing a static, global, incremental, supervised and top-down discretization algorithm. For the rest of the present paper, our discussion of proposed discretization algorithms will follow the axis of top-down versus bottom-up. More detailed discussions about the five axes can be found in [24].

CAIM is the newest top-down discretization algorithm. In comparison with six state-of-the-art top-down discretization algorithms, experiments showed that on the average, CAIM can generate a better discretization scheme. These experiments also showed that a classification algorithm, which uses CAIM as a preprocessor to discretize the training data, can on the average, produce the least number of rules and reach the highest classification accuracy [21]. However, the general goals of a discretization algorithm should be: (a) generating a high quality discretization scheme to help the experts understand the data more easily (the quality of a discretization scheme can be measured by*CAIR criterion* which is discussed in Section 2); (b) the generated discretization scheme should lead to the improvement of accuracy and the efficiency of a learning algorithm (for a decision tree algorithm, the efficiency is evaluated by the number of rules and training time); and, (c) the discretization process should be as fast as possible. Although CAIM outperforms the other top-down methods in these aspects, it still has two drawbacks. First of all, CAIM algorithm gives a high factor to the number of generated intervals when it discretizes an attribute. Thus, CAIM usually generate a simple discretization scheme in which the number of intervals is very close to the number of target classes. Secondly, for each discretized interval, CAIM considers only the class with the most samples and ignores all the other target classes. Such a consideration would decrease the quality of the produced discretization scheme in some cases. The two observations motivated us to propose our Class-Attribute Contingency Coefficient (CACC) discretization algorithm. The detailed discussions and examples about CAIM were presented in Section 2.3.

CACC is inspired by the *contingency coefficient*. The main contribution of CACC is that it can generate a better discretization scheme (i.e., higher *cair* value) and its discretization scheme can lead to the improvement of classifier accuracy like that of C5.0. As regards to the time complexity of discretization, the number of generated rules and the execution time of a classifier, our approach also achieved promising results. The rest of the paper is organized as follows. In Section 2, we review some related works. Section 3 presents our Class-Attribute Contingency Coefficient discretization algorithm. The experimental comparisons of seven discretization algorithms on 13 real datasets and four artificial datasets and a further evaluation of CACC are presented in Section 4. Finally, the conclusions are presented in Section 5.

2. Related works

In this section, we review some of the related works. Since we evaluated the performance of several discretization algorithms in Section 4 by using the famous classification algorithm C5.0, we first gave a brief introduction of classification in Section 2.1. Then, we reviewed the proposed discretization algorithms on the axis of top-down versus bottom-up in Section 2.2. Finally, the detailed discussions of CAIM were given in Section 2.3.

2.1. Classification

In the field of classification, there are many branches that are developing *decision trees* [9],*bayesian classification* [37], *neural networks* [3], and *genetic algorithms* [32]. Among them, the decision tree has become a popular tool for several reasons [30]: (a) compared to neural networks or a bayesian based approach, it is more easily interpreted by humans; (b) it is more efficient for large training data than neural networks which would require a lot of time on thousands of iterations; (c) a decision tree algorithm does not require a domain knowledge or prior knowledge; and, (d) it displays good classification accuracy as compared to other techniques. A decision tree like C5.0 [29] is a flow-chart-like tree structure, which is constructed by a recursive divide-and-conquer algorithm that generates a partition of the data. In a decision tree, each *internal node* denotes a test on an attribute, each branch represents an outcome of the test, and each *leaf node* is associated with a *target class* (or *class*). The topmost node in a tree is called the *root*, and each path forming from the root to a leaf node represents a *rule*. Classifying an unknown example begins with the root node, and successive internal nodes are visited until this example has reached a leaf node. Then the class of this leaf node is assigned to this example as a prediction.

2.2. Discretization algorithms

Proposed discretization algorithms can be divided into top-down versus bottom-up, while the top-down methods can be further divided into *unsupervised* versus *supervised* [24]. Famous unsupervised top-down algorithms are Equal Width and Equal Frequency [5], while the state-of-the-art supervised top-down algorithms include Paterson–Niblett [27], maximum entropy [35], CADD (Class-Attribute Dependent Discretizer algorithm) [4], Information Entropy Maximization [13], Class-attribute Interdependence Maximization (CAIM) [21], and Fast Class-attribute Interdependence Maximization (FCAIM) [20]. Experiments in [21] showed that CAIM discretization algorithm is superior to the other top-down discretization algorithms since its discretization schemes can generally maintain the highest interdependence between target class and discretized attributes, result to the least number of generated rules, and attain the highest classification accuracy. FCAIM [20], which is an extension of CAIM algorithm, have been proposed to speed up CAIM. The main framework, including the discretization criterion and the stopping criterion, as well as the time complexity between CAIM and F-CAIM are all the same. The only difference is the initialization of the boundary point in two algorithms. Compared to CAIM, F-CAIM was faster and had a similar C5.0 accuracy, but obtained a slightly worse *cair* value. Since the main goal of our approach is to reach a higher *cair* value and attain an improvement in the accuracy of classification, we compared our approach to CAIM instead of F-CAIM in our experiments. Of course, CACC can be easily extended to F-CACC with the same considerations as in F-CAIM if the readers consider that a faster discretization is more important than the quality of a discretization scheme.

In the bottom-up branch, famous algorithms include ChiMerge [19], Chi2 [23], Modified Chi2 [34] and Extended Chi2 [33]. The computational complexity of bottom-up methods is usually larger than top-down ones, since they start with the complete list of all the continuous values of the attribute as cut-points and then removing some of them by merging intervals in each step. Another common characteristic of these methods is in the use of the significant test to check if two adjacent intervals should be merged. ChiMerge [19] is the most typical bottom-up algorithm. In addition to the problem of high computational complexity, the other main drawback of ChiMerge is that users have to provide several parameters during the application of this algorithm that include the significance level as well as the maximal and minimal intervals. Hence, Chi2 was proposed based on the ChiMerge. Chi2 improved the ChiMerge by automatically calculating the value of the significance level. However, Chi2 still requires the users to provide an inconsistency rate to stop the merging procedure and does not consider the freedom which would have an important impact on the discretization schemes. Thereafter, Modified Chi2 takes the freedom into account and replaces the inconsistency checking in Chi2 by the quality of approximation after each step of discretization. Such a mechanism makes the Modified Chi2 a completely automated method to attain a better predictive accuracy than Chi2. After the Modified Chi2, the Extended Chi2 takes into consideration that the classes of instances often overlap in the real world. Extended Chi2 determines the predefined misclassification rate from the data itself and considers the effect of variance in two adjacent intervals. With these modifications, Extended Chi2 can handle an uncertain dataset. Experiments on these bottom-up approaches by using C5.0 also showed that the Extended Chi2 outperformed the other bottom-up discretization algorithms since its discretization scheme, on the average, can reach the highest accuracy [33].

2.3. CAIM discretization algorithm and CAIR criterion

Given the two-dimensional *quanta matrix* (also called a *contingency table*) in Table 1, CAIM defined the interdependency between the target class and the discretization scheme of a continuous attribute A as

$$caim = \frac{\sum_{r=1}^{n} \frac{max_r^2}{M_{+r}}}{n}, \tag{1}$$

where q_{ir} $(i = 1, 2, \ldots, S, r = 1, 2, \ldots, n)$ denotes the total number of examples belonging to the ith class that are within interval $(d_{r-1}, d_r]$, M_{i+} is the total number of examples belonging to the ith class, M_{+r} is the total number of examples that are within the interval $(d_{r-1}, d_r]$, n is the number of intervals, max_r is the maximum value among all q_{ir} values, and M_{+r} is the total number of continuous values of attribute A that are within the interval $(d_{r-1}, d_r]$.The larger the value of *caim* is, the better the generated discretization scheme D will be. It is worth

Table 1
The quanta matrix for attribute F and discretization scheme D

Class	Interval $[d_0,d_1]\ldots(d_{r-1},d_r]\ldots(d_{n-1},d_n]$	Sum of class
C_1	$q_{11}\cdots q_{1r}\cdots q_{1n}$	M_{1+}
:	$:\ldots:\ldots:$	:
C_i	$q_{i1}\cdots q_{ir}\cdots q_{in}$	M_{i+}
:	$:\ldots:\ldots:$	:
C_s	$q_{s1}\cdots q_{sr}\cdots q_{sn}$	M_{s+}
Sum of intervals	$M_{+1}\ldots M_{+r}\ldots M_{+n}$	M

noting that in order to generate a simpler discretization scheme, $\sum_{r=1}^{n}\frac{max_r^2}{M_{+r}}$ is divided by the number of intervals $n$in Formula 1.

CAIM is a progressing discretization algorithm that does not require users to provide any parameter. For a continuous attribute, CAIM will test all possible cutting points and then generate one in each loop. The loop is stopped until a specific condition is met. For each possible cutting point in each loop, the corresponding *caim* value is computed according to Formula 1, and the one with the highest *caim* value is chosen. Since finding the discretization scheme with the globally optimal *caim* value would require a lot computation cost, CAIM algorithm only finds a local maximum *caim* to generate a sub-optimal discretization scheme.

In the experiments, CAIM adopts the CAIR criterion [4,35] as shown in Formula 2 to evaluate the quality of a generated discretization scheme. The CAIR criterion is used in the CADD algorithm. CADD has several disadvantages, such as the need for a user-specified number of intervals and requires training for the selection of a confidence interval. Experimental results also showed that the CAIR criterion is not a good discretization formula since it can suffer from the overfitting problem [21]. However, the CAIR criterion can effectively represent the interdependency between the target class and discretized attributes, and thus, is widely used to measure the quality of a discretization scheme.

$$cair = \sum_{i=1}^{s}\sum_{r=1}^{n} p_{ir}\log_2\frac{p_{ir}}{p_{i+}p_{+r}} \Bigg/ \sum_{i=1}^{s}\sum_{r=1}^{n} p_{ir}\log_2\frac{1}{p_{ir}}, \tag{2}$$

where $p_{ir}=\frac{q_{ir}}{M}$, $p_{i+}=\frac{M_{i+}}{M}$, and $p_{+r}=\frac{M_{+r}}{M}$ in Table 5.

Although CAIM outperformed the other top-down methods, it still has two drawbacks. In the first place, CAIM gives a high factor to the number of generated intervals when it discretizes an attribute. Hence, CAIM usually generates a simple discretization scheme in which the number of intervals is very close to the number of

Table 2
Age dataset

ID	Age	Target class
1	3	Care
2	5	Care
3	6	Care
4	15	Edu
5	17	Edu
6	21	Edu
7	35	Work
8	45	Work
9	46	Work
10	51	Edu
11	56	Edu
12	57	Edu
13	66	Care
14	70	Care
15	71	Care

target classes. For example, if we take the age dataset in Table 2 as the training data, the discretization scheme of CAIM is presented in Table 3. In Table 3 CAIM divided the age dataset into three intervals: $[3.00, 10.50]$, $(10.50, 61.50]$, and $(61.50, 71.00]$. Interval $[3.00, 10.50]$ contains samples 1–3, interval $(10.50, 61.50]$ contains samples 4–12, and interval $(61.50, 71.00]$ has samples 13–15. However, this discrete result is not good and the age dataset should obviously be discretized into five intervals: samples 1–3, 4–6, 7–9, 10–12, and 13–15. If a classifier is learning with such a discretized dataset produced by CAIM, the accuracy would be worse. Secondly, CAIM considers only the distribution of the major target class. Such a consideration is also unreasonable in some cases. Take Table 4 as an example, for the interval I_1 of both datasets D_{31} and D_{32}. Since the CAIM discrete formula uses only the five samples belonging to target class C_1 to compute the *caim* value (the two samples with class C_2 and the three samples with class C_3 are ignored), the two datasets have the same *caim* value in spite of the different data distribution. Such an unreasonable condition also occurs when the CAIR criterion is considered. As shown in Table 5, the two datasets D_{41} and D_{42} have the same *caim* value even if their *cair* values are different.

Table 3
The discretization scheme of age dataset by CAIM

Class	Interval			*sum*
	$[3.00, 10.50]$	$(10.50, 61.50)$	$(61.50, 71.00]$	
Care	3	0	3	6
Edu	0	6	0	6
Work	0	3	0	3
Sum	3	9	3	15

Table 4
Two datasets with equal *caim* values but different data distribution

Class	Interval		*sum*
	I_1	I_2	
Dataset D_{31}: $caim(I_1) = caim(I_2) = 2.5$			
C_1	5	5	10
C_2	2	3	5
C_3	3	2	5
sum	10	10	20
Dataset D_{32}: $caim(I_1) = caim(\mathrm{I}_2) = 2.5$			
C_1	5	5	10
C_2	1	4	5
C_3	4	1	5
Sum	10	10	20

Table 5
Two datasets with equal *caim* value but different *cair* values

Class	Interval		*sum*
	I_1	I_2	
Dataset D_{41}: $caim(I_1) = caim(I_2) = 5$; $cair(I_1) = cair(I_2) = 0$.			
C_1	5	5	10
C_2	0	0	0
sum	5	5	10
Dataset $D_{42:}$ $caim(I_1) = caim(I_2) = 5$; $cair(I_1) = cair(I_2) = 1$.			
C_1	5	0	5
C_2	0	5	5
sum	5	5	10

3. Class-Attribute Contingency Coefficient discretization algorithm

As stated in the Introduction, a good discretization algorithm should generate a discretization scheme which maintains a high interdependence between the target class and the discretized attribute. As described in Section 2.3, CAIM gives a high factor to the number of generated intervals and does not consider the data distribution, whereas CDD can suffer from the overfitting problem. Thus, both methods could generate irrational discrete results in some cases. Let us review the age dataset in Table 3, wherein CAIM discretizes samples 4 to 6 (with class "Edu"), samples 7 to 9 (with class "Work"), and samples 10–12 (with class "Edu") into the same interval. Such a result seriously changes the original data distribution. However, if we discretize the 15 samples into 15 intervals to actually represent the distribution of the original dataset, there will be an overfitting. In summary, a discrete formula should not only avoid overfitting but also consider the distribution of all samples to generate an ideal discretization scheme.

Given the quanta matrix in Table 1, researchers usually use the *contingency coefficient* as shown in Formula 3 to measure the strength of dependence between the variables.

$$C = \sqrt{\frac{y}{y+M}}, \tag{3}$$

where $y = M\left[\left(\sum_{i=1}^{S}\sum_{r=1}^{n}\frac{q_{ir}^2}{M_{i+}M_{+r}}\right) - 1\right]$, M is the total number of samples, n is the number of intervals, q_{ir} is the number of samples with class i ($i = 1,2,\ldots,S$, and $r = 1,2,\ldots,n$) in the interval $(d_{r-1}, d_r]$, M_{i+} is the total number of samples with class i, and M_{+r} is the total number of samples in the interval $(d_{r-1}, d_r]$. From Formula 3, we can see that the contingency coefficient indeed takes the distribution of all samples into account by using $[(q_{ir})^2/\ M_{i+}M_{+r}]$. In other words, if we regard the target class and discretized attribute as two variables, the contingency coefficient is a very good criterion to measure the interdependence between them. However, in the present paper we do not directly use the *contingency coefficient* C but instead, we divide y by $\log(n)$ and define the *cacc* value as

$$cacc = \sqrt{\frac{y'}{y'+M}}, \tag{4}$$

where $y' = M\left[\left(\sum_{i=1}^{S}\sum_{r=1}^{n}\frac{q_{ir}^2}{M_{i+}M_{+r}}\right) - 1\right]/\log(n)$.

We divide the y by $\log(n)$ for two main reasons: (a) speed up the discretization process; (b) as described in the first paragraph of Section 3, a discretization scheme containing too many intervals could suffer from an overfitting problem. In fact, CAIM also took these reasons into account, so that in the CAIM criterion in Eq. (1), the summed value was divided by the number of intervals n. However, as described in the example in Table 2, CAIM makes its discretization schemes unreasonable due to the huge influence of variable n. In our experiments in Section 4, we can also find that CAIM almost always generate a discretization scheme in which the number of intervals is very close to the number of target classes. Hence, we use $\log(n)$ in Eq. (4) instead of n to reduce its influence.

With Formula 4, we can now detail our Class-Attribute Contingency Coefficient (CACC) discretization algorithm. The pseudo-code of CACC is shown in Fig. 1. Given a dataset with i continuous attributes, M examples, and S target classes, for each attribute A_i, CACC first finds the maximum d_n and minimum d_0 of A_i in Line 4 and then forms a set of all distinct values of A_i in the ascending order in Line 5. As a result, all possible interval boundaries B with the minimum and the maximum, and all the midpoints of all the adjacent boundaries in the set are obtained in Lines 6 and 7. Then, CACC would iteratively partition the attribute A_i from Line 10 to Line 18. In the kth loop, CACC would compute for all possible cutting points to find the one with the maximum *cacc* value and then partition this attribute accordingly into $k+1$ intervals. In order to reduce the computation cost of discretization, CACC also uses a greedy method as in CAIM to generate the sub-optimal discretization scheme. In other words, for every loop, CACC not only finds the best division point but also records a *Globalcacc* value. If the generated *cacc* value in loop $k+1$ is less than the *Globalcacc* obtained in loop k, CACC would terminate and output the discretization scheme. Besides, to generate a rational discrete result, such a greedy mechanism is ignored if the number of generated intervals is less than

```
1  Input:  Dataset with $i$ continuous attribute, $M$ examples and $S$ target classes;
2  Begin
3     For each continuous attribute $A_i$
4        Find the maximum $d_n$ and the minimum $d_0$ values of $A_i$;
5        Form a set of all distinct values of $A$ in ascending order;
6        Initialize all possible interval boundaries $B$ with the minimum and maximum
7        Calculate the midpoints of all the adjacent pairs in the set;
8        Set the initial discretization scheme as $D$: $\{[d_0,d_n]\}$and $Globalcacc = 0$;
9        Initialize $k = 1$;
10           For each  inner boundary  $B$ which is not already in  scheme $D$,
11                Add it into $D$;
12                Calculate the corresponding $cacc$ value;
13           Pick up the scheme $D'$ with the highest $cacc$ value;
14           If $cacc > Globalcacc$ or $k < S$  then
15               Replace $D$ with D';
16               $Globalcacc = cacc$;
17               $k = k + 1$;
18               Goto Line 10;
18           Else
19              $D' = D$;
20           End If
21     Output the Discretization scheme $D'$  with $k$ intervals for continuous attribute $A_i$;
22  End
```

Fig. 1. The pseudo-code of CACC.

the number of target classes. Since the main framework of CACC is similar to that of CAIM, the complexity of CACC for discretizing a single attribute is still O($m\log(m)$), where m is the number of distinct values of the discretized attribute.

Note that, the main goal and contribution of CACC is to propose a criterion to generate better discretization schemes that can lead to the improvement of accuracy of a learning algorithm. In order to make the readers easily understand the difference between CACC and CAIM, we did not obviously use different pseudo-codes to confuse the readers. Similar conditions had occurred in the research field of discretization algorithms. For example, the pseudo-code of Chi2 is similar to that of ChiMerge since the former only adds a procedure that automatically calculates the significance level. The pseudo-code of Modified Chi2 is also similar to that of Chi2 except that the Modified Chi2 replaces the inconsistency checking criterion in Chi2 with its approximation measurement.

To clearly explain the process of our CACC algorithm, we again use the age dataset in Table 2 as the example. First, CACC finds the minimum ($d_0 = 3$) and maximum ($d_n = 71$) of the age attribute, and then sorts all values in ascending order. The *Globalcacc* is set to 0 as default. In the first loop, CACC gets the cutting point for which the maximum *cacc* (=0.5045) is age = 10.50. Since 0.5045 > *Globalcacc* (=0), CACC updates the *Globalcacc* = 0.5045 and runs the second loop. At this point, the attribute age is discretized into two intervals: [3.00, 10.50] and (10.50, 71]. Similarly, CACC generates the second cutting point at 61.50 and its corresponding *cacc* (=0.6473) > *Globalcacc* (=0.5045), so that *Globalcacc* is updated to 0.6473 and the third loop is processed. CACC continues to follow the same process for the third cutting point (age = 28.00) with the corresponding *Globalcacc* = 0.6612, and for the fourth cutting point (age = 48.50) with the corresponding

Table 6
The discrete result for the age dataset in every loop

# of loop	# of intervals	Cutting point	Maximum *cacc*
1	2	10.50	0.5045
2	3	61.50	0.6473
3	4	28.00	0.6612
4	5	48.50	0.7263

Table 7
The discretization scheme of the age dataset by CACC

Class	Interval					Total
	[3.00, 10.50]	(10.50, 28.00]	(28.00, 48.50]	(48.50, 61.50]	(61.50, 71.00]	
Care	3	0	0	0	3	6
Edu	0	3	0	3	0	6
Work	0	0	3	0	0	3
Total	3	3	3	3	3	15

Globalcacc = 0.7263. However, in the fifth loop, the maximum *cacc* generated is less than *Globalcacc* = 0.7263 and thus, CACC terminates. The discrete result and the corresponding *cacc* of the age dataset are detailed in Table 6. Table 7 is the final discrete result for the age dataset. We find CACC groups ages 15, 17, 21 in interval (10.50, 28.00], ages 35, 45, 46 in interval (28.00, 48.50], and ages 51, 56, 57 in interval (48.50, 61.50]. This result is obviously much more reasonable than that generated by CAIM in Table 2.

4. Performance analysis

In this section, we compare the following seven discretization algorithms in Microsoft Visual C++ 6.0 for performance analysis.

1. Equal Width and Equal Frequency: two typical unsupervised top-down methods;
2. CACC: the method proposed in this paper;
3. CAIM: the newest top-down method;
4. IEM: a famous and widely used top-down method;
5. ChiMerge: a typical bottom-up method;
6. Extended Chi2: the newest bottom-up approach.

Among the seven discretization algorithms, Equal Width, Equal Frequency and ChiMerge require the user to specify in advance some parameters of discretization. For the ChiMerge algorithm, we set the level of significance to 0.95. For the Equal Width and Equal Frequency methods, we adopted the heuristic formula used in CAIM to estimate the number of discrete interval [21,35]. All experiments were run on a PC equipped with Windows XP operating system, Pentium IV 1.8 GHz CPU, and 512mb SDRAM memory.

Our experimental data includes 13 UCI real datasets and four artificial datasets. As regards to the 13 UCI real dataset, seven of them were used in CAIM and the rest were gathered from the U.C. Irvine repository [1]. The details of the 13 UCI experimental datasets are listed in Table 8. In order to further analyze the

Table 8
The summary of 13 UCI real datasets in our experiments

Dataset	Number of continuous attributes	Number of attributes	Number of classes	Number of examples
breast	9	9	2	699
bupa	6	6	2	345
glass	10	10	6	214
hea	6	13	2	270
ion	32	34	2	351
iris	4	4	3	150
optdigit	64	64	10	5620
page-blocks	10	10	5	5473
pendigit	16	16	10	10992
pid	8	8	2	768
sat	36	36	6	6435
thy	6	21	3	7200
wav	21	21	3	5000

performance of CACC, we also encoded a program to generate four artificial datasets. The details of the artificial datasets are introduced in Section 4.3. The 10-fold cross-validation test method was applied to all experimental datasets. In other words, each dataset was divided into ten parts of which nine parts were used as training sets and the remaining one as the testing set. The discretization was done using the training sets and the testing sets were discretized using the generated discretization scheme. In addition, we also used C5.0 [29] to evaluate the generated discretization schemes. In our experiments, C5.0 was chosen since it was conveniently available and widely used as a standard for comparison in machine learning literature. Finally as suggested by Demsar [10], we used the Friedman test and the Holm's post-hoc tests with significance level $\alpha = 0.05$ to statistically verify the hypothesis of improved performance.

4.1. The comparison of discretization schemes

In this section we used the seven discretization algorithms to discretize the 10-fold training sets of each dataset in Table 8. The comparisons of the generated discretization schemes are shown in Table 9. Due to the content limit, we only showed for each dataset the mean of *cair* value, the mean of execution time and the mean number of discrete intervals. Quick comparisons of the seven methods can be obtained by checking the mean ranks in the last column in Table 9. With this column, we then used the Friedman test to check if the measured mean ranks reached statistically significant differences. If the Friedman test showed that there was a significant difference, the Holm's post-hoc test was used to further analyze the comparisons of all the methods against CACC. Although we also showed the number of discrete intervals in this experiment, it was not our main concern. Recall that in the Introduction, we stated that the general goals of a discretization algorithm should be: (a) generate a discretization scheme with a higher *cair* value; (b) the generated discretization scheme should lead to the improvement of accuracy and efficiency of a learning algorithm; and, (c) the discretization process should be as fast as possible. A discretization scheme with fewer intervals may not only lead to a worse quality of discretization scheme and a decrease in the accuracy of a classifier, but also increase the produced rules in a classifier. This condition was demonstrated in next sub-section using C5.0.

The comparison results in Table 9 showed that on the average, CACC reached the highest *cair* value from among the seven discretization algorithms. This was a very exhilarating result that demonstrated that the CACC criterion can indeed produce a high quality discretization scheme. In order to obtain the statistical support, the Friedman and the Holm's post-hoc test was used. The corresponding value of Friedman test was 58.714 (p-value < 0.0001), which was larger than the threshold 12.592. The visualizations of the Holm's post-hoc test are illustrated in Fig. 2. In Fig. 2 the top line in the diagram is the axis on which we plotted the average ranks of all the methods while a method on the right side means that it performs better. A method with rank outside the marked interval in Fig. 2 means that it is significantly different from CACC. From Fig. 2a we can see that the mean *cair* of CACC was statistically comparable to that of CAIM and significantly better than that of all the other five methods. The comparison between CAIM and CACC did not achieve significant difference since we compared all seven algorithms. If we removed the two unsupervised algorithms from this comparison, we can obtain Fig. 2b in which CACC performed significantly better than all of the other four methods. It is also worth noting that although we only showed the mean *cair* in the present paper, for all of the 228 continuous attributes in Table 8, the *cair* value of CACC is always equal to or better than that of CAIM.

Regarding the number of discrete intervals, on the average CAIM generated the least number of intervals. This result was not surprising since CAIM usually generated a simple discretization scheme in which the number of intervals was very close to the number of classes. The corresponding value of Friedman test was 8.192 (p-value = 0.228), which was smaller than the threshold 12.592, and meant that there were no significant differences among the number of generated intervals of the seven algorithms. However, if we removed the two unsupervised algorithms, in which the number of generated intervals was decided in advance, from this comparison, the Friedman test reached statistical significance and we obtained Fig. 2c. From Fig. 2c, we can see that the generated number of intervals of CACC was significantly less than that of ChiMerge and comparable to that of CAIM, IEM and Extended chi2.

Finally, the two unsupervised methods were the fastest since they did not consider the processing of any class related information. The discretization time of CACC was a little longer than that of CAIM but the

Table 9
The comparison of discretization schemes on UCI real datasets

Criterion	Algorithm	Dataset													Mean rank
		breast	bupa	glass	hea	ion	iris	optdigit	page-blocks	pendigit	pid	sat	thy	wav	
Mean cair value	Equal-W	0.198	0.013	0.156	0.076	0.084	0.376	0.053	0.042	0.116	0.051	0.201	0.050	0.047	6.1
	Equal-F	0.214	0.011	0.147	0.074	0.089	0.398	0.056	0.037	0.117	0.048	0.177	0.024	0.043	6.2
	CACC	**0.323**	**0.024**	**0.261**	**0.122**	**0.185**	**0.554**	0.058	**0.138**	**0.130**	**0.839**	**0.269**	**0.174**	**0.074**	**1.1**
	CAIM	0.310	0.015	0.239	0.103	0.169	0.535	0.057	0.125	0.129	0.813	0.264	0.169	0.071	2.6
	IEM	0.283	0.004	0.213	0.096	0.175	0.507	**0.061**	0.097	0.128	0.794	0.219	0.135	0.063	3.5
	ChiMerge	0.241	0.020	0.243	0.056	0.132	0.485	0.055	0.060	0.118	0.032	0.169	0.090	0.047	5.1
	Ex-Chi2	0.292	0.021	0.234	0.103	0.177	0.488	0.054	0.063	0.121	0.823	0.217	0.119	0.056	3.5
Mean number of intervals	Equal-W	14.0	10.0	8.0	10.0	20.0	4.0	13.0	18.0	18.0	14.0	7.0	21.0	20.0	5.0
	Equal-F	14.0	10.0	8.0	10.0	20.0	4.0	13.0	18.0	18.0	14.0	7.0	21.0	20.0	5.0
	CACC	**2.0**	3.7	14.6	6.4	4.3	**3.0**	8.9	**5.0**	**10.0**	11.2	**6.0**	5.7	18.1	2.7
	CAIM	**2.0**	2.0	6.0	2.0	**2.0**	**3.0**	8.9	**5.0**	**10.0**	**2.0**	**6.0**	**3.0**	**3.0**	**1.5**
	IEM	2.4	**1.2**	**2.7**	**1.5**	4.0	**3.0**	**4.0**	6.3	10.2	2.1	11.4	3.8	5.0	2.1
	ChiMerge	4.6	6.3	5.3	7.8	21.4	3.5	5.7	41.9	19.1	25.6	19.2	17.7	28.5	5.4
	Ex-Chi2	3.3	12.2	5.0	2.3	8.8	7.5	10.4	36.0	15.9	20.0	16.4	11.3	12.2	4.7
Mean Discretization time (s)	Equal-W	0.26	0.18	**0.29**	**0.12**	**1.72**	**0.02**	**31.63**	4.92	21.94	**0.33**	29.84	5.33	9.06	**1.5**
	Equal-F	**0.27**	**0.17**	0.33	**0.12**	1.84	0.03	31.89	**4.89**	**21.86**	**0.33**	**29.09**	**5.31**	**9.33**	**1.5**
	CACC	0.58	0.24	0.84	0.22	3.62	0.08	58.75	14.44	70.66	0.90	58.28	12.22	61.41	4.2
	CAIM	0.58	0.21	0.63	0.20	3.43	0.08	58.83	15.42	71.53	0.80	58.34	12.13	52.38	3.6
	IEM	0.58	0.21	0.52	0.20	3.81	0.08	56.77	16.44	72.83	0.91	57.75	11.09	54.52	3.7
	ChiMerge	0.66	0.41	0.68	0.39	4.28	0.09	61.13	18.39	105.41	0.94	73.61	22.81	64.33	5.9
	Ex-Chi2	1.91	1.53	1.30	1.68	11.11	0.11	173.59	61.16	189.72	3.23	155.78	89.83	136.03	7.0

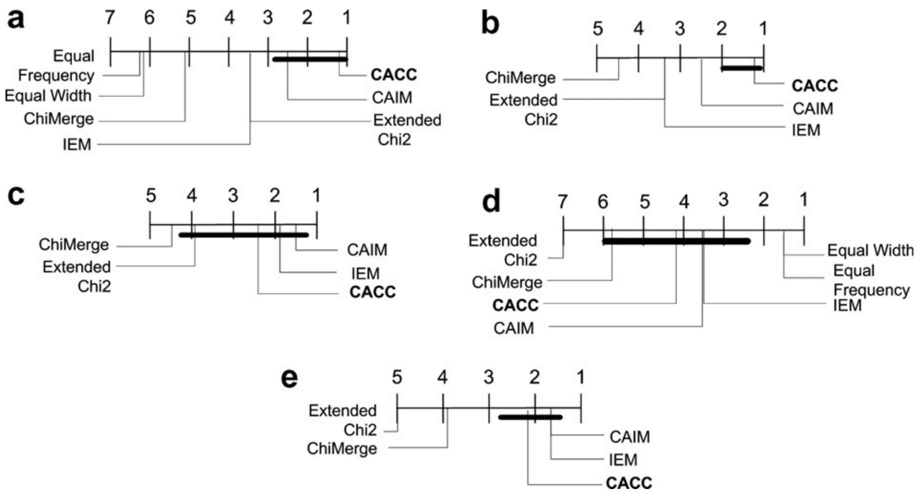

Fig. 2. The comparison of CACC against the other discretization methods with the Holm's post-hoc tests ($\alpha = 0.05$): (a) and (b) *cair* value; (c) number of intervals; (d) and (e) execution time.

difference did not reach statistical significance. If we compare all seven algorithms, the Holm's post-hoc test in Fig. 2d showed that CACC was significantly faster than Extended Chi2, significantly slower than Equal Width and Equal Frequency, and comparable to CAIM, IEM and ChiMerge. When we removed the two unsupervised algorithms from this comparison, we obtained a little different result as shown in Fig. 2e. In Fig. 2e, CACC was significantly faster than both bottom-up approaches Extended Chi2 and ChiMerge, and comparable to CAIM, IEM. This result corresponded to our discussions in Section 2.2 that the computational complexity of the bottom-up methods is usually worse than that of the top-down methods. It is also worth noting that compared to the ChiMerge algorithm, although the Extended Chi2 algorithm had a better discretization quality and generated fewer intervals, it required more execution time to check the merged inconsistency rate in every step.

4.2. The comparison of discretization schemes by using C5.0

To evaluate the effect of generated discretization schemes on the performance of the classification algorithm, we used the discretized datasets in Section 4.1 to train C5.0. The testing datasets were then used to calculate the accuracy, the number of rules, and the execution time as shown in Table 10. Similarly, the Friedman test and the Holm's post-hoc tests with significance level $\alpha = 0.05$ were used to check if these comparisons reached significant differences.

The comparison results in Table 10 show that on the average, CACC reached the highest accuracy from among the seven discretization algorithms. It is worth noting that CACC always reaches a higher C5.0 accuracy than the CAIM in all 13 datasets. This was a very exhilarating result that demonstrated that the discretization schemes generated by CACC can indeed improve the accuracy of classification. Since the Friedman test achieved the statistical significance, we then used the Holm's post-hoc tests to further analyze the comparisons of all the methods against CACC. The visualizations of the Holm's post-hoc test are illustrated in Fig. 3a. In Fig. 3a we can see that the accuracy of CACC was significantly better than Equal Width, Equal Frequency and ChiMerge, and comparable to CAIM, IEM and Extended Chi2. However, when we removed the two unsupervised methods and the two slowest bottom-up methods from this comparison, we obtained a little different result. The mean rank of CACC, CAIM and IEM was 1.2, 2.3, and 2.5 respectively. The Friedman test and the Holm's post-hoc tests in Fig. 3b showed that among the tree top-down approaches, the accuracy of CACC was significantly better than that of CAIM and IEM.

As regards to the number of generated rules of C5.0, the CAIM reached the best performance and CACC was ranked secondly. The Friedman test and the Holm's post-hoc tests in Fig. 3c showed that C5.0 produced

Table 10
The comparison of C5.0 performance on 13 UCI real datasets

Criterion	Algorithm	Dataset													Mean rank
		breast	bupa	glass	hea	ion	iris	optdigit	page-blocks	pendigit	pid	sat	thy	wav	
Mean accuracy (%)	Equal-W	91.3	63.0	86.1	70.3	88.0	91.4	69.7	93.4	86.0	70.1	74.2	92.4	73.2	6.4
	Equal-F	90.8	56.1	87.0	72.6	86.6	90.8	70.4	94.7	86.5	72.8	73.8	92.6	69.3	6.3
	CACC	**94.1**	**69.7**	**90.9**	**78.6**	**91.5**	93.5	76.6	**96.2**	**90.6**	**77.0**	77.2	**97.5**	**79.2**	**1.3**
	CAIM	93.8	69.1	90.6	77.1	91.2	93.0	75.0	95.7	90.4	75.6	77.0	96.7	78.4	3.0
	IEM	93.6	63.7	89.6	75.2	90.8	92.5	**79.8**	96.0	90.2	75.4	**77.4**	97.2	77.9	3.2
	ChiMerge	93.0	67.3	88.5	76.5	89.8	90.7	72.5	94.9	87.1	76.1	74.7	96.4	69.1	4.9
	Ex-Chi2	92.3	69.4	90.8	78.4	90.2	**94.6**	77.3	95.1	88.9	76.8	77.2	97.1	76.2	2.8
Mean number of rules	Equal-W	12.6	18.2	20.2	38.6	22.6	6.2	435.6	44.4	617.6	66.8	342.6	29.8	72.6	5.7
	Equal-F	10.1	16.2	22.1	44.4	26.8	5.3	394.0	63.5	544.3	72.3	366.2	42.2	65.4	5.8
	CACC	**6.8**	12.8	13.2	26.4	**10.4**	**3.2**	369.2	25.6	484.2	23.4	312.4	**9.4**	**44.6**	2.1
	CAIM	9.2	11.4	12.6	32.1	**10.4**	**3.2**	357.4	**23.4**	**442.5**	**21.2**	**309.6**	11.2	52.8	**1.9**
	IEM	7.4	10.2	**9.4**	**16.4**	11.1	**3.2**	**338.6**	40.6	478.2	23.6	378.4	28.4	98.8	2.8
	ChiMerge	12.4	19.3	14.4	33.2	26.2	4.3	402.1	39.6	621.4	36.6	385.8	33.2	62.6	5.5
	Ex-Chi2	7.4	**9.2**	10.2	32.4	18.4	3.5	388.4	36.2	525.6	25.6	380.2	31.8	82.6	3.8
Mean building time (s)	Equal-W	1.04	1.03	1.04	1.06	1.04	1.02	41.69	3.77	62.45	1.08	16.44	3.56	9.25	4.8
	Equal-F	1.05	1.03	1.04	1.07	1.06	1.02	36.31	3.83	52.75	1.08	15.94	3.66	11.44	5.3
	CACC	**1.02**	**1.02**	1.03	1.05	**1.03**	**1.00**	35.74	**3.13**	44.41	1.03	16.52	**2.53**	**8.25**	2.0
	CAIM	**1.02**	**1.02**	1.02	1.05	**1.03**	**1.00**	34.64	**3.13**	**44.14**	**1.02**	**16.42**	2.57	9.22	**1.6**
	IEM	1.06	**1.02**	**1.00**	**1.04**	**1.03**	**1.00**	**27.64**	3.16	45.41	**1.02**	17.47	3.56	11.22	2.7
	ChiMerge	1.04	1.03	1.03	1.06	1.06	1.01	32.40	3.95	65.70	1.14	19.92	3.54	12.49	5.3
	Ex-Chi2	**1.02**	**1.02**	1.01	1.05	1.04	**1.00**	36.74	3.73	58.42	1.08	19.46	3.54	10.25	3.3

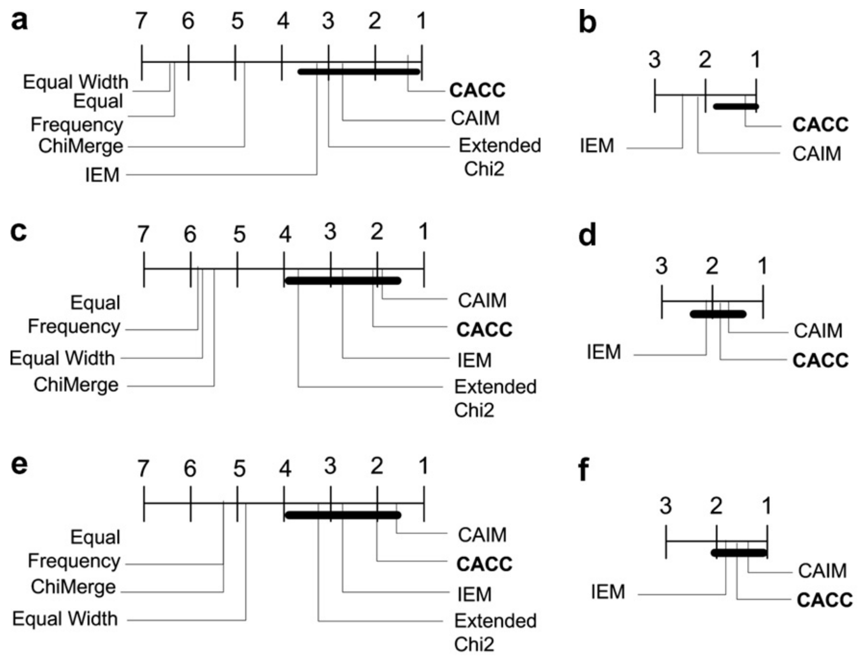

Fig. 3. The comparison of C5.0 performance on CACC against C5.0 performance on the other discretization methods with the Holm's post-hoc test ($\alpha = 0.05$): (a) and (b) accuracy; (c) and (d) number of rules; (e) and (f) execution time.

significantly more rules when it used the discretization schemes of ChiMerge, Equal Width and Equal Frequency. Fig. 3c also showed that C5.0 generated statistically comparable numbers of rules when it used the discretization schemes of CACC, CAIM, IEM and Extended Chi2. When we only compared the three top-down approaches, the Holm's post-hoc tests also showed that there were no significant differences among them as shown in Fig. 3d. Note that in Section 4.1, we stated that a discretization scheme with fewer intervals does not mean that it will result to a simpler decision tree. On the contrary, it might even increase the produced rules. Our inference can be found in Table 10. For example, CACC generated more intervals than CAIM but resulted to fewer rules in the datasets thy, wav and hea.

Finally as illustrated in Fig. 3e, when C5.0 used the training data discretized by CACC, CAIM, IEM and Extended Chi2, the training times were statistically comparable. C5.0 required significantly more training time when the training data were discretized by ChiMerge, Equal Width and Equal Frequency. When we only compared the three top-down approaches, the Holm's post-hoc tests also showed that there were no significant differences among CACC, CAIM and IEM.

4.3. Artificial datasets

In this section, we encoded a program to generate four artificial datasets as shown in Table 11 to further evaluate the difference between CACC and the newest top-down method, CAIM. Every artificial dataset contains ten continuous attributes, one target class attribute, and 1000 examples. In order to produce a meaningful artificial dataset which contains patterns for mining, each example in all datasets was generated independently, i.e., in each loop of our data generator, a sample was generated. The attribute values and class of this sample were randomly selected from the attribute domain in Table 12. As a result, each artificial dataset formed a *Bernoulli distribution*. The domains of all the attributes are shown in Table 12 with the four datasets 1, 2, 3 and 4 containing 2, 3, 5, and 8 target classes, respectively.

The comparison of the *cair* value between CAIM and CACC is presented in Table 13 wherein the result of each attribute is given. Just like the results in Section 4.1, the cair *value* of all 40 attributes discretized by CACC is always equal to or better than those discretized by CAIM, and the number of intervals and the discretization time of CACC are higher than or equal to those of CAIM. Again, C5.0 was used to evaluate the discretized schemes of CACC and CAIM. The comparisons of the accuracy, number of rules, and execution

Table 11
Four artificial datasets

Dataset	# of attributes	# of samples	# of target class
Dataset 1	10	1000	2
Dataset 2	10	1000	3
Dataset 3	10	1000	5
Dataset 4	10	1000	8

Table 12
The interval of each attribute of artificial datasets

Attribute	Interval
Attribute 1	−1.0–1.0
Attribute 2	0.0–1.0
Attribute 3	2.0–4.0
Attribute 4	3.0–6.0
Attribute 5	4.0–7.0
Attribute 6	5–8
Attribute 7	0–10
Attribute 8	10–20
Attribute 9	20–80
Attribute 10	120–150

Table 13
The comparison of discretization schemes on four artificial datasets

Criterion	Algorithm	Attribute									
		1	2	3	4	5	6	7	8	9	10
Dataset 1 (2 target classes)											
cair value	CACC	**0.0017**	**0.0026**	**0.0084**	**0.0038**	**0.0042**	**0.0008**	**0.0030**	**0.0031**	**0.0055**	**0.0048**
	CAIM	0.0008	0.0015	0.0013	0.0015	0.0008	**0.0008**	0.0024	0.0028	0.0016	0.0018
number of intervals	CACC	9	3	8	16	8	3	**2**	**2**	18	**2**
	CAIM	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**
Time (s)	CACC	1.282									
	CAIM	**1.043**									
Dataset 2: (3 target classes)											
cair value	CACC	**0.0066**	**0.0019**	**0.0039**	**0.0054**	**0.0069**	**0.0006**	**0.0026**	**0.0022**	**0.0132**	**0.0067**
	CAIM	0.0016	0.0009	0.0015	0.0018	0.0018	**0.0006**	0.0019	0.0014	0.0031	0.0024
number of intervals	CACC	17	**3**	12	8	18	**3**	**3**	8	32	16
	CAIM	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
Time (s)	CACC	2.274									
	CAIM	**1.547**									
Dataset 3: (5 target classes)											
cair value	CACC	**0.0069**	**0.0041**	**0.0484**	**0.0112**	**0.0088**	**0.0012**	**0.0042**	**0.0046**	**0.0251**	**0.0096**
	CAIM	0.0032	0.0025	0.0250	0.0033	0.0026	**0.0012**	0.0022	0.0034	0.0066	0.0066
number of rules	CACC	18	6	12	19	26	**4**	9	11	40	14
	CAIM	**5**	**5**	**5**	**5**	**5**	**4**	**5**	**5**	**5**	**5**
Time (s)	CACC	3.282									
	CAIM	**2.013**									
Dataset 4: (8 target classes)											
cair value	CACC	**0.0132**	**0.0037**	**0.1302**	**0.0169**	**0.0220**	**0.0018**	**0.0093**	**0.0074**	**0.0426**	**0.0180**
	CAIM	0.0079	0.0033	0.0866	0.0086	0.0105	**0.0018**	0.0081	0.0051	0.0118	0.0080
number of rules	CACC	15	**8**	14	28	30	**4**	11	12	54	29
	CAIM	**8**	**8**	**8**	**8**	**8**	**4**	**8**	**8**	**8**	**8**
Time (s)	CACC	4.463									
	CAIM	**3.222**									

time are shown in Table 14. Obviously, the accuracy of CACC is significantly higher than that of CAIM. As regards to the number of rules, both CACC and CAIM achieved statistically comparable results. Finally the C5.0 building times of the two algorithms were very close and showed no significant differences.

4.4. Detailed analysis of CACC

To avoid computing all possible discretization schemes, CACC uses the greedy approach to generate a sub-optimal discrete result. To evaluate the effectiveness of such a mechanism, we randomly selected one continuous attribute from each of the UCI datasets in Table 8. Discretization was done for the 13 selected continuous attributes even if the condition $cacc > Globalcacc$was met. Among the 13 attributes, eleven of them had the highest *cacc* when the discretization was terminated. The remaining two attributes were selected from the sat and thy dataset.

From the analysis, we found that the randomly selected attribute from the sat dataset had the highest *cacc* value when the number of intervals was less than the number of classes. In other words, although CACC terminated without the optimal *cacc* value, it could get a more reasonable discrete result. For the randomly selected attribute of the thy dataset, CACC terminated when the number of intervals was three, although the highest *cacc* value occurred when the number of intervals was five. By analyzing the distribution of its original data as shown in Fig. 4, we found that the interval $[0, 0.026]$ contains the target classes $C1$, $C2$, $C3$, the interval $(0.026, 0.041]$ contains $C2$, $C3$, and the interval $(0.041, 0.18]$ contains only $C3$. Since the target classes were seriously overlaid and very disorderly distributed, it was hard for any discretization algorithm to produce a good discretization scheme. Moreover, the number of instances when $C1$, $C2$, and $C3$ were within

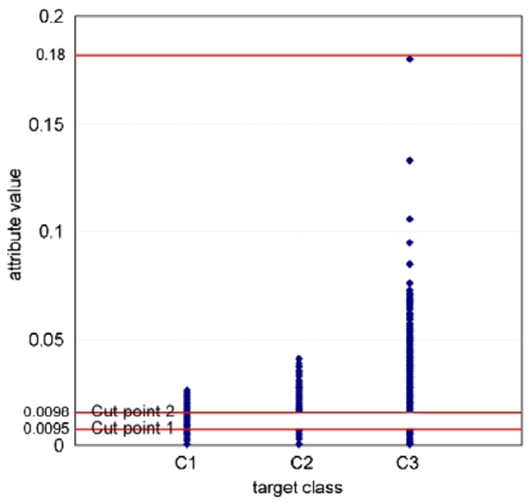

Fig. 4. The data distribution of a randomly selected attribute in thy dataset.

Table 14
The comparison of C5.0 performance on four artificial datasets

Performance	Algorithm	Artificial dataset			
		Dataset 1	Dataset 2	Dataset 3	Dataset 4
Mean accuracy (%)	CACC	**61.8.0**	**59.4**	**58.9**	**58.5**
	CAIM	58.2	57.4	57.2	56.7
Mean number of rules	CACC	28.6	163.5	**121.4**	**136.8**
	CAIM	18.6	103.2	185.9	182.4
Mean building time (s)	CACC	1.28	1.69	**1.55**	**1.71**
	CAIM	**1.20**	**1.58**	1.72	1.84

Table 15
The discretization scheme of randomly selected attribute from the dataset thy by CACC: (a) three intervals; (b) five intervals

Class	Interval			Total
	[0.00, 0.0095]	(0.0095, 0.0098]	(0.0098, 0.18]	
(a)				
*C*3	300	0	6366	6666
*C*2	30	0	338	368
*C*1	71	42	53	166
Total	401	42	6757	7200

	Interval					Total
	[0.00, 0.0095]	(0.0095, 0.0098)	(0.0098, 0.0172]	(0.0172, 0. 0177]	(0.0177, 0.18]	
(b)						
*C*3	300	0	1330	0	5036	6666
*C*2	30	0	177	34	127	368
*C*1	71	42	33	0	20	166
Total	401	42	1540	34	5183	7200

(0.0098, 0.041] were 53, 338, and 6154, respectively. Due to the serious overlap and disorderly distribution of the three classes, CACC did not form any interval within (0.0098, 0.041]. Since (0.0098, 0.041] was dominated by $C3$, the interval [0.041, 0.18] which contains purely $C3$ was combined with the interval (0.0098, 0.041] to form one interval [0.0098, 0.041]. The corresponding discretization schemes containing three and five intervals are illustrated in Table 15. In summary, we can conclude from this experiment that CACC can, on the average, generate a good discretization scheme even if it uses a greedy approach.

5. Conclusions

Discretization algorithms have played an important role in data mining and knowledge discovery. They not only produce a concise summarization of continuous attributes to help the experts understand the data more easily, but also make learning more accurate and faster. In this paper, we proposed a static, global, incremental, supervised and top-down discretization algorithm called CACC, in order to raise the quality of the generated discretization scheme by extending the idea of contingency coefficient and combining it with the greedy method. Empirical evaluation of seven discretization algorithms on 13 real datasets and four artificial datasets showed that our CACC algorithm generated a better discretization scheme to obtain the improvement of accuracy of C5.0. As regards to the execution time of discretization, the number of generated rules, and the execution time of C5.0, CACC also achieved promising results.

Acknowledgement

This research was partially supported by the National Science Council of Republic of China under Grant No. NSC 96-2520-S-024-001. We are grateful to the authors of the CAIM algorithm because of its important role on our paper. We are also grateful to Chih-Wei Ku and Ya-Ru Yang for the help of experiments in this paper.

References

[1] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, University of California, Department of Information and Computer Science, Irvine, CA, 1998. Available from: <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
[2] M. Boulle, Khiops: a statistical discretization method of continuous attributes, Machine Learning 55 (1) (2004) 53–69.
[3] G. Cardoso, F. Gomide, Newspaper demand prediction and replacement model based on fuzzy clustering and rules, Information Sciences 177 (21) (2007) 4799–4809.
[4] J.Y. Ching, A.K.C. Wong, K.C.C. Chan, Class-dependent discretization for inductive learning from continuous and mixed mode data, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (7) (1995) 641–651.
[5] D. Chiu, A. Wong, B. Cheung, Information discovery through hierarchical maximum entropy discretization and synthesis, in: G. Piatetsky-Shapiro, W.J. Frawley (Eds.), Knowledge Discovery in Databases, AAAI Press, 1991, pp. 125–140.
[6] K.J. Cios, L.A. Kurgan, Hybrid inductive machine learning: an overview of clip algorithms, in: L.C. Jain, J. Kacprzyk (Eds.), New Learning Paradigms in Soft Computing, Physica-Verlag (Springer), 2001, pp. 276–322.
[7] K.J. Cios, L.A. Kurgan, CLIP4: hybrid inductive machine learning algorithm that generates inequality rules, Information Science 163 (1-3) (2004) 37–83.
[8] P. Clark, T. Niblett, The CN2 algorithm, Machine Learning 3 (4) (1989) 261–283.
[9] S. Cohen, L. Rokach, O. Maimon, Decision-tree instance-space decomposition with grouped gain-ratio, Information Sciences 177 (17) (2007) 3592–3612.
[10] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[11] J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features, in: Proceeding of Twelfth International Conference on Machine Learning, 1995, pp. 194–202.
[12] U.M. Fayyad, K.B. Irani, On the handling of continuous-valued attributes in decision tree generation, Machine Learning 8 (1992) 87–102.
[13] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proceeding of Thirteenth International Conference on Artificial Intelligence, 1993, pp. 1022–1027.
[14] F. Berzal, J.C. Cubero, N. Marín, D. Sánchez, Building multi-way decision trees with numerical attributes, Information Sciences 165 (1–2) (2004) 73–90.
[15] D. Gómez, J. Montero, J. Yáñez, A coloring fuzzy graph approach for image classification, Information Sciences 176 (24) (2006) 3645–3657.
[16] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufman, 2001.

[17] T. Hidekazu, A.E. Sayed, F. Masao, M. Kazuhiro, T. Kazuhiko, J.I. Aoe, Estimating sentence types in computer related new product bulletins using a decision tree, Information Sciences 168 (1–4) (2004) 185–200.
[18] K.A. Kaufman, R.S. Michalski, Learning from inconsistent and noisy data: the AQ18 approach, in: Proceeding of Eleventh International Symposium on Methodologies for Intelligent Systems, 1999.
[19] R. Kerber, ChiMerge: discretization of numeric attributes, in: Proceeding of Ninth International Conference on Artificial Intelligence, 1992, pp. 123–128.
[20] L. Kurgan, K.J. Cios, Fast Class-Attribute Interdependence Maximization (CAIM) Discretization Algorithm, in: Proceeding of International Conference on Machine Learning and Applications, 2003, pp. 30–36.
[21] L. Kurgan, K.J. Cios, CAIM discretization algorithm, IEEE Transactions on Knowledge and Data Engineering 16 (2) (2004) 145–153.
[22] C.I. Lee, C.J. Tsai, Y.R. Yang, W.P. Yang, A top-down and greedy method for discretization of continuous attributes, in: Proceedings of Fourth International Conference on Fuzzy Systems and Knowledge Discovery, Haikou, China, 2007.
[23] H. Liu, R. Setiono, Feature selection via discretization, IEEE Transactions on Knowledge and Data Engineering 9 (4) (1997) 642–645.
[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] S. Mehta; S. Parthasarathy, H. Yang, Correlation preserving discretization data mining, in: Proceeding of Fourth IEEE International Conference on Data Mining, 2004, pp. 479–482.
[26] R.S. Michalski, I. Mozetic, J. Hong, N. Lavrac, The multipurpose incremental learning system AQ15 and its testing application to three medical domains, in: Proceeding of Fifth National Conference on Artificial Intelligence, 1986, pp. 1041–1045.
[27] A. Paterson, T.B. Niblett, ACLS manual, Edinburgh: Intelligent Terminals, Ltd., 1987.
[28] B. Pfahringer, Compression-based discretization of continuous attributes, in: Proceeding of Twelfth International Conference on Machine Learning, 1995, pp. 456–463.
[29] R. Quinlan, Data Mining Tools, 2005. <http://www.rulequest.com/see5-info.html>.
[30] R. Rastogi, K. Shim, PUBLIC: a decision tree classifier that integrates building and pruning, in: Proceedings of the 24th International Conference on Very Large Databases, 1998, pp. 404–415.
[31] D.H. Shih, H.S. Chiang, C.D. Yen, Classification methods in the detection of new malicious emails, Information Sciences 172 (1–2) (2005) 241–261.
[32] K.G. Srinivasa, K.R. Venugopal, L.M. Patnaik, A self-adaptive migration model genetic algorithm for data mining applications, Information Sciences 177 (20) (2007) 4295–4313.
[33] C.T. Su, J.H. Hsu, An extended chi2 algorithm for discretization of real value attributes, IEEE Transactions on Knowledge and Data Engineering 17 (3) (2005) 437–441.
[34] F. Tay, L. Shen, A modified chi2 algorithm for discretization, IEEE Transactions on Knowledge and Data Engineering 14 (3) (2002) 666–670.
[35] A.K.C. Wong, D.K.Y. Chiu, Synthesizing statistical knowledge from incomplete mixed-mode data, IEEE Transactions on Pattern Analysis and Machine Intelligence 9 (1987) 796–805.
[36] Q. Wu, D.A. Bell, T.M. McGinnity, G. Prasad, G. Qi, X. Huang, Improvement of decision accuracy using discretization of continuous attributes, in: Proceedings of the Third International Conference on Fuzzy Systems and Knowledge Discovery, Lecture Notes in Computer Science 4223, 2006, pp. 674–683.
[37] R.R. Yager, An extension of the naive Bayesian classifier, Information Sciences 176 (5) (2006) 577–588.
[38] S. Zadrożny, J. Kacprzyk, Computing with words for text processing: an approach to the text categorization, Information Sciences 176 (4) (2006) 415–437.