Statistical Preprocessing for Decision Tree Induction

Sreerama K. Murthy *

Abstract

Some apparently simple numeric data sets cause significant problems for existing decision tree induction algorithms, in that no method is able to find a small, accurate tree, even though one exists. One source of this difficulty is the goodness measures used to decide whether a particular node represents a good way to split the data. This paper points out that the commonly-used goodness measures are not equipped to take into account some patterns in numeric attribute spaces, and presents a framework for capturing some such patterns into decision tree induction. As a case study, it is demonstrated empirically that supervised clustering, when used as a preprocessing step, can improve the quality of both univariate and multivariate decision trees.

1 Introduction

Decision trees have been studied widely in the statistics, pattern recognition and machine learning literature. A typical decision tree induction algorithm can be (roughly) sketched as follows. The input is a set S of n instances, each instance having d attributes and belonging to one of c classes. To induce the split at a node, the space of all possible splits of S is searched (exhaustively or using suitable heuristics) to find the *best* split H, which divides the set into two subsets, L and R. (No split is induced if all instances at the current node belong to the same class.) The search is repeated recursively on L and R. Clearly, the criterion used to measure the goodness of a split (known as the *goodness measure* or *impurity measure*) is important in determining the quality of the resulting tree [1, 4, 3].

It is worth noting that several commonly used goodness measures were originally proposed for symbolic domains. The "adoption" of these measures into numeric domains may be less than ideal [2, 8, 9], as numeric domains have their own peculiarities. This paper argues that existing goodness measures are indeed

*Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218. murthy@cs.jhu.edu

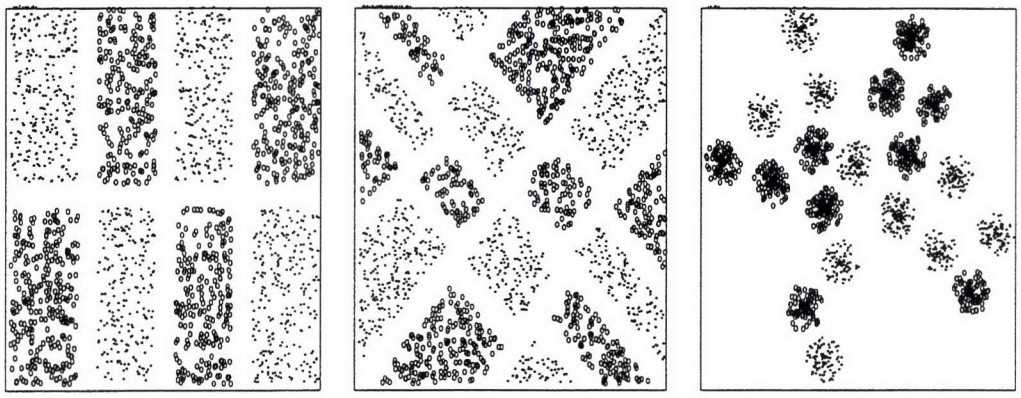

Figure 1: The CB, RCB and RGC data sets

inadequate for some numeric domains, and suggests statistical preprocessing as an effective solution. Section 2 presents some apparently simple artificial domains for which several common goodness measures fail to produce good trees. Section 3 suggests a framework in which data is processed by statistical methods prior to tree induction. Section 4 empirically demonstrates that using clustering as a preprocessing step helps both univariate and multivariate decision tree methods produce better trees. Section 5 concludes the paper by discussing open issues.

2 Three Data Sets

Fig. 1 displays three synthetic, no-noise 2-D data sets, each having 2000 objects belonging to two classes, 0 or $*$. The CB (checker board) data set can be described perfectly by an axis-parallel decision tree with 8 leaves. The RCB (rotated checkerboard) data can be described exactly by an oblique decision tree [5] of 16 leaves. The RGC (randomly generated clusters) data consists of 20 circular clusters, and need not necessarily have a clear decision tree partitioning. But since the generation process produced clusters, trees that separate each cluster into a distinct region are clearly preferable.

Each of these artificial data sets has well-separated, dense, homogeneous regions of the attribute space that *call out* to be separated. Now consider typical decision trees induced on these datasets by existing tree induction methods. Figure 2 displays the decision trees generated for the CB data by C4.5 [7], for the RCB data by OC1 [5] and for the RGC data by C4.5. C4.5 used gain ratio as the goodness measure and OC1 used gini index [1].

This figure shows that some otherwise successful tree induction methods have trouble in these apparently simple domains. The source of this difficulty is that the *only* information available to the goodness measures used is the distribution of object classes across the splits. However, building the ideal tree requires knowing that there are well-defined homogeneous clusters in the attribute space. Existing decision tree methods cannot use any such "structure" information.

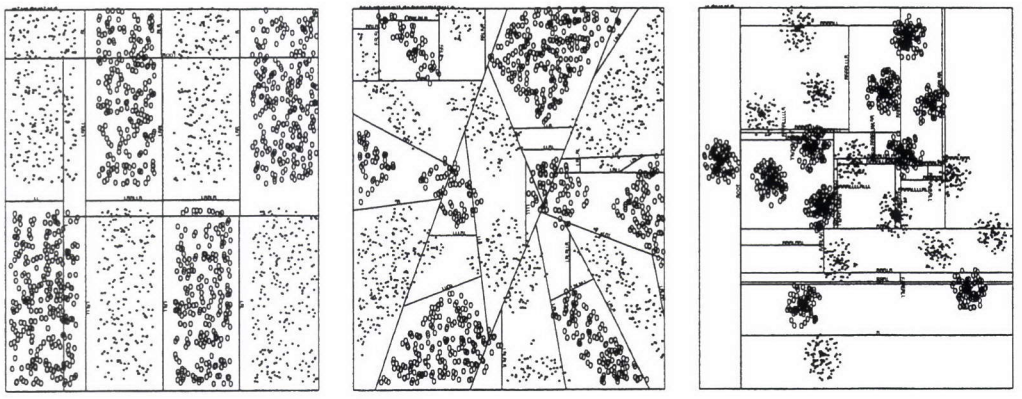

Figure 2: Trees induced on CB, RCB and RGC data by C4.5 and OC1

Some thought should convince the reader that this problem is specific to numeric attribute spaces. In nominal-valued domains, attribute similarity is used for generating the partitions and class similarity follows due to instance space proximity. But in numeric attribute spaces, this process is reversed. Class similarity guides the generation of decision regions and the proximity of instances is a side effect of the divide-and-conquer process.

One solution to this problem is to augment the definition of the goodness measures, to somehow take into account the "structure" of the examples in addition to the class distribution. Van de Merckt [9] used this approach to define a selection criterion that combines the proximity with class entropy. Though this certainly is a step towards using structure, it leaves open some potential problems.

- [9] considers only one kind of structure information, namely clusters. It is not clear how to deal with other important kinds of structure information, for eg. empty regions in attribute space.
- [9] uses unsupervised clustering. This approach fails when each class is clearly multimodal, but the entire set of examples is not. (Consider, for example, variations of fig. 1 datasets with no "space" between clusters.)
- As [9] incorporates structure information into the definition of the goodness measure, this information needs to be calculated once for every split considered. This can be very expensive, especially for multivariate tree methods [5] that consider large numbers of candidate splits.

An alternative way of incorporating structure, that overcomes the the above problems, is presented in the next section.

3 A Framework

Given that structure information is important for constructing good decision trees in some numeric domains, an effective strategy is to find the structure using

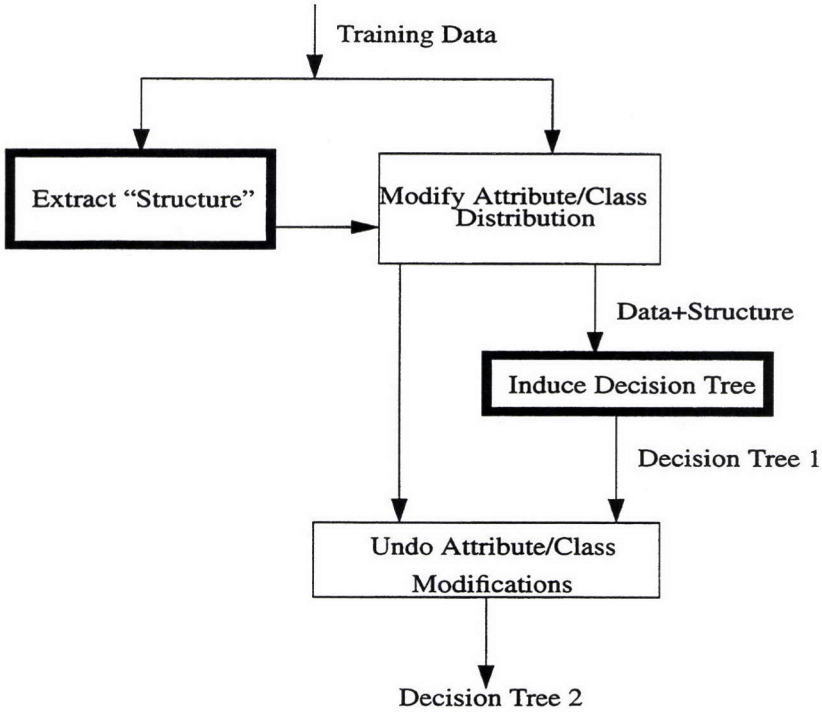

Figure 3: Statistical Preprocessing for Decision Tree Induction

well-known statistical methods and to incorporate it into the information that the tree induction methods *can* use – namely, attributes or classes. This suggests the two-layered architecture in Fig. 3, in which tree induction is preceded by statistical preprocessing.

The training data is first fed into the statistical "structure extraction" module, which outputs information about patterns in the data. Patterns can include clusters, attributes ineffective in classifying objects, large empty regions in attribute space, sudden variations in instance distribution etc. The structure information is incorporated into the training data, by adding/modifying attributes and/or class labels. For example, if we know which subsets of instances form well-separated clusters, we can change the training set by marking each homogeneous cluster as a distinct class, to ensure that the decision tree separates out the clusters. If it is known that there exist large "voids" or empty spaces in the attribute space, we can generate "null" instances in these voids, so that the decision tree is forced not to overgeneralize. Once a decision tree is produced with the modified training data, it may be necessary to postprocess the tree.

An advantage of the above framework is the clear separation between the structure extraction and tree induction stages. Structure extraction methods of varying complexity can be used in conjunction with univariate, multivariate and/or incremental decision tree methods in this model. The complexity of the resulting system is only a sum of the complexities of the preprocessing and tree building stages, as opposed to a product as in [9].

The next section gives a concrete example of the framework in fig. 3.

4 A Concrete Example

This section illustrates that five decision tree methods (three axis-parallel and two oblique) benefit by using Euclidean minimum spanning tree (EMST) clustering as a preprocessing step, on the CB, RCB and RGC domains. Due to space restrictions, the user is referred to [6] for a description of our clustering method. We perform *supervised* clustering – all clusters are constrained to be homogeneous.

I ran two experiments, each using three artificial data sets CB, RCB and RGC (fig. 1). The first experiment induced decision trees in the conventional way, i.e., with no preprocessing. In the second experiment, the data was preprocessed as follows before inducing the trees. Data was clustered using supervised EMST clustering, and each cluster was assigned to a distinct class. Trees induced in the second experiment were postprocessed by restoring the class labels of examples and, in a bottom-up traversal, removing the nodes that were splitting examples from the same category.

The decision tree induction programs used are C4.5 [7], CART [1], and OC1 [5]. Both the univariate and multivariate versions of CART and OC1 were used, unless the correct bias for a data set was known. C4.5 used gain ratio as the goodness measure, and CART and OC1 used the twoing rule. I implemented a version of multivariate CART based on [1], with no backward feature elimination.

Table 1 summarizes the results of both the experiments, giving classification accuracies and tree sizes (#leaves) with and without the use of pre-processing. Each entry lists the mean and standard deviation of ten 5-fold cross-validation experiments. A k-fold cross validation consists of dividing the training set into k disjoint partitions of equal size, and, for each partitions p, building a tree on data outside p, and testing it on p. The classification accuracy on the entire data is reported and the tree size is averaged over the k folds.

The results strongly suggest that preprocessing enables all the five methods to get smaller, more accurate trees on these domains. For the CB data, all three methods found the perfect tree every time, where without clustering they never found the right tree. For the RCB data, only the oblique methods were used. These showed a similarly dramatic improvement with clustering: in many cases they found the minimal tree with 16 nodes. OC1 only had one non-essential leaf node on average. For the RGC data the results were improved but not as dramatic. This is to be expected because clusters are not the only kind of structure present in the RGC data. In order to obtain the perfect trees for this data, we need to use the descriptions of the "empty" regions in the attribute space, so that each cluster is forced to belong to a distinct decision region.

It should be noted that accuracy and tree size alone may not completely capture the quality of a decision tree in a continuous attribute space. For example, the C4.5 tree for CB data displayed in Figures 2 is quite small and accurate, but imposes an incorrect structure on the data. In our experiments,

	OC1		C4.5	CART	
Data	Univariate	Multivariate		Univariate	Multivariate
With No Preprocessing					
CB	99.7±1.08		99.7±0.27	99.9±0.13	
	13.6±3.63		23.6±2.70	16.0±3.50	
RCB		97.1±0.76			97.0±0.46
		25.7±1.75			33.7±2.9
RGC	98.5±0.34	97.4±1.55	98.8±0.21	98.3±1.21	96.6±1.64
	18.1±2.87	17.3±2.65	26±3.54	18±1.22	17.8±2.77
With Clustering as a preprocessing step					
CB	100±0.0		100±0.0	100±0.0	
	8±0.0		8±0.0	8±0.0	
RCB		99.6±0.44			99.5±0.7
		17.2±1.2			21.0±2.53
RGC	99.4±0.2	99.6±0.29	99.6±0.16	99.5±0.21	99.4±0.22
	17.7±1.39	17.2±0.45	17.1±1.89	17.6±1.33	17.3±1.2

Table 1: Effect of Preprocessing on Decision Tree Induction

after clustering information was given by preprocessing, the trees induced were consistently identical to the original concept descriptions.

5 Conclusions

In this paper I presented a problem that many existing decision tree algorithms seem to have with numeric data, and proposed a statistical preprocessing framework as a solution. The effectiveness of this approach was illustrated using Euclidean minimum spanning tree clustering as a preprocessing step for three univariate and two multivariate decision tree induction methods.

Two natural next steps are the following. Firstly, all the data used in these experiments is synthetic. The effectiveness of this approach needs to be evaluated on real-world data. ([9] includes some such evaluations.) Secondly, my experiments only explored incorporating structure information into class distributions. It will be interesting to incorporate structure into attributes instead. By doing this in conjunction with feature selection, it may be possible to identify what kinds of structure information are most useful for specific problems.

It is not yet known what other kinds of structure information besides clusters might be beneficial for tree induction, or to what extent. One direction for future work is to identify and quantify the kinds of useful structure information, and then attempt to take advantage of this structure in building classifiers.

Acknowledgements

I thank Steven Salzberg for numerous helpful suggestions and comments.

References

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth International Group, 1984.

[2] U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8(2):87–102, 1992.

[3] W. Z. Liu and A. P. White. The importance of attribute selection measures in decision tree induction. *Machine Learning*, 15:25–41, 1994.

[4] J. Mingers. An empirical comparison of selection measures for decision tree induction. *Machine Learning*, 3:319–342, 1989.

[5] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–33, August 1994.

[6] Sreerama K. Murthy and Steven Salzberg. Clustering astronomical objects using minimum spanning trees. Technical report, Dept. of Computer Science, Johns Hopkins University, July 1992.

[7] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[8] Thierry Van de Merckt. NFDT: A system that learns flexible concepts based on decision trees for numerical attributes. In *Proceedings of the Ninth International Workshop on Machine Learning*, pages 322–331, 1992.

[9] Thierry Van de Merckt. Decision trees in numerical attribute spaces. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1016–1021, 1993.