Expert Systems with Applications 36 (2009) 5327–5332

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Ameva: An autonomous discretization algorithm

L. Gonzalez-Abril [a,*], F.J. Cuberos [b], F. Velasco [a], J.A. Ortega [c]

[a] Applied Economics I Department, Seville University, Seville 41018, Spain
[b] Planning Department – RTVA, Seville, Spain
[c] Computer Languages and Systems Department, Seville University, Spain

ARTICLE INFO

Keywords:
Knowledge discovery
Supervised discretization
Machine learning
Genetic algorithm

ABSTRACT

This paper describes a new discretization algorithm, called Ameva, which is designed to work with supervised learning algorithms. Ameva maximizes a contingency coefficient based on Chi-square statistics and generates a potentially minimal number of discrete intervals. Its most important advantage, in contrast with several existing discretization algorithms, is that it does not need the user to indicate the number of intervals.

We have compared Ameva with one of the most relevant discretization algorithms, CAIM. Tests performed comparing these two algorithms show that discrete attributes generated by the Ameva algorithm always have the lowest number of intervals, and even if the number of classes is high, the same computational complexity is maintained. A comparison between the Ameva and the genetic algorithm approaches has been also realized and there are very small differences between these iterative and combinatorial approaches, except when considering the execution time.

1. Introduction

Knowledge extraction and automated processing of databases are important tasks to be performed by machine learning algorithms. Several of these algorithms are expressly designed to handle numerical or nominal data such as the CN2 algorithm (Clark & Niblett, 1989; Su & Hsu, 2005), CLIP algorithms (Cios & Kurgan, 2002) and AQ algorithms (Kaufman & Michalski, 1999), hence such algorithms cannot be applied to mixed-mode (both continuous and discrete) data unless the continuous features are first discretized. Other algorithms perform better with discrete-value attributes, despite the fact that they can also handle continuous attributes (Catlett, 1991; Kerber, 1992).

The discretization process converts continuous attributes into discrete ones by yielding intervals in which the attribute value can reside instead of singleton values, and by associating a discrete, numerical value with each interval. The usual approach for learning tasks which use mixed-mode data is to perform discretization prior to the learning process as a preprocessing step (Catlett, 1991; Dougherty, Kohavi, & Sahami, 1995; Fayyad & Irani, 1992). Therefore the discretization process first finds the number of discrete intervals, and the width or boundaries for the intervals, given the range of values a continuous attribute has (Kurgan & Cios, 2004; Macskassy, Hirsh, Banerjee, & Dayanik, 2003). Furthermore, discretization itself may be viewed as a discovery of knowledge in that critical values in a continuous domain may be revealed.

A desirable feature for practical discretization is that discretized attributes have fewer values possible since a large number of possible attribute values contributes to a slow and ineffective process of inductive machine learning (Catlett, 1991). However, the advantages of discretizing during the learning process have not yet been shown although some comparisons have been made (Dougherty et al., 1995). For example, in Macskassy et al. (2003) it was shown that, even on numerical-valued data, the results of text classification on the derived text-like representation outperforms the more naive numbers-as-tokens representation and, more importantly, is competitive with numerical classification methods such as C4.5 (Quinlan, 1993), Ripper (Cohen, 1995) and SVM (Angulo, Anguita, Gonzalez-Abril, & Ortega, 2008; Gonzàlez, Angulo, Velasco, & Català, 2006).

A proliferation of discretization methods can be found in the literature; from unsupervised algorithms such as the equal-width interval, equal-frequency interval k-mean clustering or unsupervised MCC algorithms (Dougherty et al., 1995); to supervised algorithms such as CAIM (Kurgan & Cios, 2004), ChiSplit (Bertier & Bouroche, 1981), ChiMerge (Kerber, 1992), Chi2 (Liu & Setiono, 1997; Tay & Shen, 2002), Khiops (Boullé, 2004), CADD (Ching, Wong, & Chan, 1995), maximum entropy (Kumar & Zhang, 2007; Le & Satoh, 2007) or 1R (Holte, 1993). An extensive list may be found in Dougherty et al. (1995).

In this paper, a new discretization method called Ameva based on Chi-square statistics (χ^2) is presented. It is well known that

* Corresponding author. Tel.: +34 954554345; fax: +34 954551636.
E-mail address: luisgon@us.es (L. Gonzalez-Abril).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.06.063

there are other approaches based on a Chi-square statistics: ChiMerge, ChiSplit and Chi2. All these algorithms need an expert to provide parameters in order to use the algorithms which leaves a wide margin for error (e.g. in the cases of noise, experts' absence, measurement errors, and bad data). For this reason the Ameva algorithm is compared with the CAIM algorithm (Kurgan & Cios, 2004), which is one of the most relevant discretization algorithms, since both share similar characteristics and do not need any user to provide any parameter for the algorithms.

The remainder of the paper is organized as follows. Section II presents the problem and the notation. Sections III and IV the Ameva criterion and Ameva algorithm are introduced. Section V compares the Ameva method with other related methods. In the following section an extensive experimental evaluation is carried out. Finally, some conclusions are drawn.

2. Definition of the problem

Let $X = \{x_1, \ldots, x_N\}$ be a training data set of a continuous attribute $\mathscr{X}$ of mixed-mode data such that each example x_i belongs to only one of the ℓ classes of the class variable denoted by $\mathscr{C} = \{C_1, \ldots, C_\ell\}$. A continuous attribute discretization is a function $\mathscr{D} : \mathscr{X} \to \mathscr{C}$ which assigns a class $C_i \in \mathscr{C}$ to each value $x \in \mathscr{X}$.

Let us consider a discretization $\mathscr{D}$ which discretizes $\mathscr{X}$ into k discrete intervals $\{L_1, \ldots, L_k\}$ where $L_1 = [(d_0, d_1]$ and $L_j = (d_{j-1}, d_j]$ for $j = 2, \ldots, k$ such that d_k is the maximal value and d_0 is the minimal value of attribute X and the values d_i are arranged in ascending order. Thus, a discretization variable is defined as $\mathscr{L}(k; X, \mathscr{C}) = \{L_1, \ldots, L_k\}$ which verifies that, for all $x_i \in X$, a unique L_j exists such that $x_i \in L_j$ for $i = 1, \ldots, N$ and $j = 1, \ldots, k$. The discretization variable $\mathscr{L}(k; X, \mathscr{C})$ (denoted as $\mathscr{L}(k)$ for the sake of simplicity) of attribute $\mathscr{X}$ and the class variable $\mathscr{C}$ are treated from a descriptive point of view and Table 1 is drawn up where n_{ij} denotes the total number of continuous values belonging to the ith class that are within the jth interval, $n_{i\cdot}$ is the total number of instances belonging to the ith class, and $n_{\cdot j}$ is the total number of instances belonging to the jth interval for $i = 1, \ldots, \ell$ and $j = 1, \ldots, k$ (Table 2).

3. Ameva discretization

Given discrete attributes $\mathscr{C}$ and $\mathscr{L}(k)$, the contingency coefficient, denoted by $\chi^2(k) \overset{\text{def}}{=} \chi^2(\mathscr{L}(k), \mathscr{C} \mid X)$, defined as

$$\chi^2(k) = N\left(-1 + \sum_{i=1}^{\ell}\sum_{j=1}^{k}\frac{n_{ij}^2}{n_{i\cdot}n_{\cdot j}}\right) \tag{1}$$

is considered. It is straightforward to prove that

$$\max_{X,\mathscr{L}(k),\mathscr{C}} \chi^2(k) = N(\min\{\ell, k\} - 1). \tag{2}$$

Hence, the Ameva coefficient, $\text{Ameva}(k) \overset{\text{def}}{=} \text{Ameva}(\mathscr{L}(k), \mathscr{C} \mid X)$, is defined as follows:

$$\text{Ameva}(k) = \frac{\chi^2(k)}{k(\ell - 1)} \tag{3}$$

for $k, \ell \geqslant 2$. The Ameva criterion has the following properties:

Table 1
Contingency table for attribute X and discretization variable $\mathscr{L}(k)$

$C_i \mid L_j$	L_1	$\cdots$	L_j	$\cdots$	L_k	$n_{i\cdot}$
C_1	n_{11}	$\cdots$	n_{1j}	$\cdots$	n_{1k}	$n_{1\cdot}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_i	n_{i1}	$\cdots$	n_{ij}	$\cdots$	n_{ik}	$n_{i\cdot}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C_ℓ	$n_{\ell 1}$	$\cdots$	$n_{\ell j}$	$\cdots$	$n_{\ell k}$	$n_{\ell\cdot}$
$n_{\cdot j}$	$n_{\cdot 1}$	$\cdots$	$n_{\cdot j}$	$\cdots$	$n_{\cdot k}$	N

Table 2
Columns which are different in the contingency table of X for discretization schemes $\mathscr{L}(k)$ and $\mathscr{L}(k+1)$

	L_k^*		L_k		L_{k+1}
C_1	n_{1k}^*	=	n_{1k}	+	n_{1k+1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
C_i	n_{ik}^*	=	n_{ik}	+	n_{ik+1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
C_ℓ	$n_{\ell k}^*$	=	$n_{\ell k}$	+	$n_{\ell k+1}$
	$n_{\cdot k}^*$	=	$n_{\cdot k}$	+	$n_{\cdot k+1}$

- The minimum value of Ameva(k) is 0 and when this value is achieved then both discrete attributes $\mathscr{C}$ and $\mathscr{L}(k)$ are statistically independent and viceversa.
- The maximum value of Ameva(k) indicates the best correlation between the class labels and the discrete intervals. If $k \geqslant \ell$ then, for all $x \in C_i$ a unique j_0 exists such that $x \in L_{j_0}$ (the remaining intervals $(k - \ell)$ have no elements); and if $k < \ell$ then, for all $x \in L_j$, a unique i_0 exists such that $x \in C_{i_0}$ (the remaining classes have no elements), i.e. the highest value of the Ameva coefficient is achieved when all values within a particular interval belong to the same associated class for each interval.
- The aggregated value is divided by the number of intervals k, hence the criterion favours discretization schemes with the lowest number of intervals.
- In order to make a comparison with the CAIM coefficient, the aggregated value is divided by $\ell - 1$.
- From (2), it is followed that $\text{Ameva}_{\max}(k) \overset{\text{def}}{=} \max_{X,\mathscr{L}(k),\mathscr{C}} \text{Ameva}(k) = \frac{N(k-1)}{k(\ell-1)}$ if $k < \ell$ and $\frac{N}{k}$ otherwise. Hence, $\text{Ameva}_{\max}(k)$ is an *increasing* function of k if $k \leqslant \ell$, and a *decreasing* function of k if $k > \ell$. Therefore, $\max_{k \geqslant 2}\text{Ameva}_{\max}(k) = \text{Ameva}_{\max}(\ell)$, i.e. the maximum of the Ameva coefficient is achieved in the optimal situation (all values of C_i are in a unique interval L_j and viceversa).

Therefore, the aim of the Ameva method is to maximize the dependency relationship between the class labels $\mathscr{C}$ and the continuous-values attribute $\mathscr{L}(k)$, and at the same time to minimize the number of discrete intervals k.

4. The Ameva algorithm: two approaches

Usually the problem of finding a discretization scheme $\mathscr{L}(k)$ with a globally optimal value is a highly combinatorial problem. Hence, a first approach is given which by starting with a single interval, the algorithm, see Appendix A, works in a top-down way, repeatedly dividing one of the existing intervals into two new intervals by using a criterion which results in achieving the optimal value of (3) after the split. This approach performs the discretization task at reasonable computational cost, hence it may be applied to continuous attributes with a large number of unique values, and it then finds local maximum values of the Ameva criterion.

The number of labels is small in some problems and, as the Ameva criterion provides few intervals, it is possible to define a genetic algorithm for these problems (two approach) which finds global maximum values of the Ameva criterion which has no excessive computational cost.

Both approaches find the lowest number of intervals which provides the best "number of interval/association coefficient" ratio based on the Ameva value.

5. The Ameva criterion versus other criteria

In this section, the Ameva method is compared with other discretization methods from a theoretical point of view.

5.1. Criteria based on Chi-square statistics

ChiSplit, ChiMerge, Chi2, Khiops and Ameva are based on Chi-square statistics for which the following lemma is very important.

Lemma 5.1. *Let $X = \{x_1, \ldots, x_N\}$ be a training data set of a continuous attribute $\mathscr{X}$, and $\mathscr{C}$ be a class variable with ℓ classes. Let $\mathscr{L}(k)$ and $\mathscr{L}(k+1)$ be two discretization schemes where $\mathscr{L}(k+1)$ is obtained from $\mathscr{L}(k)$ by splitting an interval into two intervals. Hence*

$$\chi^2(k+1) \geqslant \chi^2(k), \quad \text{for } k \geqslant 2$$

Proof. For the sake of simplicity the last interval is split into two intervals. In this case the contingency table of $\mathscr{L}(k)$ and $\mathscr{L}(k+1)$ are the same except in the last columns. In Table 1 can be seen this situation and some equalities where L_k^* is the k interval of $\mathscr{L}(k)$, and where L_k and L_{k+1} are the k and $k+1$ intervals of $\mathscr{L}(k+1)$) Following this notation and from (1) it is verified that

$$\chi^2(k+1) - \chi^2(k) = N \sum_{i=1}^{\ell} \frac{1}{n_{i\cdot}} \frac{(n_{ik} n_{\cdot k+1} - n_{ik+1} n_{\cdot k})^2}{n_{\cdot k} n_{\cdot k+1} (n_{\cdot k} + n_{\cdot k+1})} \tag{4}$$

whereby the lemma is proved. □

On the other hand, the ChiSplit, ChiMerge and Chi2 algorithms need a stopping criterion which is used to measure if the difference between iterations is irrelevant. In these algorithms the stopping criterion must be chosen by an expert from a set of parameters. Nevertheless, in the Ameva algorithm, the stopping criterion is automatically chosen. Let us see: Following (4), it holds that:

$$\text{Ameva}(k+1) = \frac{k}{k+1}\text{Ameva}(k) + \frac{AA(k+1)}{k+1} \tag{5}$$

where $AA(k+1) = \frac{N}{(\ell-1)} \sum_{i=1}^{\ell} \frac{1}{n_{i\cdot}} \frac{(n_{ik} n_{\cdot k+1} - n_{ik+1} n_{\cdot k})^2}{n_{\cdot k} n_{\cdot k+1} (n_{\cdot k} + n_{\cdot k+1})}$. That is, $\text{Ameva}(k+1)$ is a weighted arithmetic mean of Ameva(k), with weight $\frac{k}{k+1}$, and of $AA(k+1)$ with weight $\frac{1}{k+1}$. Thus, the stopping criterion is automatically chosen owing to (5), whereby step 2.4 of the Ameva algorithm may be written $\chi^2(k) < (\ell-1)kAA(k+1)$. On the other hand, it is important to note that if $\mathscr{C}$ and $\mathscr{L}(k)$ are considered as random variables then it may be proved that the random variable $\chi^2(k)$ possesses a distribution which is approximately a Chi-square for a sufficiently large N. This approach is used in Khiops which is a bottom-up method whose stopping criterion is based on the confidence level with the Chi-square statistic.

In this paper no comparison has been made with the above methods, however in Risvik (1999) comparisons between four other discretization algorithms were carried out and the results of the Ameva algorithm and the ChiMerge algorithm in classification accuracy can be compared. Furthermore, in Boullé (2004) the results of the Ameva algorithm and the Chi2, ChiSplit and Khiops algorithms in classification accuracy can also be compared.

5.2. Ameva versus CAIM

The CAIM criterion (Kurgan & Cios, 2004) uses the class attribute dependency information as the criterion for the optimal discretization. It is denoted as $\text{CAIM}(k) \stackrel{\text{def}}{=} \text{CAIM}(\mathscr{L}(k), \mathscr{C} \mid X)$ and, given Table 1, is defined as $\text{CAIM}(k) = \frac{1}{k}\sum_{j=1}^{k} \frac{\max_{i=1,\ldots,\ell}\{n_{ij}^2\}}{n_j}$. The value of $\text{CAIM}(k)$ has the following properties: (i) The algorithm favours discretization schemes where each interval has all its values grouped within a single class label; (ii) $\min_{X,\mathscr{L}(k),\mathscr{C}}\text{CAIM}(k) = \frac{N}{k\ell^2}$; (iii) The aggregated value is divided by the number of intervals k, and therefore the criterion favours discretization schemes with the lowest number of intervals; and (iv) the maximum value of $\text{CAIM}(k)$ is the best correlation between the class labels and the discrete intervals. It is proved (Kurgan & Cios, 2004) that given k, $\max_{X,\mathscr{L}(k),\mathscr{C}}\text{CAIM}(k) \stackrel{\text{def}}{=} \text{CAIM}_{\max}(k) = \frac{N}{k} \leqslant \text{CAIM}_{\max}(2)$; and (v) If the attributes $\mathscr{C}$ and $\mathscr{L}(k)$ are statistically independent then $\text{CAIM}(k) = \frac{1}{kN}\max_{i=1,\ldots,\ell} n_{i\cdot}^2$.

The main objective of the CAIM method is the same than the Ameva method and the Ameva algorithm is similar to the CAIM algorithm except that in step 2.4 a second condition is presented. The complete conditional clause is (CAIM > GlobalCAIM or $k < \ell$). This constraint overrides the fact that usually $\text{CAIM}(k) > \text{CAIM}(\ell+1)$ for all $k = 2, 3, \ldots, \ell$, and therefore guarantees that $\mathscr{L}(k)$ has at least as many intervals as the number of classes.

This clause causes a serious drawback since takes into account that $\text{CAIM}_{\max}(k)$ plunges hyperbolically, hence the CAIM algorithm cannot always provide the minimum number of discrete intervals because it is impossible to find a discretization scheme such that $k < \ell$. It is empirically proved that if, in the conditional clause, the constraint $k < \ell$ is omitted then there is a high probability that the number of intervals is two or three.

On the other hand, it is seldom necessary to continue the process for $k > \ell$ since almost always, $\text{CAIM}(\ell) > \text{CAIM}(\ell+1)$. This may be seen by taking in account the values of $\text{CAIM}_{\max}(\ell)$ and $\text{CAIM}_{\max}(\ell+1)$, and the next section.

6. Empirical comparisons

The results of the Ameva algorithm compared with the CAIM algorithm on several well-known, continuous and mixed-mode data sets are presented in this section. To this end, three experiments have been carried out.

In the first experiment Ameva has been compared with CAIM with the same data sets as used in Kurgan & Cios (2004). For the second experiment, four other data sets are used which have a greater number of classes than in the data sets in the first experiment. The last experiment is a comparison between the two separate approaches of the Ameva algorithm, Ameva and Ameva^R proposed in this paper. In these experiments the CAIM and Ameva algorithms are applied to every data set whereby new discretized sets are obtained by following a 10-fold cross-validation scheme. The discretized sets are used in a learning task performed by the *C*4.5 algorithm (Quinlan, 1993).

The measure of the discretization quality of both algorithms is given by the total number of intervals, the time spent in the discretization process, the number of nodes included in the decision tree (generated by the *C*4.5 algorithm) and the accuracy achieved in the identification.

6.1. First experiment

The names of the data sets (the number of classes and the short forms) used are: Iris Plants (3, IRIS), John Hopkins University Ionosphere (3, ION), Statlog Project Heart Disease data set (2, HEA), Pima Indians Diabetes (2, PID), Statlog Project Satellite Image (6, SAT), Thyroid Disease (3, THY), Waveform (3, WAV) and Attitudes Towards Workplace Smoking Restrictions (3, SMO). All data sets are obtained from the UCI Machine Learning Repository (Blake & Merz, 1998) except the last data set which was obtained from the Statlib data set archive (Vlachos, 2000).

We compare the Ameva algorithm with the CAIM algorithm by means of these data sets since in Kurgan & Cios (2004), CAIM was compared with six other well-known discretization algorithms by means of the same data sets and was found to be superior to them.

Table 3 shows the values of the quality criteria selected for those data sets discretized by the CAIM and Ameva algorithms. The best result in each case is shown in bold. Since a cross-validation method is followed, the mean value and standard deviation are presented.

Table 3
Results obtained in the first experiment

Criterion	Discretization method	Dataset							
		IRIS		SAT		THY		WAV	
		Mean	StdD.	Mean	StdD.	Mean	StdD.	Mean	StdD.
Total number of intervals	CAIM	12	0.000	216	0.000	18	0.000	63	0.000
	Ameva	**10.0**	0.000	**100.4**	.699	**15.6**	.699	**47.1**	3.81
Execution time (in s)	CAIM	0.012	0.004	3.69	0.011	0.924	0.007	9	0.021
	Ameva	**0.010**	0.000	**3.30**	0.015	**0.896**	0.010	**7.44**	0.363
Tree size C4.5	CAIM	4.3	0.949	884	67.8	29.3	8.29	493	43.2
	Ameva	4.3	0.949	**528**	48.9	**17.6**	1.26	**319**	27.4
% Accuracy C4.5	CAIM	93.3	5.4	**85.8**	1.9	98.7	0.520	76.2	2.6
	Ameva	93.3	5.4	82.8	1.4	**99.1**	0.340	**76.8**	1.3
		ION		SMO		HEA		PID	
Total number of intervals	CAIM	**67.0**	.000	**6.0**	.000	**12.0**	0.000	**16.0**	0.000
	Ameva	107.1	2.13	7.1	1.91	13.6	1.9	18.8	0.919
Execution time (in s)	CAIM	**0.76**	0.006	**0.115**	0.006	**0.031**	0.003	**0.192**	0.004
	Ameva	1.49	0.038	0.134	0.020	0.042	0.008	0.226	0.005
Tree size C4.5	CAIM	21.2	4.37	1.9	1.45	31.8	4.71	**16.0**	4.74
	Ameva	**18.7**	3.02	**1.00**	0	**31.2**	5.29	23.2	7.48
% Accuracy C4.5	CAIM	**88.9**	5.5	69.5	.20	80.8	4.9	72.9	3.3
	Ameva	88.0	4.6	**69.6**	.10	80.8	6.7	**76.0**	3.7

Due to the drawback of the CAIM method, in all these data sets, the number of intervals can be previously estimated by multiplying the number of attributes (s) by the number of classes (ℓ). In the ION data set, this rule is broken, whereby 67 intervals are obtained from a theoretical value of 68 ($34 \cdot 2$); the reason being that one of the attributes in this data set presents only one single value and is therefore included in one interval.

By comparing the number of intervals generated by the CAIM and Ameva methods, the best results are provided by CAIM in four data sets and by Ameva in the other four data sets. However, it is interesting note that CAIM is superior in the three data sets with only two classes and only one of the four data sets which presents 3 classes. Methods have similar behaviour with a clear advantage of Ameva over CAIM in the SAT and WAV data sets, and a lower number of intervals for CAIM in the ION data set.

The same situation is found in the execution time since a lower number of intervals implies fewer iterations and the evaluation of fewer candidate intervals.

From only the "number of nodes" perspective, there is a clear advantage for Ameva whereby the results are better than CAIM in 6 cases, equal to CAIM in 1 case and inferior in 1 case. This is an important factor since a lower number of nodes conveys a faster implementation of the identification task. Obviously, the more intervals there are, the longer the time required and the greater the tree. In the SAT and WAV data sets the *C*4.5 learning algorithm generates 884 and 493 nodes over the CAIM discretized data sets as opposed to 528 and 319 nodes generated for Ameva respectively.

It is important to note that, in general, a lower number of intervals implies a smaller tree size, but is not a sufficient condition as can be seen in the ION, SMO and HEA data sets, in the *C*4.5 algorithm.

Finally, the accuracy is nearly identical in six data sets, with differences lower than 1%. In the two remaining data sets there is a 3% advantage for that method which produces a greater number of intervals, CAIM for SAT and Ameva for PID.

The results shows a possible relation between the number of classes and the number of intervals obtained with each method: when the number of classes is equal to 2, CAIM produces fewer intervals, and when the number of classes increases the method which generates fewer intervals is Ameva.

Very few differences can be found between these two methods in terms of the number of intervals, execution time and accuracy in classification. The conclusion from this experiment is that there is an equivalent behaviour of these two methods, with Ameva being better than CAIM in the number of intervals obtained.

6.2. Second experiment

As previously stated, the CAIM algorithm is obliged to produce at least the same number of intervals as classes in the data set. All the sets presented in Kurgan& Cios (2004) have only two or three classes except the SAT data set (six classes). Thus, in this second experiment, the influence of the CAIM constraint, which produces as many intervals as classes, is studied when the number of classes increases.

Four new data sets from the UCI ML Repository have been selected with a number of classes greater than 3. These data sets (the number of classes and their short forms) are: Glass Identification Database (6, GLA), Image Segmentation database (7, SEG), Vehicle Silhouettes (4, VEH), and Vowel Recognition (11, VOW). The results of the experiment using these data sets are presented in Table 4.

The results show that with these four data sets the Ameva algorithm produces the lowest number of intervals since it is not influenced by the CAIM constraint which dictates that the number of intervals must be equal to or greater than the number of classes. The fewer the intervals found, the fewer the iterations of the algorithm. Hence, the Ameva implementation has a shorter execution time, from a 20% reduction in the VEH to a 66% reduction in the VOW data set. Analogously, a reduced number of intervals produces a smaller tree in the C4.5 algorithm. The number of nodes obtained from the Ameva discretized data set are at most half of its CAIM counterpart (and a third in the VOW data set). The number of intervals generated by CAIM with the SEG data set is lower than the theoretical value calculated by following the aforementioned reasoning for the ION set.

The accuracy is slightly better in two of the four CAIM discretized data sets. The differences in accuracy are wider in the SEG data set (about 4%) and in the VOW data set (nearly 6%). However, if the loss of accuracy for these data sets is contrasted with the radical reduction in the number of intervals and the tree size, ($\pm 50\%$ for the SEG and $\pm 70\%$ for the VOW in both criteria) it can be concluded that, in some applications, this slight reduction in identification is a minor inconvenience.

These results are coherent with the restriction that when the number of intervals is equal to the number of classes then informa-

Table 4
Comparison of results obtained by CAIM, Ameva and Ameva^R algorithms in data sets with a high number of classes

Criterion	Discretization method	Data sets							
		GLA		SEG		VEH		VOW	
		Mean	StdD.	Mean	StdD.	Mean	StdD.	Mean	StdD.
Total number of intervals	CAIM	54	0	119.9	.316	72	0	110	0
	Ameva	**25.5**	1.84	**50.8**	.422	**44**	1.63	**30.2**	0.919
	Ameva^R	54.9	0.748	127	0.000	72.1	0.316	110.1	0.316
Execution time (in s)	CAIM	0.415	0.005	64.7	0.371	0.538	0.007	54.1	0.192
	Ameva	**0.280**	0.036	**30.2**	0.596	**0.444**	0.037	**16.2**	0.543
	Ameva^R	0.554	0.019	59.0	0.316	0.582	0.007	53.9	0.368
Tree size C4.5	CAIM	58	11.7	187.2	20.1	164.2	27.0	635.7	47.5
	Ameva	**31.7**	4.0	**88.8**	12.7	**100.5**	10.4	**225.7**	30.6
	Ameva^R	54.6	8.59	204.7	16.3	170.4	8.54	601.9	36.0
% Accuracy C4.5	CAIM	67.3	6.63	**94.8**	1.69	67.4	2.22	**68.5**	3.92
	Ameva	**68.3**	6.89	90.5	2.26	**67.9**	3.89	62.5	5.07
	Ameva^R	65.4	9.30	95.0	1.80	68.1	3.00	70.8	4.5

Table 5
Results obtained by Ameva and the genetic algorithm (GA)

Criterion	Discretization method	Dataset							
		IRIS		SAT		THY		WAV	
		Mean	StdD.	Mean	StdD.	Mean	StdD.	Mean	StdD.
Total number of intervals	Ameva	10	0.000	100.4	0.699	15.6	0.699	**47.1**	3.81
	GA	10	0.000	100.4	0.843	**15.5**	.972	47.2	1.55
Execution time (in s)	Ameva	**0.010**	0.000	**3.30**	0.015	**0.896**	0.010	**7.44**	0.363
	GA	2.74	.218	234	5.07	9.17	.406	54.7	15.9
Tree size	Ameva	**4.30**	0.949	**527.9**	48.9	17.6	1.26	**318.7**	27.4
C4.5	GA	4.5	1.58	565.2	43.7	**16.9**	3.70	337	18.8
% Accuracy	Ameva	93.3	.056	**82.8**	0.056	99.1	0.056	76.8	0.056
C4.5	GA	**96.0**	0.056	81.7	0.056	99.1	0.056	**77.1**	0.056
		ION		SMO		HEA		PID	
Total number of intervals	Ameva	107.1	2.13	**7.10**	1.91	**13.6**	1.90	18.8	0.919
	GA	**104.6**	2.07	8.1	0.568	13.8	1.03	**18.6**	0.699
Execution time (in s)	Ameva	**1.49**	0.038	**0.134**	0.020	**0.042**	0.008	**0.226**	0.005
	GA	13.6	0.404	1.92	.203	1.34	0.107	2.78	0.275
Tree size C4.5	Ameva	**18.7**	3.02	1	0.000	31.2	5.29	**23.2**	7.48
	GA	20.4	4.35	1	0.000	**26.9**	4.01	24.3	6.16
% Accuracy C4.5	Ameva	88	0.056	69.6	0.056	80.8	0.056	**76.0**	0.056
	GA	**89.4**	0.056	69.6	0.056	**81.9**	0.056	75.6	0.056

tion is better-preserved for classification tasks (Kurgan, 2004). Hence, this reduction in accuracy can be justified by the selection of a lower number of intervals. Table 4 displays the results obtained when the Ameva method includes the CAIM constraint and generates at least as many intervals as classes. This version of the algorithm is called Ameva^R.

The results of Ameva^R are similar to those obtained with CAIM and present smaller differences in the four criteria under consideration than with the original Ameva.

In the number of intervals, the only difference is a 5% increase for the SEG data set. The execution time is the criterion which presents the greatest differences where Ameva^R takes from 8% less time to 25% longer than CAIM in the SEG and GLA data sets respectively.

The number of nodes in the generated tree is very similar, varying from 7% smaller to 8% greater. The accuracy also has very minor differences, where Ameva^R is more accurate than CAIM in three data sets.

This comparison demonstrates that the information lost due to the lower number of intervals selected is the reason for the accuracy reduction in some data sets using Ameva^R.

Hence, the conclusions to be drawn from this experiment are that with a high number of classes, the Ameva method drastically reduces the number of intervals, the execution time and the size of the classification tree. On the other hand, a slight reduction in accuracy can be found in some data sets due to the lower number of intervals.

6.3. Third experiment

A genetic algorithm has been designed which finds global maximum values of the Ameva criterion for $k = 2, \ldots, 2\ell$ and the same data sets have been used as in the first experiment.

A comparison between the Ameva and the genetic algorithm approaches has been realized and the results are shown in Table 5. There are very small differences between these iterative and combinatorial approaches, except when considering the execution time. The number of intervals from the discretized data sets are nearly identical. The execution time increases exponentially since the combinatorial approach must find the best value for all the possible number of intervals, and does not stop when a value is lower than the previous one. Hence execution time grows dramatically, from a 7× factor in WAV data set to a 274× factor for IRIS.

The trees generated by the C4.5 algorithm are bigger in 4 data sets, equal in 1 and smaller in 3 when applying the combinatorial approach rather than with the original Ameva, although these differences are no higher than 10%.

The improvement over the genetic algorithm, in terms of accuracy, is very low. There is an improvement in only four data sets which is never higher than 3% and usually lower than 1%. This implies that the iterative algorithm, designed in Kurgan & Cios (2004), is efficient since it finds a near-optimal solution of the maximum Ameva values. With little, if any, improvement in accuracy over Ameva and such a long execution time there is no reason for a practical implementation of the genetic algorithm.

7. Conclusions

A new algorithm for the discretization of continuous variables featuring the selection of a low number of intervals and their limits has been presented.

The Ameva algorithm uses a measure based on χ^2 as the criterion for the optimal discretization which has the minimum number of discrete intervals and minimum loss of class attribute interdependence. The algorithm uses a greedy approach, which finds local maximum values of Ameva criterion and a stopping criterion, which depends on the data with no user interaction required, that promotes a low number of intervals. When the number of classes in the data set increases, Ameva computes very reduced sets of intervals which lead to shorter execution times and smaller decision trees than those generated by CAIM. Although the identification accuracy might be slightly reduced in some situations, a drastic improvement of the other three criteria is guaranteed. A greatly improved algorithm is presented for those tasks which involve more than 2 or 3 classes and for which a quick response is required. Furthermore, even when the number of classes is only 2 or 3, the Ameva algorithm matches or is an improvement on CAIM in efficiency.

Acknowledgements

This work was partly supported by the FAMENET-InCare (TSI2006-13390-C02-02) from the Spanish Ministry of Education and Science and CUBICO (P06-TIC-02141) from Andalusian Government.

Appendix A. The Ameva algorithm

Input: Data consisting of N examples, ℓ classes, and continuous variables X_i

For every X_i do:

Step 1: Initialization of the candidate interval boundaries and the initial discretization scheme.

1.1 Find the maximum (d_k) and minimum (d_o) values of X_i.

1.2 Form a set of all distinct values of X_i, in ascending order, and initialize all possible interval boundaries B with the minimum, maximum and all the midpoints of all the adjacent pairs in the set.

1.3 Set the initial discretization scheme to $\mathscr{L}$: $\{[d_0, d_k]\}$, set GlobalAmeva $= 0$.

Step2. Consecutive additions of a new boundary which results in the locally highest value of the Ameva criterion.

2.1 Initialize $k = 1$;

2.2 Tentatively add an inner boundary, which is not already in $\mathscr{L}$, from B, and calculate the corresponding Ameva value.

2.3 After all the tentative additions have been tried, accept the one with the highest value of Ameva.

2.4 If (Ameva $>$ GlobalAmeva) then update $\mathscr{L}$ with the accepted boundary in step 2.3 and set GlobalAmeva $=$ Ameva , else terminate.

2.5 Set $k = k + 1$ and go to 2.2

Output: Discretization scheme $\mathscr{L}(k)$.

References

Angulo, C., Anguita, D., Gonzalez-Abril, L., & Ortega, J. (2008). Support vector machines for interval discriminant analysis. *Neurocomputing, 71*(7–9), 1220–1229.

Bertier, P., & Bouroche, J. (1981). Analyse des données multidimensionnelles, Presses Universitaires de France (in French).

Blake, C., & Merz, C. (1998). Uci repository of machine learning databases. UC Irvine, Information and Computer Science Department. Available from <http://www.ics.uci.edu/mlearn/MLRepository.html>.

Boullé, M. (2004). Khiops a statistical discretization methods of continuous attributes. *Machine Learning, 55*, 53–69.

Catlett, J. (1991). On changing continuous attributes into ordered discrete attributes. In: Proceedings of European working session on learning.

Ching, J., Wong, A., & Chan, K. (1995). Class-dependent discretization for inductive learning for continuous and mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 17*(7), 641–651.

Cios, K., & Kurgan, L. (2002). Hybrid inductive machine learning algorithm that generates inequalities rules. *Information Science, 163*, 37–83. Special issue of soft computing data mining.

Clark, P., & Niblett, Y. (1989). The CN2 algorithm. *Machine Learning, 3*, 261–283.

Cohen, W. (1995). Fast effective rule induction. In: Proceedings of the twelfth international conference on machine learning.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In A. Preditis & S. Russell (Eds.), *Machine learning: Proceedings of the twelfth international conference*. San Francisco: Morgan Kaufmann Publishers.

Fayyad, U., & Irani, K. (1992). On the handling of continuous valued attributes in decision tree generation. *Machine Learning, 8*, 87–102.

Gonzàlez, L., Angulo, C., Velasco, F., & Català, A. (2006). Dual unification of bi-class support vector machine formulations. *Pattern Recognition, 39*(7), 1325–1332.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning, 11*, 63–90.

Kaufman, K. A., & Michalski, R. S. (1999). Learning from inconsistent and noisy data: The AQ18 approach. In: Proceedings of 11th international symposium on methodologies for intelligence systems.

Kerber, R. (1992). Chimerge: Discretization of numeric attributes. In: *Proceedings of* the AAAI-9, 9th international conference on artificial intelligence. MIT Press.

Kumar, A., & Zhang, D. (2007). Hand-geometry recognition using entropy-based discretization. *IEEE Transactions on Information Forensics and Security, 2*(2), 181–187.

Kurgan, L. (2004). Personal communication.

Kurgan, L., & Cios, K. (2004). CAIM discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering, 16*(2), 145–153.

Le, D.-D., & Satoh, S. (2007). Ent-boost: Boosting using entropy measures for robust object detection. *Pattern Recognition Letters, 28*, 419–425.

Liu, H., & Setiono, R. (1997). Feature selection via discretization. *IEEE Transactions on Knowledge and Data Engineering, 9*(4), 642–645.

Macskassy, A., Hirsh, H., Banerjee, A., & Dayanik, A. (2003). Converting numerical classification into text classification. *Artificial Inteligence*(143), 51–77.

Quinlan, J. (1993). *C4.5 programs for machine learning*. Morgan Kaufmann.

Risvik, K. (1999). Discretization of numerical attribute – Preprocessing for machine learning, Project 45073. Knowledge Systems Group. Department of Computer and Information Science, Norwegian University of Science and Technology. N-7034 Trondheim, Norway. Available from citeseer.ist.psu.edu/249760.html.

Su, C.-T., & Hsu, J.-H. (2005). An extended chi2 algorithm for discretization of real value attributes. *IEEE Transactions on Knowledge and Data Engineering, 17*(3), 437–441.

Tay, F., & Shen, L. (2002). A modified Chi2 algorithm for discretization. IEEE Transactions on Knowledge and Data Engineering, 14(3), 666–670. Available from http://dx.doi.org/10.1109/TKDE.2002.1000349.

Vlachos, P. (2000). Statlib project repository. Available from <http://lib.stat.cmu.edu/datasets/csb/>.