Noname manuscript No.
(will be inserted by the editor)

ur-CAIM: Improved CAIM Discretization for Unbalanced and Balanced Data

Alberto Cano · Dat T. Nguyen · Sebastián Ventura · Krzysztof J. Cios

Received: date / Accepted: date

Abstract Supervised discretization is one of basic data preprocessing techniques used in data mining. CAIM (Class-Attribute Interdependence Maximization) is a discretization algorithm of data for which the classes are known. However, new arising challenges such as the presence of unbalanced data sets, call for new algorithms capable of handling them, in addition to balanced data. This paper presents a new discretization algorithm named ur-CAIM, which improves on the CAIM algorithm in three important ways. First, it generates more flexible discretization schemes while producing a small number of intervals. Second, the quality of the intervals is improved based on the data classes distribution, which leads to better classification performance on balanced and, especially, unbalanced data. Third, the runtime of the algorithm is lower than CAIM's. The algorithm has been designed free-parameter and it self-adapts to the problem complexity and the data class distribution. The ur-CAIM was compared with 9 well-known discretization methods on 28 balanced, and 70 unbalanced data sets. The results obtained were contrasted through non-parametric statistical tests, which show that our proposal outperforms CAIM and many of the other methods on both types of data but especially on unbalanced data, which is its significant advantage.

Keywords Supervised discretization · class-attribute interdependency maximization · unbalanced data · classification

A. Cano and S. Ventura are with the Department of Computer Science and Numerical Analysis. University of Cordoba, Spain.
S. Ventura is also with the Computer Sciences Department.
Faculty of Computing and Information Technology.
King Abdulaziz University. 21589 Jeddah (Saudi Arabia).
E-mail: {acano,sventura}@uco.es

D. T. Nguyen and K. J. Cios
Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284, USA
K. J. Cios is also with the IITiS Polish Academy of Sciences, Poland.
E-mail: {nguyendt22,kcios}@vcu.edu

1 Introduction

Discretization is a data preprocessing technique which transforms continuous attributes into discrete ones by dividing the continuous values into intervals, or bins [12,28,59]. There are two basic types of discretization methods: unsupervised and supervised [18]. Unsupervised discretization methods, such as Equal-Width (EW) and Equal-Frequency (EF) [8] do not take advantage of the class labels (even if known) in the discretization process. On the other hand, supervised methods make use of this information to generate intervals that are correlated with the data classes.

Class-Attribute Interdependence Maximization (CAIM) [44] is a top-down discretization algorithm that generates good discretization schemes. Data discretized by CAIM and used as the input of a classifier, produced high predictive accuracy on many data sets [44]. Although CAIM outperforms many other methods it has two drawbacks [54]. First, it generates discretization schemes where the number of intervals is equal or very close to the number of classes. This behavior biases the outcome of discretization regardless of the data distribution and the problem properties. Second, the formula of the CAIM criterion only takes into account the data class with the highest number of instances while it ignores all other classes. This behavior may lower the quality of the discretization scheme, in particular for unbalanced data sets. The problem of learning from unbalanced data is a challenging task in data mining that has attracted attention of both academical and industrial researchers [34,40]. Unbalanced data problems concern the performance of learning algorithms in the presence of severe class distribution skews (some classes have many times more instances than other classes). The CAIM criterion formula is biased to majority class instances and it is not capable of handling such highly unbalanced data.

This paper introduces a new algorithm, named ur-CAIM, which solves the aforementioned issues of the original CAIM algorithm. We will analyze the behavior and the performance of the original CAIM on unbalanced data. We will discuss how we can address this issue and propose an heuristic which takes into account the data class distributions. We will show that the new algorithm outperforms the original CAIM on both balanced and especially, unbalanced data sets, while generating a small number of intervals and better discretization schemes (as measured by the subsequently used classifiers), and at the lower computational cost. The ur-CAIM algorithm is free-parameter, which means that it does not require any parameter settings introduced by the user. The algorithm is capable to select automatically the most appropriate number of discrete intervals. Moreover, it overcomes the bias of the CAIM algorithm of choosing a number of intervals very close to the number of classes, which provides more flexible discretization schemes that adapt better to the specific data problem properties.

The algorithm is evaluated and compared with 9 other discretization algorithms, including well-known and recently published methods [28], on 28 balanced and 70 unbalanced data sets. Many different performance measures are used to evaluate performance of the algorithms and the discretization intervals they generate. The results from the experimental study show that it performs very well, as measured by the number of intervals, execution time, accuracy, Cohen's kappa rate [3,4] and area under the curve (AUC) [6,36]. The experimental results are contrasted through the analysis of non-parametric statistical tests [17,26], namely the Friedman [14], Holm's [35] and Wilcoxon [56] tests that evaluate whether there are statistically significant differences between the algorithms.

The remainder of this paper is organized as follows. The next section reviews related works on discretization methods. Section 3 presents the ur-CAIM algorithm. Section 4 describes the experiments performed, whose results are discussed in Section 5. Finally, Section 6 presents some concluding remarks.

2 Background

The literature review provide a vast number of related works on discretization methods. These methods are based on a wide number of heuristics such as information entropy [21], or likelihood [5], or statistics [47]. Specifically, *Kotsiantis et al.* [43] and *García et al.* [28] presented two recent surveys on discretization methods. From the theoretical perspective, they developed a categorization and taxonomy based on the main properties pointed out in previous research, and unified the notation. Empirically, they conducted an experimental study in supervised classification involving the most representative and the newest discretizers, different types of classifiers, and a large number of data sets. They concluded with a selection of some best performing discretizers, which we included in our experimental study. This set of discretization algorithms include Information Entropy Maximization (IEM) [21], Class-Attribute Interdependence Maximization (CAIM) [44], ChiMerge [41], Modified-χ^2 [53], Ameva [30], Hypercube Division-based Discretization (HDD) [58], Class-Attribute Contingency Coefficient (CACC) [54], and Interval Distance-Based Method for Discretization (IDD) [52]. These top-ranked algorithms are reviewed next.

Fayyad et al. [21] used entropy minimization heuristic for discretizing the range of a continuous-valued attribute into multiple intervals (IEM). They presented theoretical evidence for the appropriateness of this heuristic in the binary discretization algorithm used in ID3, C4, CART, etc. IEM is known to achieve both good accuracy and low number of intervals. The entropy-based heuristic defined in Equation 1 measures the class information entropy of an interval. It is based on the probabilities P in a set of examples T to belong to the class i, where C is the number of classes. The algorithm measures the entropy of partitions it may generate. The cut point of intervals are selected as the ones which minimize the entropy measure. Even the algorithm was not specifically designed for unbalanced data, the entropy takes into account the data class probabilities. Therefore, it is expected to produce appropriate discretization intervals under the presence of unbalanced data.

$$Entropy(T) = -\sum_{i=1}^{C} P(T,i)\ log\ (P(T,i)) \tag{1}$$

Kerber [41] presented ChiMerge, a general and robust algorithm that employed the χ^2 statistic to discretize numeric attributes. While the χ^2 statistic is general and should have nearly the same meaning regardless of the number of classes or examples, ChiMerge does tend to produce more intervals when there are more examples. Another shortcoming of ChiMerge is its lack of global evaluation. When deciding which intervals to merge, the algorithm only examines adjacent intervals, ignoring other surrounding intervals.

Tay et al. [53] proposed a modified χ^2 algorithm as an automated discretization method. It replaced the inconsistency check in the original χ^2 algorithm using a level of consistency which maintains the fidelity of the training set after discretization. In contrast to the original χ^2 algorithm, this modified algorithm takes into consideration the effect of the degree of freedom, that consequently results in greater accuracy. The formula for computing the χ^2 value considers the expected frequency of examples belonging to each of the data classes. Therefore, it should create appropriate discretization intervals under the presence of unbalanced data.

Gonzalez et al. [30] introduced Ameva, an autonomous discretization algorithm designed to work with supervised learning algorithms. It maximizes a contingency coefficient based on χ^2 statistics and generates a potentially minimal number of discrete intervals. The maximum value of the Ameva coefficient indicates the best correlation between the class labels and the discrete intervals, 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. Therefore, we would expect that examples from minority classes should be intervalized into partitions separated from majority classes.

Yang et al. [58] introduced a hypercube division-based (HDD) top-down algorithm for supervised discretization. The algorithm considers the distribution of both the class and continuous attributes and the underlying correlation structure in the data set to divide the continuous attribute space into a number of hypercubes. Objects within each hypercube belong to the same decision class. HDD is known to perform slow and generate high number of intervals. The algorithm is motivated after the performance of class-attribute interdependence maximization. The bias of this criterion to data classes with the most samples would decrease the quality of the produced discretization scheme.

Tsai et al. [54] proposed a static, global, incremental, supervised top-down discretization algorithm called CACC, to raise quality of the generated discretization scheme by extending the idea of contingency coefficient and combining it with the greedy search. The contingency coefficient takes into account the distribution of all samples and it is a very good criterion to measure the interdependence between partitions. However, CACC requires very long runtimes, which reduces its appeal when applying on real-world problems.

Ruiz et al. [52] introduced a method for supervised discretization based on interval distances, using a concept of neighborhood in the target's space (IDD). The method takes into consideration order of the class attribute, when it exists, so that it can be used with ordinal classes. However, the neighborhood concept suffers from data class skews and therefore, it may not be capable of producing appropriate discretization intervals under the presence of unbalanced data. Moreover, IDD may create high number of intervals depending on the distances between data examples.

There are many other discretization methods based on multiple heuristics. *Chmielewski and Grzymala-Busse* [11] presented a method of transforming any local discretization method into a global one based on cluster analysis. *Elomaa and Rousu* [19] presented a multisplitting approach and they demonstrate that the cumulative functions information gain and training set error as well as the non-cumulative function gain ratio and normalized distance measure are all well-behaved. *Grzymala-Busse* [31,32] also presented entropy driven methodologies based on dominant attribute and multiple scanning.

In spite of the large number of discretization algorithms and publications, little attention has been given to the unbalanced data discretization problem. *Janssens et al.* [38] included the concept of misclassification costs (cost-based discretization) to find an optimal multi-split. This idea followed the cost-based classification principles [46] that class-distributions may vary significantly. In order to test its performance, they compared against entropy-based and error-based discretization methods with decision tree learning.

2.1 CAIM discretization

Kurgan et al. [44] presented CAIM, a supervised discretization algorithm which maximizes the class-attribute interdependence and generates minimal number of discrete intervals. However, CAIM has two drawbacks [54]. First, the algorithm is designed to generate a number of intervals very close to the number of classes. This behavior is not flexible and it does not adapt to the specific properties of each data set distribution. Second, the CAIM criterion formula only takes into account the data class with the highest number of instances. Therefore, discretization schemes generated by CAIM for unbalanced data are biased to majority class examples. Next, we analyze the behavior of the CAIM criterion, giving special attention to its performance on unbalanced data.

Supervised discretization builds a model from a training data set, where classes are known. The data consists of M instances, where each instance belongs to only one of the S classes; F indicates any continuous attribute. We can define a discretization scheme D on F, which discretizes a continuous attribute F into n discrete intervals bounded by the pairs of numbers:

$$D : \{[d_0, d_1], (d_1, d_2], ..., (d_{\text{n-1}}, d_{\text{n}}]\} \tag{2}$$

where d_0 is the minimal value and d_n is the maximal value of attribute F, and the values in Equation 2 are arranged in ascending order.

The class variable and the discretization variable of attribute F are treated as two random variables defining a two-dimensional frequency/quanta matrix that is shown in Table 1, where q_{ir} is the total number of continuous values belonging to the *i*-th class that are within interval $(d_{r-1}, d_r]$. M_{i+} is the total number of objects belonging to the *i*-th class and M_{+r} is the total number of continuous values of attribute F that are within the interval $(d_{r-1}, d_r]$, for $i = 1, 2, ..., S$ and $r = 1, 2, ..., n$.

The Class-Attribute Interdependency Maximization criterion defines dependence between the class variable C and the discretization scheme D for attribute F as follows:

Table 1: Discretization Quanta Matrix.

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

$$CAIM(C,D|F) = \frac{1}{n} \cdot \sum_{r=1}^{n} \frac{{max_r}^2}{M_{+r}} \tag{3}$$

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

The CAIM criterion shown in Equation 3 is a heuristic measure used to quantify the interdependence between classes and the discretized attribute, and it favors a lower number of intervals for which ${max_r}^2$ is maximized. The theoretical and mathematical analysis of the formula shows that it focuses on the data class for which the number of instances is highest (majority class). However, it does not take into account data class distribution of instances in the intervals, i.e., given the same max_r and M_{+r} but different minority data class distributions, the CAIM value remains a constant value. The outcome is a discretization scheme that may not be the best for unbalanced data, and this is an important disadvantage of CAIM. Therefore, it is necessary to improve the heuristic to address the unbalanced data problem, which is the main motivation of this work.

3 ur-CAIM algorithm

This section introduces the definitions of three well-known class-attribute interdependence criteria and shows how they can be used in tandem to achieve the goal of designing a robust discretization criterion, the ur-CAIM criterion. Next, the ur-CAIM algorithm, based on the ur-CAIM criterion, is described in detail.

The estimated joint probability of the occurrence that attribute F values are within the interval $D_r = (d_{r-1}, d_r]$ and they belong to class C_i is calculated as:

$$p_{ir} = p(C_i, D_r|F) = \frac{q_{ir}}{M} \tag{4}$$

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 $D_r = (d_{r-1}, d_r]$ p_{+r} are as follows:

$$p_{i+} = p(C_i) = \frac{M_{i+}}{M} \tag{5}$$

$$p_{+r} = p(D_r|F) = \frac{M_{+r}}{M} \tag{6}$$

The class-attribute mutual information between the class variable C and the discretization variable D for attribute F, given the frequency matrix shown in Table 1, is defined as:

$$I(C,D|F) = \sum_{i=1}^{S}\sum_{r=1}^{n} p_{ir} \cdot \log_2 \frac{p_{ir}}{p_{i+} \cdot p_{+r}} \tag{7}$$

Similarly, the class-attribute information [20] and the Shannon's entropy are defined, respectively, as:

$$INFO(C,D|F) = \sum_{i=1}^{S}\sum_{r=1}^{n} p_{ir} \cdot \log_2 \frac{p_{+r}}{p_{ir}} \tag{8}$$

$$H(C,D|F) = \sum_{i=1}^{S}\sum_{r=1}^{n} p_{ir} \cdot \log_2 \frac{1}{p_{ir}} \tag{9}$$

Given Equations 7, 8, and 9, the Class-Attribute Interdependence Redundancy (CAIR) [57] criterion and Class-Attribute Interdependence Uncertainty (CAIU) [37] criteria are defined as follows:

$$CAIR(C,D|F) = \frac{I(C,D|F)}{H(C,D|F)} \tag{10}$$

$$CAIU(C,D|F) = \frac{INFO(C,D|F)}{H(C,D|F)} \tag{11}$$

The CAIR criterion is used to measure the interdependence between classes and the discretized attribute (the larger the value the better correlated are the class labels with discrete intervals). It is 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. Namely, CAIU optimizes for discretization schemes with larger number of intervals. Both CAIR and CAIU values are in the range [0,1]. On the other hand, the CAIM criterion can be normalized to the range [0,1] as follows:

$$CAIM_N(C, D|F) = \frac{1}{M} \cdot \sum_{r=1}^{n} \frac{{max_r}^2}{M_{+r}} \tag{12}$$

To address the unbalanced data problem, we introduced the unbalanced ratio factor (class probability p_i) into the formulas by means of the class-attribute mutual information (I) defined in Equation 7. Thus, we redefine the class-attribute mutual information in Equation 13, and consequently the CAIR factor is modified to handle unbalanced data more appropriately.

$$I'(C, D|F) = \sum_{i=1}^{S} \sum_{r=1}^{n} (1 - p_{i+}) \cdot p_{ir} \cdot \log_2 \frac{p_{ir}}{p_{i+} \cdot p_{+r}} \tag{13}$$

All of the above described criteria serve for different discretization goals and cover different aspects of the discretization task. We decided to combine them to propose the new criterion, called ur-CAIM, which combines the CAIM, CAIR (modified) and CAIU into one, which is defined as follows:

$$ur\text{-}CAIM = CAIM_N \cdot \; CAIR \cdot \; (1 - CAIU) \tag{14}$$

This way, the ur-CAIM criterion takes into account possibly unbalanced classes, so that minority class instances are not “squashed” by instances from classes with much larger number of instances. As a result, it improves generation of intervals for the under-represented classes with small number of instances.

Figure 1 shows and analyzes the discretization behavior of CAIM and ur-CAIM on an extremely unbalanced simple data. Positive class (minority) and negative class (majority) instances are located in a continuous attribute domain. We want discretization to successfully separate the minority class instance by taking into consideration the data class distribution. The minority class is under-represented and overlapped with the other class. Thus, it should be discretized even though it would mean covering a higher number of instances from the negative class. The question is: is the success ratio for the positive class worth the failure ratio of the negative class?. The answer is that if the interval had not been discretized, a classification algorithm, subsequently used, would be set for almost certain failure because the minority class instance is included in intervals where majority class instances prevail. It is better to fail the prediction of the two negative examples that failing the prediction of the minority positive example.

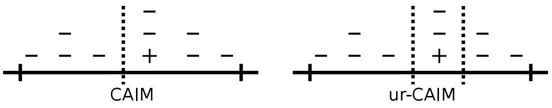

Fig. 1: CAIM and ur-CAIM unbalanced discretization.

Algorithm 1 ur-CAIM algorithm

Input: Data of M instances, S classes, and F attributes

```
1: for every $F_i$ do
2:     Sort all distinct values of $F_i$ in ascending order.
3:     Find the minimum $d_{min}$, maximum $d_{max}$ values of $F_i$.
4:     Initialize interval boundaries $B$ with $d_{min}$, $d_{max}$, and all midpoints of adjacent pairs in the set.
5:     Set discretization scheme $D$ = $\{[d_{min}, d_{max}]\}$.
6:     $\text{ur-CAIM}_{\text{D}} \leftarrow$ ur-CAIM value of $D$.
7:     Evaluate the ur-CAIM value of the tentative schemes using $D$ and the points from $B$.
8:     $\text{ur-CAIM}_{\max} \leftarrow$ Select the highest valued midpoint.
9:     if ($\text{ur-CAIM}_{\max} > \text{ur-CAIM}_{\text{D}}$) then
10:        Update $D$ with the midpoint from $\text{ur-CAIM}_{\max}$.
11:        Go to step 6.
12:    else
13:        return Discretization scheme $D$ for attribute $F_i$.
14:    end if
15: end for
```

Output: Discretization scheme for all attributes

The ur-CAIM criterion represents a trade-off for dealing with the number of intervals. The CAIM part of the formula advocates for a more generalized scheme with lower number of intervals, whereas the CAIR and CAIU advocate for the larger number. The ur-CAIM criterion thus allows for evaluating different behaviors of different metrics and presents a single-value quality measure of the discretization scheme that works well on both balanced and unbalanced data, as will be shown in the experimentation.

The ur-CAIM algorithm is based on the ur-CAIM criterion, which evaluates the quality of the tentative discretization schemes and finds the one with the highest ur-CAIM value. Discretization schemes are iteratively improved by splitting the feature domain into intervals. The algorithm procedure is shown in Algorithm 1. It follows a top-down scheme, similar to CAIM, IEM and HDD algorithms. It first initializes the tentative discretization intervals based on the attribute values present on the data set. It evaluates the ur-CAIM formula for each of the intervals and it selects the one with the highest value. The stop criterion is triggered when the ur-CAIM value is not further improved.

In contrast to the CAIM algorithm, the ur-CAIM does not use any assumption, such as that every discretized attribute needs at least the number of intervals that are equal to the number of classes. Therefore, the ur-CAIM algorithm is free-parameter and it self-adapts automatically to the data problem properties. The ur-CAIM complexity is O($m\ log\ m$), where m is the number of distinct values of the discretized attribute. We designed a fast implementation of the ur-CAIM criterion computation that minimizes the number of calculations by reusing the quanta matrix values. Moreover, the discretization process for each attribute is an independent operation and therefore, current multi-core CPUs can take advantage of the concurrent computation of discretization for each attribute. This makes the ur-CAIM algorithm fast and scalable to large data. Details about execution times are provided in the experiments section.

4 Experiments

This section describes experiments performed to evaluate the performance and compare ur-CAIM with other discretization algorithms. First, performance measures used in evaluation of the algorithms are presented. Second, information about the data sets and algorithms is detailed. Finally, the tests for the statistical analysis are presented.

4.1 Performance measures

There are many performance measures to evaluate discretization methods and the quality of the discretization schemes generated. Different measures allow to observe different behavior of algorithms. Evaluating different complementary measures increases the strength of the experimental study.

Two direct measures are the number of intervals created and the execution time of the algorithms. The number of intervals evaluates complexity of the discretization scheme. The lower the number of intervals the simpler discretization, but it its important to highlight that too simple discretization schemes may lead to worse classification performance. The computational cost of the algorithms is especially relevant for their scalability to large data sets, not only in terms of the number of data instances but also their dimensionality. The quality of the intervals generated is usually evaluated in terms of the classification error [45].

The most frequently used performance measure for classification is accuracy, but unfortunately it may be misleading when classes are strongly unbalanced. In this situation a default-hypothesis classifier could achieve a very good accuracy by just predicting the majority class. Therefore, we should perform evaluation of the discretization by using also other measures. These measures are based on the values of the confusion matrix, where each column of the matrix represents the count of instances in a predicted class, while each row represents the number of instances in the actual class.

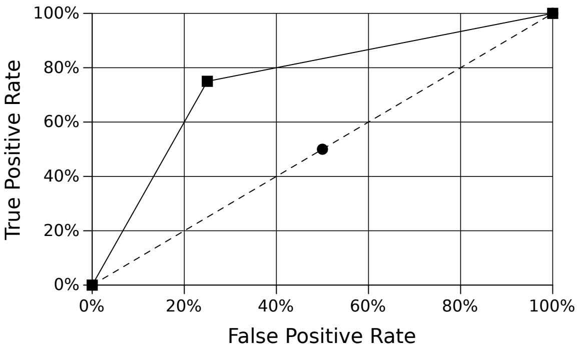

Fig. 2: Example of ROC plot. The solid line is a good performing classifier whereas the dashed line represents a random classifier.

Cohen's kappa rate [3,4] is an alternative measure to predictive accuracy that compensates for random hits. The kappa measure evaluates the merit of the classifier, i.e., the actual hits (coverage of true positives) that can be attributed to the classifier and not to a mere chance. Kappa statistic values range from -1 (total disagreement) through 0 (random classification) to 1 (total agreement). It is calculated from the confusion matrix as follows:

$$Kappa = \frac{N\sum_{i=1}^{k} x_{ii} - \sum_{i=1}^{k} x_{i.}x_{.i}}{N^2 - \sum_{i=1}^{k} x_{i.}x_{.i}} \quad (15)$$

where x_{ii} is the count of cases in the main diagonal of the confusion matrix, N is the number of instances, and $x_{.i}$, $x_{i.}$ are the column and the row total counts, respectively. Kappa penalizes all-positive or all-negative predictions (default hypothesis), which is crucial to consider when dealing with unbalanced data sets.

The area under the ROC curve (AUC) [6,36] is also commonly used as it shows the trade-off between the true positive rate (TP_{rate}) and the false positive rate (FP_{rate}) as demonstrated in [22,25,48,49]. The way to build the ROC space is to plot on a two-dimensional chart the true positive rate (Y-axis) against the false positive rate (X-axis) as shown in Figure 2. The points (0,0) and (1,1) are trivial classifiers in which the class is always predicted as negative and positive, respectively, while the point (0,1) represents perfect classification. AUC is calculated using the graphic's area as:

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2} \quad (16)$$

4.2 Data sets and algorithms

The data sets used in the experiments were collected from the KEEL [1] and UCI [50] machine learning repositories. These data sets are very different in terms of complexity, number of classes, number of attributes, number of instances, and unbalance ratio (ratio of size of the majority class to the minority class). There are 28 balanced and 70 unbalanced data sets. Detailed information about the data sets is provided as the additional material that can be found at this link online[1]. The balanced data sets are partitioned using the stratified 10-fold cross-validation procedure [42,55]. The unbalanced data sets are partitioned using the stratified 5-fold cross-validation procedure to ensure the presence of minority class instances in the test sets.

Discretization algorithms used in comparisons were run from the KEEL [2] and WEKA [33] software tools, that facilitate the replicability of the experiments. Algorithms employed are the ones reviewed in the background section and were recommended by García et al. [28]: Equal-Width (EW) [8], Equal-Frequency (EF) [8], Information Entropy Maximization (IEM) [21], Class-Attribute Interdependence Maximization (CAIM) [44], Ameva [30], Modified-χ^2 [53], Hypercube Division-based Discretization (HDD) [58], Class-Attribute Contingency Coefficient (CACC) [54], and Interval Distance-Based Method for Discretization (IDD) [52].

The source code of ur-CAIM is made publicly available at this link online[1]. Moreover, it is provided as a WEKA plugin to enable its utilization in the WEKA software tool.

Quality of discretization intervals is evaluated by means of the classification performance of the subsequently used classifiers. In order to avoid the bias of particular classification algorithms to data, 8 different classification algorithms belonging to different families are used to evaluate the classification performance, which increases the strength of the experimental study. They are NaiveBayes [39], SVM [9], KNN [15], AdaBoost [24], JRip [13], PART [23], C45 [51], and RandomForest [7]. Details about the algorithms and experimental settings are also available online[1].

4.3 Statistical analysis

The statistical analysis supports the results obtained through the experimental study. We use hypothesis testing techniques to find significant differences between algorithms [26]. We employ non-parametric tests according to the recommendations made in [16,17,26,27].

The Friedman test [14] identifies statistical differences among a group of results and can be used to test the hypothesis of equality of medians between the results of the algorithms. If the Friedman test hypothesis of equality is rejected (that is, a low p-value is obtained), then it is assumed that there are significant differences among the different algorithms of the experiment. These differences can then be assessed by using a post-hoc method. The Holm [35] post-hoc test finds which algorithms are distinctive among a $1 \times n$ comparison. Moreover, we compute the p-value associated with each comparison, which represents the lowest level of significance of a hypothesis that results in a rejection. That is the adjusted p-value. This way, we can know whether two algorithms are significantly different and how different they are. We also obtain the average ranking of the algorithms, according to the Friedman procedure, which shows the performance of an algorithm with respect to the others and it is based on the ranking of the algorithms in each data set. Finally, we perform the Wilcoxon [56] test, which aims to detect significant differences between pairs of algorithms.

5 Results

This section presents and discusses the experimental study in order to compare the performance of ur-CAIM in a scenario of both balanced and unbalanced data sets. First, the number of intervals, execution time, and accuracy for balanced data sets are analyzed. Second, the number of intervals, execution time, AUC and the Cohen's kappa rate for unbalanced data sets are analyzed. Finally, the performance of ur-CAIM is compared with regards of the unbalance ratio.

Due to the article's space limitations and the large number of data sets and methods employed, we show only the results of the statistical tests. Tables with the results of the cross validation, for each data set and for each method, are available online for the readers[1].

5.1 Balanced data sets

Table 2 show the results of the statistical analysis for the balanced data sets. Algorithms are ranked according to the Friedman's ranking procedure for each measure. The lower the rank value the better performance of the algorithm. The Friedman test run on all the measures outcomes a p-value lower than 0.01 (except for AdaBoost accuracy which is 0.015), which is low enough to reject the null equality hypothesis with a high confidence level ($\geq 99\%$). Therefore, as we know there are significant differences, we proceed with the application of the Holm's post-hoc procedure. In Table 2

[1] The data sets description along with their partitions, the ur-CAIM source code and WEKA plugin, the experimental settings and results for all data sets and algorithms are fully described and publicly available to facilitate the replicability of the experiments and future comparisons at the website:

`http://www.uco.es/grupos/kdis/wiki/ur-CAIM`

Table 2: Friedman ranks and p-values using Holm's post-hoc procedure for the balanced data sets.

Number of intervals			Runtime		
Algorithm	Rank	p-value	Algorithm	Rank	p-value
IEM	3.5357		EW	1.0893	
ur-CAIM	3.5536	0.9824	EF	2.3750	0.1121
CAIM	4.1964	0.4142	IEM	3.4643	0.0033
CACC	4.3929	0.2895	ur-CAIM	3.8929	5.3E-4
Ameva	4.6786	0.1578	IDD	4.3571	5.4E-5
Modified-χ^2	5.6964	0.0076	Ameva	6.4643	0.0000
EW	6.5000	2.5E-4	Modified-χ^2	7.6429	0.0000
EF	6.5000	2.5E-4	CAIM	7.7143	0.0000
IDD	7.1429	8.0E-6	HDD	8.5357	0.0000
HDD	8.8036	0.0000	CACC	9.4643	0.0000

Accuracy

AdaBoost			KNN			C45			JRip		
Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value
IEM	4.2321		ur-CAIM	3.8571		ur-CAIM	2.8036		IEM	3.3393	
ur-CAIM	4.4464	0.7912	IEM	3.8571	1.0000	IEM	3.5179	0.3774	ur-CAIM	3.3929	0.9472
IDD	4.9286	0.3894	CAIM	4.6250	0.3427	CAIM	4.4286	0.0446	CAIM	4.6250	0.1121
Modified-χ^2	5.1607	0.2511	Modified-χ^2	4.7143	0.2895	CACC	5.4821	9.3E-4	Modified-χ^2	5.1786	0.0230
CACC	5.2857	0.1929	Ameva	5.5357	0.0380	Modified-χ^2	5.6607	4.1E-4	CACC	5.4286	0.0098
CAIM	5.3214	0.1782	CACC	5.8571	0.0135	Ameva	5.8036	2.1E-4	Ameva	5.6429	0.0044
Ameva	6.0357	0.0258	EF	6.1607	0.0044	EW	6.2679	1.9E-5	IDD	6.5357	7.8E-5
EW	6.4821	0.0054	IDD	6.3214	0.0023	IDD	6.4464	7.0E-6	EF	6.5714	6.5E-5
EF	6.5000	0.0051	EW	7.0179	9.4E-5	EF	6.9286	0.0000	EW	7.1429	3.0E-6
HDD	6.6071	0.0033	HDD	7.0536	7.8E-5	HDD	7.6607	0.0000	HDD	7.1429	3.0E-6

NaiveBayes			PART			RamdomForest			SVM		
Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value
CAIM	3.7321		IEM	3.2143		IEM	3.9821		IEM	3.0000	
IEM	3.8929	0.8426	ur-CAIM	3.6964	0.5513	CAIM	4.0536	0.9297	Modified-χ^2	3.8214	0.3100
Modified-χ^2	4.1786	0.5812	CAIM	4.5179	0.1072	ur-CAIM	4.4464	0.5661	ur-CAIM	4.7500	0.0306
ur-CAIM	5.2321	0.0638	Modified-χ^2	5.1607	0.0162	Ameva	4.6250	0.4269	CAIM	4.8036	0.0258
Ameva	5.3036	0.0521	EW	5.6429	0.0027	Modified-χ^2	4.7679	0.3315	Ameva	5.2679	0.0051
EW	5.4464	0.0341	Ameva	5.7857	0.0015	EF	5.6250	0.0423	IDD	5.8036	5.3E-4
IDD	5.7321	0.0135	CACC	5.9286	7.9E-4	IDD	5.8571	0.0205	EW	5.9643	2.5E-4
HDD	6.6786	2.7E-4	EF	6.2321	1.9E-4	EW	5.9821	0.0135	EF	5.9643	2.5E-4
CACC	7.2321	1.5E-5	IDD	6.7500	1.2E-5	CACC	7.1429	9.4E-5	CACC	7.0893	0.0000
EF	7.5714	2.0E-6	HDD	8.0714	0.0000	HDD	8.5179	0.0000	HDD	8.5357	0.0000

we also show the adjusted p-values that were calculated using Holm's post-hoc procedure. The algorithm which obtains the lower rank turns into the control method, and it is compared against all the other algorithms. The adjusted p-values associated to the methods which are lower than 0.05 and 0.01 are said to reject the null-hypothesis with a high confidence level ($\geq 95\%$ and $\geq 99\%$, respectively).

The results indicate that IEM and ur-CAIM produce the lower number of intervals with a very similar rank, whereas Modified-χ^2, EW, EF, IDD, and HDD obtain much higher number of intervals and there are statistical differences since their p-values are lower than 0.01. On the other hand, EW and EF are the fastest methods, as expected since they are the simplest algorithms of all used. On the contrary, Ameva, Modified-χ^2, CAIM, HDD and CACC demand significantly longer runtimes with p-values lower than 1.0E-6. It is also interesting to point out that ur-CAIM is ranked to be faster than CAIM.

The accuracy performance is evaluated with regards of each of the 8 classification methods. IEM achieves the lowest ranks in 5 methods, ur-CAIM in 2 methods, and CAIM in just one. Although IEM is ranked better many times, there are no statistical significant differences with ur-CAIM except for SVM with a p-value of 0.03. Moreover, ur-CAIM is ranked better than CAIM for 6 of the 8 classifiers. On the other hand, HDD performs significantly worse than the rest of the algorithms, and many times it is ranked the worst. The high number of intervals created causes bad classification performance and penalizes the accuracy results.

Table 3 shows the p-values of the Wilcoxon test for the balanced data sets in order to compute multiple pairwise comparisons among ur-CAIM and the other methods. The ur-CAIM approach outperforms the original CAIM method and achieves statistical differences with p-values lower than 0.05 with regards of the number of intervals, the runtime, and two classifiers.

Table 3: Wilcoxon test for the balanced data sets.

ur-CAIM vs	Intervals	Runtime	Accuracy							
			AdaBoost	KNN	C45	JRip	NaiveBayes	PART	RandomForest	SVM
EW	4.2E-5	≥ 0.2	0.0036	2.2E-5	1.1E-5	2.8E-6	≥ 0.2	0.0024	0.0066	0.0280
EF	4.2E-5	≥ 0.2	0.0006	0.0002	2.2E-5	5.5E-6	0.0010	0.0004	0.0662	0.0280
IEM	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2	≥ 0.2
CAIM	0.0229	7.4E-9	≥ 0.2	≥ 0.2	0.0007	0.0162	≥ 0.2	0.0794	≥ 0.2	≥ 0.2
Ameva	0.0057	0.0002	0.0187	0.0245	4.5E-5	0.0103	≥ 0.2	0.0042	0.1256	0.1536
Modified-χ^2	0.0071	0.0002	0.1936	≥ 0.2	0.0001	0.0698	≥ 0.2	0.0521	≥ 0.2	≥ 0.2
HDD	7.4E-8	7.4E-9	0.0136	0.0005	3.7E-7	2.4E-5	0.0214	1.3E-6	8.0E-6	3.7E-6
CACC	0.1375	7.4E-9	0.1638	0.0039	1.9E-5	0.0204	0.0004	0.0009	1.1E-5	5.9E-5
IDD	6.7E-6	≥ 0.2	≥ 0.2	0.0007	1.0E-5	1.5E-6	0.0426	4.2E-5	0.1936	0.0595

5.2 Unbalanced data sets

Table 4 show the results of the statistical analysis for the unbalanced balanced data sets. The Friedman test run on all the measures computes a p-value lower than 0.01, which is low enough to reject the null equality hypothesis with a high confidence level ($\geq 99\%$). Therefore, we proceed with the application of the Holm's post-hoc procedure and we show the adjusted p-values.

The results indicate that IEM produces the lower number of intervals and it achieves statistical significant differences with all the other discretization methods. However, we will show that it generates an excessively low number of intervals which eventually, leads classification algorithms to obtain higher classification errors. Similarly to the performance analysis on balanced data, EW and EF are the fastest methods and ur-CAIM is also faster than CAIM.

The AUC and Cohen's kappa performances are evaluated with regards of each of the 8 classification methods. The results show that ur-CAIM consistently achieves better AUC and kappa ranks than the other discretization methods for almost all the classification algorithms. This good performance on unbalanced data is one of the major advantages of ur-CAIM, especially when compared with CAIM. Specifically, ur-CAIM is ranked in the first place for AUC in 7 of the 8 classifiers, whereas for Cohen's kappa, it performs best for all the classifiers evaluated. It is also important to note the bad performance of EW and EF on unbalanced datasets as measured by their ranks for most of the classification methods.

Table 5 shows the p-values of the Wilcoxon test for the AUC and Cohen's kappa. It is interesting to point out that ur-CAIM clearly outperforms both IEM and CAIM on unbalanced data, achieving statistical significant differences on many of the classifiers. These results are in contrast with the balanced data scenario, in which IEM outperformed ur-CAIM, and ur-CAIM had better but very close performance to the original CAIM. This is the main contribution of the ur-CAIM algorithm, to improve the CAIM performance on balanced, but especially, on unbalanced data sets, as seen in the experimental results.

5.2.1 Performance with data re-sampling

Unbalanced data sets are also commonly evaluated after applying a data class re-sampling method [29,49]. SMOTE (Synthetic Minority Over-sampling Technique) [10] is commonly used data re-sampling algorithm based on the oversampling of the minority class. SMOTE was used after data were discretized. It creates synthetic instances taking each minority class sample and introduces new samples.

Based on the results for particular data sets which are available online, SMOTE demonstrates good re-sampling of data classes since AUC results are much better than without using re-sampling for all the classification algorithms.

Table 6 shows the p-values of the Wilcoxon test for the AUC and Cohen's kappa after re-sampling with SMOTE. It is interesting to point out that the p-values for CAIM are generally lower with re-sampling than without re-sampling for both AUC and Cohen's kappa. Thus, after re-sampling with SMOTE, ur-CAIM results are even better than those from the original CAIM.

Table 7 show the results of the Friedman test for the unbalanced balanced data sets after applying SMOTE re-sampling. The Friedman test run on all the measures computes a p-value lower than 0.01, which is low enough to reject the null equality hypothesis with a high confidence level ($\geq 99\%$). Therefore, we proceed with the application of the Holm's post-hoc procedure and we show the adjusted p-values.

Similarly to the results without re-sampling, ur-CAIM consistently achieves better AUC and kappa ranks than the other discretization methods for almost all the classification algorithms. On the other hand, EF, EW, and IDD are commonly ranked among the worst methods for unbalanced data, both raw and re-sampled. If we look together these ranks with regards of the number of intervals, we see that discretization methods that create excessive number of intervals also obtain higher classification errors. Moreover, IEM, which obtained significantly lower number of intervals, was also overcame by ur-CAIM. Therefore, we can conclude that it is important not to generate too few nor too many number of intervals to minimize the classification error.

Table 4: Friedman ranks and p-values using Holm's post-hoc procedure for the unbalanced data sets.

Number of intervals			Runtime		
Algorithm	Rank	p-value	Algorithm	Rank	p-value
IEM	1.9857		EW	1.3071	
CAIM	3.1500	0.0229	EF	2.3143	0.0491
ur-CAIM	4.0571	5.2E-5	IEM	2.4643	0.0238
HDD	4.6714	0.0000	ur-CAIM	4.0643	0.0000
Ameva	5.0143	0.0000	IDD	5.1143	0.0000
CACC	6.0929	0.0000	CAIM	6.4429	0.0000
Modified-χ^2	6.5643	0.0000	Ameva	7.4857	0.0000
IDD	7.0357	0.0000	HDD	7.4929	0.0000
EW	8.2143	0.0000	CACC	8.9714	0.0000
EF	8.2143	0.0000	Modified-χ^2	9.3429	0.0000

Area Under the Curve (AUC)

AdaBoost			KNN			C45			JRip		
Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value
ur-CAIM	4.1286		ur-CAIM	3.5929		ur-CAIM	4.1643		ur-CAIM	4.1714	
CACC	4.3929	0.6056	IEM	4.2000	0.2355	Ameva	4.4357	0.5958	CACC	4.6071	0.3946
Ameva	4.4714	0.5029	CAIM	4.8643	0.0130	IEM	4.4786	0.5391	CAIM	4.8000	0.2194
CAIM	4.9571	0.1054	Ameva	5.1357	0.0026	CACC	4.6429	0.3497	Ameva	4.9357	0.1353
IEM	5.2429	0.0295	EF	5.3286	6.9E-4	CAIM	4.6929	0.3017	IEM	5.2214	0.0402
HDD	5.4929	0.0077	Modified-χ^2	5.5571	1.2E-4	Modified-χ^2	5.2571	0.0327	Modified-χ^2	5.5429	0.0074
IDD	5.7786	0.0013	HDD	5.6571	5.5E-5	HDD	5.4857	0.0098	HDD	5.5500	0.0071
Modified-χ^2	5.7929	0.0011	CACC	5.7357	2.8E-5	IDD	6.3357	2.2E-5	IDD	6.0000	3.5E-4
EF	6.8929	0.0000	IDD	7.3857	0.0000	EW	7.4143	0.0000	EF	6.3071	3.0E-5
EW	7.8500	0.0000	EW	7.5429	0.0000	EF	8.0929	0.0000	EW	7.8643	0.0000

NaiveBayes			PART			RamdomForest			SVM		
Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value
EF	4.5429		ur-CAIM	3.9286		ur-CAIM	3.8143		ur-CAIM	4.5286	
ur-CAIM	4.5929	0.9222	Ameva	4.1071	0.7271	Ameva	4.2071	0.4427	Ameva	4.6214	0.8560
IEM	4.9643	0.4102	CAIM	4.8643	0.0675	IEM	4.8143	0.0507	Modified-χ^2	4.6786	0.7694
Modified-χ^2	4.9857	0.3869	IEM	4.9214	0.0524	CAIM	4.8214	0.0491	IEM	4.7643	0.6451
Ameva	5.2714	0.1545	CACC	4.9500	0.0459	Modified-χ^2	5.7714	1.3E-4	IDD	5.3214	0.1213
CACC	5.5071	0.0595	Modified-χ^2	5.4857	0.0023	HDD	5.8500	7.0E-5	CAIM	5.5214	0.0524
HDD	6.1071	0.0022	HDD	5.5357	0.0017	CACC	5.9929	2.1E-5	EW	5.9714	0.0048
IDD	6.2286	9.9E-4	IDD	5.7571	3.5E-4	EF	6.2214	3.0E-6	HDD	6.3143	4.8E-4
CAIM	6.3500	4.1E-4	EW	7.6857	0.0000	IDD	6.4500	0.0000	CACC	6.5857	5.8E-5
EW	6.4500	1.9E-4	EF	7.7643	0.0000	EW	7.0571	0.0000	EF	6.6929	2.3E-5

Cohen's Kappa

AdaBoost			KNN			C45			JRip		
Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value
ur-CAIM	4.2000		ur-CAIM	3.6357		ur-CAIM	4.0357		ur-CAIM	3.7357	
Ameva	4.4071	0.6856	IEM	4.2857	0.2040	Ameva	4.4000	0.4766	CAIM	4.5786	0.0996
CACC	4.4286	0.6551	CAIM	4.8286	0.0198	CACC	4.4857	0.3792	CACC	4.9429	0.0183
CAIM	4.8857	0.1803	Ameva	4.9286	0.0115	CAIM	4.5500	0.3149	Ameva	5.0500	0.0102
IEM	5.3429	0.0255	CACC	5.5071	2.6E-4	IEM	4.7857	0.1428	IEM	5.1000	0.0077
HDD	5.4143	0.0177	Modified-χ^2	5.5071	2.6E-4	HDD	5.2714	0.0158	HDD	5.4214	9.9E-4
IDD	5.7571	0.0023	HDD	5.5786	1.5E-4	Modified-χ^2	5.5286	0.0035	Modified-χ^2	5.8000	5.5E-5
Modified-χ^2	6.0000	4.4E-4	EF	5.6571	7.8E-5	IDD	6.2500	1.5E-5	IDD	5.8357	4.1E-5
EF	6.7000	1.0E-6	IDD	7.2857	0.0000	EW	7.4857	0.0000	EF	6.4571	0.0000
EW	7.8643	0.0000	EW	7.7857	0.0000	EF	8.2071	0.0000	EW	8.0786	0.0000

NaiveBayes			PART			RamdomForest			SVM		
Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value
ur-CAIM	4.6571		ur-CAIM	3.8571		ur-CAIM	3.7714		ur-CAIM	4.4429	
Ameva	4.9500	0.5672	Ameva	4.1786	0.5300	CAIM	4.4857	0.1628	Modified-χ^2	4.5714	0.8016
Modified-χ^2	4.9857	0.5209	CAIM	4.6857	0.1054	Ameva	4.5714	0.1180	Ameva	4.8071	0.4766
IEM	5.0000	0.5029	IEM	4.8571	0.0507	IEM	4.6714	0.0786	IEM	5.0714	0.2194
CACC	5.1429	0.3426	CACC	5.0643	0.0183	HDD	5.7500	1.1E-4	IDD	5.0786	0.2142
EF	5.6286	0.0577	HDD	5.4429	0.0019	Modified-χ^2	5.8286	5.8E-5	CAIM	5.6571	0.0177
HDD	5.8357	0.0213	IDD	5.5286	0.0011	EF	6.1214	4.0E-6	EW	5.8214	0.0071
IDD	5.9429	0.0120	Modified-χ^2	5.7500	2.2E-4	CACC	6.2071	2.0E-6	CACC	6.4071	1.2E-4
CAIM	6.0857	0.0053	EF	7.8000	0.0000	IDD	6.4000	0.0000	HDD	6.5429	4.1E-5
EW	6.7714	3.6E-5	EW	7.8357	0.0000	EW	7.1929	0.0000	EF	6.6000	2.5E-5

Table 5: Wilcoxon test for the AUC and Cohen's kappa on unbalanced data sets.

ur-CAIM vs	Area Under the Curve (AUC)							
	AdaBoost	KNN	C45	JRip	NaiveBayes	PART	RandomForest	SVM
EW	0.0000	0.0000	0.0000	0.0000	0.0013	0.0000	0.0000	0.0280
EF	0.0000	0.0029	0.0000	3.0E-5	1.0000	0.0000	5.0E-5	0.0280
IEM	0.0220	0.2918	0.2180	0.0205	0.2887	0.0071	0.1286	≥ 0.2
CAIM	0.0400	5.0E-5	0.2470	0.0939	0.0007	0.0297	0.0293	≥ 0.2
Ameva	0.0401	1.0E-5	0.5723	0.0207	0.0269	0.3527	0.1278	0.1536
Modified-χ^2	4.0E-5	4.0E-5	0.0109	0.0003	0.6881	0.0002	0.0006	≥ 0.2
HDD	0.0073	0.0000	0.0092	0.0182	0.0064	0.0002	0.0004	3.7E-6
CACC	0.4804	0.0000	0.0720	0.2798	0.0258	0.0094	0.0000	5.9E-5
IDD	7.0E-5	0.0000	2.0E-5	5.0E-5	0.0017	0.0002	0.0000	0.0595

ur-CAIM vs	Cohen's kappa							
	AdaBoost	KNN	C45	JRip	NaiveBayes	PART	RandomForest	SVM
EW	0.0000	0.0000	0.0000	0.0000	4.0E-5	0.0000	0.0000	0.0362
EF	1.0E-5	4.0E-5	0.0000	0.0000	0.0297	0.0000	5.0E-5	0.0001
IEM	0.0065	0.1249	0.0354	0.0048	0.4531	0.0053	0.0743	0.1249
CAIM	0.0442	0.0002	0.3431	0.0394	0.0029	0.0288	0.0519	0.0002
Ameva	0.1730	0.0003	0.4209	0.0002	0.1619	0.2687	0.0208	0.2526
Modified-χ^2	1.0E-5	5.0E-5	0.0003	3.0E-5	0.8154	0.0000	0.0004	0.5782
HDD	0.0040	1.0E-5	0.0136	0.0034	0.0166	0.0003	0.0006	2.0E-5
CACC	0.4824	0.0000	0.0649	0.0020	0.1700	0.0031	0.0000	0.0001
IDD	0.0002	0.0000	1.0E-5	1.0E-5	0.0781	0.0002	1.0E-5	0.2421

Table 6: Wilcoxon test for the AUC and Cohen's kappa with SMOTE on unbalanced data sets.

ur-CAIM vs	Area Under the Curve (AUC) with SMOTE							
	AdaBoost	KNN	C45	JRip	NaiveBayes	PART	RandomForest	SVM
EW	0.0000	0.0000	0.0000	0.0000	3.0E-5	0.0000	0.0000	0.0005
EF	0.0000	0.0613	0.0000	0.0000	4.0E-5	0.0000	0.0000	0.0000
IEM	0.6539	0.2741	0.0767	1.0000	0.5895	0.5309	0.4636	0.2705
CAIM	0.0111	0.0002	0.0290	0.0856	0.0141	0.0082	0.0146	0.0016
Ameva	0.1421	0.0029	0.0008	0.0021	0.0447	0.0038	0.0080	0.0008
Modified-χ^2	0.0002	0.0187	2.0E-5	0.0002	0.0416	0.0000	4.0E-5	0.0243
HDD	0.0011	2.0E-5	0.0023	0.0009	0.0159	0.0003	0.0017	2.0E-5
CACC	0.0268	2.0E-5	0.0003	4.0E-5	0.0044	6.0E-5	0.0000	2.0E-5
IDD	0.0013	0.0000	0.0000	3.0E-5	0.0026	0.0000	0.0000	0.0015

ur-CAIM vs	Cohen's kappa with SMOTE							
	AdaBoost	KNN	C45	JRip	NaiveBayes	PART	RandomForest	SVM
EW	0.0000	0.0000	0.0000	0.0000	0.0168	0.0000	0.0002	0.3299
EF	0.0019	0.0000	2.0E-5	1.0E-5	0.0318	2.0E-5	0.0408	0.0087
IEM	0.1331	0.1485	0.0115	0.5564	0.0254	0.2073	0.0638	0.0538
CAIM	0.0019	0.0005	0.0056	0.1241	0.0013	0.0025	0.0066	0.0015
Ameva	0.1086	0.0050	0.0004	0.0023	0.3650	0.0079	0.0344	0.0002
Modified-χ^2	0.0929	0.3123	0.0455	0.4993	1.0000	0.0042	1.0000	1.0000
HDD	0.0002	2.0E-5	0.0006	0.0019	0.0068	0.0001	0.0008	0.0000
CACC	0.1733	0.0024	0.0008	0.0084	0.6734	0.0432	0.0004	0.0013
IDD	0.0033	0.0000	0.0000	9.0E-5	0.0027	0.0000	1.0E-5	0.0178

Table 7: Friedman ranks and p-values using Holm's post-hoc procedure for the unbalanced data sets with re-sampling.

Area Under the Curve (AUC) with SMOTE

AdaBoost			KNN			C45			JRip		
Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value
ur-CAIM	4.0143		ur-CAIM	4.0071		ur-CAIM	3.7643		ur-CAIM	4.0286	
Ameva	4.4286	0.4182	IEM	4.4714	0.3643	IEM	4.7143	0.0634	IEM	4.1071	0.8780
IEM	4.5500	0.2952	Modified-χ^2	4.9857	0.0559	CAIM	4.7714	0.0491	CAIM	4.5500	0.3083
CACC	4.7000	0.1803	Ameva	5.0286	0.0459	Ameva	5.0000	0.0158	Ameva	4.9214	0.0810
CAIM	5.0714	0.0389	EF	5.2286	0.0170	HDD	5.2714	0.0032	CACC	5.3929	0.0077
HDD	5.5929	0.0020	CAIM	5.4000	0.0065	CACC	5.3500	0.0019	HDD	5.6429	0.0016
IDD	5.6714	0.0012	CACC	5.5500	0.0026	Modified-χ^2	5.8786	3.6E-5	IDD	6.0714	6.6E-5
Modified-χ^2	5.9643	1.4E-4	HDD	6.0357	7.4E-5	EW	6.6214	0.0000	Modified-χ^2	6.1357	3.8E-5
EF	6.7786	0.0000	IDD	7.0643	0.0000	EF	6.7929	0.0000	EW	6.8071	0.0000
EW	8.2286	0.0000	EW	7.2286	0.0000	IDD	6.8357	0.0000	EF	7.3429	0.0000

NaiveBayes			PART			RamdomForest			SVM		
Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value
ur-CAIM	4.2714		ur-CAIM	3.6500		ur-CAIM	3.5571		ur-CAIM	3.9286	
IEM	4.8214	0.2825	IEM	4.3286	0.1849	IEM	4.3357	0.1282	IEM	4.4286	0.3286
Ameva	5.2429	0.0577	Ameva	4.3929	0.1466	Ameva	4.6143	0.0389	Ameva	5.0714	0.0255
CAIM	5.3143	0.0416	CAIM	4.6429	0.0524	CAIM	4.9429	0.0068	Modified-χ^2	5.0786	0.0246
Modified-χ^2	5.4786	0.0183	CACC	5.0143	0.0077	HDD	5.7143	2.5E-5	CAIM	5.1071	0.0213
CACC	5.4857	0.0177	HDD	5.5714	1.7E-4	CACC	5.8286	9.0E-6	CACC	5.7714	3.2E-4
HDD	5.5286	0.0140	Modified-χ^2	6.2429	0.0000	Modified-χ^2	6.1143	1.0E-6	IDD	5.8286	2.1E-4
IDD	5.8286	0.0023	IDD	6.5857	0.0000	EF	6.1786	0.0000	EW	6.1500	1.4E-5
EF	6.4571	1.9E-5	EF	6.9357	0.0000	IDD	6.7357	0.0000	HDD	6.1929	1.0E-5
EW	6.5714	7.0E-6	EW	7.6357	0.0000	EW	6.9786	0.0000	EF	7.4429	0.0000

Cohen's Kappa with SMOTE

AdaBoost			KNN			C45			JRip		
Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value
ur-CAIM	4.3143		ur-CAIM	3.9286		ur-CAIM	3.8714		ur-CAIM	4.3929	
CACC	4.5786	0.6056	Modified-χ^2	4.4143	0.3426	Modified-χ^2	4.9286	0.0389	IEM	4.6643	0.5958
Ameva	4.7571	0.3869	IEM	4.5571	0.2194	IEM	5.0786	0.0183	CAIM	4.8357	0.3869
IEM	4.8786	0.2702	Ameva	4.5857	0.1991	CACC	5.1286	0.0140	Modified-χ^2	4.8429	0.3792
Modified-χ^2	5.0571	0.1466	CACC	4.9643	0.0430	Ameva	5.1357	0.0135	Ameva	5.1714	0.1282
CAIM	5.6643	0.0083	CAIM	5.2786	0.0083	CAIM	5.1429	0.0130	CACC	5.3429	0.0634
IDD	5.8643	0.0025	HDD	6.0643	3.0E-5	HDD	5.6643	4.6E-4	HDD	5.9143	0.0029
EF	6.0000	9.9E-4	IDD	6.4786	1.0E-6	IDD	6.5643	0.0000	IDD	6.0786	9.9E-4
HDD	6.1786	2.7E-4	EF	7.0929	0.0000	EF	6.6500	0.0000	EF	6.7714	3.0E-6
EW	7.7071	0.0000	EW	7.6357	0.0000	EW	6.8357	0.0000	EW	6.9857	0.0000

NaiveBayes			PART			RamdomForest			SVM		
Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value	Algorithm	Rank	p-value
Modified-χ^2	4.3857		ur-CAIM	4.0500		ur-CAIM	4.4000		Modified-χ^2	4.2714	
ur-CAIM	4.6857	0.5577	CACC	4.7857	0.1505	Modified-χ^2	4.6929	0.5672	ur-CAIM	4.4714	0.6959
CACC	5.0000	0.2300	Ameva	4.8214	0.1317	IEM	4.9000	0.3286	IEM	5.3071	0.0430
Ameva	5.1286	0.1466	IEM	4.8286	0.1282	Ameva	5.0214	0.2246	CAIM	5.4714	0.0190
EF	5.7643	0.0071	CAIM	5.2143	0.0229	CAIM	5.5214	0.0284	EW	5.5286	0.0140
IEM	5.8000	0.0057	Modified-χ^2	5.3071	0.0140	EF	5.5286	0.0274	Ameva	5.6714	0.0062
CAIM	5.8357	0.0046	HDD	6.0000	1.4E-4	CACC	5.6143	0.0177	CACC	5.7071	0.0050
HDD	5.8429	0.0044	EF	6.4214	4.0E-6	HDD	6.2429	3.2E-4	IDD	5.8286	0.0023
EW	6.1571	5.4E-4	IDD	6.4286	3.0E-6	IDD	6.5214	3.4E-5	EF	6.0214	6.3E-4
IDD	6.4000	8.3E-5	EW	7.1429	0.0000	EW	6.5571	2.5E-5	HDD	6.7214	2.0E-6

Table 8: Wilcoxon test for the number of intervals and runtime on unbalanced data sets.

ur-CAIM vs	Intervals	Runtime
EW	0.0000	1.0000
EF	0.0000	1.0000
IEM	1.0000	1.0000
CAIM	1.0000	0.0000
Ameva	0.0000	0.0000
Modified-χ^2	0.0000	0.0000
HDD	0.9071	0.0000
CACC	0.0000	0.0000
IDD	0.0000	0.0000

Finally, Table 8 shows the p-values of the Wilcoxon test for the number of intervals and the execution time on unbalanced datasets. These values are equal or very close to 0 or 1 because algorithms generally generate always more/less discretization intervals as well as they run faster/slower by means of pairwise comparisons.

5.3 Unbalance ratio and performance

The ur-CAIM criterion was made to consider the data classes distribution in order to improve classification performance, especially on unbalanced data. The experimental results in the previous section showed the good performance of the ur-CAIM algorithm. However, to evaluate whether the new criterion actually improves the performance it its necessary to perform a more detailed analysis with regards of the unbalance ratio of the data sets.

The unbalance ratio is the relation of majority class instances to minority class instances. The 70 unbalanced data sets were categorized into two groups regarding to their unbalance ratio. The former group comprised low unbalanced data sets with unbalance ratios from 1 (equal number of majority and minority class instances) to 5 (there are 5 times more majority class instances than minority class instances). The latter group comprised high unbalanced data sets with unbalance ratios from 5 to 129 (the highest unbalance ratio among all of the data sets).

When comparing the results available online of ur-CAIM and CAIM on low unbalanced data, the AUC and Cohen's kappa values are very close but in favour of ur-CAIM's. On the other hand, when handling high unbalanced data, ur-CAIM shows that its improved criterion clearly achieves better classification performance than CAIM's with larger differences. Table 9 shows the results of the Wilcoxon test for the comparison of ur-CAIM with CAIM on the two unbalanced data groups, considering the results from all data sets and classifiers. Results indicate the better performance of ur-CAIM on both groups, but especially on high unbalanced data. Specifically, the p-values reported are all lower than 0.01, i.e., statistical confidence higher than 99%.

Table 9: Wilcoxon test with regards of unbalance ratio.

Unbalance ratio	Measure	p-value
Low unbalanced [1,5)	AUC	0.0014
	Kappa	0.0002
High unbalanced [5,129)	AUC	3.9E-9
	Kappa	1.1E-8

6 Conclusion

In this paper we presented the ur-CAIM algorithm for supervised discretization. The ur-CAIM criterion extended the CAIM criterion to account for the class-attribute interdependency, redundancy and uncertainty. The new approach considered the balance of the data classes distribution in terms of the number of instances in each class in order to improve the discretization process, especially under the presence of unbalaced data sets. This resulted in significantly better discretization schemes than original CAIM algorithm. Experiments carried out over many balanced and unbalanced data sets demonstrated the good performance of the algorithm in terms of obtaining high predictive accuracy, high kappa rate and high AUC, while keeping low number of intervals, all at the lower computational cost. Classification performance was evaluated and compared using 8 different classification algorithms from different families to avoid bias of algorithms to data. The results were validated using non-parametric statistical tests, which support the better performance of the ur-CAIM algorithm than the original CAIM and many other discretization methods used in comparisons. ur-CAIM proved to be a significant improvement over the original CAIM, achieving better and faster results, especially when handling unbalanced data. The performance differences were specially noteworthy on highly unbalanced data sets, in which the other discretization methods were not capable of handling such data appropriately.

Acknowledgements This work has been supported by the National Institutes of Health grant 1R01HD056235-01A1 (KJC), the NSF GANN grant (DTN), the Regional Government of Andalusia and the Ministry of Science and Technology project TIN-2011-22408 (SV,AC), and the Ministry of Education FPU grant AP2010-0042 (AC).

References

1. J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing*, 17:255–287, 2011.
2. J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. Fernández, and F. Herrera. KEEL: A Software Tool to Assess Evolutionary Algorithms for Data Mining Problems. *Soft Computing*, 13:307–318, 2009.

3. A. Ben-David. About the relationship between ROC curves and Cohen's kappa. *Engineering Applications of Artificial Intelligence*, 21(6):874–882, 2008.
4. A. Ben-David. Comparison of classification accuracy using Cohen's weighted kappa. *Expert Systems with Applications*, 34(2):825–832, 2008.
5. M. Boullé. MODL: A Bayes optimal discretization method for continuous attributes. *Machine Learning*, 65(1):131–165, 2006.
6. A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7), 1997.
7. L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
8. J. Catlett. On changing continuous attributes into ordered discrete attributes. In *Machine Learning - EWSL91*, volume 482 of *Lecture Notes in Computer Science*, pages 164–178. 1991.
9. C.C. Chang and C.J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.
10. N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling TEchnique. *Artificial Intelligence Research*, 16:321–357, 2002.
11. M.R. Chmielewski and J.W. Grzymala-Busse. Global discretization of continuous attributes as preprocessing for machine learning. *International Journal of Approximate Reasoning*, 15:319–331, 1996.
12. K.J. Cios, W. Pedrycz, R.W. Swiniarski, and Lukasz A. Kurgan. *Data Mining: A Knowledge Discovery Approach*. Springer, 2007.
13. W.W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, 1995.
14. W.J. Conover. *Practical nonparametric statistics*. Applied probability and statistics. Wiley, 1999.
15. T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.
16. J. Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30, 2006.
17. J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.
18. J. Dougherty, R. Kohavi, and M. Sahami. Supervised and Unsupervised Discretization of Continuous Features. In *Proceedings of the 12th International Conference Machine Learning*, pages 194–202, 1995.
19. T. Elomaa and J. Rousu. General and efficient multisplitting of numerical attributes. *Machine Learning*, 36(3):201–244, 1999.
20. U. Fayyad and K. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102, 1992.
21. U.M. Fayyad and K.B. Irani. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *Proceedings of the 13th International Joint Conference on Uncertainly in Artificial Intelligence*, pages 1022–1029, 1993.
22. A. Fernández, M.J. del Jesus, and F. Herrera. On the 2-Tuples Based Genetic Tuning Performance for Fuzzy Rule Based Classification Systems in Imbalanced Data-Sets. *Information Sciences*, 180(8):1268–1291, 2010.
23. E. Frank and I.H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the 15th International Conference on Machine Learning*, pages 144–151, 1998.
24. Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, 1996.
25. M. Galar, A Fernández, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484, 2012.
26. S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064, 2010.
27. S. García and F. Herrera. An Extension on Statistical Comparisons of Classifiers over Multiple Data Sets for all Pairwise Comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.
28. S. García, J. Luengo, J. Saez, V. Lopez, and F. Herrera. A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):734–750, 2013.
29. V. García, J. S. Sánchez, and R. A. Mollineda. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems*, 25(1):13–21, 2012.
30. L. Gonzalez-Abril, F.J. Cuberos, F. Velasco, and J.A. Ortega. Ameva: An autonomous discretization algorithm. *Expert Systems with Applications*, 36:5327–5332, 2009.
31. J.W. Grzymala-Busse. A multiple scanning strategy for entropy based discretization. In *Foundations of Intelligent Systems*, volume 5722 of *Lecture Notes in Computer Science*, pages 25–34. 2009.
32. J.W. Grzymala-Busse. Discretization based on entropy and multiple scanning. *Entropy*, 15:1486–1502, 2013.
33. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemannr, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11:10–18, 2009.
34. H. He and E.A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.
35. S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.
36. J. Huang and C.X. Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, 2005.
37. W. Huang. *Discretization of Continuous Attributes for Inductive Machine Learning*. University of Toledo., 1997.
38. D. Janssens, T. Brijs, K. Vanhoof, and G. Wets. Evaluating the performance of cost-based discretization versus entropy- and error-based discretization. *Computers & Operations Research*, 33(11):3107–3123, 2006.
39. G. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, 1995.
40. H. Kaizhu, Y. Haiqin, K. Irwinng, and M.R. Lyu. Imbalanced learning with a biased minimax probability machine. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(4):913–923, 2006.
41. R. Kerber. ChiMerge: Discretization of Numeric Attributes. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 123–128, 1992.
42. R. Kohavi. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, volume 2, pages 1137–1143, 1995.
43. S. Kotsiantis and D. Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.
44. L.A. Kurgan and K.J. Cios. CAIM Discretization Algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):145–153, 2004.
45. L.A. Kurgan, K.J. Cios, and S. Dick. Highly scalable and robust rule learner: performance evaluation and comparison. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1):32–53, 2006.

46. T. Landgrebe, P. Paclik, S. Tax, D.and Verzakov, and R. Duin. Cost-based classifier evaluation for imbalanced problems. *Lecture Notes in Computer Science*, 3138:762–770, 2004.
47. H. Liu and R. Setiono. Feature selection via discretization. *IEEE Transactions on Knowledge and Data Engineering*, 9:642–645, 1997.
48. V. López, A. Fernández, S. García, V. Palade, and F. Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.
49. J. Luengo, A. Fernández, S. García, and F. Herrera. Addressing data complexity for imbalanced data sets: analysis of smote-based oversampling and evolutionary undersampling. *Soft Computing*, 15:1909–1936, 2011.
50. D.J. Newman and A. Asuncion. UCI machine learning repository, 2007.
51. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffman Publishers, 1993.
52. F.J. Ruiz, C. Angulo, and N. Agell. IDD: A Supervised Interval Distance-Based Method for Discretization. *IEEE Transactions on Knowledge and Data Engineering*, 20(9):1230–1238, 2008.
53. F. Tay and L. Shen. A Modified Chi2 Algorithm for Discretization. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):666–670, 2002.
54. C.J. Tsai, C.I. Lee, and W.P. Yang. A discretization algorithm based on Class-Attribute Contingency Coefficient. *Information Sciences*, 178(3):714–731, 2008.
55. T.S. Wiens, B.C. Dale, M.S. Boyce, and G.P. Kershaw. Three way k-fold cross-validation of resource selection functions. *Ecological Modeling*, 212(3-4):244–255, 2008.
56. F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, 1945.
57. A. Wong and T.S. Liu. Typicality, Diversity, and Feature Pattern of an Ensemble. *IEEE Transactions on Computers*, 24(2):158–181, 1975.
58. P. Yang, J. S. Li, and Y. X. Huang. HDD: a hypercube division-based algorithm for discretisation. *International Journal of Systems Science*, 42(4):557–566, 2011.
59. Y. Yang, G.I. Webb, and X. Wu. Discretization Methods. In *Data Mining and Knowledge Discovery Handbook*, pages 101–116. 2010.