Proposal and Empirical Comparison of a Parallelizable Distance-Based Discretization Method

Jesús Cerquides[1] **and Ramon López de Màntaras**
Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
08193, Bellaterra, Barcelona, Spain
{cerquide,mantaras}@iiia.csic.es

Abstract

Many classification algorithms are designed to work with datasets that contain only discrete attributes. Discretization is the process of converting the continuous attributes of the dataset into discrete ones in order to apply some classification algorithm. In this paper we first review previous work in discretization, then we propose a new discretization method based on a distance proposed by López de Màntaras and show that it can be easily implemented in parallel, with a high improvement in its complexity. Finally we empirically show that our method has an excellent performance compared with other state-of-the-art methods.

Introduction

Discretization is a process that transforms continuous attributes into discrete ones. Performing this process, we can apply discrete classification methods to datasets containing continuous values.

In this work we introduce a discretization method and show that it is parallelizable and that it achieves a top performance. The work begins introducing the problem and reviewing some work done in discretization. Then we explain our algorithm, and perform its parallelization. Next we give the results of a set of empirical comparisons between the different discretization methods, to end up with a set of conclusions in the final section.

Current discretization methods

This section introduces into more detail the discretization problem and five solutions that have been given from different viewpoints.

[1]Jesús Cerquides research is supported by a doctoral scholarship of the CIRIT (Generalitat de Catalunya).

Discretization methods classification

In (Dougherty, Kohavi, & Sahami 1995) three different axis (Global vs. Local, Supervised vs. Unsupervised and Static vs. Dynamic) are used to make a classification of discretization methods. We will add two more axis:

Direct vs. Incremental
Direct methods divide the range in k intervals simultaneously, needing an additional criterion to determine the value of k. Incremental methods begin with a simple discretization and pass though a improvement process, needing an additional criterion to know when to stop the discretization.

Bottom-Up vs. Top-Down
Incremental methods usually can be divided into Top-Down and Bottom-Up. Top-Down methods begin with an empty discretization and its improvement process is simply to add a new cutpoint to the discretization. Bottom-Up methods begin with a discretization that has all the possible cutpoints and its improvement process consists in merging two intervals (delete a cutpoint).

Some discretization methods

Equal size The simplest discretization method is an unsupervised direct method named *equal size discretization*. It calculates the maximum and the minimum for the attribute that is being discretized and partitions the range observed into k equal sized intervals.

Equal frequency is another unsupervised direct method. It counts the number of values we have from the attribute that we are trying to discretize and partitions it into intervals containing the same number of examples.

ChiMerge is a supervised, incremental, bottom-up method described in (Kerber 1992). ChiMerge uses χ^2 statistic to determine the *independence* of the class from the two adjacent intervals, combining them if it is *independent*, and allowing them to be separate otherwise.

Entropy is a supervised, incremental, top-down method described in (Fayyad & Irani 1992),(Fayyad & Irani 1993). Entropy discretization recursively selects the cutpoints minimizing entropy until a stopping criterion based on the Minimum Description Length criterion ends the recursion.

D-2 is a supervised, incremental, top-down method described in (Catlett 1991). D-2 recursively selects the cutpoints maximizing Quinlan's Gain until a stopping criterion based on a set of heuristic rules ends the recursion.

Distance-Based discretization

Our algorithm, based on Mantaras distance between partitions (López de Màntaras 1991), is global, supervised, static and Top-Down incremental. This means that it is required to have two main components, a cutpoint selection criterion and a stopping criterion. Once the examples have been sorted by the attribute value, the main loop of our implementation of the method is:

```
function MDiscretization(Set S,Attribute A)
Discretization = $\emptyset$
NewCutPoint = SelectNewCutPoint(S,A,Discretization)
While (ImprovesDiscretization(S,A,Discretization,NewCutPoint))
        Discretization = Discretization $\cup$ {NewCutPoint}
        NewCutPoint = SelectNewCutPoint(S,A,Discretization)
return Discretization
```

Our algorithm is iterative, considering the whole set for the selection of each new cutpoint, while previously seen Top-Down incremental methods were recursive divide and conquer algorithms.

Cutpoint selection criterion

In the ID3 algorithm, we estimate the classification power of an attribute by some measure (Gain, Gain Ratio, 1 - Distance,...). We want to generate a set of cutpoints so that the classification power of the resulting discretized attribute is the highest possible. Our idea is to follow a greedy heuristic in this search.

Each discretization can be identified with a set of cutpoints. We denote by P_D the partition induced by a discretization D. We will note $P_{D\cup\{T\}}$ the partition induced in our dataset when the discretization applied to our attribute is the result of adding cutpoint T to the discretization D. In these terms the requirement is to find a cutpoint T_A so that it accomplishes:

$$\forall T, Dist(P_C, P_{D\cup\{T\}}) \geq Dist(P_C, P_{D\cup\{T_A\}}) \qquad (1)$$

Where P_C is the partition generated in the dataset by the class attribute and $Dist$ stands for Mantaras normalized Distance which is defined as:

$$Dist(P_C, P_D) = \frac{I(P_C|P_D) + I(P_D|P_C)}{I(P_C \cap P_D)} \qquad (2)$$

where $I(P_C|P_D)$,$I(P_C \cap P_D)$,$I(P_D)$ are the standard Shannon measures of information. For more details see (López de Màntaras 1991).

Once T_A is found, the next step is checking whether the cutpoint improvement is significant enough to accept it or if otherwise no further cutpoints are considered necessary for the discretization.

The stopping criterion

We needed a heuristic to evaluate improvement. We developed a stopping criterion based on the Minimum Description Length Principle (MDLP).

The development followed to apply MDLP to our problem is parallel to that in (Fayyad & Irani 1993). The problem that needs to be solved is a communication problem. We have to communicate a classifier method, that allows the receiver to determine the class of each example. The sender knows all the attributes of the examples, plus the class, and the receiver knows all the attributes of the examples but not the class. The sender must choose the shortest description for sending a message that allows the receiver to correctly classify each example.

The encoding length of communicating the set of classes based on a p-cutpoint discretization can be decomposed as $Len(Disc) + Len(Classes|Disc)$. If we note N the number of examples of the dataset, k the number of classes, k_i the number of classes in the interval i of the discretization, S_i the set of examples in the same interval and $Ent(S)$ Shannon Entropy for the set S, we have:

$$Len(Disc) = p\ log(N-1) + (p+1)k + \sum_{i=0}^{p} k_i Ent(S_i)$$

$$Len(Classes|Disc) = \sum_{i=0}^{p} |S_i| Ent(S_i)$$

Given two discretizations, one with p and the other with p+1 cutpoints, we will choose that with the minimal length. If it is the one with p cutpoints, then we stop our algorithm and no more cutpoints are added to the discretization, otherwise we consider including another cutpoint.

Computational complexity

The computational complexity of the method is not easily measurable, because the stopping criterion depends on the data in which we are working. The complexity of the sorting step is $O(N\ logN)$. The complexity of the function *SelectNewCutPoint* in our implementation is $O(k\ i\ N)$ where k is the number of classes in the dataset, N the number of examples and i the number of intervals of the discretization in this run. The complexity of *ImprovesDiscretization* is $O(k\ i)$. We will not consider it, because $O(k\ i) \subset O(k\ i\ N)$. If we discretize the attribute with p cutpoints, the total complexity of the method is given by:

$$O(N\ logN + \sum_{i=1}^{p} ikN) = O((logN + p^2k)\ N) \qquad (3)$$

k is constant, and very small with respect to N. To ease the evaluation of the complexity, we can use a heuristic restriction as the one imposed in D-2, and say that discretizations cannot have more than a fixed number of intervals. With this assumptions, complexity is bounded by the sorting step, as for Entropy, D-2 or ChiMerge.

Parallelization of the method

We have found that the complexity of the algorithm, without including the sorting step, is mainly related with the complexity of the function *SelectNewCut-Point*. We will parallelize this function to obtain a high improvement in the performance of the algorithm.

To simplify the explanation we suppose we have assign a processor to each example in the dataset. The parallelized version of the algorithm is as follows:

- Step 1. The sorting step can be parallelized with N processors in time $O(logN)$. From now on we assume the values are sorted by the attribute being discretized.

- Step 2. We have to calculate a contingency table for each processor, in order for the processor to be able to evaluate the Distance between the partition generated by the class and the partition generated by fixing the cutpoint just between its value and the value of the neighbour processor on its left side. This can be done in $O(k\ logN)$ time in two steps, the first one by adding the information of all the processors following a processor binary tree until it arrives to the root, and the second one by descending the processors tree, distributing the information we have previously put together in the first step.

- Step 3. Now we have that each processor has its corresponding contingency table. Each processor evaluates independently the Distance measure for its cutpoint. This is done in time $O(k\ i)$.

- Step 4. We have to calculate the processor with minimal Distance. We use again the binary processor tree, and this gives us a time $O(logN)$.

- Step 5. The root processor evaluates the MDL criterion. If it turns out that the new cutpoint is not good enough, broadcasts it to the other processors, and the algorithm stops here. Otherwise the new cutpoint is annotated by the root processor. The information of where the cutpoint has been fixed, and the contingency table of the processor whose cutpoint has been selected is broadcasted to all the processors. This can be bounded in time by $O(k\ i\ logN)$

- Step 6. Each processor transforms its contingency table considering that a new cutpoint has been fixed. This step is bounded by $O(k\ i)$

- Step 7. We return to step 3.

The time complexity, when having h processors ($h \leq N$) and the attribute discretized with p cutpoints, is bounded by $O(k\ p^2\ \frac{N}{h} log\ h)$, just assigning $\frac{N}{h}$ of the examples to each processor. Concretely, when having N processors, the time is $O(k\ p^2\ logN)$, which is clearly better than the time found for the sequential procedure. A parallel version of the method has been implemented in MPI and its code can be examined in (Cerquides 1997).

Empirical comparison

Comparison design

We will use the accuracy of two classification algorithms to measure the discretization goodness. The two algorithms will be ID3 (Quinlan 1986) (with no pruning) and Naive-Bayes (Langley, Iba, & Thompson 1992). We will run each algorithm in 9 different domains with different characteristics (see Table 1). For each learning algorithm, discretization method and dataset we do 50 runs, each one with 70% of the examples as training set and the remainder 30% as test set. We take the average of the 50 runs as a measure of performance. We also keep the results of the 50 runs to make two statistical significance tests: Rank and Signed Rank. In (Cerquides 1997),(Gibbons 1971) one can find a complete explanation of this tests.

Comparison results

Average accuracies comparison For each dataset and classification algorithm we rank the 6 discretization methods, from the first place (the most accurate) to the sixth one. The results are displayed in the two tables that appear below. The rows are ordered with the best method on the top and the worst on the bottom. In the tables 55555 means the algorithm ranked

Dataset	Attributes	Instances	Classes	Missing
CRX	15	690	2	few
ECHO	7	131	2	some
GLASS	10	214	7	none
HEARTC	13	303	2	several
HEARTH	13	294	2	some
HEP	19	155	2	some
HORSE	27	368	2	30 %
IRIS	4	150	3	none
WINE	13	178	3	none

Table 1: Domains

five times in the position specified by the column under which it appears, 4444 four times, 333 three times and so on.

	First	Second	Third	Fourth	Fifth	Sixth
DISTANCE	55555	4444				
ENTROPY	4444	55555				
SIZE			55555	333	1	
D2			22	4444	22	1
CHIMERGE			22	1	1	55555
FREQUENCY				1	55555	333

Table 2: Average accuracy ID3 results

	First	Second	Third	Fourth	Fifth	Sixth
DISTANCE	55555	22		22		
ENTROPY	1	333	22	1	1	1
FREQUENCY	22		333	1	22	1
SIZE	1	333		22	1	22
D2		22	333	1	22	1
CHIMERGE				22	333	4444

Table 3: Average accuracy Naive-Bayes results

We can extract some conclusions from this two tables:

- Distance seems to be globally the best performer, while ChiMerge seems to be the worst.
- For the ID3 classification algorithm, either Distance or Entropy have always the first place.

Significance test comparison For each dataset and each classification algorithm we have performed the pairwise comparison of accuracies for the six discretization methods. This has given us a partial ordering of the methods for each dataset. The full results are in (Cerquides 1997). We will try to extract the most important conclusions in a few statements:

- Rank and Signed Rank tests differ only in one over twenty of the comparisons. We have decided to analyze only Signed Rank results.
- For ID3, Entropy and Distance are better than the rest in a statistically significant way for all the datasets.
- For the Bayesian classifier, Distance seems to perform better than the rest, but in most of the cases it is not statistically significantly better than Entropy.

Conclusions

We have introduced a new method for discretization of continuous values. We have seen that our new method has as good time complexity as the other existing methods. We have also shown that our method is easily parallelizable, and we have implemented a parallel version of it. This characteristic is specially important for its use in Knowledge Discovery in Databases (KDD), because databases in that area are supposed to have a high number of registers. The time complexity constraints of algorithms used in the KDD area are very strong, and parallelism seems the better way for reducing it.

We have also compared our discretization method, in terms of accuracy, with other five methods, and for two different classification algorithms observing that it has better average accuracy than the best of the methods proposed until now (the Entropy method), but this difference is not statistically significant. We can say then that our method can be a good alternative to Entropy discretization, especially for very large datasets, where a time complexity $O(N\ logN)$ may be unacceptable.

Future work

The datasets we have used for comparing the performance of our method have at most several hundreds of elements. A new study must be done for larger datasets.

References

Catlett, J. 1991. On Changing Continuous Attributes into Ordered Discrete Attributes. In Kodratoff, Y., ed., *Proceedings of the European Working Session on Learning*, 164–178. Springer-Verlag.

Cerquides, J. 1997. Mantaras Distance for Discretization. Proposal and Empirical Comparison of a New Parallelizable Discretization Method. Technical report, IIIA-97-03.

Dougherty, J.; Kohavi, R.; and Sahami, M. 1995. Supervised and Unsupervised Discretization of Continuous Features. In Prieditis, A., and Rusell, S., eds., *Machine Learning: Proceedings of the Twelfth International Conference.*

Fayyad, U. M., and Irani, K. B. 1992. On the Hadling of Continuous-Valued Attributes in Decision Tree Generation. *Machine Learning* 8:87–102.

Fayyad, U. M., and Irani, K. B. 1993. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *13th International Joint Conference of Artificial Intelligence*, 1022–1027.

Gibbons, J. 1971. *Nonparametric statistical inference.* Series in probability and statistics. New York: McGraw-Hill.

Kerber, R. 1992. ChiMerge: Discretization of Numeric Attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 123–128. MIT Press.

Langley, P.; Iba, W.; and Thompson, K. 1992. An Analysis of Bayesian Classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 223–228. AAAI Press and MIT Press.

López de Màntaras, R. 1991. A Distance Based Attribute Selection Measure for Decision Tree Induction. *Machine Learning* 6:81–92.

Quinlan, J. 1986. Induction of decision trees. *Machine Learning* 1:81–106.