CAIM Discretization Algorithm

Lukasz A. Kurgan, *Member*, *IEEE*, and Krzysztof J. Cios, *Senior Member*, *IEEE*

Abstract—The task of extracting knowledge from databases is quite often performed by machine learning algorithms. The majority of these algorithms can be applied only to data described by discrete numerical or nominal attributes (features). In the case of continuous attributes, there is a need for a discretization algorithm that transforms continuous attributes into discrete ones. This paper describes such an algorithm, called CAIM (class-attribute interdependence maximization), which is designed to work with supervised data. The goal of the CAIM algorithm is to maximize the class-attribute interdependence and to generate a (possibly) minimal number of discrete intervals. The algorithm does not require the user to predefine the number of intervals, as opposed to some other discretization algorithms. The tests performed using CAIM and six other state-of-the-art discretization algorithms show that discrete attributes generated by the CAIM algorithm almost always have the lowest number of intervals and the highest class-attribute interdependency. Two machine learning algorithms, the CLIP4 rule algorithm and the decision tree algorithm, are used to generate classification rules from data discretized by CAIM. For both the CLIP4 and decision tree algorithms, the accuracy of the generated rules is higher and the number of the rules is lower for data discretized using the CAIM algorithm when compared to data discretized using six other discretization algorithms. The highest classification accuracy was achieved for data sets discretized with the CAIM algorithm, as compared with the other six algorithms.

Index Terms—Supervised discretization, class-attribute interdependency maximization, classification, CLIP4 machine learning algorithm.

1 Introduction

In this section, we describe the existing discretization methods, provide basic definitions, and formulate goals for the CAIM algorithm.

We observe exponential growth of the amount of data and information available on the Internet and in database systems. Researchers often use machine learning (ML) algorithms to automate the processing and extraction of knowledge from data. Inductive ML algorithms are used to generate classification rules from class-labeled examples that are described by a set of numerical (e.g., 1,2,4), nominal (e.g., black, white), or continuous attributes. Some of the inductive ML algorithms like the AQ algorithm [20], [15], CLIP algorithms [5], [6], or CN2 algorithm [8], [9] can handle only numerical or nominal data, while some others can handle continuous attributes but still perform better with discrete-valued attributes [1], [17]. This drawback can be overcome by using a discretization algorithm as a front-end for a machine learning algorithm.

Discretization transforms a continuous attribute's values into a finite number of intervals and associates with each interval a numerical, discrete value. For mixed-mode (continuous and discrete) data, discretization is usually performed prior to the learning process [1], [11], [12], [22]. Discretization can be broken into two tasks. The first task is to find the number of discrete intervals. Only a few discretization algorithms perform this; often, the user must specify the number of intervals or provide a heuristic rule [3]. The second task is to find the width, or the boundaries, of the intervals given the range of values of a continuous attribute. The proposed CAIM algorithm automatically selects a number of discrete intervals and, at the same time, finds the width of every interval based on the interdependency between class and attribute values.

Discretization algorithms can be divided into two categories:

1. *Unsupervised* (or class-blind) algorithms discretize attributes without taking into account respective class labels. The two representative algorithms are equal-width and equal-frequency discretizations [4]. The equal-width discretization algorithm determines the minimum and maximum values of the discretized attribute and then divides the range into the user-defined number of equal width discrete intervals. The equal-frequency algorithm determines the minimum and maximum values of the discretized attribute, sorts all values in ascending order, and divides the range into a user-defined number of intervals so that every interval contains the same number of sorted values.
2. *Supervised* algorithms discretize attributes by taking into account the interdependence between class labels and the attribute values. The representative algorithms are: maximum entropy [27], Patterson and Niblett [21], which is built into a decision trees algorithm [23], Information Entropy Maximization (IEM) [13], and other information-gain or entropy-based algorithms [11], [29], statistics-based algorithms like ChiMerge [17] and Chi2 [19], class-attribute interdependency algorithms like CADD [3] and clustering-based algorithms like K-means discretization [26].

- *L.A. Kurgan is with the Department of Electrical and Computer Engineering, University of Alberta. E-mail: lkurgan@ece.ualberta.ca.*
- *K.J. Cios is with the Department of Computer Science and Engineering, University of Colorado at Denver, Campus Box 109, PO Box 173364, Denver, Colorado 80217. E-mail: Krys.Cios@cudenver.edu.*

Manuscript received 18 May 2001; revised 19 Mar. 2002; accepted 2 Oct. 2002.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 114171.

TABLE 1
2D Quanta Matrix for Attribute F and Discretization Scheme D

Class	Interval					Class Total
	$[d_0, d_1]$	…	$(d_{r-1}, d_r]$	…	$(d_{n-1}, d_n]$	
C_1	q_{11}	…	q_{1r}	…	q_{1n}	M_{1+}
:	:	…	:	…	:	:
C_i	q_{i1}	…	q_{ir}	…	q_{in}	M_{i+}
:	:	…	:	…	:	:
C_S	q_{S1}	…	q_{Sr}	…	q_{Sn}	M_{S+}
Interval Total	M_{+1}	…	M_{+r}	…	M_{+n}	M

Quantization methods [18] are also used to design discretization algorithms, e.g., the adaptive quantizer algorithm [2]. Since a large number of possible attribute values slows and makes inductive learning ineffective [1], one of the main goals of a discretization algorithm is to significantly reduce the number of discrete intervals of a continuous attribute. At the same time, the algorithm should maximize the interdependency between discrete attribute values and class labels, as this minimizes the information loss due to discretization. A satisfactory trade off between these two goals needs to be achieved.

The CAIM algorithm discretizes an attribute into the smallest number of intervals and maximizes the class-attribute interdependency and, thus, makes the ML task subsequently performed much easier. The algorithm automatically selects the number of discrete intervals without any user supervision. The CAIM algorithm uses class-attribute interdependency as defined in [3].

The CAIM algorithm is compared with six well-known discretization algorithms, almost always resulting in the smallest number of discrete intervals and the highest class-attribute interdependency. The CAIM algorithm and the six algorithms were used to discretize several continuous and mixed-mode data sets. The data sets were used with two ML algorithms—the CLIP4 [5], [6], and C5.0 [10] algorithms—to generate the rules. The accuracy of the rules shows that the application of the CAIM algorithm as a front-end discretization algorithm significantly improves the classification performance and reduces the number of generated rules.

2 CAIM Discretization Algorithm

In this section, we describe the newly developed discretization criterion and algorithm.

2.1 Definitions of Class-Attribute Interdependent Discretization

The CAIM algorithm's goal is to find the minimum number of discrete intervals while minimizing the loss of class-attribute interdependency. The algorithm uses class-attribute interdependency information as the criterion for the optimal discretization. We introduce several definitions after [3] to define the criterion.

A supervised classification task requires a training data set consisting of M examples, where each example belongs to only one of S classes. F indicates any of the continuous attributes from the mixed-mode data. Next, there exists a discretization scheme D on F, which discretizes the continuous domain of attribute F into n discrete intervals bounded by the pairs of numbers:

$$\mathrm{D} : \{[\mathrm{d}_0, \mathrm{d}_1], (\mathrm{d}_1, \mathrm{d}_2], \ldots, (\mathrm{d}_{n-1}, \mathrm{d}_n]\}, \tag{1}$$

where d_0 is the minimal value and d_n is the maximal value of attribute F, and the values in (1) are arranged in ascending order. These values constitute the boundary set $\{d_0, d_1, d_2, \ldots, d_{n-1}, d_n\}$ for discretization D.

Each value of attribute F can be classified into only one of the n intervals defined in (1). Membership of each value within a certain interval for attribute F may change with the change of the discretization D. The class variable and the discretization variable of attribute F are treated as two random variables defining a two-dimensional frequency matrix (called quanta matrix) that is shown in Table 1.

In Table 1, q_{ir} is the total number of continuous values belonging to the ith class that are within interval $(\mathrm{d}_{r-1}, \mathrm{d}_r]$. M_{i+} is the total number of objects belonging to the ith class and M_{+r} is the total number of continuous values of attribute F that are within the interval $(\mathrm{d}_{r-1}, \mathrm{d}_r]$, for $i = 1, 2 \ldots, S$ and $r = 1, 2, \ldots, n$.

The estimated joint probability of the occurrence that attribute F values are within the interval $D_r = (\mathrm{d}_{r-1}, \mathrm{d}_r]$ and belong to class C_i can be calculated as:

$$p_{ir} = p(C_i, D_r|F) = \frac{q_{ir}}{M}. \tag{2}$$

The estimated class marginal probability that attribute F values belong to class C_i, p_{i+}, and the estimated interval marginal probability that attribute F values are within the interval $D_r = (\mathrm{d}_{r-1}, \mathrm{d}_r]$ p_{+r} are as follows:

$$p_{i+} = p(C_i) = \frac{M_{i+}}{M}, \tag{3}$$

$$p_{+r} = p(D_r|F) = \frac{M_{+r}}{M}. \tag{4}$$

The Class-Attribute Mutual Information between the class variable C and the discretization variable D for attribute F given the 2D frequency matrix shown in Table 1 is defined as:

$$I(C, D|F) = \sum_{i=1}^{S} \sum_{r=1}^{n} p_{ir} \log_2 \frac{p_{ir}}{p_{i+}p_{+r}}. \tag{5}$$

Similarly, the Class-Attribute Information [12] and the Shannon's entropy of the given matrix are defined, respectively, as:

$$INFO(C, D|F) = \sum_{i=1}^{S} \sum_{r=1}^{n} p_{ir} \log_2 \frac{p_{+r}}{p_{ir}}, \quad (6)$$

$$H(C, D|F) = \sum_{i=1}^{S} \sum_{r=1}^{n} p_{ir} \log_2 \frac{1}{p_{ir}}. \quad (7)$$

Given (5), (6), and (7), the Class-Attribute Interdependence Redundancy (CAIR) criterion [28] (8) and Class-Attribute Interdependence Uncertainty (CAIU) [14] (9) criterion are defined as follows:

$$R(C, D|F) = \frac{I(C, D|F)}{H(C, D|F)}, \quad (8)$$

$$U(C, D|F) = \frac{INFO(C, D|F)}{H(C, D|F)}. \quad (9)$$

The CAIR criterion is used in the Class-Attribute Dependent Discretizer (CADD) algorithm [3]. The CAIR criterion [7] is used to measure the interdependence between classes and the discretized attribute (the larger its value, the better correlated are the class labels and the discrete intervals). It is also independent of the number of class labels and the number of unique values of the continuous attribute. The same holds true for the CAIU criterion, but with a reverse relationship. The CADD algorithm has the following disadvantages:

- It uses a user-specified number of intervals when initializing the discretization intervals.
- It initializes the discretization intervals using a maximum entropy discretization method; such initialization may be the worst starting point in terms of the CAIR criterion.
- The significance test used in the algorithm requires training for selection of a confidence interval.

The CAIU and CAIR criteria were both used in the CAIUR discretization algorithm [14]. The CAIUR algorithm avoided the disadvantages of the CADD algorithm generating discretization schemes with higher CAIR values, but at the expense of very high-computational cost, making it inapplicable for discretization of continuous attributes that have a large number of unique values.

The CAIM algorithm aims to:

- maximize the interdependency between the continuous-valued attribute and its class labels,
- achieve the minimum number of discrete intervals possible, and
- perform the discretization task at reasonable computational cost so that it can be applied to continuous attributes with large number of unique values.

The CAIM algorithm avoids the disadvantages of the CADD and CAIUR algorithms. It works in a top-down manner, dividing one of the existing intervals into two new intervals using a criterion that results in achieving the optimal class-attribute interdependency after the split, and starts with a single, $[\mathrm{d_o}, \mathrm{d_n}]$ interval. The idea of the CAIM algorithm and initial benchmarking results were first reported in [16]. The discretization criterion and the CAIM algorithm are described in the next two sections.

2.2 Discretization Criterion

The Class-Attribute Interdependency Maximization (CAIM) criterion measures the dependency between the class variable C and the discretization variable D for attribute F, for a given quanta matrix (see Table 1), and is defined as:

$$CAIM(C, D|F) = \frac{\sum_{r=1}^{n} \frac{\max_r{}^2}{M_{+r}}}{n}, \quad (10)$$

where n is the number of intervals, r iterates through all intervals, i.e., $\mathrm{r} = 1, 2, \ldots, \mathrm{n}$, max_r is the maximum value among all $\mathrm{q_{ir}}$ values (maximum value within the rth column of the quanta matrix), $\mathrm{i} = 1, 2, \ldots, S$, $\mathrm{M_{+r}}$ is the total number of continuous values of attribute F that are within the interval $(\mathrm{d_{r-1}}, \mathrm{d_r}]$.

The CAIM criterion is a heuristic measure that is used to quantify the interdependence between classes and the discretized attribute. It has the following properties:

- The larger the value of CAIM, the higher the interdependence between the class labels and the discrete intervals. The bigger the number of values belonging to class $\mathrm{C_i}$ within a particular interval (if the number of values belonging to $\mathrm{C_i}$ within the interval is the largest, then $\mathrm{C_i}$ is called the leading class within the interval), the higher the interdependence between $\mathrm{C_i}$ and the interval. The goal of maximizing the interdependence between classes and the discrete intervals can be translated into achieving the largest possible number of values that belong to a leading class within all intervals. The CAIM criterion accounts for the trend of maximizing the number of values belonging to a leading class within each interval by using $\mathrm{max_r}$. The value of CAIM grows when values of $\mathrm{max_r}$ grow, which relates to the increase of the interdependence between the class labels and the discrete intervals. The highest interdependence between the class labels and the discrete intervals (and, at the same time, the highest value of CAIM) is achieved when all values within a particular interval belong to the same class for all intervals. In this case, $\mathrm{max_r} = \mathrm{M_{+r}}$ and CAIM=M/n.
- It takes on real values from the interval [0, M], where M is the number of values of the continuous attribute F.
- The criterion generates discretization schemes where each interval has all of its values grouped within a single class label. This observation motivated us to use the $\mathrm{max_r}$ values within each of the n intervals and summing them for all intervals.
- The squared $\mathrm{max_r}$ value is divided by the $\mathrm{M_{+r}}$ for two reasons:

- To account for the negative impact that values belonging to classes other than the class with the maximum number of values within an interval have on the discretization scheme. The more such values the bigger the value of M_{+r}, which, in turn, decreases the value of CAIM.
- To scale the max_r^2 number. Because the division factor M_{+r} is always greater or equal to max_r, the overflow error will not happen during calculations. To avoid the overflow, the calculation is performed by first dividing max_r by M_{+r} and then multiplying the result by max_r, i.e.,

$$\frac{max_r{}^2}{M_{+r}} \; is\ calculated\ as \; \frac{max_r}{M_{+r}} max_r.$$

- Because the algorithm favors discretization schemes with smaller numbers of intervals, the summed value is divided by the number of intervals n.
- The M_{i+} values from the quanta matrix are not used because they are defined as the total number of objects belonging to the ith class, which does not change with different discretization schemes.

The value of the CAIM criterion is calculated with a single pass over the quanta matrix. The CAIM criterion has similar properties to the CAIR criterion, but the experimental results show that the CAIM criterion tends to generate a much smaller number of intervals and using it results in achieving higher interdependency. The CAIM algorithm uses the CAIM criterion as its discretization criterion.

2.3 The CAIM Algorithm

The optimal discretization scheme can be found by searching over the space of all possible discretization schemes to find the one with the highest value of the CAIM criterion. Such a search for a scheme with the globally optimal value of CAIM is highly combinatorial and time consuming. Thus, the CAIM algorithm uses a greedy approach, which searches for the approximate optimal value of the CAIM criterion by finding locally maximum values of the criterion. Although this approach does not guarantee finding the global maximum, it is both computationally inexpensive and well-approximates finding the optimal discretization scheme, which is shown in the results section. The algorithm consists of these two steps:

- initialization of the candidate interval boundaries and the initial discretization scheme and
- consecutive additions of a new boundary that results in the locally highest value of the CAIM criterion.

The pseudocode of the CAIM algorithm follows.

Given: Data consisting of M examples, S classes, and continuous attributes F_i

For every F_i do:

Step 1.

1.1 Find maximum (d_n) and minimum (d_o) values of F_i.

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

1.3 Set the initial discretization scheme as $D : \{[d_0, d_n]\}$, set GlobalCAIM=0.

Step 2.

2.1 Initialize $k = 1$.

2.2 Tentatively add an inner boundary, which is not already in D, from B, and calculate corresponding CAIM value.

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

2.4 If (CAIM $>$ GlobalCAIM or $k < S$) then update D with the accepted in Step 2.3 boundary and set GlobalCAIM=CAIM, else terminate.

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

Output: Discretization scheme D

The algorithm starts with a single interval that covers all possible values of a continuous attribute and divides it iteratively. From all possible division points that are tried (with replacement) in 2.2., it chooses the division boundary that gives the highest value of the CAIM criterion. The algorithm assumes that every discretized attribute needs at least the number of intervals equal to the number of classes since this assures that the discretized attribute can improve subsequent classification.

In what follows, we estimate the complexity of the algorithm for discretizing a single attribute. The CAIM algorithm's time bound is determined by the calculation of the CAIM criterion in Step 2.2. In the worst case, the CAIM criterion is calculated in $O(M \cdot S)$ time, where M is the number of distinct values of the discretized attribute and S is the number of classes in the problem. The CAIM algorithm starts with a single interval and as experimental results in Table 3 show, the expected number of intervals per attribute is $O(S)$. Thus, the time bound for calculation of the CAIM value can be estimated as $O(S^2)$. The CAIM values are calculated for $O(M)$ boundaries in Step 2.2. This gives the total time of Step 2.2 as $O(M \cdot S^2)$. Step 2.2 is executed in the worst-case $O(M)$ times, and the results show that the expected number of intervals is again $O(S)$, thus we can estimate that Step 2.2 is executed $O(S)$ times. Thus, the time bound for Step 2 of the CAIM algorithm is $O(S) \cdot O(M \cdot S^2) = O(M \cdot S^3)$. Sorting in Step 1.2 takes $O(M \cdot \log(M))$ time and determines the time for Step 1. Depending on the value of S, which for most inductive machine learning applications is a small constant, the expected running time of the algorithm is $O(M \log(M))$. This makes the CAIM algorithm applicable to large problems.

The remaining costs of the algorithm include building the quanta matrix given the discretization scheme in $O(M)$ time (this time adds to calculating the CAIM value), updating the discretization scheme in Step 2.4 in $O(M)$ time, and updating the global CAIU value in $O(M)$ time. All these costs can be omitted from the estimation.

The CAIM algorithm achieves a balance between a reasonable computational cost and finding the optimal discretization scheme. Despite the greedy manner in which the algorithm works, the discretization schemes it generates have very high class-attribute interdependency and always

TABLE 2
Major Properties of Data Sets Considered in the Experimentation

Properties	Datasets							
	iris	sat	thy	wav	ion	smo	hea	pid
# of classes	3	6	3	3	2	3	2	2
# of examples	150	6435	7200	3600	351	2855	270	768
# of training / testing examples	10 x cross-validation	10 x cross-validation	10 x cross-validation	10 x cross-validation	10 x cross-validation	10 x cross-validation	10 x cross-validation	10 x cross-validation
# of attributes	4	36	21	21	34	13	13	8
# of continuous attributes	4	36	6	21	32	2	6	8

TABLE 3
Comparison of the Seven Discretization Schemes Using Eight Continuous and Mixed-Mode Data Sets (Bolded Values Indicate the Best Results)

Criterion	Discretization Method	Dataset																RANK
		iris	std	sat	std	thy	std	wav	std	ion	std	smo	std	hea	std	pid	std	mean
CAIR mean value through all intervals	Equal Width	0.40	0.01	0.24	0	0.071	0	0.068	0	0.098	0	0.011	0	0.087	0	0.058	0	4.0
	Equal Frequency	0.41	0.01	0.24	0	0.038	0	0.064	0	0.095	0	0.010	0	0.079	0	0.052	0	5.4
	Paterson-Niblett	0.35	0.01	0.21	0	0.144	0.01	**0.141**	0	0.192	0	0.012	0	0.088	0	0.052	0	3.5
	Maximum Entropy	0.30	0.01	0.21	0	0.032	0	0.062	0	0.100	0	0.011	0	0.081	0	0.048	0	5.9
	CADD	0.51	0.01	**0.26**	0	0.026	0	0.068	0	0.130	0	**0.015**	0	0.098	0.01	0.057	0	3.4
	IEM	0.52	0.01	0.22	0	0.141	0.01	0.112	0	**0.193**	0.01	0.000	0	0.118	0.02	0.079	0.01	3.1
	CAIM	**0.54**	0.01	**0.26**	0	**0.170**	0.01	0.130	0	0.168	0	0.010	0	**0.138**	0.01	**0.084**	0	**1.9**
total # of intervals	Equal Width	16	0	252	0	126	0.48	630	0	640	0	22	0.48	56	0	106	0	4.8
	Equal Frequency	16	0	252	0	126	0.48	630	0	640	0	22	0.48	56	0	106	0	4.8
	Paterson-Niblett	48	0	432	0	45	0.79	252	0	384	0	17	0.52	48	0.53	62	0.48	4.0
	Maximum Entropy	16	0	252	0	125	0.52	630	0	572	6.70	22	0.48	56	0.42	97	0.32	4.4
	CADD	16	0.71	246	1.26	84	3.48	628	1.43	536	10.26	22	0.48	55	0.32	96	0.92	3.6
	IEM	**12**	0.48	430	4.88	28	1.60	91	1.50	113	17.69	**2**	0	**10**	0.48	17	1.27	2.1
	CAIM	**12**	0	**216**	0	**18**	0	**63**	0	**64**	0	6	0	12	0	**16**	0	**1.3**
time [s]	Equal Width	**0.02**	0.01	26.63	2.02	**4.74**	0.05	**8.04**	0.26	**1.21**	0.02	0.32	0.01	**0.08**	0.01	**0.27**	0.01	**1.3**
	Equal Frequency	0.03	0.01	**26.11**	0.55	4.85	0.16	8.46	0.21	1.29	0.06	**0.31**	0.01	**0.08**	0	**0.27**	0.01	1.5
	Paterson-Niblett	0.12	0.01	82.52	2.36	32.98	1.60	176.8	4.13	14.35	2.75	1.44	0.06	0.42	0.01	2.44	0.01	6.4
	Maximum Entropy	0.03	0	30.36	1.42	11.01	0.65	26.76	1.66	3.33	0.39	0.45	0.05	0.14	0.02	0.58	0.05	3.5
	CADD	0.08	0.01	54.19	1.81	65.46	13.62	435.5	24.68	17.88	0.65	1.05	0.16	0.69	0.05	3.91	0.30	6.6
	IEM	0.05	0	49.90	1.77	10.56	0.35	59.14	3.16	2.57	0.32	0.53	0.01	0.14	0.01	0.77	0.05	4.1
	CAIM	0.05	0.01	53.36	1.90	11.50	0.47	46.13	3.68	2.51	0.25	0.64	0.01	0.13	0.01	0.70	0.01	4.1

a small number of discretization intervals. For the data sets used in the experimental section, the CAIM algorithm generates discretization schemes with the (possibly) smallest number of intervals that assures low-computational cost and always achieves very high class-attribute interdependency, which results in significant improvement in the subsequently performed classification task.

3 Results

In the following sections, the results of the CAIM algorithm along with six other leading discretization algorithms on the eight well-known continuous and mixed-mode data sets are presented.

3.1 The Experimental Setup

The eight data sets used to test the CAIM algorithm are:

1. Iris Plants data set (*iris*),
2. Johns Hopkins University Ionosphere dataset (*ion*),
3. Statlog Project Heart Disease data set (*hea*),
4. Pima Indians Diabetes data set (*pid*)
5. Statlog Project Satellite Image data set (*sat*),
6. Thyroid Disease data set (*thy*),
7. Waveform data set (*wav*), and
8. Attitudes Towards Workplace Smoking Restrictions data set (*smo*).

The first seven data sets were obtained from the UC Irvine ML repository [25]; the last data set was obtained from the StatLib data set archive [24]. A detailed description of the data sets is shown in Table 2.

Tests were performed for the CAIM algorithm and six other discretization algorithms. The six discretization algorithms were:

- two unsupervised algorithms: equal-width and equal frequency and
- four supervised algorithms: Patterson-Niblett, IEM, Maximum Entropy, and CADD.

The unsupervised algorithms require the user to specify the number of discrete intervals. In our experiments, we used the following heuristic formula [27] to estimate the number of intervals: $n_{Fi} = M/(3C)$, where n_{Fi} is the number of intervals for attribute Fi, M is the number of examples,

and C is the number of classes. The supervised algorithms apply their own criteria to generate an appropriate number of discrete intervals.

All seven algorithms were used to discretize the eight data sets. The goodness of the discretization algorithm was evaluated based on the CAIR criterion value, the number of generated intervals, and the execution time.

To quantify the impact of the selection of a discretization algorithm on the classification task performed subsequently by a machine learning algorithm, the discretized data sets were used to generate classification rules by ML algorithms. The ML algorithms can be divided into rule, decision tree, and hybrid algorithms [7]. We used the CLIP4 algorithm [5], [6] to represent the family of hybrid algorithms that generate rules, and the C5.0 algorithm [10] to represent the family of decision tree algorithms. The classification goodness was measured using accuracy and the number of rules. The results were compared among the seven discretization algorithms for all data sets and both learning algorithms.

3.2 Analysis of the Results

3.2.1 Analysis of the Discretization Results

Evaluation of the discretization algorithms was performed using the CAIR criterion [28] since one of the goals of discretization is to maximize the class-attribute interdependence. After [3], this can be done by finding a discretization scheme, D_{MAX}, out of all possible discretization schemes, D, such that: $CAIR(D_{MAX}) \geq CAIR(D_i) \forall (D_i \in D)$.

The CAIM criterion has the same properties as the CAIR criterion, but, since it is a new heuristic measure, the CAIR criterion was used instead. The higher the value of the CAIR criterion, the higher the interdependence between the class labels and the discrete intervals. Table 3 shows the CAIR value, the number of discrete intervals, and the execution time for the 10-fold cross validation tests on eight data sets and the seven discretization schemes. The discretization was done using the training folds and the testing folds were discretized using the already generated discretization scheme. The discretized data sets were used in Section 3.2.2. The direct comparison of results can be performed by looking at the rank column in Table 3. The rank value is defined as each algorithm's rank for a particular data set among the seven algorithms, averaged over the eight data sets.

The CAIM algorithm achieved the highest class-attribute interdependency for five out of eight data sets, and, for *wav* and *ion*, data sets had the second and third highest, respectively. The CAIM algorithm was behind the competitors for only the *smo* data set, but this data set has only two continuous attributes out of 13. For this test, the CAIM algorithm achieved the highest rank (1.9) among all compared algorithms and this rank is significantly better than 3.1 achieved by the Information Entropy Maximization algorithm, which was the second best. The results show that the greedy approach combined with the CAIM criterion work, in practice, resulted, on average, in higher interdependence between class and attribute variables than the interdependence achieved by other algorithms.

The CAIM-generated discretization schemes have the following mean (through all intervals and experiments) values of the CAIM criterion (mean CAIM value/upper boundary of CAIM value): for *iris* 33.3/135 (std 0.5), for *sat* 562.3/5791 (std 5.4), for *thy* 5554.5.6/6480 (std 11.8), for *wav* 457.0/3240 (std 2.9), for *ion* 129.5/315 (std 2.2), for *smo* 1242.1/2569 (std 10.4), for *pid* 293.0/691 (std 6.1), and for *hea* 75.0/243 (std 2.9). The upper boundary is the number of examples, as shown in Section 2.2. For some of the data sets, like *thy*, *pid*, and *smo*, the achieved CAIM value was high in comparison to its upper boundary, but, in general, its value depends on the distribution of the values belonging to different classes. Since the computation of the globally optimal value of the CAIM criterion is computationally expensive (the number of all possible discretization schemes that need to be considered is highly combinatorial), we did not compare the achieved CAIM values to the optimal values. Instead, we used the CAIR criterion to show the performance, in terms of interdependence between the class and attribute variables, for the CAIM algorithm.

The CAIM algorithm generated a discretization scheme with the smallest number of intervals for six data sets, as compared with six other discretization algorithms. For the *smo* and *hea* data sets, it generated the second smallest number of intervals. Again, the rank of CAIM was significantly better than the ranks of other discretization algorithms. Smaller numbers of discrete intervals reduce the size of the data and helps to better understand the meaning of the discretized attributes. This is a significant advantage of the CAIM algorithm that further shows its usefulness.

Unsupervised discretization algorithms achieved the shortest execution time since they do not process any class related information; they require less computation time and generate results that are less suited for the subsequent ML tasks. Among supervised algorithms, the Maximum Entropy algorithm achieved the best average rank. The second fastest were IEM and CAIM algorithms; they worked well on larger data sets like *thy* or *wav*, which is important for real-life applications. The results for IEM, CAIM, and Maximum Entropy algorithms show that they are the fastest among supervised methods, with comparable performance.

The above results show that the CAIM algorithm generates small numbers of intervals that are highly interdependent with class labels, with speeds comparable to the fastest state-of-the-art supervised discretization algorithms.

3.2.2 Analysis of the Classification Results Using the Discretized Data Sets

The discretized data sets generated in Section 3.2.1, were used as input to CLIP4 and C5.0 algorithms to generate classification rules. The accuracy and the number of rules were compared for the seven discretization algorithms. Since C5.0 can generate data models from continuous attributes, we compared its performance while it generated rules from raw data against the results achieved using discretized data using the seven algorithms. Direct

TABLE 4
Comparison of the Accuracies Achieved by the CLIP4 and C5.0 Algorithms for the Eight Data Sets Using the Seven Discretization Schemes (Bolded Values Indicate the Best Results)

Algor.	Discretization Method	Datasets																RANK mean
		iris		Sat		thy		wav		ion		smo		hea		pid		
		acc	std	acc	std	acc	std	acc	std	acc	std	acc	std	acc	std	acc	std	
CLIP4 accuracy	Equal Width	88.0	6.9	**77.5**	2.8	91.7	1.9	68.2	2.2	86.9	6.4	68.6	2.0	64.5	10.1	65.5	6.5	4.6
	Equal Frequency	91.2	7.6	76.3	3.4	95.7	2.4	65.4	2.9	81.0	3.7	68.9	2.8	72.6	8.2	63.3	6.3	4.8
	Paterson-Niblett	87.3	8.6	75.6	3.9	97.4	0.6	60.9	5.2	**93.7**	3.9	68.9	2.7	68.5	13.6	72.7	5.1	4.3
	Maximum Entropy	90.0	6.5	76.4	2.7	97.3	0.9	63.5	2.9	82.9	4.8	68.7	2.8	62.6	9.8	63.4	5.1	5.3
	CADD	**93.3**	4.4	**77.5**	2.6	70.1	13.9	61.5	3.4	88.8	3.1	68.8	2.5	72.2	11.4	65.5	4.2	3.9
	IEM	92.7	4.9	77.2	2.7	**98.8**	0.5	75.2	1.7	92.4	6.9	66.9	2.6	75.2	8.6	72.2	4.2	2.9
	CAIM	92.7	8.0	76.4	2.0	97.9	0.4	**76.0**	1.9	92.7	3.9	**69.8**	4.0	**79.3**	5.0	**72.9**	3.7	**1.8**
C5.0 accuracy	Equal Width	94.7	5.3	86.0	1.6	95.0	1.1	57.7	8.2	85.5	6.4	69.2	5.4	74.7	5.2	70.8	2.8	5.3
	Equal Frequency	94.0	5.8	85.1	1.5	97.6	1.2	57.5	7.9	81.0	12.4	70.1	1.7	69.3	5.7	70.3	5.4	6.0
	Paterson-Niblett	94.0	4.9	83.0	1.0	97.8	0.4	74.8	5.6	85.0	8.1	70.1	3.2	**79.9**	7.1	71.7	4.4	4.3
	Maximum Entropy	93.3	6.3	85.2	1.5	97.7	0.6	55.5	6.2	86.5	8.8	70.2	3.9	73.3	7.6	66.4	5.9	5.6
	CADD	93.3	5.4	86.1	0.9	93.5	0.8	56.9	2.1	77.5	11.9	70.2	4.7	73.6	10.6	71.8	2.2	5.4
	IEM	**95.3**	4.5	84.6	1.1	99.4	0.2	**76.6**	2.1	**92.6**	2.9	69.7	1.6	73.4	8.9	**75.8**	4.3	3.3
	CAIM	**95.3**	4.5	86.2	1.7	98.9	0.4	72.7	4.2	89.0	5.2	**70.3**	2.9	76.3	8.9	74.6	4.0	**2.1**
	Built-in	92.7	9.4	**86.4**	1.7	**99.8**	0.4	72.6	3.6	87.0	9.5	70.1	1.3	76.8	9.9	73.7	4.9	3.3

TABLE 5
Comparison of the Number of Rules/Leaves Generated by the CLIP4 and C5.0 Algorithms for the Eight Data Sets Using the Seven Discretization (Bolded Values Indicate the Best Results)

Algor.	Discretization Method	Datasets																RANK mean
		iris		sat		thy		wav		ion		smo		pid		hea		
		#	std	#	std	#	std	#	std	#	std	#	std	#	std	#	std	
CLIP4 # rules	Equal Width	4.2	0.4	47.9	1.2	**7.0**	0.0	**14.0**	0.0	**1.1**	0.3	20.0	0.0	7.3	0.5	7.0	0.5	3.8
	Equal Frequency	4.9	0.6	47.4	0.8	**7.0**	0.0	**14.0**	0.0	1.9	0.3	19.9	0.3	7.2	0.4	6.1	0.7	3.5
	Paterson-Niblett	5.2	0.4	**42.7**	0.8	**7.0**	0.0	**14.0**	0.0	2.0	0.0	19.3	0.7	**1.4**	0.5	7.0	1.1	2.6
	Maximum Entropy	6.5	0.7	47.1	0.9	**7.0**	0.0	**14.0**	0.0	2.1	0.3	19.8	0.6	7.0	0.0	**6.0**	0.7	3.6
	CADD	4.4	0.7	45.9	1.5	**7.0**	0.0	**14.0**	0.0	2.0	0.0	20.0	0.0	7.1	0.3	6.8	0.6	3.5
	IEM	4.0	0.5	44.7	0.9	**7.0**	0.0	**14.0**	0.0	2.1	0.7	18.9	0.6	3.6	0.5	8.3	0.5	3.0
	CAIM	**3.6**	0.5	45.6	0.7	**7.0**	0.0	**14.0**	0.0	1.9	0.3	**18.5**	0.5	1.9	0.3	7.6	0.5	**2.1**
C5.0 # rules	Equal Width	6.0	0.0	348.5	18.1	31.8	2.5	69.8	20.3	32.7	2.9	**1.0**	0.0	249.7	11.4	66.9	5.6	4.9
	Equal Frequency	4.2	0.6	367.0	14.1	56.4	4.8	56.3	10.6	36.5	6.5	**1.0**	0.0	303.4	7.8	82.3	0.6	5.8
	Paterson-Niblett	11.8	0.4	**243.4**	7.8	15.9	2.3	**41.3**	8.1	18.2	2.1	**1.0**	0.0	58.6	3.5	58.0	3.5	3.3
	Maximum Entropy	6.0	0.0	390.7	21.9	42.0	0.8	63.1	8.5	32.6	2.4	**1.0**	0.0	306.5	11.6	70.8	8.6	5.8
	CADD	4.0	0.0	346.6	12.0	35.7	2.9	72.5	15.7	24.6	5.1	**1.0**	0.0	249.7	15.9	73.2	5.8	4.9
	IEM	**3.2**	0.6	466.9	22.0	34.1	3.0	270.1	19.0	12.9	3.0	**1.0**	0.0	**11.5**	2.4	**16.2**	2.0	3.5
	CAIM	**3.2**	0.6	332.2	16.1	**10.9**	1.4	58.2	5.6	**7.7**	1.3	**1.0**	0.0	20.0	2.4	31.8	2.9	**1.9**
	Built-in	3.8	0.4	287.7	16.6	11.2	1.3	46.2	4.1	11.1	2.0	1.4	1.3	35.0	9.3	33.3	2.5	3.1

comparison of results can be seen by looking at the RANK column in Table 4 that shows the accuracy.

On average, the best accuracy for the two learning algorithms was achieved for the data that was discretized using the CAIM algorithm. Using CLIP4 and C5.0 to generate a data model, the difference between the rank achieved by the CAIM algorithm and the next best IEM algorithm, and built-in discretization, in the case of C5.0, is over 1.0. In the case of using the CLIP4 algorithm to generate a data model, the average accuracy of the rules was the highest for data discretized with the CAIM algorithm. The second best accuracy was achieved for the data discretized with the IEM algorithm, while accuracies using the remaining discretization algorithms were lower and comparable to each other.

The averaged accuracy of rules generated by the C5.0 algorithm shows that the best results are achieved after discretization of data with the CAIM algorithm. The second best results were achieved by discretizing data using the IEM algorithm and C5.0 with its built-in discretization. Discretization using the remaining algorithms resulted in achieving significantly worse accuracies on average. The accuracy results show that the CAIM algorithm generates the discrete data that results in improved performance of subsequently used supervised learning algorithms when compared to the data generated by the other discretization algorithms. Table 5 shows the classification results in terms of number of generated rules.

The rank achieved by the CAIM algorithm, for experiments performed with CLIP4 and C5.0 algorithms, shows that, on average, it had the smallest number of rules. Closer analysis of the results shows that the CLIP4 algorithm generates a small number of rules for all data sets discretized using the seven discretization algorithms. The

average rank results show that discretizing data using Paterson-Niblett algorithm resulted in an average number of rules similar to the number of rules for data models generated using data discretized with the CAIM algorithm. On the other hand, the number of leaves (rules) generated by the C5.0 algorithm varied significantly over the data sets. The three discretization algorithms that work best with the C5.0 algorithm are: the CAIM algorithm, the Paterson-Niblett algorithm, and the IEM algorithm. Also, similarly low numbers of leaves were generated when using the C5.0's built-in discretization. Among these four discretization algorithms, discretizing the data using the CAIM algorithm resulted in, on average, the smallest number of leaves.

The comparison between the CAIM algorithm and the other six state-of-the-art discretization algorithms shows that it generates discretization schemes with the highest, on average, interdependency between the discrete intervals and the class labels and the smallest number of intervals. The classification results using the discretized data show that the CAIM algorithm significantly improves accuracy of the results achieved by the subsequently used ML algorithms and reduces the number of rules generated by the CLIP4 algorithm and the size of trees generated by the decision tree algorithm. The same results were achieved when comparing the CAIM algorithm to the C5.0 built-in discretization. CAIM discretizes the data with speeds that are comparable to the speeds of the two fastest supervised discretization algorithms used in the experiments, which shows its potential for large applications.

The future work will include the expansion of the CAIM algorithm so it can remove irrelevant or redundant attributes after the discretization is performed. This task can be performed by the application of the χ^2 methods [17], [19]. This would, in turn, reduce the dimensionality of the discretized data in addition to the already reduced number of values for each attribute.

4 Summary and Conclusions

In this paper, we proposed the CAIM algorithm that handles continuous and mixed mode attributes. The CAIM algorithm is a supervised discretization algorithm. The tests show that, when the proposed algorithm is applied as a front-end tool, it improves the performance of supervised ML algorithms. The algorithm works with any class-labeled data and is not limited to a particular learning algorithm. The CAIM algorithm does not require user interaction and performs automatic selection of the number of discrete intervals, in contrast to some other discretization algorithms.

The CAIM algorithm maximizes mutual class-attribute interdependence and possibly generates the smallest number of intervals for a given continuous attribute. It was tested on several well-known data sets and compared with six other state-of-the-art discretization algorithms. The comparison shows that the CAIM algorithm generates discretization schemes with, on average, the lowest number of intervals and the highest dependence between class labels and discrete intervals, thus outperforming other discretization algorithms. The execution time of the CAIM algorithm is much shorter than the execution time of some other supervised discretization algorithms, being at the same time comparable to the time of the two fastest supervised discretization algorithms. The CAIM and six other discretization algorithms were also tested with two ML algorithms. The tests show that the CAIM algorithm significantly improved the accuracy of rules generated by CLIP4 and C5.0 algorithms achieving results much better than the other algorithms. Significant reduction in the number of leaf nodes and rules is also achieved when using the CAIM algorithm in conjunction with the decision tree algorithm and CLIP4 algorithm, respectively. The analysis of performance of the CAIM algorithm shows that the small number of intervals that the algorithm generates helps to reduce the size of the data and improves the accuracy and the number of subsequently generated rules.

In a nutshell, the CAIM algorithm is very effective and easy to use supervised discretization algorithm that can be applied to problems that require discretization of large data sets.

Acknowledgements

The authors would like to thank the reviewers for their suggestions and Mr. Michael Trombley for his assistance in improving the readability of the paper.

References

[1] J. Catlett, "On Changing Continuous Attributes into Ordered Discrete Attributes," *Proc. European Working Session on Learning,* pp. 164-178, 1991.

[2] C.C. Chan, C. Bartur, and A. Srinivasasn, "Determination of Quantization Intervals in Rule Based Model for Dynamic Systems," *Proc. IEEE Conf. System, Man, and Cybernetics,* pp. 1719-1723, 1991.

[3] J.Y. Ching, A.K.C. Wong, and K.C.C. Chan, "Class-Dependent Discretization for Inductive Learning from Continuous and Mixed Mode Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 7, pp. 641-651, July 1995.

[4] D. Chiu, A. Wong, and B. Cheung, "Information Discovery through Hierarchical Maximum Entropy Discretization and Synthesis," *Knowledge Discovery in Databases,* G. Piatesky-Shapiro and W.J. Frowley, ed., MIT Press, 1991.

[5] K.J. Cios and L. Kurgan, "Hybrid Inductive Machine Learning Algorithm That Generates Inequality Rules," *Information Sciences,* special issue on soft computing data mining, accepted, 2002.

[6] K.J. Cios and L. Kurgan, "Hybrid Inductive Machine Learning: An Overview of CLIP Algorithms," *New Learning Paradigms in Soft Computing,* L.C. Jain and J. Kacprzyk, ed., pp. 276-322, Physica-Verlag (Springer), 2001.

[7] K.J. Cios, W. Pedrycz, and R. Swiniarski, *Data Mining Methods for Knowledge Discovery.* Kluwer, 1998, http://www.wkap.nl/book.htm/0-7923-8252-8.

[8] P. Clark and T. Niblett, "The CN2 Algorithm," *Machine Learning,* vol. 3, pp. 261-283, 1989.

[9] P. Clark and R. Boswell, "Rule Induction with CN2: Some Recent Improvements," *Proc. European Working Session on Learning,* 1991.

[10] Data Mining Tools, http://www.rulequest.com/see5-info.html, 2003.

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

[12] U.M. Fayyad and K.B. Irani, "On the Handling of Continuous-Valued Attributes in Decision Tree Generation," *Machine Learning,* vol. 8, pp. 87-102, 1992.

[13] U.M. Fayyad and K.B. Irani, "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning," *Proc. 13th Int'l Joint Conf. Artificial Intelligence,* pp. 1022-1027, 1993.

[14] W. Huang, "Discretization of Continuous Attributes for Inductive Machine Learning," master's thesis, Dept. Computer Science, Univ. of Toledo, Ohio, 1996.
[15] K.A. Kaufman and R.S. Michalski, "Learning from Inconsistent and Noisy Data: The AQ18 Approach," *Proc. 11th Int'l Symp. Methodologies for Intelligent Systems,* 1999.
[16] L. Kurgan and K.J. Cios, "Discretization Algorithm that Uses Class-Attribute Interdependence Maximization," *Proc. 2001 Int'l Conf. Artificial Intelligence (IC-AI-2001),* pp. 980-987, 2001.
[17] R. Kerber, "ChiMerge: Discretization of Numeric Attributes," *Proc. Ninth Int'l Conf. Artificial Intelligence (AAAI-91),* pp. 123-128, 1992.
[18] Y. Linde, A. Buzo, R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Comm.,* vol. 28, no. 1, pp. 84-95, 1980.
[19] H. Liu and R. Setiono, "Feature Selection via Discretization," *IEEE Trans. Knowledge and Data Eng.,* vol. 9, no. 4, pp. 642-645, July/Aug. 1997.
[20] R.S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The Multi-purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains," *Proc. Fifth Nat'l Conf. Artificial Intelligence,* pp. 1041-1045, 1986.
[21] A. Paterson and T.B. Niblett, *ACLS Manual.* Edinburgh: Intelligent Terminals, Ltd, 1987.
[22] B. Pfahringer, "Compression-Based Discretization of Continuous Attributes," *Proc. 12th Int'l Conf. Machine Learning,* pp. 456-463, 1995.
[23] J.R. Quinlan, *C4.5 Programs for Machine Learning.* Morgan-Kaufmann, 1993.
[24] P. Vlachos, StatLib Project Repository, http://lib.stat.cmu.edu/datasets/csb/, 2000.
[25] C.L. Blake and C.J. Merz, UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, UC Irvine, Dept. Information and Computer Science, 1998.
[26] J.T. Tou and R.C. Gonzalez, *Pattern Recognition Principles.* Addison-Wesley, 1874.
[27] A.K.C. Wong and D.K.Y. Chiu, "Synthesizing Statistical Knowledge from Incomplete Mixed-Mode Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 9, pp. 796-805, 1987.
[28] A.K.C. Wong and T.S. Liu, "Typicality, Diversity and Feature Pattern of an Ensemble," *IEEE Trans. Computers,* vol. 24, pp. 158-181, 1975.
[29] X. Wu, "A Bayesian Discretizer for Real-Valued Attributes," *The Computer J.,* vol. 39, 1996.

Lukasz A. Kurgan received the MSc degree (recognized by an Outstanding Student Award) in automatics and robotics in 1999 from the AGH University of Science and Technology, Krakow, Poland, and the PhD degree in computer science in 2003 from the University of Colorado at Boulder. During his studies, he was awarded with several awards, the most prestigious was the Gold Medal of Stanislaw Staszic (1999). Currently, he is an assistant professor in the Electrical and Computer Engineering Department at the University of Alberta, Edmonton, Canada. His research interests are in knowledge discovery and data mining, bioinformatics, software engineering, machine learning, and image processing. He authored and coauthored several machine learning and data mining algorithms with focus on supervised learning paradigms. He is a member of the ACM, the IEEE, and IEEE Computer Society.

Krzysztof (Krys) J. Cios is a professor and chair of the Computer Science and Engineering Department at the University of Colorado at Denver, and codirector of the CU Center for Computational Biology. He is also an affiliated professor at the CU Health Sciences Center and at CU Boulder. His primary research interests are in the area of learning algorithms, mainly inductive machine learning and networks of spiking neurons, and in data mining and knowledge discovery. He developed a family of hybrid decision-tree and rule-based inductive machine learning algorithms (for one of which he got the Norbert Wiener Outstanding Paper Award from Kybernetes) that are very efficient, and have the built-in discretization schemes and ranking of the features that allows for feature selection. His work in networks of spiking neurons resulted in their first application to some optimization problems; most notably they find a clustering structure in data without prior specification of the number of clusters. NASA, The US National Science Foundation, American Heart Association, Ohio Aerospace Institute, NATO, Colorado Institute of Technology, and the US Air Force have all funded Dr. Cios's research for a total of more than $2.4 million. He published two books on data mining, more than 100 journal articles, book chapters, and peer-reviewed conference papers. He serves on editorial boards of *IEEE Transactions on Systems, Man, and Cybernetics*, *Neurocomputing*, *IEEE Engineering in Medicine and Biology Magazine*, and *Integrative Neuroscience*. Dr. Cios has been the recipient of the Neurocomputing Best Paper Award, the University of Toledo Outstanding Faculty Research Award, and the Fulbright Senior Scholar Award. He is a senior member of the IEEE and the IEEE Computer Society.