Relative Unsupervised Discretization for Regression Problems

Marcus-Christopher Ludl[1] and Gerhard Widmer[1,2]

[1] Austrian Research Institute for Artificial Intelligence, Vienna
[2] Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, Austria

Abstract. The paper describes a new, context-sensitive discretization algorithm that combines aspects of unsupervised (class-blind) and supervised methods. The algorithm is applicable to a wide range of machine learning and data mining problems where continuous attributes need to be discretized. In this paper, we evaluate its utility in a regression-by-classification setting. Preliminary experimental results indicate that the decision trees induced using this discretization strategy are significantly smaller and thus more comprehensible than those learned with standard discretization methods, while losing only minimally in numerical prediction accuracy. This may be a considerable advantage in machine learning and data mining applications where comprehensibility is an issue.

1 Introduction

In the area of classification learning, there has been quite some research on attribute discretization in recent years, both regarding unsupervised (class-blind) and supervised methods – see, e.g., [1,5,6,9,11,13]. Some authors have also produced detailed studies of different discretization criteria used in "on-the-fly" discretization in induction, for instance, in decision tree learning algorithms [2] or in Bayesian classifiers [3]. While discretization is strictly necessary for induction algorithms that cannot handle numeric attributes directly (e.g., decision table algorithms or simple Bayesian classifiers), it has been shown that pre-discretizing continuous attributes — even when used in induction algorithms that can actually handle continuous features — can improve both the classification accuracy and the interpretability of the induced models.

Whereas in unsupervised discretization the attribute in question is discretized with simple, class-blind procedures, supervised discretization also takes class information into account, thereby possibly constructing split points that might be missed by a class-blind algorithm. [1] gives a good overview.

Recently, there have also been some investigations into the use of discretization for a *regression-by-classification* paradigm [12], where regression is converted into a classification problem by abstracting the continuous target attribute into discrete intervals. The work presented here falls into this latter category. We describe a new, context-sensitive discretization algorithm that can be used in both supervised and unsupervised settings. We evaluate the algorithm by using

R. López de Mántaras, E. Plaza (Eds.): ECML 2000, LNAI 1810, pp. 246–254, 2000.

it as the basis for a regression-by-classification system. Preliminary experimental results to be presented in section 4 demonstrate that the decision trees induced using this discretization strategy are significantly smaller than those learned with standard discretization methods, while losing only minimally in prediction accuracy (measured in terms of numeric error). We think this can be a considerable advantage in machine learning and data mining applications where comprehensibility is an issue.

2 Regression via Classification

The regression problem is usually seen as the task of predicting a (more or less) exact numeric target value for previously unseen examples. Thus, the target attribute is not specified by discrete symbols, but by many distinct values from a fixed range. Specialized algorithms have been invented for this task, but the question arises whether algorithms capable of doing classification (i.e. predicting discrete symbols) couldn't possibly be applied here as well. The basic idea would be to discretize the target attribute by splitting its range into some pre-defined number of intervals, and learn to classify examples with a classification learner (like C4.5). Then, instead of just predicting the class label of an unseen example, an exact value from the according interval can be predicted (e.g. the *mean* or the *median*). [12] is one of the first detailed studies in this direction.

Unfortunately it seems that there is a theoretical limit as to what *can* be achieved by this approach to the regression problem (in terms of a lowering of the summed errors): Increasing the number of intervals usually means that the deviations *within* the intervals become smaller, but also that the accuracy of the class predictions decreases. Decreasing the number of intervals on the other hand usually goes along with higher intra-interval deviations.

What shall be shown in this paper is that by using RUDE, a method that is capable of projecting the structure of source attributes onto the continuous target attribute without demanding discreteness (as supervised methods do), we can improve the regression behaviour of the learning step in comparison to unsupervised methods. This improvement can be seen in terms of *absolute deviation* and/or *tree size* (readability).

3 RUDE – Relative Unsupervised Discretization

3.1 Goals

Originally, the algorithm RUDE described in this paper was developed as a strategy for discretizing datasets where a specified target attribute does not exist (like, e.g. when inducing *functional dependencies*) or where the target attribute itself is continuous. RUDE combines aspects of both unsupervised and supervised discretization algorithms. What sets RUDE apart from other supervised discretization algorithms is that it is not constrained to using information from only one discrete (class) attribute when deciding how to split the attribute in

question (see section 3.2). "RUDE" is actually short for **R***elative* **U***nsupervised* **D***iscr***E***tization*, which quite exactly summarizes what this procedure does: The procedure may be called unsupervised in the sense that there is no need to specify one particular class attribute beforehand, nonetheless the split points are not constructed independently of the "other" attributes (hence "relative").

3.2 RUDE – The Top-Level

The basic idea when discretizing a given attribute (the *target*) is to use information about the value distribution of all attributes other than the target (the *source attributes*). Intuitively, a "good" discretization would be one that has split points that correlate strongly with changes in the value distributions of the source attributes. The process that tries to accomplish this (the central component of RUDE) is called *structure projection*. Here is the top level of RUDE:

1. **Preprocessing:** Discretize (via some unsupervised method) all source attributes that are continuous (see section 3.3);
2. **Structure Projection:** Project the structure of each source attribute a onto the target attribute t:
   (a) Filter the dataset by the different values of attribute a.
   (b) For each such filtering perform a **clustering** procedure on values of t (see section 3.4) and gather the split points thereby created.
3. **Postprocessing:** Merge the split points found.

The time complexity of the RUDE algorithm (discretizing one continuous attribute) is $O(nm \log m)$, with n the number of attributes and m the number of examples. A complete discretization of all continuous attributes can therefore be performed in time $O(n^2 m \log m)$. Please refer to [7] for the proof.

3.3 The Main Step: Structure Projection

The intuition behind the concept of structure projection is best illustrated with an example (see Figure 1). Suppose we are to discretize a target attribute t with a range of, say, [0..1], which happens to be uniformly distributed in our case. The values of t in our learning examples have been drawn along the lowest line in Figure 1. The two lines above indicate the same examples when filtered for the values 1 and 2, respectively, of some particular binary source attribute a. Given the distribution of t, any unsupervised discretizer would return a rather arbitrary segmentation of t that would not reflect the (to us) obvious distribution changes in the source attribute a. The idea of structure projection is to find points where the distribution of the values of a changes drastically, and then to map these "edges" onto the target t. The algorithm we have developed for that purpose was in fact inspired by the concept of edge detection in grey-scale image processing (see section 3.4). The basic discretization algorithm can now be stated in Fig. 2.

RUDE successively maps the "structure" of all source attributes onto the sequence of t's values, thereby creating split points only at positions where some

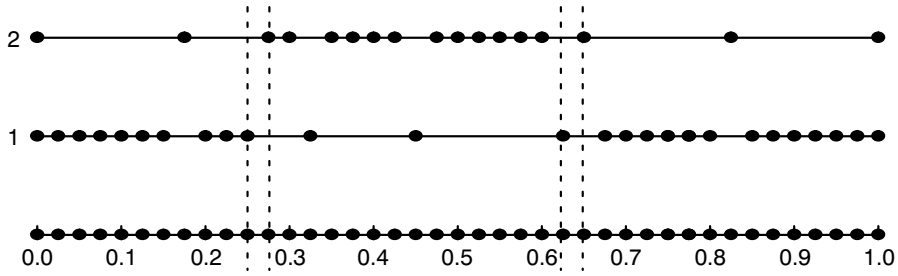

Fig. 1. Structure Projection: An Example

Given:

- a database containing our training examples;
- a set of (possibly continuous) source attributes $a_1, \ldots, a_n$
- information on what attribute should be discretized (the target t);

The algorithm:

1. Sort the database in ascending order according to attribute t.
2. For each attribute a_i with $a_i \neq t$ do the following:
   (a) If continuous, discretize attribute a_i by *equal width*
   (b) For each symbolic value (interval) v thereby created do the following:
      i. Filter the database for value v in attribute a_i.
      ii. Perform *clustering* on the corresponding values of t in the filtered database.
      iii. Gather the split points thereby created in a *split point list* for attribute t.

Fig. 2. RUDE – The basic discretization algorithm

significant distribution changes occur in some of the a_i. For pre-discretizing continuous source attributes in item 2(a) above, we have decided to use *equal-width discretization*, because it not only provides a most efficient (linear) method, but also has some desirable statistical properties (see [7] for details).

The critical component in all this is the clustering algorithm that groups values of the target t into segments that are characterized by more or less common values of some source attribute a_i. Such segments correspond to relatively densely populated areas in the range of t when filtered for some value of a_i (see Figure 1). Thus, an essential property of this algorithm must be that it tightly delimits such dense areas in a given sequence of values.

3.4 A Characterizing Clustering Algorithm

The clustering algorithm we developed for this purpose has its roots in the concept of *edge detection* in grayscale image processing ([8]). The central problem in edge detection is to find boundaries between areas of markedly different degrees of darkness. Typical edge detection algorithms amplify the *contrast* where it exceeds a certain threshold. The analogy to our clustering problem is fairly obvious and has led us to develop an algorithm that basically works by opening

Given:

- A split point list $s_1, s_2, \ldots$.
- A merging parameter (minimal difference s).

The algorithm:

1. Sort the sequence of split points in ascending order.
2. Run through the sequence until you find split points s_i and s_{i+1} with $s_{i+1} - s_i \leq s$.
3. Starting at $i+1$ run through the sequence until you find two split points s_j and s_{j+1} with $s_{j+1} - s_j > s$.
4. Calculate the *median* m of $[s_i, \ldots, s_j]$.
   - If $s_j - s_i \leq s$ **merge** all split points in $[s_i, \ldots, s_j]$ to m.
   - If $s_j - s_i > s$ **triple** the set of split points in $[s_i, \ldots, s_j]$ to $\{s_i, m, s_j\}$.
5. Start at s_{j+1} and go back to step 2.

Fig. 3. Merging the split points

a "window" of a fixed size around each of the values in an ordered sequence and determining whether this value lies at an "edge", i.e. whether one half of the window is "rather empty" and the other is "rather full". The notions of "rather full" and "rather empty" are operationalized by some user-defined parameters. One advantage of the algorithm is that it autonomously determines the appropriate number of clusters/splits, which is in contrast to simpler clustering methods like, e.g., k-means clustering. The details of the algorithm are described in [7].

3.5 Post-processing: Merging the Split Points

Of course, due to the fact that RUDE projects multiple source attributes onto a single target attribute, usually many "similar" split points will be formed during the projections. It is therefore necessary to *merge* the split points in a post-processing phase. Figure 3 shows an algorithm for doing that. At step 3 we have found a subset of split points with successive differences lower than or equal to a certain pre-defined value s. Now, if all these split points lie closer than s (very dense), they are merged down to only one point (the median). If not, the region is characterized by the median and the two outer borders.

4 Experimental Results

Generally, evaluating discretization algorithms is not a straightforward task, as the quality of the discretization *per se* can hardly be measured. Therefore, analogously to [12], we have chosen to apply RUDE to the problem of regression. We measure the *mean average deviation* as well as the *mean tree size* that can be achieved by applying RUDE to a dataset with a continuous target attribute, learning a decision tree via C4.5 [10], and using the median of a predicted interval as the numeric class label for test examples. The results are compared to those achievable by *Equal Width* and *K-Means* discretization with the same classification learner. Table 1 summarizes the databases used for the experiments.

Table 1. The UCI datasets used in the experiments

Dataset	size	attributes continuous	attributes nominal
Abalone	4177	7	1
Auto-mpg	398	4	3
Housing	506	12	1
Machine	209	6	0
Servo	167	0	4

All results were achieved by 10-fold cross-validation. Within each of the 10 runs, a discretization of the target attribute (i.e. a split point list and the according medians) was learned on the training set, these intervals were applied to the test set, and C4.5 was run on these transformed files. We report results in terms of *mean average deviation (MAD)* and *mean tree size*.

For each method, different parameter settings were tried. Table 2 shows selected results for runs with the same number of intervals: The best RUDE run (in terms of MAD) was compared to the values achieved by equal width (EW) and k-means (KM), when set to the same number of intervals.

In table 3, the "best" results achievable by each algorithm are compared. However, simply defining the "best" runs by the lowest MAD value would have resulted in the observation that the deviations achieved by RUDE are nearly always slightly higher than with EW or KM, but the tree sizes are drastically lower! Therefore this figure shows runs with slightly higher deviations than necessary, but much better tree sizes – a good compromise was intended.

Table 2. Selected results from running EW, KM and RUDE on the same datasets, comparing values for the same number of intervals (best RUDE run) against each other. The values in bold print are the best ones (differences are not necessarily significant)

Dataset	EW MAD & Size	KM MAD & Size	RUDE MAD & Size	Intervals
Abalone	1.95 ± 0.04 871.3 ± 35.3	1.94 ± 0.06 1444.9 ± 47.3	**1.93 ± 0.08** **497.8 ± 408.4**	7
Auto-MPG	**2.76 ± 0.43** 153.2 ± 8.9	2.85 ± 0.36 163.6 ± 5.4	3.47 ± 0.36 **129.7 ± 15.3**	8
Housing	**3.08 ± 0.34** 167.6 ± 4.7	3.13 ± 0.30 197.6 ± 13.5	3.32 ± 0.41 **138.0 ± 28.6**	9
Machine	57.91 ± 22.03 **36.4 ± 10.8**	61.91 ± 32.52 129.9 ± 16.2	**45.59 ± 15.14** 86.9 ± 25.0	7
Servo	0.44 ± 0.17 **35.0 ± 0.0**	**0.34 ± 0.13** 60.0 ± 4.0	0.39 ± 0.15 62.0 ± 4.0	6

Table 3. Comparing the "best" runs of EW, KM and RUDE

Dataset	EW		KM		RUDE	
	MAD, Size	# Ints.	MAD, Size	# Ints.	MAD, Size	# Ints.
Abalone	2.31 ± 0.05 133.6 ± 16.2	2	**2.10 ± 0.06** 259.1 ± 27.7	2	2.13 ± 0.09 **32.8 ± 58.64**	4
Auto-MPG	**2.83 ± 0.33** 133.4 ± 6.4	6	3.59 ± 0.32 76.4 ± 13.68	3	3.96 ± 0.34 **51.4 ± 3.32**	3
Housing	4.27 ± 0.36 66.8 ± 9.04	4	**4.00 ± 0.33** 71.0 ± 6.8	3	4.07 ± 0.34 **65.8 ± 6.96**	4
Machine	49.80 ± 13.10 **13.6 ± 9.76**	4	**39.63 ± 7.80** 59.5 ± 14.8	4	51.49 ± 16.25 32.8 ± 8.72	5
Servo	0.44 ± 0.16 **20.0 ± 0.0**	2	0.44 ± 0.16 20.5 ± 0.9	2	0.44 ± 0.16 21.5 ± 2.1	3

As can be seen, the mean average deviation achieved by RUDE is usually slightly higher than with the other two methods (or about equal). The reason for this could be that there is a theoretical limit as to what *can* be achieved by applying classification methods to regression problems; equal width usually achieves low numeric error, because the medians are quite equally distributed, even though the classification accuracy might not be very high (resulting in a higher tree size). With RUDE, on the other hand, tree size usually decreases significantly. This effect is apparently more visible the larger the dataset is.

In summary, RUDE seems to be able to tune the interval boundaries better than the two unsupervised methods compared here. With the same number of intervals, RUDE creates better split points (with regard to lower tree sizes and thus better understandability), even compared to k-means. Comparing the lowest MAD achieved (not caring about the number of intervals), RUDE admittedly loses. Nonetheless, even in these cases, RUDE can improve readability.

5 Discussion

What we have presented is a new method for discretizing continuous attributes by using information about the "structure" of multiple source attributes. Preliminary experimental results show that in a regression-by-classification setting, this algorithm does not improve the summed numerical error of the predictions, but can lower the tree sizes substantially, especially in large databases.

One of the main problems with the current system is that the user-specified parameters still need to be fine-tuned when dealing with a new dataset. Up to now there is no good standard set of parameter settings that works well every time. Also, unfortunately some of the parameters represent absolute values; the problem of defining relative threshold measures (like percentages) is also a current research topic.

RUDE was originally designed with *association rules* and *functional dependencies* in mind. Algorithms for inducing the latter type of knowledge can, by

definition, only work on nominal data, which makes them unsuitable for numerical databases. We are currently testing the efficacy of RUDE in this setting. Devising quantitative measures of success in such applications is a non-trivial problem, which we are currently trying to solve.

Acknowledgments

This research is supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant no. P12645-INF. We would like to thank Johannes Fürnkranz and Bernhard Pfahringer for invaluable suggestions.

References

1. Dougherty J., Kohavi R., Sahami M.: Supervised and Unsupervised Discretization of Continuous Features. In *Proceedings of the 12th International Conference on Machine Learning (ML95)*, Morgan Kaufmann, San Francisco, CA, 1995. 246
2. Fayyad U., Irani K.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *Proc. of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, Morgan Kaufmann, San Francisco, 1993. 246
3. Friedman N., Goldszmidt M., Lee T.J.: Bayesian Network Classification with Continuous Attributes: Getting the Best of Both Discretization and Parametric Fitting. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*, Morgan Kaufmann, San Francisco, CA, pp.179-187, 1998. 246
4. Jun B.H., Kim C.S., Song H.Y., Kim J.: A New Criterion in Selection and Discretization of Attributes for the Generation of Decision Trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(12), 1371- 1375, 1997.
5. Kerber R.: ChiMerge: Discretization of Numeric Attributes. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI'92)*, AAAI Press, Menlo Park, 1992. 246
6. Kohavi R., Sahami M.: Error-Based and Entropy-Based Discretization of Continuous Features. In *KDD-96: Proceedings 2nd International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, pp.114-119, 1996. 246
7. Ludl M.-C.: *Relative Unsupervised Discretisation of Continuous Attributes.* Master's thesis, Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, 2000 (forthcoming). 248, 249, 250
8. Pavlidis T.: *Algorithms for Graphics and Image Processing*, Computer Science Press, Inc., Rockville, Maryland USA, 1982. 249
9. Pfahringer B.: Compression-Based Discretization of Continuous Attributes, in *Proceedings of the* 12^{th} *International Conference on Machine Learning (ML95)*, Morgan Kaufmann, San Francisco, CA, 1995. 246
10. Quinlan, J.R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993. 250
11. Richeldi M., Rossotto M.: Class-Driven Statistical Discretization of Continuous Attributes. In *Machine Learning: ECML-95*, Springer, Berlin, pp.335-338, 1995. 246
12. Torgo, M., Gama, J.: Regression Using Classification Algorithms. *Intelligent Data Analysis* 1, 275-292, 1997. 246, 247, 250

13. Wang K., Goh H.C.: Minimum Splits Based Discretization for Continuous Features. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, Morgan Kaufmann, San Francisco, CA, pp.942-951, 1997. 246