Predicting Ordinal Classes in ILP

Gerhard Widmer[1,2], Stefan Kramer[2], Bernhard Pfahringer[2]
and Michael de Groeve[3]

[1] Department of Medical Cybernetics and Artificial Intelligence,
University of Vienna, Freyung 6/2, A-1010 Vienna
[2] Austrian Research Institute for Artificial Intelligence,
Schottengasse 3, A-1010 Vienna, Austria
[3] Department of Computer Science, Katholieke Universiteit Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium

Abstract. This paper is devoted to the problem of learning to predict ordinal (i.e., ordered discrete) classes in an ILP setting. We start with a relational regression algorithm named SRT (Structural Regression Trees) and study various ways of transforming it into a first-order learner for ordinal classification tasks. Combinations of these algorithm variants with several data preprocessing methods are compared on two ILP benchmark data sets to verify the relative strengths and weaknesses of the strategies and to study the trade-off between optimal categorical classification accuracy (hit rate) and minimum distance-based error. Preliminary results indicate that this is a promising avenue towards algorithms that combine aspects of classification and regression in relational learning.

1 Introduction

Learning to predict discrete classes from preclassified examples has long been, and continues to be, a central research topic in Inductive Logic Programming (e.g., [Quinlan, 1990, Pazzani & Kibler, 1992, Muggleton, 1995, Fürnkranz, 1994]). Recently, there has also been increased interest in relational regression, i.e., the task of learning to predict continuous numeric variables from relational data (e.g., [Karalič, 1995,Karalič & Bratko, 1997,Kramer, 1996]). A class of problems between classification and regression is learning to predict *ordinal classes*, that is, discrete classes with a linear ordering. It has not received much attention so far, which seems somewhat surprising, as there are many classification problems in the real world that fall into that category.

Consider, for instance, the *Mesh Design* task, one of the by now classical benchmark problems in the area of ILP. The problem consists in predicting the optimal granularity of a finite element (FE) model of a given physical structure. More precisely, the task is to predict the appropriate number of FEs along a given edge. In the widely used data set first described in [Dolšak & Muggleton, 1992], there are 13 classes (1, ..., 12 and 17 FEs). It seems obvious that in this domain, classification error should be regarded as a gradual phenomenon. Prescribing 1 FE for an edge that should have 12 is a worse mistake than predicting class 4

when the correct class is 5. In other words, the classes are *ordered*, and the classification error should depend on the relative *distance* between predicted and actual class. However, in almost all experiments with this data set described in the literature, classification learners (mostly first-order) were applied and only classification accuracies (as percentages of correct predictions) were reported. A summary of the results reported by various researchers is given in [Dolšak et al., 1998].

Current machine learning research that seems most relevant to the problem of predicting ordinal classes is work on *cost-sensitive learning*. In the domain of propositional learning, some induction algorithms have been proposed that can take into account matrices of misclassification costs (e.g., [Schiffers, 1997,Turney, 1995]. Such cost matrices can be used to express relative distances between classes. In the area of statistics, there are methods directly relevant to our problem (e.g., *Ordinal Logistic Regression* [McCullagh, 1980]); some of these have also been studied in the field of neural networks (e.g., [Mathieson, 1996]). However, our goal is to develop induction algorithms that (1) produce interpretable, symbolic models and (2) are applicable to relational, first-order learning tasks like the mesh design problem.

The purpose of the research described in this paper is to study ways of learning to predict ordinal classes in an ILP setting. We will start with a relational regression algorithm and turn it into an ordinal learner by some simple modifications. This seems a natural strategy because regression algorithms by definition have a notion of relative distance of target values, while classification algorithms usually do not. More precisely, we start with the algorithm SRT (*Structural Regression Trees* [Kramer, 1996] and study several modifications of the basic algorithm that turn it into a distance-sensitive classification learner. Combinations of these algorithm variants with several pre-processing methods are compared on two data sets to verify the relative strengths and weaknesses of the strategies and to study the trade-off between optimal categorical classification accuracy (hit rate) and minimum distance-based error.

2 Relational Regression: The SRT Algorithm

Structural Regression Trees (SRT) [Kramer, 1996] is an algorithm that learns a first-order theory for the prediction of numerical values from examples and relational background knowledge. The algorithm constructs a tree containing a positive literal (an atomic formula) or a conjunction of literals in each node, and assigns a numerical value to each leaf. Since the original publication, SRT has been turned into a full-fledged relational version of CART [Breiman et al., 1984]. After the tree growing phase, the tree is pruned using so-called error-complexity pruning for regression or cost-complexity pruning for classification [Breiman et al., 1984]. These types of pruning are based on a separate "prune set" of examples.

For the construction of a tree, SRT follows the general procedure of top-down decision tree induction algorithms [Quinlan, 1993]. It recursively builds a binary

tree, selecting a positive literal or a conjunction of literals (as defined by user-defined schemata [Silverstein & Pazzani, 1991]) in each node of the tree until a stopping criterion is fulfilled. The algorithm keeps track of the examples in each node and the positive literals or conjunctions of literals in each path leading to the respective nodes. This information can be turned into a clausal theory.

As a regression algorithm, SRT is designed to predict a numeric (real) value in each node and, in particular, in each leaf. In the original version of the algorithm the target value predicted in a node (let us call this the *center value* from now on) is simply the mean of the numeric class values of the instances covered by the node. A natural choice for the *evaluation measure* for rating candidate splits during tree construction is then the *Mean Squared Error (MSE)* of the example values relative to the means in the two new nodes created by the split:

$$MSE = \frac{1}{n_1 + n_2} \sum_{i=1}^{2} \sum_{j=1}^{n_i} (y_{ij} - \bar{y_i})^2 \qquad (1)$$

where n_i is the number of instances covered by branch i, y_{ij} is the value of the dependent variable of training instance e_j in branch i, and $\bar{y_i}$ is the mean of the target values of all training instances in branch i.

In constructing a single tree, the simplest possible stopping criterion is used to decide whether the tree should be further refined: SRT stops extending the tree given some node when no literal(s) can be found that produce(s) two partitions of the training instances in the node with a required minimum cardinality. In the following this parameter will be called *Minimum Coverage*.

SRT has been shown to be competitive with other regression algorithms. Its main advantages are that it offers the full power and flexibility of first-order (Horn clause) logic, provides a rich vocabulary for the user to explicitly represent a suitable language bias (e.g. through the provision of schemata), and produces trees that are interpretable as well as good predictors.

As our goal is to predict discrete ordered classes, SRT cannot be used directly for this task. We will, however, include results with standard SRT in the experimental section to find out how paying attention to ordinal classes influences the mean squared error achievable by a learner.

3 Learning Clausal Theories for Ordinal Classes

In the following, we describe a few simple modifications that turn SRT into a learning algorithm for ordinal classification problems. In section 3.2, we additionally consider some pre-processing methods that might further improve the results.

3.1 Adapting SRT to Ordinal Class Prediction

The most straightforward way of adapting a regression algorithm like SRT to classification tasks is to simply run the algorithm on the given data as if the

ordinal classes (represented by integers) were real values, and then to apply some sort of *post-processing* to the resulting rules or regression tree that translates real-valued predictions into discrete class labels.

An obvious post-processing method is *rounding*. SRT is run on the training data, producing a regular regression tree. The real values predicted in the leaves of the tree are then simply rounded to the nearest of the ordinal classes (not to the nearest integer, as the classes may be discontiguous; after pre-processing, they might indeed be non-integers — see section 3.2 below).

More complex methods for mapping predicted real values to symbolic (ordinal) class labels are conceivable. In fact, we did experiment with an algorithm that greedily searches for a mapping, within a defined class of functions, that minimizes the mean squared error of the resulting (mapped) predictions on the training set. Initial experiments were rather inconclusive; in fact, there were indications of the algorithm overfitting the training data. However, more sophisticated methods might turn out to be useful. This is one of the goals of our future research.

An alternative to post-processing is to modify the way SRT computes the target values in the nodes of the tree *during tree construction*. We can force SRT to always predict integer values (or more generally: a valid class from the given set of ordinal classes) in any node of the tree. The leaf values will thus automatically be valid classes, and no post-processing is necessary.

It is a simple matter to modify SRT so that instead of the *mean* of the class values of instances covered by a node (which will in general not be a valid class value), it chooses one of the class values represented in the examples covered by the node as the center value that is predicted by the node, and relative to which the node evaluation measure (e.g., the mean squared error, see Section 2 above) is computed. Note that in this way, we modify SRT's *evaluation heuristic* and thus its *bias*.

There are many possible ways of choosing a center value; we have implemented three: the *median*, the *rounded mean*, and the *most frequent class*. Let E_i be the set of training examples covered by node N_i during tree construction and C_i the multiset of the class labels of the examples in E_i, with $|E_i| = |C_i| = n$. In the MedianClass strategy, SRT selects the class $\hat{c}_i$ as center value that is the median of the class labels in C_i; in other words, if we assume that the example set E_i is sorted with respect to the class values of the examples, MedianClass chooses the class of the $(n/2)^{th}$ example.[1] In contrast, the MeanClass strategy chooses the class closest to the (real-valued) mean $\overline{c}$ of the class values in C_i. Finally, in the MostFrqClass strategy the center value $\hat{c}_i$ for node N_i is chosen to be the class with the highest frequency in C_i.

Table 1 summarizes the variants of SRT that will be put to the test in Section 4 below.

[1] In this case the *Mean Absolute Deviation (MAD)* is used as distance metric instead of the Mean Squared Error, because the former measure is the one that is known to be minimized by the median.

Table 1. Variants of SRT for learning ordinal classes.

Name	Formula
RealMean + PostProc	$\hat{c_i}$ = mean of the $c_{ij} \in C_i$; real values in leaves of learned tree are rounded to nearest class in C_i
MedianClass	$\hat{c_i}$ = median of class labels in multiset C_i
MeanClass	$\overline{c}$ = mean of the $c_{ij} \in C_i$, $\hat{c_i} = \overline{c}$ rounded to nearest class $c_{ij} \in C_i$
MostFrqClass	pc_i = most frequent class in C_i

3.2 Preprocessing

The results of regression algorithms can often be improved by applying various transformations to the raw input data before learning. The basic idea underlying different data transformations is that numbers may represent fundamentally different types of measurements. [Mosteller & Tukey, 1977] distinguish, among others, the broad classes of *amounts and counts* (which cannot be negative), *ranks* (e.g., 1 = smallest, 2 = next-to-smallest, ...), and *grades* (ordered labels, as in A, B, C, D, E). They suggest the following types of pre-processing transformations: for amounts and counts, translate value v to $tv = \log(v + c)$; for ranks, $tv = \log((v - 1/3)/(N - v + 2/3))$, where N is the maximum rank; and for grades, $tv = (\phi(P) - \phi(p))/(P - p)$, where P is the fraction of observed values that are at least as big as v, p is the fraction of values $> v$, and $\phi(x) = x \log x + (1 - x) \log(1 - x)$. We have tentatively implemented these three pre-processing methods in our experimental system and have tested them in various combinations with the different SRT variants. Table 2 summarizes them in succinct form, in the notational frame of our learning problem.

Note that these transformations do not by themselves contribute to the goal of learning rules for ordinal classes. They are simply tested here as additional enhancements to the methods described above. In fact, pre-processing usually transforms the original ordinal classes into real numbers. That is no problem,

Table 2. Pre-processing types (c = original class value; tc = transformed class value)

Name	Formula
Raw	No pre-processing ($tc = c$)
Counts	$tc = \log(c + 1 - \min(Classes))$
Ranks	$tc = \log((c - 1/3)/(N - c + 2/3))$, where $N = \max(Classes)$
Grades	$tc = (\phi(P) - \phi(p))/(P - p)$, where $\phi(x) = x \log x + (1 - x) \log(1 - x)$, P = fraction of observed class values $\geq c$, p = fraction of observed class values $> c$

however, as the number of distinct values remains unchanged. Thus, the transformed values can still be treated as discrete class values without changing the learning algorithms.

4 Experiments

4.1 Algorithms compared

In the following, we experimentally compare all combinations of SRT variants and preprocessing methods on two ILP benchmark datasets. Two quantities will be measured: the *Root Mean Squared Error (RMSE)* $\sqrt{1/n \sum_{i=1}^{n} (c_i - \hat{c})^2}$ of the predictions on the test set, as a measure of the average distance of the algorithms' predictions from the true class, and the *Classification Accuracy* as the percentage of exact class hits.

As ordinal class prediction is somewhere 'between' classification and regression, we additionally include two 'extreme' algorithms in the experimental comparison. One, called SRTREGRESS, is simply the original SRT as a regression algorithm that acts as if the task were to predict real values; we are interested in finding out how much paying attention to the discreteness of the classes costs in terms of achievable RMSE. (Of course, the percentage of exact class hits achieved by SRTREGRESS will be close to zero.) The other extreme, SRTCLASSIF, is a variant of SRT designed for categorical classification. SRTCLASSIF chooses the most frequent class in a node as center value and uses the *Gini index of diversity* [Breiman et al., 1984] as evaluation measure; it does not pay attention to the distance between classes. And finally, we will also list the *Default* or *Baseline Accuracy* for each algorithm on each data set (i.e., the accuracy of a one-node, root-only tree produced with the respective center value selection strategy) and the corresponding *Baseline RMSE*.

4.2 Data sets

The algorithms were compared on two sets of relational data that are characterized by a clear linear ordering among the classes. The first is the well-known *Mesh Design* data set [Dolšak & Muggleton, 1992]. In the original version of this data set, there are 278 instances in 13 classes (1 FEs per edge, 2FEs, ..., 12 FEs, 17FEs). The class distribution is rather inhomogeneous, with about 50 % of the examples distributed between classes 1 and 2, but also sizeable numbers of instances in classes 3, 6, 8, and 12.

The second dataset, which we will call *biodegradability data set* in the following, describes 62 chemical substances in the familiar 'atoms and bonds' representation [Srinivasan et al., 1995]. The task is to predict the half-rate of surface water aerobic aqueous biodegradation in hours [Džeroski, personal communication]. For previous experiments, we had already discretized this quantity and mapped it to the four classes *fast*, *moderate*, *slow*, and *resistant*, represented as 1, 2, 3, and 4. The 62 instances are distributed over the four classes with frequencies 5/18/26/13.

4.3 Results

In the following tables, we summarize the results (RMSE and classification accuracy) of 10-fold stratified cross-validation runs on these data sets.

The first and most fundamental obervation we make is that the learners improve upon the baseline values in almost all cases, both in terms of RMSE and in terms of classification accuracy. In other words, they really learn something. Furthermore, in some cases they are already competitive with the specialized algorithms SRTRegress (the regression specialist) and SRTClassif (the classification specialist) at the level of raw data, and usually improve their performance with pre-processing.

As expected, there seems to be a fundamental tradeoff between the two goals of error minimization and accuracy maximization. This tradeoff shows most clearly in the results of the 'extreme' algorithms SRTRegress and SRTClassif: SRTClassif, which solely seeks to optimize the hit rate during tree construction but has no notion of class distance, is among the best class predictors in both domains, but among the worst in terms of RMSE. SRTRegress, on the other hand, is rather successful as a minimizer of the RMSE, but unusable as a classifier. Interestingly, neither of the two solves its particular problem optimally: there are ordinal learners that beat SRTClassif in terms of accuracy and at the same time achieve a lower RMSE than the 'specialist' SRTRegress (e.g., the MedianClass/Ranks combination in the mesh domain and MeanClass/Grades in the biodegradability data)! This result is most motivating; it shows that it is possible to develop learners that achieve good predictive accuracy while at the same time keeping an eye on the class-distance-weighted error.

Which of the four ordinal SRT variants is best? The question cannot be answered in a general way; it seems to depend very much on characteristics of the data. For instance, consider the results in the Mesh domain (Tables 3 and 4): here MedianClass and MostFrqClass seem clearly superior to the two strategies that predict the mean. A closer look at the data explains that. The mesh data are extremely skewed: around 50 % of the examples are in classes 1 and 2. Frequency-counting methods like MedianClass and MostFrqClass will tend to select these classes as center values in their nodes, while the average-based methods RealMean and MeanClass try to balance frequency and absolute values of class labels, which is inappropriate in this domain (see also the differences in baseline accuracy). Preprocessing is of great help here.

Generally, the effects of the preprocessing methods are rather mixed, but that should have been expected. In most cases, the learning algorithms already perform quite well on the raw data, but substantial improvements are possible if an adequate transformation is chosen, where adequacy depends on what types of measurements the ordinal classes represent. For instance, the 'grades' interpretation of the four biodegradation classes both makes intuitive sense and produces the best results in this domain for almost all algorithms, in terms of both error and accuracy. On the other hand, transformation methods that are inadequate can actually hurt, as can be seen in a few cases (e.g., the effect of the

Table 3. Summary of results on Mesh domain — RMSE (across: learning strategy (split evaluation and/or post-processing); down: pre-processing methods).

	RealMean +PostProc	Median Class	Mean Class	MostFrq Class	SRT Regress	SRT Classif
Baseline	*4.0287*	*4.6321*	*4.0287*	*4.6321*	*3.9997*	*4.6321*
Raw	2.2393	2.1600	2.2767	2.9437	2.2241	2.6184
Counts	1.9183	2.3011	2.0099	2.7997		
Ranks	2.3574	2.0411	2.2553	2.5991		
Grades	2.2728	2.5741	2.4968	2.9161		

Table 4. Summary of results on Mesh domain — Classification accuracy in %

	RealMean +PostProc	Median Class	Mean Class	MostFrq Class	SRT Regress	SRT Classif
Baseline	*4.32*	*26.25*	*4.32*	*26.25*	*00.00*	*26.25*
Raw	24.82	47.48	19.42	41.01	01.08	53.96
Counts	29.14	55.04	33.45	48.92		
Ranks	36.33	59.65	31.65	51.80		
Grades	39.57	48.20	48.20	50.72		

Table 5. Summary of results on Biodegradability domain — RMSE

	RealMean +PostProc	Median Class	Mean Class	MostFrq Class	SRT Regress	SRT Classif
Baseline	*0.9070*	*0.9070*	*0.9070*	*0.9070*	*0.8741*	*0.9070*
Raw	0.8980	0.9070	0.9246	0.9070	0.8448	0.9672
Counts	0.8231	0.9158	0.8519	0.9333		
Ranks	1.0000	0.8032	0.9070	0.8707		
Grades	0.8799	0.8032	0.7829	0.8032		

Table 6. Summary of results on Biodegradability domain — Classification accuracy

	RealMean +PostProc	Median Class	Mean Class	MostFrq Class	SRT Regress	SRT Classif
Baseline	*41.94*	*41.94*	*41.94*	*41.94*	*00.00*	*41.94*
Raw	46.77	41.94	43.55	41.94	09.68	50.00
Counts	46.77	45.16	46.77	41.94		
Ranks	33.87	45.16	46.77	48.39		
Grades	41.94	45.16	53.20	50.00		

'ranks' assumption on the performance of RealMean in the biodegradability domain).

Drawing more general conclusions from these limited experimental data seems unwarranted. Our results so far show that first-order learning algorithms for predicting ordinal classes can be naturally derived from relational regression algorithms, but more extensive experiments with larger data sets from diverse areas will be needed to establish the precise capabilities and relative advantages of these algorithms. We are currently applying the algorithms to several complex learning problems in various domains, from carcinogenicity prediction to music.

5 Conclusion

In this paper, we have taken first steps towards effective methods for learning to predict ordinal classes in relational domains, which we believe to be a natural problem class. We have shown how one can derive algorithms for learning ordered discrete classes by simple modifications to a relational regression algorithm. Preliminary experiments in two ILP benchmark domains have shown that the resulting algorithms seem to be able to achieve good predictive accuracy while at the same time keeping the class-distance-weighted error low. But the results also hint at complex interactions between the learner's bias (in particular, its evaluation heuristic, including the strategy for selecting a 'center value'), the distribution of classes and instances, and assumptions about the nature of the data embodied in the choice of pre-processing methods (if any). Our immediate research goals at the moment are to study the algorithms in a variety of complex application domains to learn more about these interactions and the relative strenghts of different learner-preprocessor combinations.

Finally, although our algorithms are based on a first-order learning algorithm and were tested in relational domains, the general methods should be valuable for propositional regression algorithms like CART [Breiman et al., 1984] or M5 [Quinlan, 1992] as well.

Acknowledgments

This research is part of the project "Carcinogenicity Detection by Machine Learning", supported by the Austrian Federal Ministry of Science and Transport. Michael de Groeve was supported by a SOCRATES (ERASMUS) grant. Thanks to Sašo Džeroski for providing the biodegradability dataset.

References

[Breiman et al., 1984] Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984). *Classification and Regression Trees.* Belmont, CA: Wadsworth International Group.

[Dolšak & Muggleton, 1992] Dolšak, B. & Muggleton, S. (1992). The Application of Inductive Logic Programming to Finite-Element Mesh Design. In S. Muggleton (ed.), *Inductive Logic Programming*. Academic Press.

[Dolšak et al., 1998] Dolšak, B., Bratko, I., & Jezernik, A. (1998). Application of Machine Learning in Finite Element Computation. In R.S. Michalski, I. Bratko & M. Kubat (eds.), *Machine Learning and Data Mining: Methods and Applications*. Chichester, UK: Wiley.

[Džeroski, personal communication] Džeroski, S. Biodegradability data set. Personal communication.

[Džeroski & Bratko, 1992] Džeroski, S. and Bratko, I. (1992). Handling noise in Inductive Logic Programming. In *Proceedings ILP-92*, Tokyo, Japan.

[Fürnkranz, 1994] Fürnkranz, J. (1994). FOSSIL: A robust relational learner". In *Proceedings of the 7th European Conference on Machine Learning (ECML-94)*. Berlin: Springer Verlag.

[Karalič, 1995] Karalic, A. (1995). *First Order Regression*. Ph.D. Thesis, University of Ljubljana.

[Karalič & Bratko, 1997] Karalič, A. & Bratko, I. (1997). First Order Regression. *Machine Learning* 26(2/3), 147–177.

[Kramer, 1996] Kramer, S. (1996). Structural Regression Trees. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*. Cambridge, MA: AAAI Press/MIT Press.

[Mathieson, 1996] Mathieson, M. (1996). Ordered Classes and Incomplete Examples in Classification. In M. Mozer et al. (eds.), *Advances in Neural Information Processing Systems 9*. Cambridge, MA: MIT Press.

[McCullagh, 1980] McCullagh, P. (1980). Regression Models for Ordinal Data. *Journal of the Royal Statistical Society Series B* 42, 109–142.

[Mosteller & Tukey, 1977] Mosteller, F. & Tukey, J.W. (1977). *Data Analysis and Regression - A Second Course in Statistics*. Reading, MA: Addison-Wesley.

[Muggleton, 1995] Muggleton, S. (1995). Inverse Entailment and Progol. *New Generation Computing* 13, 245–286.

[Pazzani & Kibler, 1992] Pazzani, M. & Kibler, D. (192). The Utility of Knowledge in Inductive Learning. *Machine Learning* 9(1), 57–94.

[Quinlan, 1990] Quinlan, J.R. (1990). Learning logical definitions from relations. *Machine Learning* 5, 239–266.

[Quinlan, 1992] Quinlan, J.R. (1992). Learning with Continuous Classes. In *Proceedings AI'92*. Singapore: World Scientific.

[Quinlan, 1993] Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

[Schiffers, 1997] Schiffers, J. (1997). A Classification Approach Incorporating Misclassification Costs. *Intelligent Data Analysis* 1(1).

[Silverstein & Pazzani, 1991] Silverstein, G. & Pazzani, M.J. (1991). Relational Clichés: Constraining Constructive Induction During Relational Learning. In *Proceedings of the 8th International Workshop on Machine Learning (ML-91)*. San Mateo, CA: Morgan Kaufmann.

[Srinivasan et al., 1995] Srinivasan, A., Muggleton, S., and King, R.D. (1995). Comparing the use of background knowledge by Inductive Logic Programming systems. In *Proceedings ILP-95*, Katholieke Universiteit Leuven, Belgium.

[Turney, 1995] Turney, P.D. (1995). Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research* 2, 369–409.