Discretization oriented to Decision Rules Generation

R. Giráldez[*] J.S. Aguilar–Ruiz J.C. Riquelme F.J. Ferrer–Troyano
D.S. Rodríguez–Baena
Department of Computer Science, University of Seville
Avenida Reina Mercedes s/n, 41012 Sevilla, Spain
e-mail:{giraldez, aguilar, riquelme, ferrer, drodriguez}@lsi.us.es

Abstract. Many of the supervised learning algorithms only work with spaces of discrete attributes. Some of the methods proposed in the bibliography focus on the discretization towards the generation of decision rules. This work provides a new discretization algorithm called USD (Unparametrized Supervised Discretization), which transforms the infinite space of the values of the continuous attributes in a finite group of intervals with the purpose of using these intervals in the generation of decision rules, in such a way that these rules do not loose accuracy or goodness. Stands out the fact that, contrary to other methods, USD doesn't need parameterization.

Keywords: Supervised Learning, Discretization

1 Introduction

In the bibliography, there are a great amount of studies which are focused on the development of methods of obtaining decision rules within the context of supervised learning. In general, the rules can be judged by two criteria: their classification accuracy and their complexity. The relationship between both criteria has been aim of some studies [1, 7]. Other important aspect in the area we are studying is to decrease the cardinality of the continuous attributes of a database, especially in data mining processes in which many learning algorithms only work with finite space of discrete attributes [3]. In order to handle continuous attributes with infinite range of values, the discretization of such attributes can be recommendable. This problem also has been studied widely in the literature [2, 3, 4, 5, 6, 7, 8, 9, 10].

This work is focused on the study of the discretization within the supervised learning field, presenting a novel discretization method of continuous attributes called USD (Unparametrized Supervised Discretization). This method divides the range of values of each attribute in continuous and consecutive intervals, so that each interval can be handled like a discrete value. Thus, USD decreases the cardinality of the continuous attributes, obtaining a set of intervals that conserve the information of the original database in the sense that the method tries to obtain the maximum goodness of the intervals. Among aforementioned works we stands out the algorithm proposed by Robert C. Holte in 1993 and that he called 1R (1-Rules)[7], since similarly to 1R method, USD is focused to the generation of decision rules,

[*]This research was supported by the Spanish Research Agency CICYT under grant TIC2001-1143-C03-02.

therefore the maximization of goodness is a fundamental aspect of the method. Because of both 1R and USD have a similar purpose, the result of this work will be compared with the results obtained by 1R.

Other considerable aspect is that most discretization methods need any user parameter in order to carry out their task. For instance, before running 1R, user has to establish a parameter that Holte names SMALL [7]. This parameter is fundamental for carrying out the discretization. In this sense, USD does not need parametrization. This characteristic means a advantage opposite to other methods with similar aim.

2 Definitions

Before stating the description of USD algorithm is necessary to establish several definitions in order to clarify the notation used in this paper. We name *pure value*, for any attribute, when such value has the same class for every occurrences in the different examples of the dataset. On the contrary, it is called *impure value*. We define *cutpoints* as the values that delimitate the intervals calculated during the discretization process. Every cutpount c_i is calculated as the half-sum between the greatest value of the attribute a_j for the examples contained in the interval I_i and the least value of the attribute a_j for the examples contained in the interval I_{i+1}. We call *pure interval*, for any attribute, when every examples that it contains are labelled with the same class. On the contrary, we denominate it as *impure interval*. The *majority class* of an interval is the class with more occurrences in such interval. Finally, the *goodness* of an interval is defined as the relationship between the goals and the errors of this interval. The expression that defines the goodness can vary depending on the penalty per error that we want to consider. A possible expression of the goodness of an interval I_i is shown by Equation 1, where the penalty per error is high, since the number of errors is in the denominator.

$$Goodness(I_i) = \frac{goals(I_i)}{1 + errors(I_i)} \tag{1}$$

3 Algorithm

Our aim is to divide the continuous attributes in a finite number of intervals with maximum goodness, so that the average-goodness of the final set of intervals will be the highest. The user does not need to give any parameter to the algorithm in order to complete the process, since the latter is carried out without any user parameter or additional information. The way of calculating the intervals makes the algorithm to be determinist.

Figure 1 shows the USD algorithm. The main process is divided in two different parts. The first part (line 2) calculates the initial intervals that will be refined in the second part later (lines 3–18), depending on the goodnesses obtained after carrying out the possible actions: to join two intervals or to leave them independent.

3.1 Calculating the initial intervals

The method that calculates the initial intervals (Figure 1, line 2) could be a simple discretization method in itself. This process maximizes the purity of the intervals in order to obtain the best goodness, regardless of the examples it contains. This fact makes the number of intervals to be relatively high, although it does not suppose a disadvantage, because the refinement

```
1. Procedure USD($DB$)
2.   $InitializeCutpoints(DB)$
3.   for every continuous attribute of DB
4.     for every interval $I_i$ less the last one
5.       if $UnionCondition(I_i, I_{i+1})$=TRUE
6.         $MarkPossibleUnion(I_i, I_{i+1})$ and its goodness
7.       end if
8.     end for
9.     while there are possible Unions
10.      i = Join possible union with maximum goodness
11.      if $UnionCondition(I_i, I_{i+1})$=TRUE
12.        $MarkPossibleUnion(I_i, I_{i+1})$ and its goodness
13.      end if
14.      if $UnionCondition(I_{i-1}, I_i)$=TRUE
15.        $MarkPossibleUnion(I_{i-1}, I_i)$ and its goodness
16.      end if
17.    end while
18.  end for
19. end USD.

20. UnionCondition($I_i$, $I_{i+1}$) = [($I_i$ has the same majority class than $I_{i+1}$)OR (there is a tie in $I_i$ or $I_{i+1}$)] AND
21. [the goodness of the union between $I_i$ and $I_{i+1}$ is greater or equal than the average of goodness of $I_i$ and goodness of $I_{i+1}$]
```

Figure 1: Algorithm USD.

Table 1: Example of database.

N	a_k	$frec(C_A)$	$frec(C_B)$	Majority Class
1	1.0	5	0	A
2	1.2	4	0	A
3	1.4	0	3	B
4	1.6	0	4	B
5	1.8	6	0	A
6	2.0	5	1	A
7	2.2	5	2	A
8	2.4	0	4	B
9	2.6	1	6	B
10	3.0	1	5	B
11	3.2	0	4	B
12	3.4	6	2	A
13	3.6	1	3	B
14	3.8	8	1	A
15	4.0	6	0	A
16	4.2	2	7	B
17	4.4	8	0	A

Table 2: Coloration of the values in an interval.

Pair	Cl_i	Cl_{i+1}	Pure(v_i)	Pure(v_{i+1})	Action
1,2	A	A	Y	Y	join
2,3	A	B	Y	Y	cut
3,4	B	B	Y	Y	join
4,5	B	A	Y	Y	cut
5,6	A	A	Y	N	cut
6,7	A	A	N	N	join
7,8	A	B	N	Y	cut
8,9	B	B	Y	N	cut
9,10	B	B	N	N	join
10,11	B	B	N	Y	cut
11,12	B	A	Y	N	cut
12,13	A	B	N	N	cut
13,14	B	A	N	N	cut
14,15	A	A	N	Y	cut
15,16	A	B	Y	N	cut
16,17	B	A	N	Y	cut

process will reduce considerably this number later. In order to make the understanding easier of the calculation process of the initial intervals, we expound a simple example [1].

Let suppose that we have the data represented in Table 1, where we point out the values of a particular continuous attribute (a_k), its frequency and the majority class. Notice that the dataset has been previously sorted by value of the attribute a_k. Table 1 collects every situations that can be found in a database, specifically this database has 100 examples and 2 classes. The situations correspond to every combination between two consecutive values that have same and different class and, in turn, the value is pure and impure. From Table 1 we will calculate the cutspoints, treating the values in pairs in a consecutive way. Thus, Table 2 shows the actions to carry out (to fix a cutpoint or not) in every situation. The first column (Pair) gives the examples that are being considered; the two next columns (Cl) show their majority class; the next two (Pure) point out if the value is pure or not; and the last one, specifies the action to carry out with the pair of examples. Using a Karnaugh map, we obtain the condition shown by Equation 2, that shows what conditions must be fulfilled to fix a cutpoint.

$$Cl_i \neq Cl_{i+1} \textbf{ OR } Pure(v_i) \neq Pure(v_{i+1}) \tag{2}$$

3.2 Refining the intervals

Once the initial intervals have been obtained according to the conditions that Equation 2 presents, USD processes these intervals in order to reduce their cardinality without losing the global goodness. In Figure 1, the refinement process is carried out in lines 3–18. Basically, the refinement consists on looking over the intervals for every attribute and to evaluate, for every pair of consecutive intervals, if it is possible to join them depending on the condition given by lines 20–21 in Figure 1. In this condition, the first term of the boolean conjunction avoids two intervals with different majority class to be joined, because this union could not be advantageous for the global goodness. The second term of the boolean conjunction compares the goodness of the union with the half-sum of the goodness of the intervals that take part in the pair. This half-sum represents the average goodness that we would obtain if the join of the two intervals does not take place. If a pair of intervals fulfill the union-condition, we mark the union as possible-union, also storing the goodness of such union (Figure 1, line 6). Once the possible-unions for the attribute under consideration have been marked, the while-loop is run in the next step (line 9), joining in every iteration only those intervals whose goodness of possible-union is maximum (line 10). Such union produces a new set of intervals where the new possible-unions and goodness of the two pairs of intervals, in which the new interval takes part, are calculated (figure 2, lines 11–16). The process is carried out while the set of possible-unions is not empty. When in an iteration any union does not take place, the next continuous attribute is treated. When USD finishes, it is obtained a set of intervals with maximal global goodness.

4 Experiments

Some exhaustive tests of the USD method have been carried out, comparing it with the 1R algorithm and using the UCI Repository databases. The test process has consisted in the random generation of decision rules by means of the intervals obtained by USD and 1R, comparing the relationship between goals and errors of both kinds of rules. The way of generating the random rules assures each rule contains at least one example. In general, the examples that are covered by a rule are those whose have values included between both bounds that such rule establishes for each attribute a_i.

In order to generate a rule, one example of dataset is randomly selected. For each attribute, both bounds, upper and lower, are generated in a pseudo-random way using the cutpoints obtained by each algorithm. Thus, we sure that the generated rule contains at least one example, that it is the one chosen at random. Following this rules generation process, 100.000 rules have been generated at random for each algorithm and for each database. The purpose of these tests is to compare the goodness of the intervals obtained by USD with those obtained by 1R for different values of the SMALL parameter which 1R needs for its running. This goodness is based fundamentally on the percentage of goals of the rules created using the intervals obtained by both method. The more goodness of the obtained intervals, the more accuracy the rules will have. In this sense, an important measure is the percentage of goals for the generated rules by each method.

Table 4 shows the results obtained by applying the test to both methods, USD and 1R, with the values of the SMALL parameter from 1 to 7 (value in brackets in the method column MET). The databases used for carrying out the test are *iris, pima, heart, ionosphere, wine, glass, vehicle and letter,* which are indicated in the table by their two first letters. The two

Table 3: Results.

	IR		PI		HE		IO		WI		CL		VE		LE	
MET	%GO	NC	%GO	NC	%GO	NC	%GO	NC	%GO	NC	%GO	NC	%GO	NC	%GO	NC
USD:	78.2	7.2	76.1	52.6	85.0	11.7	99.9	68.4	99.6	36.0	65.6	54.4	69.8	37.7	25.4	13.1
1R(1):	73.3	6.0	75.6	56.1	82.8	13.3	99.9	68.5	99.4	34.4	65.8	52.2	69.6	38.7	24.4	13.6
1R(2):	73.7	5.7	75.9	45.0	83.1	11.8	99.9	40.1	99.5	23.6	67.4	30.1	70.9	34.3	24.3	13.6
1R(3):	73.9	5.7	75.8	36.4	83.1	10.5	99.9	30.8	99.4	19.4	69.4	20.8	71.3	31.1	24.4	13.6
1R(4):	75.8	5.2	75.9	32.5	83.2	9.1	99.9	25.9	99.1	15.8	69.1	17.4	73.0	28.7	24.1	13.6
1R(5):	75.2	5.2	76.5	28.5	83.0	8.1	99.9	23.3	99.0	13.8	68.7	14.5	72.8	25.6	24.5	13.6
1R(6):	75.1	4.7	76.5	25.7	82.7	7.7	99.9	20.9	98.9	12.5	68.3	12.8	73.5	23.7	23.8	13.6
1R(7):	75.4	4.7	76.2	23.1	82.5	7.1	99.9	19.3	98.5	11.0	67.9	11.3	74.4	22.1	24.4	13.6

columns of values, under each databases, give the percentages goals (%GO) and the average number of cutpoints (NC) respectively for each method.

As it can be observed, for the half of the databases the percentage of goal obtained by USD is greater than the percentage by 1R for any value of SMALL. With regard to the average number of cutpoints obtained in each case, USD obtains a number of cuts appreciably bigger than 1R. However, with USD we do not have to worry about establishing a more favorable value for SMALL and, therefore, we ignore the realization of multiples experiments.

5 Conclusions

As result of our research, it has been obtained an unparametric supervised discretization algorithm (USD) that reduces the cardinality of the continuous attributes of a labelled database. USD divides the search space of the continuous attributes into intervals, trying that these intervals attain the maximum goodness, keeping the number of intervals approximately of the same order than other methods. An important advantage that our proposal has, opposite to other discretization algorithms, is the unparametrization, that is, our method does not need any user parameter. The comparative results show that the USD algorithm behaves in a very satisfactory way.

References

[1] J. S. Aguilar. *Discovering Hierarchical Decision Rules with Evolutionary Algorithms in Supervised Learning*. PhD thesis, University de Seville, 2001.

[2] C. Apte and S. Hong. *Predicting equity returns from securities data*. Ch.22, 542–560. In (Fayyad 1996).

[3] P. Berka and I. Bruha. *Empirical comparison of various discretization procedures*. Technical Report LISP-95-04, Laboratory of Intelligent Systems, Prage, 1995.

[4] J. Catlett. *On changing continuous attributtes into ordered discrete attributes*. Proceedings of European Working Session on Learning. 1991, pp. 164-178, Berlin, Springer-Verlag.

[5] J. Dougherty, R. Kohavi, and M. Sahami. *Supervised and Unsupervised Discretization of Continuous Features*. Proceedings of the Twelfth International Conference on Machine Learning. pág. 194–202, 1995.

[6] U. Fayyad and K.B. Irani. *Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning*. Proc. of 13th International Joint Conference on Artificial Intelligence, pág. 1022-1027. 1993.

[7] R. C. Holte. *Very Simple Classification Rules Perform Well on Most Commonly Used Datasets.*. Machine Learning, 11:63-91, 1993.

[8] R. Kohavi and M. Sahami. *Error-based and entropy-based discretization of continuous features*. Proc. 2nd International Conference on Knowledge Discovery and Data Mining, pág. 114-119, 1996. AAAI Press.

[9] H. Liu and R. Setiono. *Chi2: Feature selection and discretization of numeric attributes*. Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence, 1995.

[10] A. K. C. Wong and D. K. Y. Chiu. *Synthesizing statistical knowledge from incomplete mixed-mode data*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 9, N^o 6, pp. 205-218, 1987.