LR-SDiscr: a novel and scalable merging and splitting discretization framework using a lexical generator

Habiba Drias, Hadjer Moulai and Nourelhouda Rehkab

Department of Computer Science, Laboratory of Research in Artificial Intelligence, USTHB, Algiers, Algeria

ABSTRACT
In this paper, we propose a novel supervised discretization method namely LR-SDiscr. It is based on a Left to Right (LR) scanning technique, which partitions automatically the input stream into intervals. Its originality resides in the fact it handles both merging and division operations in the same process, and hence it benefits of the use of a large spectrum of statistic measures. The second strength of our proposal is the reduction of the discretization complexity by processing the data in only one pass. Extensive experiments were conducted on various cut-point functions using public benchmarks that include small, large and medical datasets. LR-SDiscr outperforms several classical and recent discretization methods.

ARTICLE HISTORY
Received 23 June 2018
Accepted 23 November 2018

KEYWORDS
Supervised discretization; classifier; massive data; machine learning; data preprocessing; lexical generator

1. Introduction
In machine learning, most of the time, algorithms are designed to manipulate discrete attributes whereas in real life, data are often presented in a continuous format. In data mining, data preprocessing is the preliminary step before going into the mining process and discretization is one of its key focuses. The need to discretize continuous attributes, is of a great utility for several applications. Discretization aims at improving the performance of classifiers, as handling discrete data yields more effective outcomes than handling continuous data. Another important objective is the scalability as by dividing continuous data into intervals, the cardinality of discrete values is much smaller than that of the continuous values. A last goal is that discrete data are easier to interpret and use in contrast to continuous data.

Discretization consists in dividing distinct values of a continuous variable into a number of disjoint ranges, each one having its own label. There are two main ways to undertake this task. The first schema corresponding to a top-down approach, starts with the whole set of data and keeps dividing it into contiguous subsets until some termination criteria. The last subsets constitute the intervals. The second bottom-up strategy starts with a set of intervals made respectively by a single value and keeps merging adjacent intervals until reaching some termination conditions. The main issue is to determine a cut-point in
case of division, a grouping-point in case of merging sub-intervals and stopping criteria to terminate the process.

The approach we propose is novel as it gets completely out of these schemes. It uses at the same time partitioning of the top-down strategy and merging of the bottom-up framework, while processing the discretization. The idea is that, while the process progresses, each point separating two values is tested if it is a cut-point or a grouping point using a statistic measure. Then the deduced adequate action is performed.

The present paper is an extension of the article published in the proceedings of the 2018 ACIIDS conference (Drias, Moulai, & Rehkab, 2018). It provides more details on the proposal as well as a theoretical development and extensive experiments for the approach validation. More precisely, the extension includes the following points. The related work section is enriched with other efforts. Also, a theoretical part is developed to validate the proposal: a theorem showing that just one look-ahead value is sufficient to perform the discretization. On the other hand, the discretization algorithm is generated automatically by a lexical analyser generator, which is Lex. This tool is described and its corresponding programme adapted to discretization, is shown. The implementation is simple and fast and the generated algorithm is accurate, short and rapid. Moreover, the calculation of the time and space complexities of the algorithm is provided. Besides, the possible statistics for use in LR-SDiscr are extended to other measure functions such as the distance of Mantares and Zita, in addition to $\chi^2$ and the entropy. Finally, in the experiments section, other datasets were tested, in total 23 benchmarks were tested whereas only 9 were considered initially. More comparisons with state-of-the-art methods were performed in order to assess the superiority of the proposed algorithm.

1.1. Top-down discretization methods

A top-down discretization method starts with an empty list of cut-points and as the algorithm moves forward, it adds those calculated and used to partition the intervals. Prior to this process, the values are sorted, then the best cut-point is searched. Afterwards, the set of sorted values is divided into two distinct intervals according to the cut-point. The process is iterated for each range until the stop criterion is met. The division operation is handled by the means of a measure such as binning, entropy, correlation and precision.

Binning is a mechanism that consists merely in distributing the set of sorted values in several bins. These partitions can be obtained using the equal-width or equal-frequency techniques, in which a number of intervals is defined. In equal-width method, all ranges have the same width or size, whereas in equal-frequency technique, each interval contains about the same number of learning examples. Since the number of intervals is fixed at the beginning, there is no need for a stop criterion.

The Shannon entropy is the measure the most often used in supervised discretization methods (Liu, 2002; Shannon & Weaver, 1949; Thornton, 1992). It is defined for a variable $X$ by Equation (1).

$$H(X) = - \sum_x p_x \log p_x,$$

where $x$ represents a value of $X$ and $p_x$ the estimated probability of occurrence of $x$. This is the average amount of information per event, where the information is defined by
Information is high for unlikely events and low otherwise. Consequently, the entropy $H$ is high when all events are equally likely ($p_{x_i} = p_{x_j}$), and low when the probability of an event is close to 1 ($p_x = 1$) and that of the others to 0.

A top-down discretization method splits the intervals from 1 to the maximum number of intervals namely, $maxIntervals$. And for one split operation, the process scan all the current intervals to search for the cut-point. Therefore the time complexity of the method can be expressed as:

$$
\sum_{i=n-1}^{i=maxIntervals} i = \sum_{i=1}^{i=maxIntervals} (n - i) \\
= \sum_{i=1}^{i=maxIntervals} n - \sum_{i=1}^{i=maxIntervals} i \\
= n \cdot maxIntervals - \frac{maxIntervals \cdot (maxIntervals + 1)}{2} \\
= n \cdot maxIntervals - \frac{maxIntervals^2 + maxIntervals}{2}.
$$

The expression $(maxIntervals^2 + maxIntervals)/2$ being negligible relatively to $(n \cdot maxIntervals)$, the time complexity is then $O(n \cdot maxIntervals)$.

**1.2. Bottom-up discretization methods**

A bottom-up discretization method begins by considering each continuous value in an interval as a potential grouping point, then it merges the adjacent intervals that are similar. The corresponding algorithm can be summarized in four steps:

1. Sort the input values.
2. Put each value in an interval.
3. While the stop criterion is not satisfied
   3.1) Find the best pair of similar adjacent intervals.
   3.2) Merge the pair into a single interval.
4. stop.

$\chi^2$ is a statistic measure that evaluates the relationship between two objects with regards to the classes they belong to, respectively. In discretization, it is used to measure whether two adjacent intervals are independent of the classes their values belong to or not. Its formula is given by Equation (3).

$$
\chi^2 = \sum_{i=1}^{i=p} \sum_{j=1}^{j=p} \frac{(A_{ij} - E_{ij})^2}{E_{ij}},
$$

where:
• $p$ is the number of classes.
• $A_{ij}$ is the number of distinct values in the interval $i$ of class $j$.
• $E_{ij}$ is the expected frequency of $A_{ij} = (R_i \times C_j)/n$.
• $R_i$ is the number of instances in the interval $i$, which is equal to $\sum_{j=1}^{p} A_{ij}$.
• $C_j$ is the number of instances in class $j$, which is equal to $\sum_{i=1}^{m} A_{ij}$.
• $n$ is the total number of instances, which is equal to $\sum_{j=1}^{p} C_j$.

The worst case complexity of a bottom-up discretization method is expressed by the number of times the function $\chi^2$ is called. At the first iteration, the function $\chi^2$ is called $(n - 1)$ times for the $(n - 1)$ adjacent intervals. Knowing that a single pair of intervals is merged each time, then at the next iteration, it will be called $(n - 2)$ times, and so on until reaching a number of intervals equal to $\text{maxIntervals}$. The total number of invoking $\chi^2$ is then equal to:

$$
\sum_{i=1}^{\text{maxIntervals}} i = \sum_{i=1}^{n-1} (i) - \sum_{i=1}^{\text{maxIntervals}-1} (i)
$$

$$
= \frac{n \times (n - 1)}{2} - \frac{\text{maxIntervals} \times (\text{maxIntervals} - 1)}{2}
$$

$$
= \frac{(n^2 - n)}{2} - \frac{(\text{maxIntervals}^2 - \text{maxIntervals})}{2}
$$

$$
= \frac{(n^2 - n - \text{maxIntervals}^2 + \text{maxIntervals})}{2}
$$

The expression $(\text{maxIntervals}^2 + \text{maxIntervals})$ being very small relatively to $n^2$, the time complexity is then $O(n^2)$.

1.3. Motivation, scope and organization

Besides the main framework, we distinguish two kinds of discretization methods: supervised and unsupervised. Unlike a supervised method that uses the class attribute in the cut-point formulation, an unsupervised method ignores utterly the class attribute. Unsupervised methods such as equal-width and equal-frequency were the precursors of the discretization research.

In this paper, we propose a supervised discretization algorithm called LR-SDiscr(Left to Right Supervised Discretization). The originality of our approach is based on the use of a lexical generator to build automatically a discretization algorithm of type LR (Left to Right). The data are presented to the generated algorithm, which consists in reading the sequence from left to right and determining breakpoints when the measurement of division between two consecutive values is satisfactory to constitute intervals. A cut-point corresponds to the recognition of an interval, whereas a grouping-point corresponds to the construction of the interval. Lex (Lesk, 1975; Lesk & Schmidt, 2016) tool enables this process to be effectively and efficiently implemented.

LR-SDiscr was implemented and extensive experiments were carried out on small, large public datasets as well as on medical benchmarks. Comparison with the well-known Chi-merge method but also with recent state-of-the-art algorithms is performed.
A very short version of this work is published in Drias et al. (2018), where the novel idea of handling simultaneously both cutting and merging operations is presented with preliminary experimental results. The unsupervised version of the method can also be found in Drias, Rehkab, and Moulai (2017).

The rest of this paper is organized as follows. The next section provides a state-of-the-art of existing discretization methods. Section 3 is a detailed overview of the Chimerge and $\chi^2$ algorithms, as our proposal is challenging these techniques in particular from the time complexity point of view. In Section 4, we present our main contribution, which is the LR-SDiscr method. We explain how for the first time Lex is used to generate a discretization algorithm and then describe the various explored discretization measures. Section 5 reports the numerous experiments undertaken and the achieved outcomes. Comparisons of the results with those of Chimerge and recent state-of-the-art algorithms are performed. Finally we conclude this work and provide some future perspectives.

2. Related work

An abundant literature exists for discretization algorithms, which differ mainly by the various options offered for the design: top-down or bottom-up, supervised or unsupervised and the multitude of splitting and merging measures. A non-exhaustive chronological survey of the most important of them is presented below.

ChiMerge (Kerber, 1992) is the first effective method that proposes an approach other than division using the measurement of $\chi^2$. It is a supervised bottom-up method that exploits the relationship between the attribute and its class to merge adjacent intervals. Initially each distinct value of the continuous attribute is considered in an interval. Then, the $\chi^2$ test is performed on each pair of adjacent intervals and those with the smallest value of $\chi^2$ are merged. The process is iterated until a stop criterion is met. The value of the level of significance of $\chi^2$ acts on the discretization and determines the number of intervals. This parameter will prevent the algorithm from creating too many intervals. The discretization stops when the $\chi^2$ of all adjacent intervals is greater than a certain threshold, which is determined as the level of significance. The maximum number of intervals can also be used as a parameter to limit the number of constructed intervals.

Based on experimentations, Dougherty, Kohavi, and Sahami (1995) showed that induction algorithms are more effective when data are prior discretized. They tested three methods as a preprocessing step to the C4.5 algorithm and a Naïve-Bayes classifier: equal-width Intervals, 1RD proposed in Holte (1993) and the entropy minimization heuristic devised in Fayyad and Irani (1993). The results revealed that all these methods improved the Naïve-Bayes classifier accuracy especially the entropy heuristic. Increase in accuracy was also observed for C4.5 but not on all the tested datasets.

$\chi^2$ (Liu & Setiono, 1997) is an improved version of ChiMerge using the $\chi^2$ measure and a second measure representing data inconsistency. The statistic $\chi^2$ is used to discretize until an inconsistency is met in the data. Also, the number of inconsistencies is used to eliminate noise.

A comprehensive synthesis on discretization techniques is presented in Liu (2002). The authors introduced the issue of discretization as well as its history, the existing methods and its impact on classification and other applications. They concluded that important
efforts have been done in the domain of discretization research but still new methods are needed.

Kurgan and Cios (2004) proposed a supervised discretization algorithm, called Class-Attribute Interdependence Maximization (CAIM), with the purpose to maximize the class-attribute interdependence and to minimize the number of intervals. The experiments held showed indeed the satisfaction of both criteria when comparing the algorithm to the state-of-the-art algorithms at that time.

Su and Hsu (2005) went further in the development of the $\chi^2$ method. They proposed an extended $\chi^2$ algorithm with a technique that determines the predefined inconsistency rate based on the least upper bound of data misclassification error. In addition, the effect of variance in merging two intervals is considered.

Lee, Tsai, Yang, and Yang (2007) proposed an improved version of the CAIM algorithm. They considered the data distribution in the discretization formula used in CAIM in order to adjust the accuracy of the results and called their discretization algorithm Class-Attribute Contingency Coefficient (CACC). The experimental results showed that compared to CAIM, CACC yielded a better performance for classification purposes. The number of generated rules and execution time of a classifier are comparable for both algorithms.

Tsai, Lee, and Wei-Pang (2008) proposed an algorithm based on CACC with the aim to maintain a high interdependence between the target class and the discretized attribute for better accuracy. Experiments were performed and results were compared to those of seven methods. CACC was shown to yield the best outcomes.

Shang, Yu, Jia, and Ji (2009) explored artificial intelligence tools namely, a neural network and a genetic algorithm for the determination of optimal cut-points. The cut-points are trained through a four-layer neural network and their number is optimized by a genetic algorithm. The experimental results showed a better performance compared to another method.

Bettinger (2011) proposed an algorithm called ChiD, which is a hybrid algorithm based on Chimerge and $\chi^2$. He enriched the method by a criterion, the logworth to define the significance of a set of cut-points.

Grzymala-Busse (2013) utilized the entropy measure to discretize numeric data. A strategy based on multiple scanning of the attributes allows the selection of the attribute for the determination of the best cut-point.

Madhu, Rajinikanth, and Govardhan (2014) designed an algorithm based on the z-score standard deviation technique for continuous attributes on biomedical datasets. Experiments were undertaken on medical datasets and the results showed a better accuracy and also less classifier confusion for decision making process relatively to other methods.

Jishan, Raisul, Rashu, Haque, and Rahman (2015) presented an application of discretization in the domain of educational datamining and more precisely a prediction model for the students final grade of a particular course. They developed a discretization method called the Optimal Equal Width Binning and an over-sampling technique known as the Synthetic Minority Over-Sampling. They tested the application on real data and the result yields a good accuracy of the prediction model.

Grzymala-Busse and Mroczek (2016) experimented for a comparison purpose, four discretization methods, all based on entropy: a method using C4.5, the equal interval width method, the equal-frequency interval method and the multiple scanning method. They used as evaluation measures, the error rate obtained by ten-fold cross validation and
the number of nodes of the tree generated by C4.5. It turned out that the multiple scanning method is the best.

Through this non exhaustive synthesis on discretization research, we see that investigations on this topic have been regular over time and is still stimulating the interest of scientists of datamining community. All these efforts focused attention on the discretization effectiveness, but unfortunately not on the efficiency and scalability. Scalability is nowadays an extremely important indicator for model evaluation as data sizes are becoming more and more prohibitive. In this paper, we present an all different and novel algorithm that is capable to gather several options of the described methods in one unique platform while ensuring scalability. As basis, the algorithm uses the framework of ChiMerge and $\chi^2$, these methods being the most that influenced the studies done so far. The next section describes then the most important features of these methods.

3. A brief overview of ChiMerge and its improved variants

Equal-width and equal-frequency methods have long been used in the field of machine learning. But their performances are weak where some concepts become impossible to learn. Kerber (1992) considers a precise discretization, which ensures an intra-interval uniformity and an inter-interval difference. An interval must have a certain consistency in the relative frequency of class, whereas two adjacent intervals with similar relative class frequencies must be merged. The ChiMerge method was the first to move away from the concept of division in its discretization process, preferring to merge intervals instead of partitioning them. Using the measurement of $\chi^2$, it is possible to judge whether two adjacent intervals are similar or not. The ChiMerge algorithm is a supervised and ascending method, including the following steps:

1. Sort the attribute values in ascending order.
2. Consider each value in a distinct interval.
3. Calculate the value of $\chi^2$ for all adjacent intervals.
4. Merge the interval pairs with the smallest value of $\chi^2$.
5. Stop the process if a predefined stopping criterion is met (such as the significance level of $\chi^2$, the maximum number of intervals, the maximum inconsistency, etc) for all intervals.
6. Go to (3) otherwise.

The formula for calculating $\chi^2$ is given by Equation (3). It helps determining whether the difference in the relative frequencies between the adjacent intervals is a real relationship between the numerical attribute and the class, or the result of a coincidence.

The sorting of the continuous values of the attribute to be discretized is a very important step. Indeed, the choice of the sorting algorithm that will perform this task can influence the performance of the ChiMerge algorithm in terms of time complexity. The stopping criterion must satisfy a minimum probability of independence between the intervals. The $\chi^2$ threshold is set according to a level of significance. If a computed value of $\chi^2$ is greater than or equal to the threshold, the class and the attribute are not independent. Consequently, if the threshold is too high, the discretization will last longer, resulting in an over-discretization and a large number of intervals, whereas a too small threshold
generates a sub-discretization and a limited number of intervals. In order to cope with this issue, two other parameters were added: the minimal number of intervals (minIntervals) and the maximum number of intervals (maxIntervals).

ChiMerge is a robust algorithm that rarely ignores large intervals or chooses an interval marker when there is a better one. It is a simple to use method, finding a good setup is easy. Indeed, a good threshold level of significance, and a constraint on the number of intervals (maxIntervals, minIntervals) generally produces good discretization. The main source of the robustness of this method is the use of the class attribute during the construction of the intervals, and the adjustment of the number of intervals according to the characteristics of the data. In addition, it is applicable for multi-class learning (ie, examples are distributed over more than two classes). Another advantage of ChiMerge is that it provides an accurate summary of numerical attributes, which improves the understanding of the relationship between numerical attributes and the class attribute.

The main major disadvantage of ChiMerge is that $\chi^2$ statistic allows some noise tolerance. Consequently, the threshold parameter is not sufficient to determine the cut-points in an effective way.

$\chi^2$ in Liu and Setiono (1997), is an improved version of ChiMerge by automatically calculating the significance level of $\chi^2$ and considering the inconsistency measure. However, it still needs the user to provide the inconsistency rate to stop the merge procedure and does not consider the independence that has a significant impact on the discretization schema. The modified version of $\chi^2$ takes the independence concept into account and replaces the inconsistency test in $\chi^2$ by the quality of an approximation after each step of the discretization. This makes Modified $\chi^2$ a fully automated method with better predictive accuracy than $\chi^2$. Extended $\chi^2$ determines the predefined classification error rate from the data itself and considers the effect of the variance in two adjacent intervals. With these changes, Extended $\chi^2$ can handle an uncertain set of data. Experiments on these bottom-up approaches using C5.0 have also shown that Extended $\chi^2$ surpasses other bottom-up discretization algorithms and its average discretization schema can reach the highest accuracy.

The proposed LR-SDiscr method is based on ChiMerge and $\chi^2$ principles. The main purposes are first to palliate to the time complexity issue of the previous works and hence conceive a more efficient discretization tool and second, to increase accuracy. This approach is novel in the sense that it carries out at the same time division and merging operations during the discretization process. Therefore it can use a large spectrum of statistic measures, among which several cut-point functions other than $\chi^2$ can be explored in order to compare the discretization effectiveness and determine the best measure to exploit. Several variants of the method are designed, thanks to the possibility of choice of a profusion of statistics for the determination of the cut-points.

4. The proposed LR-SDiscr approach

One of the disadvantages of ChiMerge is the abusive calculation of the $\chi^2$ measurement, due to the repetitive instructions of the intervals generation during the discretization process. To cope with this issue, we propose a new discretization method using a lexical analysis generator (Lesk, 1975; Lesk & Schmidt, 2016).
The key idea of our approach is to scan the data in a Left to Right (LR) direction to create the intervals. Initially, after recognizing the two first values \(v_0\) and \(v_1\), the cup-point measure is calculated. If it is lower than the predefined threshold then these two values are merged into a single interval \([v_0, v_1]\). Then on recognizing the next value, the test is performed between this new interval and the interval \([v_2]\). In case, the statistic measure between \(v_0\) and \(v_1\) is greater than the threshold, a cut-point which is \(v_1\) is set. Afterwards, on recognizing the next value, the test will be performed between the intervals \([v_1]\) and \([v_2]\). The process continues for the whole input file. The only parameter used for the algorithm is the level of significance of \(\chi^2\). The maximum number of intervals, if required can be used in the experiments to tune the threshold. The general framework of LR-SDiscr can be summarized as follows:

1. Sort the values and calculate their respective frequencies using the efficient well-known heapsort algorithm (Knuth, 1997).
2. Insert each value in a separate range, that is the value \(v_i\) in the interval \([v_i]\) for \(i = 0\) to \(n\).
3. Prepare the input file according to the following form: \(v_0 f_0 v_1 f_1 \cdots v_n f_n\), where \(v_i\) is a continuous value of the attribute and \(f_i\) its frequency in the data set.
4. Use the Lex tool to generate the discretization programme.
5. Launch the discretization programme, which will scan the input file in a left to right direction, while performing the discretization.
6. Stop the process at the end of the file. Before launching the execution of the programme, the empirical \(\chi^2\) threshold parameter has to be tuned in terms of either the maximum number of intervals if required, or just the level of significance.

**Theorem 4.1** Consider two adjacent intervals \(i\) and \(j\) and the \(\chi^2\) measure defined as in Equation (3), a cut-point between the interval \(i\) and the interval \(j\) is also a cut-point between \(i\) and the interval containing only the first value of \(j\).

**Proof.** A cut-point exists between \(i\) and \(j\) if:

\[
\chi^2 = \sum_{j=1}^{i=2} \sum_{j=1}^{j=p} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \leq \text{threshold},
\]

\[
\chi^2 = \sum_{j=1}^{i=2} \sum_{j=1}^{j=p} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} = \sum_{i=1}^{i=2} \frac{(A_{i1} - E_{i1})^2}{E_{i1}} + \sum_{j=2}^{j=p} \sum_{i=1}^{i=2} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \leq \text{threshold}.
\]

When the interval \(j\) contains only one value, the latter belongs to one class. Hence, \(p\) is equal to 1 and the \(\chi^2\) is expressed as:

\[
\chi^2 = \sum_{i=1}^{i=2} \sum_{j=1}^{j=1} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} = \sum_{i=1}^{i=2} \frac{(A_{i1} - E_{i1})^2}{E_{i1}}.
\]
Then:

$$\chi^2 = \sum_{i=1}^{2} \frac{(A_{i1} - E_{i1})^2}{E_{i1}} = \sum_{i=1}^{2} \sum_{j=1}^{p} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} - \sum_{i=1}^{2} \sum_{j=1}^{p} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$$

$$\leq \text{threshold} - \sum_{i=1}^{2} \sum_{j=1}^{p} \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$$

$$\leq \text{threshold}.$$ 

In this method, only one look-ahead value is tested with the current interval for determining a cut-point. We can consider it as a LR(1) discretization method. If $k$ look-ahead values are considered to test for a cut-point, then the method is qualified as LR(k). According to Theorem 4.1, LR(1) method is sufficient to perform the discretization.

### 4.1. Lex as a tool for numeric discretization

Numeric discretization consists in recognizing values of same significance or dependency and insert them in intervals. Lex, a lexical analyzer (Lesk, 1975; Lesk & Schmidt, 2016) is a tool for automatically and rapidly implementing a lexer for a programming language or a sequence of recognizable objects. It is widely used in compilers’ construction but it is also prevalent in many areas that require patterns’ recognition, such as word processing and natural language. As the discretization mechanism is based on recognizing intervals of values to classify them according to their importance, Lex is an appropriate and convenient tool for handling this task. As shown in Figure 1, Lex generates automatically the discretization programme from a source containing specifications of entities and actions. Each entity is described by a regular expression followed by an action composed of a programme fragment. The action is executed each time the corresponding interval is recognized during the discretization process.

Afterwards, the discretization programme when executed produces the intervals. Figure 2 illustrates the role of the discretization programme. It takes as input, the data and the statistic threshold such as that of $\chi^2$ and produces the sequence of ranges.

![Figure 1](image1.png)

**Figure 1.** Using Lex to produce the discretization programme (step1).

![Figure 2](image2.png)

**Figure 2.** Executing the discretization programme for yielding the sequence of intervals (step2).
The Lex source has the following format:

{definitions}
%%
{rules}
%%
{user subroutines}

There are three parts delimited by the double symbol ‘%%’. The rules specify how to transform the source file into the target one. It may use some entities defined in the first part of the Lex programme and some procedures appearing in the last part. The first and last parts are optional as well as the second delimiter ‘%%’. In our work, rules have to be designed to transform the continuous values into intervals. A rule respects the following syntax:

Regular-expression {action}

The regular expression builds from the objects presented in the input file, a target element. In other words, it allows the partition of the input stream data according to the target objective. The action is a programme fragment to execute each time a target entity is recognized. It is written in a computer language such as C or Java. In our case, the input is represented as a sequence of pairs including a value of type real and its frequency of type integer in the dataset. Recall that the data are sorted before discretization.

The Lex programme we designed is outlined in Figure 3.

Statistic is a function that calculates the measure that allows to decide to cut or to pursue the creation of an interval and relop is the appropriate relational operator. For our experiments, we used $\chi^2$ and the entropy as measures for our method. For modifying the measure, we have just to change the function of the statistic that appears in the third part of the Lex source and letting the rest of the programme unchanged.

4.1.1. LR-SDiscr time complexity

Let $n$ be the number of instances. The heapsort algorithm was applied to sort the numerical values of an attribute to be discretized and was extended to count the frequencies of each distinct value. This operation requires a time complexity of $O(n\log n)$, which is the best for sorting. Then, the discretization of one attribute using LR-SDiscr needs scanning its values only once to create the intervals, which requires a $O(n)$ linear complexity. Therefore the total time complexity (including the sorting) is $O(n\log n)$.

4.1.2. LR-SDiscr spatial complexity

To sort $n$ values of an attribute, it is necessary to save them in an array of size $n$. After the sorting operation, the data will occupy a space of size $2^*n$ because each value will be followed by its frequency in the dataset. For the discretization of an attribute, one needs an array of size equal to the total number of intervals. In the worst case, this number is equal to $n$. The spatial complexity is therefore $O(n)$. 
Without considering the sorting step for all discretization algorithms, it is clear that the LR-SDiscr time complexity is the best of all those of the methods based on top-down or bottom-up strategies. We calculated in Subsection 1.1 the time complexity for top-down methods and found it is $O(n \times \text{maxIntervals})$ and in 1.2 the time complexity for bottom-up methods and found it is $O(n^2)$. For LR-SDiscr, it is linear because the sorted data are scanned only once to determine the intervals.

The second important strength of our approach is the ability and ease allowed by the programme implementation to change the cut-point measure. It is a very interesting advantage because it helps testing several relevant statistics and selecting the one that yields the best outcomes. Of course, the quality of the results depends also on the cut-point measure threshold. With experiments, this parameter is tuned taking into account the number of intervals for instance.

### 4.2. Possible statistics for use in LR-SDiscr

In the implementation of our discretization algorithm, we have already seen that the statistic measure can be modified easily. We can use all the functions that we can find in the

```
space                 \[[^n]+\]
digit                 [0-9]
integer               \{(digit)\}+

\%
{space}               \{
{integer}.\{integer\} \{value = atof(yytext) ;
}
{integer}             \{frequency = atoi(yytext);
                     Interval2 = [(value, frequency)];
                     if (statistic(interval1, interval2) relop threshold)
                         then interval1 = interval2
                       else interval1 = merge (interval1, interval2)

\%
function statistic (interval1,interval2);
{
}
}

function merge (interval1, interval2);
{
}

main {
    interval1 = empty;
    threshold = ...;
    yylex();
}
```

Figure 3. The Lex source for generating the discretization programme.

Without considering the sorting step for all discretization algorithms, it is clear that the LR-SDiscr time complexity is the best of all those of the methods based on top-down or bottom-up strategies. We calculated in Subsection 1.1 the time complexity for top-down methods and found it is $O(n \times \text{maxIntervals})$ and in 1.2 the time complexity for bottom-up methods and found it is $O(n^2)$. For LR-SDiscr, it is linear because the sorted data are scanned only once to determine the intervals.

The second important strength of our approach is the ability and ease allowed by the programme implementation to change the cut-point measure. It is a very interesting advantage because it helps testing several relevant statistics and selecting the one that yields the best outcomes. Of course, the quality of the results depends also on the cut-point measure threshold. With experiments, this parameter is tuned taking into account the number of intervals for instance.

### 4.2. Possible statistics for use in LR-SDiscr

In the implementation of our discretization algorithm, we have already seen that the statistic measure can be modified easily. We can use all the functions that we can find in the
literature and among them, those that are cited in the related work section. Below are some measures that we implemented for the selection of the one that yields the best outcomes.

### 4.2.1. Using $\chi^2$

In our approach, the $\chi^2$ measure is one of the statistics that is computed to decide about the fusion of adjacent intervals or not. The formula of $\chi^2$ is expressed in Equation (3).

$$ \chi^2 = \sum_{i=1}^{i=m} \left( \frac{p_{i,\text{left}} - p_{i,\text{right}}}{p_{i,\text{left}} \log p_{i,\text{left}} - p_{i,\text{right}} \log p_{i,\text{right}}} \right) $$

$m$ is the number of classes, $p_{i,\text{left}}$ and $p_{i,\text{right}}$ the respective probabilities that an instance is on the left or on the right of the cut-point. $p_{i,\text{left}}$ and $p_{i,\text{right}}$ are respectively the probabilities that an instance situated on the left or on the right part of the cut-point belongs to class $i$.

In a discretization algorithm, the cut-point is determined for a small value of the entropy, which corresponds to a large gain in information. A threshold is determined to judge if a continuous value can be considered as a cut-point or not. It represents the rate of minimum pureness that a continuous value should reach to be considered as a cut-point.

### 4.2.2. Using the entropy

We used the entropy formulation given in Quinlan (1993) to discretize continuous values and that appears in Equation (4).

$$ H = - \sum_{i=1}^{i=m} p_{i,\text{left}} \log p_{i,\text{left}} - p_{i,\text{right}} \log p_{i,\text{right}} $$

### 4.2.3. Variants of the entropy

For more effectiveness, the two following complementary strategies can be implemented when using the entropy statistic.

The Minimum Description Length Principle (MDLP) variant (Fayyad & Irani, 1993) suggests that potential cut-points are the values that form a boundary between classes, after having ordered the set of continuous values of the attribute. However, not all bounds are considered immediately as cut-points. MDLP is a strategy that is used to select the right cut-points. Once the selection is made, the process of discretization is completed. The minimum description length concept is used as a stop criterion.

The Distance of Mantaras (1991) is introduced to evaluate the potential of the cut-points. Let consider two partitions of a sorted set of continuous values and the number of classes contained in each partition. The principle is to calculate the distance of Mantaras between these two partitions using these data for all cut-points. The cut-point that minimizes this distance is then selected. This method uses the same stop criterion as MDLP to determine if more cut-points are to be added.

### 4.2.3.1. Using Zeta

Zeta (Z) Ho and Scott (1993) is a measure that evaluates the strength of association between the class and the attribute. A Zeta value for a cut-point is
computed as in Equation (5).

\[ Z = \sum_{i=1}^{i=k} n_{f(i),i} \]  (5)

where:

- \( k \) is the number of intervals specified beforehand (2 by default).
- \( f(i) \) is the index of the class with the most instances in the interval \( i \).
- \( n_{f(i),i} \) is the number of instances in the interval \( i \) with the index of class \( f(i) \).

The cut-point with the highest value of \( Z \) is chosen and the process is iterated until a stop criterion is met, which is the number of intervals to be reached for each attribute.

### 4.2.3.2. Using the accuracy

The accuracy (Chan, Batur, & Srinivasan, 1991) is a measure often used to assess the accuracy of a classifier. The quality of partitioning is tested by using a classifier to see if the partitioning scheme improves accuracy. This method takes a long time because it involves learning a classifier, an efficient method should be considered. The process stops when there is no improvement in the accuracy rate. As discretization can be seen as a classifying process, where the intervals represent the classes, this measure can be exploited.

### 5. Experimental results

In order to prove the performance of the proposed approach, we undertook extensive experiments with the aim to compare the achieved results with those of the state-of-the-art methods. Evaluation criteria such as accuracy, speed and scalability are considered for the comparison.

The Lex generator as well as the Java language were used for the implementation on a machine of Processor Intel Core i5-3317U CPU @ 1.70GHz x 4 and a RAM of 4.00 GB under the Ubuntu 14.04 LTS 64-bit operating system.

#### 5.1. The tested datasets

For a comparative purpose, the same datasets as those used in Liu (2002) although of small size, were considered in a first step. The first part of Table 1 shows these small tested datasets with their characteristics: the number of instances (#instances) and the number of attributes (#attributes). Then large datasets (UCI, Irvine CA, 2008) shown in the second part of the table were also investigated in order to test the scalability of the proposed methods. We also treated datasets belonging to the health domain, they appear at the end of the table.

#### 5.2. Parameters setting and model evaluation

The discretization method is designed for classification purposes. We applied \( k \) folds cross validation in combination with C4.5 classifier, for parameter tuning and model evaluation.
The parameters we tuned are the cut-point threshold and the maximum number of intervals either one of them used to stop the discretization process. They correspond to the minimum error rate computed by C4.5. For the model evaluation, the mean of the $k$ error rates computed by C4.5 is used to measure the discretization performance. The empirical parameters as well as the error rate are reported in Tables 2–4 exhibiting the numeric results yielded respectively by LR-SDiscr using $\chi^2$, LR-SDiscr using the entropy and Chimerge for all tested datasets.

Table 1. Popular, large and medical Datasets.

| Dataset       | #Instances | #Attributes |
|---------------|------------|-------------|
| Australian    | 690        | 14          |
| Breast        | 699        | 9           |
| Heart         | 270        | 13          |
| Vehicle       | 846        | 18          |
| Iris          | 150        | 4           |
| Wine          | 178        | 13          |
| Pima          | 768        | 8           |
| Bupa          | 345        | 6           |
| Thyroid       | 215        | 5           |
| Ionos         | 351        | 34          |
| German        | 1000       | 20          |
| Yeast         | 1484       | 8           |
| Hypothyroid   | 3772       | 29          |
| Abalone       | 4177       | 8           |
| satimage      | 6435       | 36          |
| Handwritten digits | 10,992   | 16          |
| Letter recognition | 20,000    | 16          |
| Shuttle       | 57,999     | 9           |
| Appendicitis  | 106        | 7           |
| Cleveland     | 297        | 13          |
| Hepatitis     | 155        | 19          |
| SPECTF Heart  | 267        | 44          |
| Breast (Wisconsin) | 569     | 31          |

Table 2. Numerical results yielded by LR-SDiscr using $\chi^2$.

| Benchmark             | Threshold | Error % | #Nodes | Time  | #intervals |
|-----------------------|-----------|---------|--------|-------|------------|
| Australian            | 0.6       | 13.855  | 44.57  | 0.016 | 10         |
| Breast                | 0.2       | 4.373   | 25.58  | 0.005 | 5          |
| Heart                 | 0.8       | 21.497  | 25.56  | 0.005 | 7          |
| Vehicle               | 3.8       | 32.081  | 213.16 | 0.0414| 7          |
| Iris                  | 1.2       | 5.347   | 7.83   | 0.007 | 7          |
| Wine                  | 3.2       | 7.101   | 18.29  | 0.031 | 5          |
| Pima                  | 2.4       | 26.700  | 50.05  | 0.046 | 9          |
| Bupa                  | 0.4       | 37.952  | 35.46  | 0.008 | 10         |
| Thyroid               | 1.2       | 7.790   | 22.08  | 0.011 | 10         |
| Ionos                 | 1.0       | 9.924   | 27.91  | 0.056 | 8          |
| German                | 1.2       | 28.389  | 123.81 | 0.017 | 10         |
| Yeast                 | 10.4      | 59.895  | 22.40  | 0.058 | 5          |
| Hypothyroid           | 5.2       | 0.844   | 32.30  | 0.381 | 5          |
| Abalone               | 15.0      | 76.096  | 309.43 | 0.906 | 9          |
| satimage              | 43.0      | 14.858  | 808.76 | 1.671 | 5          |
| Handwritten digits    | 48.0      | 6.761   | 1070.05| 2.363 | 5          |
| Letter                | 0.2       | 20.253  | 7324.75| 0.487 | 8          |
| Shuttle               | 8.0       | 0.063   | 77.55  | 3.384 | 5          |
| Appendicitis          | 2         | 15.100  | 11.15  | 0.011 | 10         |
| Cleveland             | 2.5       | 44.070  | 53.04  | 0.008 | 10         |
| Hepatitis             | 0.2       | 18.500  | 7.89   | 0.004 | 8          |
| SPECTF Heart          | 0.6       | 21.146  | 4.13   | 0.037 | 9          |
| Breast(Wisconsin)     | 4         | 7.318   | 35.62  | 1.787 | 9          |
For the k-fold cross validation also called rotation estimation, $k$ is set to 10, in order to be able to compare with results of previous works. The principle consists in dividing the dataset into $k$ samples, then to select one sample as a set of validation and to consider the $(k - 1)$ remaining samples as a training set. The error is computed using the C4.5 with the training set. The operation is then repeated $k$ times by selecting each time

### Table 3. Numerical results of LR-SDiscr using the entropy.

| Benchmark       | Threshold | Error % | #Nodes | Time | #Intervals |
|-----------------|-----------|---------|--------|------|------------|
| Australian      | 0.1       | 14.039  | 46.50  | 0.010| 14         |
| Breast          | 0.3       | 3.9349  | 19.02  | 0.006| 15         |
| Heart           | 0.2       | 18.722  | 24.73  | 0.044| 15         |
| Vehicle         | 0.3       | 32.129  | 188.11 | 0.022| 12         |
| Iris            | 0.02      | 5.342   | 15.80  | 0.003| 15         |
| Wine            | 0.8       | 30.846  | 70.479 | 0.010| 15         |
| Pima            | 0.3       | 35.014  | 1.90   | 0.012| 12         |
| Bupa            | 0.3       | 37.033  | 41.94  | 0.006| 15         |
| Thyroid         | 0.3       | 21.537  | 24.10  | 0.004| 12         |
| Ionos           | 0.2       | 10.226  | 52.62  | 0.027| 14         |
| German          | 0.1       | 27.420  | 127.25 | 0.005| 9          |
| Yeast           | 0.1       | 64.458  | 18.82  | 0.027| 15         |
| Hypothyroid     | 0.1       | 7.719   | 1.00   | 0.028| 9          |
| Abalone         | 0.1       | 81.275  | 25.77  | 0.068| 10         |
| Satimage        | 0.1       | 63.239  | 202.00 | 0.441| 10         |
| Handwritten Digits | 0.1     | 29.490  | 1967.50| 0.274| 10         |
| Letter          | 0.1       | 20.969  | 8535.70| 0.608| 10         |
| Shuttle         | 0.1       | 4.510   | 40.09  | 0.558| 10         |
| Appendicitis    | 0.1       | 19.875  | 6.66   | 0.002| 8          |
| Cleveland       | 0.6       | 44.743  | 46.29  | 0.004| 9          |
| Hepatitis       | 0.1       | 16.397  | 3.30   | 0.002| 11         |
| Spectf Heart    | 0.1       | 20.531  | 6.00   | 0.006| 5          |
| Breast(Wisconsin)| 0.7     | 35.107  | 5.12   | 1.471| 11         |

### Table 4. Numerical results yielded by ChiMerge.

| Benchmark       | Threshold | Error % | #Nodes | Time |
|-----------------|-----------|---------|--------|------|
| Australian      | 2.7       | 15.053  | 6.36   | 0.189|
| Breast          | 2.7       | 5.277   | 26.52  | 0.010|
| Heart           | 2.7       | 20.979  | 15.41  | 0.022|
| Vehicle         | 6.2       | 32.964  | 417.73 | 0.171|
| Iris            | 4.6       | 5.143   | 5.49   | 0.008|
| Wine            | 4.6       | 12.655  | 22.86  | 0.068|
| Pima            | 2.7       | 28.427  | 28.51  | 0.137|
| Bupa            | 2.7       | 43.311  | 52.00  | 0.022|
| Thyroid         | 4.6       | 6.881   | 20.55  | 0.023|
| Ionos           | 2.7       | 11.683  | 47.16  | 0.792|
| German          | 2.7       | 29.410  | 12.15  | 0.157|
| Yeast           | 14.7      | 68.809  | 1.00   | 0.152|
| Hypothyroid     | 6.2       | 1.300   | 55.70  | 1.050|
| Abalone         | 35.6      | 74.974  | 1332.23| 7.341|
| Satimage        | 10.6      | 19.949  | 4121.60| 3.213|
| Handwritten Digits | 14.7    | 12.539  | 2341.20| 3.095|
| Letter          | 35.6      | 22.139  | 10182.80| 0.998|
| Shuttle         | 10.6      | 0.130   | 555.00 | 11.959|
| Appendicitis    | 2.7       | 17.049  | 10.71  | 0.023|
| Cleveland       | 7.8       | 44.473  | 55.79  | 0.023|
| Hepatitis       | 2.7       | 17.915  | 6.20   | 0.014|
| Spectf Heart    | 2.7       | 24.410  | 45.75  | 0.064|
| Breast(Wisconsin)| 2.7     | 9.176   | 42.18  | 2.694|
another sample of validation among the other \((k - 1)\) samples, which were not already used as a validation for the model.

The advantage of this method is that at the end of the process, each sample would be utilized exactly once as a set of validation and \((k - 1)\) as a training set. The means of the \(k\) errors is calculated to estimate the prediction error. In addition to the computed error estimation, the number of the tree nodes built by C4.5 and the time required for the discretization process are registered. For the assessment process, the following steps are undertaken:

1. Set the cut-point threshold and the maximum number of intervals.
2. Divide the data in two sets: the training (9/10) and the test (1/10).
3. Apply the algorithm of discretization on the training set.
4. Use the cut-points obtained in the previous step to discretize the test set.
5. Call the algorithm C4.5 with the training and test sets and register the error rate as well as the number of nodes of the obtained tree.
6. Once the 10 iterations are performed, calculate the means of the registered error rates, running time and nodes numbers.

5.3. Experimental results of LR-SDiscr

LR-SDiscr was executed using the \(\chi^2\) and the entropy measures, respectively under the three categories of datasets shown previously. As numerical results, the threshold, the error rate (Error%), the number of the tree nodes (#Nodes) built by C4.5, the running time in seconds (Time) and the yielded number of intervals (#Intervals) are presented and commented next for each bunch of datasets.

5.3.1. LR-SDiscr using the \(\chi^2\) statistic

Table 2 shows the results obtained for the threshold, the error rate, the number of nodes, the runtime and the number of intervals from the execution of LR-SDiscr using the \(\chi^2\) discretization measure. The threshold and the maximum number of intervals parameters were tuned w.r.t the minimum error rate.

We can make the following observations:

1. The threshold is high for most of the large datasets relatively to the small datasets.
2. However the number of intervals is small, ranging from 5 to 10 for all datasets.
3. The number of nodes of the decision tree and the runtime are correlated, which is foreseeable.
4. The error rate is high for some large datasets such as Yeast and Abalone.

5.3.2. LR-SDiscr using the entropy

Using the entropy, the results of LR-SDiscr are exposed in Table 3. Here the threshold is lower than 1 and the number of intervals varies between 5 and 15. For most datasets, the error rate is greater than that produced by LR-SDiscr using the \(\chi^2\) statistic. On the contrary, the runtime is short and lower than that of the LR-SDiscr using the \(\chi^2\) statistic.
5.3.3. Empirical results for ChiMerge
In order to compare the results calculated by our methods with those of ChiMerge on an objective basis, we implemented ChiMerge under the same conditions as LR-SDiscr. Also the execution of the algorithm was extended for large and medical datasets, as in the original publication, only the small datasets were tested. The results of ChiMerge are exhibited in Table 4. They are different from the previous ones and denote another specificity. The threshold is greater than 1 and is significant for large datasets. The number of nodes and the time seem to be correlated as for the previous remarks.

5.3.4. On the recent state-of-the-art efforts
As mentioned previously, research on discretization is abundant and continues to stimulate the interest of the datamining and the machine learning communities. For comparison purposes, we consider the most recent methods namely, ChiD (Bettinger, 2011) and Entropy with Multiple Scanning (Grzymala-Busse & Mroczek, 2016). We call the latter Multiple Scanning for short in the rest of the paper.

5.3.5. ChiD
ChiD Bettinger (2011) is a discretization method that uses ChiMerge to calculate a set of cut-points, given a level of significance. The latter is decreased at each iteration and a new set of cut-points is computed. At each iteration the set of cut-points that maximizes the following criterion:

\[ \text{logworth}(p) = -\log 10(p) \]

is retained.

This method resembles much \( \chi^2 \), except that the stop criterion in ChiD is different and no parameter needs to be set a priori, which is great an advantage over \( \chi^2 \). However it requires to change the level of significance each time, and to execute the ordinary ChiMerge with the latter, which generates a complexity of \( O(n^2) \).

SAS Enterprise MinerTM 4.3 was used as a test platform for all the experiments carried out. Each dataset was splitted into 3 partitions, learning (60%), validation (20%) and testing (20%). The construction of the three sets was done in a random manner. To determine the accuracy, the average of the results of three runs was calculated.

5.3.6. Multiple scanning method
The Multiple Scanning method (Grzymala-Busse & Mroczek, 2016) presents two supervised bottom-up discretization techniques, based on the entropy statistic. The first technique is a recursive discretization method whose principle is as follows. The best cut-point with the smallest conditional entropy is determined. This cut-point divides the dataset into two subsets \( S_1 \) and \( S_2 \). The same strategy is applied recursively for \( S_1 \) and \( S_2 \) separately. The algorithm stops when the consistency level is reached.

The second technique requires an additional parameter \( t \), which is the total number of scans. The best cut-point using conditional entropy is determined and the process is iterated \( t \) times. If the number of scans \( t \) is reached and the data set still needs to be discretized, the first strategy is launched to continue the discretization. The algorithm stops when the consistency level reaches the value 1.
5.3.7. Comparing LR-SDiscr with the state-of-the-art algorithms

Table 5 shows the error rates for all the implemented methods: the C4.5 without discretization, LR-SDiscr under the two options, Chimerge, ChiD and Multiple scanning. The values in bold represent the best error rate among all the discretization methods. For some large datasets, ChiD and Multiple scanning were unable to calculate the error because of a memory overflow. This is expressed as Not Enough Memory (NEM).

The second column of the table exhibits the error rate calculated by the classifier without discretization. We notice that overall the results are of the same order as those of LR-SDiscr. If we go to the details, LR-SDiscr counts 10 best errors over 23 (10/23), 6 comparable results over 23 (6/23) and then in total 16 over 23 best results. The classifier on continuous data produces only 7/23 best outcomes. We conclude that the discretization treatment enhanced the performance of the classifier. Figure 4 illustrates these findings. Most of the time, The results for continuous values overlap those of LR-SDiscr using $\chi^2$ and the results of LR-SDiscr with both variants are the best for almost all the medical datasets.

If we compare now the results of the discretization methods between themselves, we remark as Table 7 shows, LR-SDiscr using $\chi^2$ yields the best outcomes because it generates an error rate, which is the lowest for 11 over 23 datasets, whereas ChiMerge produces the lowest error for just 1 case over 23, ChiD 2 cases over 23 and Multiple scanning 6 cases over 23. We notice also that the latter is the most effective for small datasets but at the same time it is not capable to run on large datasets.

Figure 5 depicts an overall view of the error rates for the five discretization methods, where we can observe the superiority of LR-SDiscr $\chi^2$ on the others.

Table 6 exposes the runtimes for LR-SDiscr, Chimerge, ChiD and Multiple scanning. As observed previously, LR-SDiscr has an instant response time and outperforms the

Table 5. Comparing error rates for Continuous data, LR-SDiscr, Chimerge, ChiD and Multiple Scanning.

| Benchmark         | Continuous data | LR-SDiscr $\chi^2$ | LR-SDiscr Entropy | Chimerge | ChiD | Multiple Scanning |
|-------------------|-----------------|--------------------|-------------------|----------|-----|-------------------|
| Australian        | 15.28           | 13.855             | 14.039            | 15.053   | 14.649 | **13.727**        |
| Breast            | 4.72            | 4.373              | **3.935**         | 5.277    | 5.083 | 5.150             |
| Heart             | 22.16           | 21.497             | 18.722            | 20.979   | 25.234 | 18.611            |
| Vehicle           | 26.87           | 32.081             | 32.129            | 32.964   | 33.442 | **29.82**         |
| Iris              | 4.34            | 5.347              | 5.342             | 5.143    | 5.274 | 4.678             |
| Wine              | 6.22            | **7.101**          | 30.846            | 12.655   | 12.301 | 7.614             |
| Pima              | 26.22           | **26.700**         | 35.014            | 28.427   | 34.872 | 26.796            |
| Bupa              | 33.13           | 37.952             | 37.033            | 43.312   | 42.609 | **34.892**        |
| Thyroid           | 8.00            | 7.790              | 21.537            | 6.881    | **6.352** | 7.244             |
| Ionos             | 9.14            | **8.920**          | 10.226            | 11.683   | 17.558 | 10.137            |
| German            | 29.50           | 28.389             | **27.420**        | 29.410   | 30.000 | 28.220            |
| Yeast             | 44.54           | 59.896             | 64.458            | 68.809   | 68.380 | 66.309            |
| Hypothyroid       | 0.42            | **0.844**          | 7.719             | 1.300    | 3.010 | NEM               |
| Abalone           | 78.84           | 76.097             | 81.275            | **74.974** | NEM | 76.945            |
| Satimage          | 13.72           | **14.850**         | 63.239            | 19.949   | NEM | NEM               |
| Handwritten Digits| 16.13           | **6.761**          | 29.490            | 12.539   | NEM | NEM               |
| Letter            | 12.02           | **20.253**         | 20.969            | 22.139   | NEM | NEM               |
| Shuttle           | 0.02            | **0.063**          | 4.510             | 0.130    | NEM | NEM               |
| Appendicitis      | 14.15           | **15.100**         | 19.875            | 17.049   | 19.576 | 15.452            |
| Cleveland         | 51.51           | **44.070**         | 44.743            | 44.473   | 48.849 | 44.809            |
| Hepatitis         | 16.12           | 18.500             | 16.397            | 17.915   | **15.716** | 18.765            |
| SPECTF Heart      | 25.09           | 21.146             | **20.531**        | 24.410   | 20.700 | 22.858            |
| Breast (Wisconsin)| 7.20            | 7.318              | 35.107            | 9.176    | 12.400 | **5.120**         |
recent state-of-the-art methods in terms of efficiency. LR-SDiscr using the entropy has the best execution time in 21 datasets out of 23. Again Table 7 exhibits this outcome. In addition, LR-SDiscr $\chi^2$ is right behind with a better time than those of the other techniques and the gap between the two variants of LR-SDiscr is very slight. It is clear that LR-SDiscr under its two versions is the fastest among all the methods including the recent methods found in the literature. These results prove the scalability of the proposed method as all the runtimes are less than 0.1 second for the majority of the instances. These outcomes are confirmed by Figure 6, which exhibits a graphical view of the runtime for the five algorithms. The main observation is that LR-SDiscr behaves the same for both variants and is faster than the others especially for the large datasets. We see also that ChiD and Multiple Scanning are out of the competition and hence not suitable for the large datasets. As an overall conclusion on LR-SDiscr, the extensive experiments we undertook reveal that it is very effective when using the $\chi^2$
measure and very efficient when using the entropy measure to discretize data. Both variants of LR-SDiscr show their superiority on all the other methods in terms of accuracy, runtime and scalability.

6. Conclusions

Traditional supervised discretization methods belong either to the top-down or bottom-up discretization schema. We begin our contributions by calculating the time complexity for both cases and for the first time these complexities were established as a $O(n \times \text{maxIntervals})$ complexity for the top-down methods and a $O(n^2)$ for the bottom-up techniques, where $n$ is the number of instances and maxIntervals is the maximum number of intervals inferred from discretization. Of course, these orders of magnitude are provided without considering the sorting step which exists in all these methods.
Then, with the aim at improving the efficiency of these methods, we propose a novel framework for discretization that presents several strengths, among them we evoke the following:

(1) For the first time, a discretization approach based on division and merging operations simultaneously is devised.

(2) This framework allows processing discretization in only one pass and hence reduces the time complexity to \(O(n)\) still without considering the sorting step.

(3) The discretization algorithm can be automatically implemented using a lexical generator.

(4) The approach allows to modify the discretization statistic with ease, thanks to the lexical generator structure.

(5) The approach is capable to treat large datasets and hence has the ability to reach scalability.

As theoretical development, besides the time complexity calculation, we set and proved a theorem that validates the framework main principle and its feasibility.

On the other hand, extensive experiments were undertaken on various datasets to assess the effectiveness and efficiency of our method. Several measures have been

Table 6. Comparing runtime for LR-SDiscr, Chimerge, ChiD and Multiple Scanning.

| Benchmark   | LR-SDiscr \(x^2\) | LR-SDiscr Entropy | Chimerge | ChiD | Multiple Scanning |
|-------------|-------------------|-------------------|----------|------|-------------------|
| Australian  | 0.016             | 0.01              | 0.189    | 12.38| 0.173             |
| Breast      | **0.005**         | 0.006             | 0.010    | 0.610 | 0.117             |
| Heart       | **0.005**         | 0.044             | 0.022    | 1.057 | 0.106             |
| Vehicle     | 0.042             | **0.022**         | 0.171    | 10.724| 1.428             |
| Iris        | 0.0068            | **0.003**         | 0.008    | 0.297 | 0.010             |
| Wine        | 0.031             | **0.010**         | 0.068    | 3.651 | 0.073             |
| Pima        | 0.046             | **0.012**         | 0.137    | 8.825 | 0.499             |
| Bupa        | 0.008             | **0.006**         | 0.022    | 1.062 | 0.041             |
| Thyroid     | 0.011             | **0.004**         | 0.023    | 1.040 | 0.026             |
| Ionos       | 0.056             | **0.027**         | 0.792    | 51.903| 1.341             |
| German      | 0.017             | **0.005**         | 0.157    | 10.518| 0.234             |
| Yeast       | 0.058             | **0.027**         | 0.152    | 8.561 | 0.637             |
| Hypothyroid | 0.381             | **0.028**         | 1.050    | 57.413| NEM               |
| Abalone     | 0.906             | **0.068**         | 7.341    | NEM   | 6.354             |
| Satimage    | 1.671             | **0.441**         | 3.213    | NEM   | NEM               |
| Handwritten Digits | 2.363 | **0.274**         | 3.095    | NEM   | NEM               |
| Letter      | 0.487             | **0.608**         | 1.028    | NEM   | NEM               |
| Shuttle     | 3.384             | **0.558**         | 11.959   | NEM   | NEM               |
| Appendicitis| 0.011             | **0.002**         | 0.023    | 1.051 | 0.019             |
| Cleveland   | 0.008             | **0.004**         | 0.023    | 1.063 | 0.129             |
| Hepatitis   | 0.004             | **0.002**         | 0.014    | 0.544 | 0.026             |
| SPECTF Heart| 0.037             | **0.006**         | 0.064    | 3.730 | 0.559             |
| Breast (Wisconsin) | 1.787 | **1.471**         | 2.694    | 167.275| 3.549             |

Table 7. Comparison of tested supervised discretization methods.

| Discretization method | #best Error | #best Time |
|-----------------------|-------------|------------|
| LR-SDiscr Chi square  | 11/23       | 2/23       |
| LR-SDiscr Entropy     | 3/23        | **21/23**  |
| Chimerge              |             |            |
| ChiD                  | 1/23        | 0/23       |
| Multiple Scanning     | 6/23        | 0/23       |
exploited and empirical results reveal that LR-SDiscr has the highest accuracy and the shortest runtime. According to all these findings, it is clear that LR-SDiscr outperforms the classical algorithms and those of the recent literature.

For the near future, we intend to design an unsupervised discretization method based on the LR(1) Discretization framework.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Note on contributors**

*Habiba Drias* is a professor of computer science at USTHB, Algiers, Algeria and director of the laboratory of research in artificial intelligence LRIA/USTHB. Her main interests are computational
intelligence, swarm intelligence, multi-agent systems, datamining, information retrieval and the satisfiability problem. She has published 3 books and more than 200 research papers on these topics.

Hadjer Moulai earned a master degree in artificial intelligence from the computer science department of USTHB in June 2016. She is currently a Ph.D student at the laboratory of research in artificial intelligence LRIA. Her research areas include computational intelligence and datamining.

NourElHouda Rehkab earned a master degree in artificial intelligence from the computer science department of USTHB in June 2016. She is currently a Ph.D student at the laboratory of research in artificial intelligence LRIA. Her research areas include swarm intelligence and the satisfiability problem.

ORCID

Habiba Drias  http://orcid.org/0000-0001-7287-5170
Hadjer Moulai  http://orcid.org/0000-0002-9910-2386

References

Bettinger, R. (2011). ChiD. A $\chi$-based discretization algorithm. Modern analytics. In Proceedings of WUSS (pp. 1–9). San Francisco, CA.

Chan, C. C., Batur, C., & Srinivasan, A. (1991). Determination of quantization intervals in rule based model for dynamic. In Proceedings of the IEEE conference on systems, man, and cybernetics (pp. 1719–1723). Charlottesville, Virginia.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In Armand Prieditis and Stuart J. Russell (Eds.), Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995. Morgan Kaufmann 1995, ISBN 1-55860-377-8.

Drias, H., Moulai, H., & Rehkab, N. (2018). LR-SDiscr: An efficient algorithm for supervised discretization. In N. Nguyen, D. Hoang, T. P. Hong, H. Pham, B. Trawiski (Eds.), Intelligent information and database systems, ACIIDS 2018 (Vol. 10751, pp. 266–275). Lecture notes in computer science. Cham: Springer.

Drias, H., Rehkab, N., & Moulai, H. (2017). ULR-Discr: A new unsupervised approach for discretization. In A. Ghosh, R. Pal, R. Prasath (Eds.), Mining intelligence and knowledge exploration MIKE 2017 (Vol. 10682, pp. 381–390). Lecture Notes in Computer Science. Cham: Springer.

Fayyad, U., & Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In Proceedings of the thirteenth international joint conference on artificial intelligence (pp. 1022–1027). San Mateo, CA.

Grzymala-Busse, J. W. (2013). Discretization based on entropy and multiple scanning, ISSN 1099–4300.

Grzymala-Busse, J. W, & Mroczek, T. (2016). A comparison of four approaches to discretization based on entropy. Entropy, 18(3), 69.

Ho, K. M., & Scott, P. D. (1993). Zeta: A global method for discretization of continuous variables. In Proceedings of KDD97: 3rd International conference of knowledge discovery and data mining (pp. 191–194). Newport Beach, CA.

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

Jishan, S. T., Raisul, R., Rashu, I., Haque, N., & Rahman, R. M. (2015). Improving accuracy of students’ final grade prediction model using optimal equal width binning and synthetic minority oversampling technique. Decision Analytics, 2, 1.

Kerber, R. (1992). ChiMerge: Discretization of numeric attributes. In Proceedings of AAAI (pp. 123–128). Newport Beach, CA.

Knuth, D. (1997). The art of computer programming, Volume 1: Fundamental Algorithms (3rd ed.). Massachusetts: Addison-Wesley.
Kurgan, L., & Cios, K. J. (2004). CAIM discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering, 16*(2), 145–153.

Lee, C. I., Tsai, C. J., Yang, Y. R., & Yang, W. P. (2007). A top-down and greedy method for discretization of continuous attributes. In *Proceedings of fourth international conference on fuzzy systems and knowledge discovery* (pp. 472–476). Haikou, China.

Lesk, M. E. (1975). *Lex: A lexical analyzer generator* (Report No. 39). Murray Hill, New Jersey: Bell Laboratories.

Lesk, M. E., & Schmidt, E. (2016). Lex: A lexical analyzer generator. Retrieved December 12 2016.

Liu, H. (2002). Discretization: An enabling technique. *Data Mining and Knowledge Discovery, 6*, 393–423.

Liu, H., & Setiono, R. (1997). Feature selection and discretization. *IEEE Transactions on Knowledge and Data Engineering, 9*, 1–4.

Madhu, G., Rajinikanth, T. V., & Govardhan, A. (2014). Improve the classifier accuracy for continuous attributes in biomedical datasets using a new discretization method. In *Proceedings of international conference on information technology and quantitative management* (pp. 671–679). Haikou, China.

Mantaras, R. L. (1991). A distance based attribute selection measure for decision tree induction. *Machine Learning, 6*, 81–92.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Shang, L., Yu, S. Y., Jia, X. Y., & Ji, Y. S. (2009). Selection and optimization of cut-points for numeric attribute values. *Computers and Mathematics with Applications, 57*, 1018–1023.

Shannon, C., & Weaver, W. (1949). *The mathematical theory of information*. Urbana: University of Illinois Press.

Su, C. T., & Hsu, J. H (2005). An extended Chi2 algorithm for discretization of real value attributes. *IEEE Transactions on Knowledge and Data Engineering, 17*(3), 437–441.

Thornton, C. J. (1992). *Techniques of computational learning: An introduction*. Chapman and Hall Computing Series. London: Chapman and Hall.

Tsai, C. J., Lee, C. I., & Wei-Pang, Y. (2008). A discretization algorithm based on class-attribute contingency coefficient. *Information Sciences, 178*, 714–731.

UCI. (2008). UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science. Retrieved from [http://www.ics.uci.edu/mlearn/MLRepository.html](http://www.ics.uci.edu/mlearn/MLRepository.html)