# Structural Regression Trees

**Stefan Kramer**
Austrian Research Institute for Artificial Intelligence
Schottengasse 3
A-1010 Vienna, Austria
stefan@ai.univie.ac.at

## Abstract

In many real-world domains the task of machine learning algorithms is to learn a theory for predicting numerical values. In particular several standard test domains used in Inductive Logic Programming (ILP) are concerned with predicting numerical values from examples and relational and mostly non-determinate background knowledge. However, so far no ILP algorithm except one can predict numbers and cope with non-determinate background knowledge. (The only exception is a covering algorithm called FORS.) In this paper we present Structural Regression Trees (SRT), a new algorithm which can be applied to the above class of problems. SRT integrates the statistical method of regression trees into ILP. It constructs a tree containing a literal (an atomic formula or its negation) or a conjunction of literals in each node, and assigns a numerical value to each leaf. SRT provides more comprehensible results than purely statistical methods, and can be applied to a class of problems most other ILP systems cannot handle. Experiments in several real-world domains demonstrate that the approach is competitive with existing methods, indicating that the advantages are not at the expense of predictive accuracy.

## Introduction

Many real-world machine learning domains involve the prediction of a numerical value. In particular, several test domains used in Inductive Logic Programming (ILP) (including the Mesh data sets (Dolsak, Bratko, & Jezernik 1994) and the problem of learning quantitative structure-activity relations (QSAR) (Hirst, King, & Sternberg 1994a) (Hirst, King, & Sternberg 1994b)) are concerned with the prediction of numerical values from examples and relational background knowledge. This kind of learning problem is called relational regression in (Džeroski 1995), and can be formulated in the "normal" ILP framework (i.e., it is not part of the non-monotonic ILP framework which includes the closed-world assumption). Nevertheless, relational regression differs from other ILP learning tasks in that there are no negative examples.

In this paper we present Structural Regression Trees (SRT), a new algorithm for relational regression. SRT can be viewed as integrating the statistical method of regression trees (Breiman *et al.* 1984) into ILP.

To simplify the presentation, we first review work in statistics and machine learning that is related to our approach. In the third section we will describe the method, including the solution for the problem of non-determinate literals. Furthermore, we present a new method for detecting outliers by analogy. Subsequently, we discuss results of experiments in several real-world domains. Finally, we draw our conclusions, and sketch possible directions of further research.

## Related Work

The classical statistical model for the prediction of numerical values is linear least-squares regression. Refinements and extensions like non-linear models are also well-known and used in many real-world applications. However, regression models have several limitations: first of all, regression models are often hard to understand. Secondly, classical statistical methods assume that all features are equally relevant for all parts of the instance space. Thirdly, regression models do not allow for easy utilization of domain knowledge. The only way to include knowledge is to "engineer" features, and to map these symbolic features to real-valued features.

In order to solve some of these problems, regression tree methods (CART (Breiman *et al.* 1984), RETIS (Karalic 1992), M5 (Quinlan 1992)) have been developed. Regression trees are supposed to be more comprehensible than traditional regression models. Furthermore, regression trees by definition do not treat all features as equally relevant for all regions of the instance space. The basic idea of regression trees according to CART is to minimize the least squared error for the next split of a node in the tree, and to predict, for unseen instances, the average of the dependent variable of all training instances covered by the matching leaf. RETIS and M5 differ in that they do not assign single values to the leaves, but linear regression models.

Sophisticated post-pruning methods have been developed for CART, since the method grows the tree until every leaf is "pure", i.e. the leaves cover instances sharing the same value of the dependent variable. The regression tree resulting from the growing phase is usually bigger than a classification tree, since it takes more nodes to achieve pure leaves.

Manago's KATE (Manago 1989) learns decision trees from examples represented in a frame-based language that is equivalent to first-order predicate calculus. KATE makes extensive use of a given hierarchy and heuristics to generate the branch tests. To our knowledge, KATE was the first system to induce first-order theories in a divide-and-conquer fashion.

Watanabe and Rendell (Watanabe & Rendell 1991) also investigated the use of divide-and-conquer for learning first-order theories. Although their so-called structural decision trees are used for the prediction of categorical classes and not continuous classes, it is the closest work found in the literature.

So far, two methods have been applied to the problem of relational regression: DINUS/RETIS (Džeroski, Todoroski, & Urbancic 1995), a combination of DINUS (Lavrac & Džeroski 1994) and RETIS (Karalic 1992), and FORS (Karalic 1995).

DINUS/RETIS transforms the learning problem into a propositional language, and subsequently applies RETIS, a propositional regression tree algorithm, to the transformed problem. In contrast to DINUS/RETIS, SRT solves the problem in its original representation, and does not require transforming the problem. Furthermore, the transformation does not work for non-determinate background knowledge, which is a strict limitation of the approach.[1]

FORS is the only other algorithm so far that can be applied to the same class of learning problems as SRT, since it can also deal with non-determinate background knowledge. FORS differs from SRT in its use of separate-and-conquer instead of divide-and-conquer (i.e., it is a covering algorithm, not a tree-based algorithm). Generally, all the advantages and disadvantages known from other algorithms of these types (tree-based vs. covering) apply. A discussion of both strategies in the context of top-down induction of logic programs can be found in (Boström 1995): on the one hand, the hypothesis space for separate-and-conquer is larger than for divide-and-conquer. Thus, more compact hypotheses might be found using separate-and-conquer. On the other hand, building a tree is computationally cheaper than searching for rules. In (Weiss & Indurkhya 1995), a comparison of tree induction and rule induction in propositional regression basically draws the same conclusions. However, the approach to rule-based regression in (Weiss & Indurkhya 1995) is different from FORS, since it involves the discretization of the numeric dependent variable.

[1]Cohen (Cohen 1994b) demonstrated that certain types of non-determinate background knowledge can be propositionalized. However, this is not generally the case.

A more detailed comparison between FORS and SRT shows that FORS may induce clauses containing linear regression models, whereas SRT does not. Linear regression models usually improve the accuracy of the induced models, but they also reduce their comprehensibility. Moreover, FORS requires setting thirteen parameters, whereas SRT only has one parameter. Seven parameters of FORS are used to control pre-pruning, whereas SRT uses a simple method for tree selection based on the minimum description length (MDL) principle (Rissanen 1978). Most importantly, we believe that the theories learned by FORS are very hard to understand, since the rules are basically ordered rule sets (Clark & Boswell 1991): to understand the meaning of a particular clause, we have to understand all preceding clauses in the theory.

| | |
|---|---|
| $I$ | set of instances covered by a leaf $l$ in a partial tree |
| $c$ | the conjunction of all literals in the path from the root of the tree to $l$ |
| $I_1$ | subset of $I$ for which proving $c \wedge t$ ($t \in T$, the set of possible tests) succeeds |
| $I_2$ | subset of $I$ for which proving $c \wedge t$ fails ( $I = I_1 \cup I_2$, $I_1 \cap I_2 = \emptyset$) |
| $n_1$ | number of instances in $I_1$ ( $n_1 = \lvert I_1 \rvert$ ) |
| $n_2$ | number of instances in $I_2$ ( $n_2 = \lvert I_2 \rvert$ ) |
| $y_{1,j}$ | value of the dependent variable of a training instance $j$ in $I_1$ |
| $y_{2,j}$ | value of the dependent variable of a training instance $j$ in $I_2$ |
| $\bar{y_1}$ | average of all instances in $I_1$ |
| $\bar{y_2}$ | average of all instances in $I_2$ |

Table 1: Definition of terms

## Description of the Method

### Overview

SRT is an algorithm which learns a theory for the prediction of numerical values from examples and relational (and even non-determinate) background knowledge. The algorithm constructs a tree containing a literal (an atomic formula or its negation) or a conjunction of literals in each node, and assigns a numerical value to each leaf.

More precisely, SRT generates a series of trees of increasing complexity, and subsequently returns one of the generated trees according to a preference criterion. This preference criterion is based on the *minimum description length* (MDL) principle (Rissanen 1978), which helps to avoid overfitting the data especially in the presence of noise. The whole process can be seen as a kind of pruning based on the MDL principle.

For the construction of a single tree, SRT uses the same method as used for the usual top-down induc-

```
activity(Drug,8.273)     :-     struct(Drug,Group1,Group2,Group3),
                                (pi_doner(Group3,X), X< 2).
activity(Drug,6.844)     :-     struct(Drug,Group1,Group2,Group3),
                                \+ (pi_doner(Group3,X), X< 2),
                                h_doner(Group1,0).
activity(Drug,6.176)     :-     struct(Drug,Group1,Group2,Group3),
                                \+ (pi_doner(Group3,X), X< 2),
                                \+ h_doner(Group1,0).
```

Table 2: Example of a structural regression tree in clausal form

tion of decision trees (Quinlan 1993a). The algorithm recursively builds a binary tree, selecting a 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. With each selected literal or conjunction, the examples covered by a node are further partitioned, depending on the success or failure of the literal(s) on the examples.

The selection of the literal or conjunction is performed as follows: let $I$ be the set of training instances covered by a leaf $l$ in a partial tree, and $c$ be the conjunction of all literals in the path from the root of the tree to $l$. (For a definition of all used terms see table 1.) Then every possible test $t$ is evaluated according to the resulting partitioning of the training instances $I$ in $l$. The instances $I$ are partitioned into the instances $I_1 \subseteq I$ for which proving $c \wedge t$ succeeds, and into the instances $I_2 \subseteq I$ for which proving $c \wedge t$ fails. For every possible test $t$ we calculate the sum of the squared differences between the actual values $y_{i,j}$ of the training instances and the average $\bar{y_i}$ of $I_i$. From all possible tests, SRT selects $t^* \in T$ which minimizes this sum of squared differences (see equation 1). When the stopping criterion (see below) is fulfilled, the average $\bar{y_i}$ is assigned to the leaf as the predicted value for unseen cases that reach the leaf.

$$Sum\ Squared\ Errors = \sum_{i=1}^{2} \sum_{j=1}^{n_i} (y_{i,j} - \bar{y_i})^2 \qquad (1)$$

From another point of view, each path starting from the root can be seen as a clause. Every time the tree is extended by a further literal or conjunction, two further clauses are generated: one of them is the current clause (i.e. the path in the current partial tree) extended by the respective literal or conjunction of literals. The other clause is the current clause extended by the negation of the literal(s). Table 2 shows a simple example of a structural regression tree in clausal form. The three clauses predict the biological activity of a compound from its structure and its characteristics. Depending on the conditions in the clauses, the theory assigns either 8.273 or 6.844 or 6.176 to every unseen instance. In the following, we will use this latter, clausal view on the process.

The simplest possible stopping criterion is used to decide if we should further grow a tree: SRT stops extending a clause when no literal(s) can produce two further clauses that both cover more than a required minimum number of training instances. In the following this parameter will be called the *minimum coverage* of all clauses in the theory. Apart from its use as a stopping criterion, the minimum coverage parameter has the following benefits: we have direct control over the complexity of the trees being built. The smaller the value of the parameter, the more complex the tree will be, since we allow for more specific clauses in the tree. In such a way we can generate a series of increasingly complex trees, and return the one which optimizes a preference function. Note that if the minimum coverage parameter had a different value, different splits might have been selected in any node of the tree.

SRT generates a series of increasingly complex trees by varying the minimum coverage parameter. The algorithm starts with a high minimum coverage, and decreases it from iteration to iteration. Fortunately, many iterations can be skipped, since nothing would change for certain values of the minimum coverage parameter: in the process of building the tree, we always select the one literal or conjunction which produces two clauses with an admissible coverage and which yields the lowest sum of squared errors. There could be literals or conjunctions yielding an even lower sum of squared errors, but with a coverage that is too low. The maximum coverage of these literals or conjunctions is the next value of the parameter for which the tree would be different from the current tree. So we choose this value as the next minimum coverage.

Finally, SRT returns the one tree from this series that obtains the best compression of the data. The compression measure is based on the minimum description length (MDL) principle (Rissanen 1978), and will be discussed in the next section.

## Tree Selection by MDL

The MDL principle tries to measure both the simplicity and the accuracy of a particular theory in a common currency, namely in terms of the number of bits needed for encoding theory and data. (Cheeseman 1990) defines the message length of a theory (called *model* in his article) as:

```
Total message length =
   Message length to describe the model +
   Message length to describe the data,
                         given the model.
```

This way a more complex theory will need more bits to be encoded, but might save bits when encoding more data correctly.

The message length of the model consists of the encoding of the literals and the encoding of the predicted values in the leaves. The message length of the data, given the model, is the encoding of the errors.

The encoding of the tree structure is simply the encoding of the choices made (for the respective literals) as the tree is built. For a single node, we encode the choice from all possible literals, so that the encoding considers predicates as well as all possible variabilizations of the predicates. In addition to the tree structure, we also have to encode the real numbers assigned to the leaves. In our coding scheme, we turn them into integers by multiplication and rounding. The factor is the minimum integer that still allows to discern the values in the training data after rounding. Then we encode all these integers in a way that the encoding requires the same amount of information for all values regardless of their magnitude.

The errors are also real numbers, and they are turned into integers in the same way as above. Subsequently, however, these integers are encoded by the *universal prior of integers* (UPI) (Rissanen 1986) — in this way the coding length of the errors roughly corresponds to their magnitude.

We chose MDL instead of cross-validation since it is computationally less expensive, and it can be used for pruning in search (Pfahringer & Kramer 1995). However, we are planning to compare both methods for model selection in the future.

## Non-Determinate Background Knowledge

Literals are non-determinate if they introduce new variables that can be bound in several alternative ways. Non-determinate literals often introduce additional parts of a structure like adjacent nodes in a graph. (Other examples are "part-of"-predicates.) Clearly, non-determinate literals do not immediately reduce the error when they are added to a clause under construction. Thus, any greedy algorithm without look-ahead would ignore non-determinate literals. The problem is how to introduce non-determinate literals in a controlled manner.

In SRT, the user has to specify which literal(s) may be used to extend a clause. Firstly, the user can define conjunctions of literals that are used for a limited look-ahead. (These user-defined schemata are similar to relational clichés (Silverstein & Pazzani 1991)). Furthermore, the user can constrain the set of possible literals depending on the body of the clause so far. The conditions concerning the body are arbitrary Prolog clauses, and therefore the user has even more possibilities to define a language than by Antecedent Description Grammars (ADGs) (Cohen 1994a). To further reduce the number of possibilities, the set of literals and conjunctions is also constrained by modes, types of variables, and variable symmetries.

## Outlier Detection by Analogy

Test instances that are outliers strongly deteriorate the average performance of learned regression models. Usually we cannot detect if test instances are outliers, because only little information is available for this task. If relational background knowledge is available, however, a lot of information can be utilized to detect, by "analogy", if test instances are outliers. Intuitively, when a new prediction is made, we check if the test instance is similar to the training instances which are covered by the clause that fires. If the similarity between these training instances and the test instance is not big enough, we should consider a different prediction than the one suggested by the clause which succeeds on the instance. In this case, we interpret the regression tree as defining a hierarchy of clusters. SRT chooses the cluster which is most similar to the test instance, and predicts the average of this cluster for the test instance.

To implement this kind of reasoning by analogy, we first have to define similarity of "relational structures" (such as labeled graphs). [2] Our simple approximation of similarity is based on *properties* of such structures. In this context, we say that an instance $i$ has a property $P$ iff $P$ is a literal or a conjunction (permitted by specified schemata) that immediately succeeds on $i$ (i.e., it succeeds without the introduction of intermediate variables). The similarity is defined as follows: let $p_{instance}(i)$ denote the set of properties of an instance $i$. Let $p_{in\_common}(I)$ be the set of properties all instances in a set $I$ have in common. Then the similarity between a test instance $i$ and a set (cluster) of training instances $I$ is

$$similarity(i, I) = \frac{|p_{instance}(i) \cap p_{in\_common}(I)|}{|p_{in\_common}(I)|}$$

The similarity is simply defined as the number of properties that the test instance and the covered training instances have in common, divided by the number of properties that the training instances have in common.

SRT uses a parameter for the *minimum similarity* to determine if the similarity between a test instance and the training instances covered by the clause that fires is large enough. The *minimum similarity* parameter is the only real parameter of SRT, since the best value for the required *minimum coverage* of a clause is determined automatically using the MDL

[2](Bisson 1992) defined a similarity measure for first-order logic, but it measures the similarity of two tuples in a relation, not of two "relational structures".

| Method | Pyrimidines Mean ($\sigma$) | Triazines Mean ($\sigma$) |
|---|---|---|
| Linear Regression on Hansch parameters and squares | 0.693 (0.170) | 0.272 (0.220) |
| Linear Regression on attributes and squares | 0.654 (0.104) | 0.446 (0.181) |
| Neural Network on Hansch parameters and squares | 0.677 (0.118) | 0.377 (0.190) |
| Neural Network on attributes and squares | 0.660 (0.225) | 0.481 (0.145) |
| GOLEM | 0.692 (0.077) | 0.431 (0.166) |
| SRT | 0.806 (0.110) | 0.457 (0.089) |

Table 3: Summary of all methods in the biomolecular domains of the inhibition of dihydrofolate reductase by pyrimidines and by triazines: performances as measured by the Spearman rank correlation coefficients

principle. Note that although we choose the cluster with the largest similarity, this similarity might be smaller than the specified *minimum similarity*.

This way of detecting and handling outliers adds an instance-based aspect to SRT. However, it is just an additional possibility, and can be turned off by means of the minimum similarity parameter.

## Experimental Results

We performed experiments in five real-world domains. For each domain, we performed experiments with (*minimum similarity* = 0.75 ) and without outlier detection by analogy (*minimum similarity* = 0 ). In cases where outlier detection affects the results, we will mention it in the discussion.

A common step in pharmaceutical development is forming a quantitative structure-activity relationship (QSAR) that relates the structure of a compound to its biological activity. Two QSAR domains, namely the inhibition of *Escherichia coli* dihydrofolate reductase (DHFR) by pyrimidines (Hirst, King, & Sternberg 1994a) and by triazines (Hirst, King, & Sternberg 1994b) have been used to test SRT.

The pyrimidine dataset consists of 2198 background facts and 55 instances (compounds), which are partitioned into 5 cross-validation sets. For the triazines, the background knowledge are 2933 facts, and 186 instances (compounds) are used to perform 6-fold cross validation. Hirst et al. made comprehensive comparisons of several methods in these domains, but they concluded there is no statistically significant difference between these methods.

Table 3 shows the results of the methods compared in (Hirst, King, & Sternberg 1994a) and in (Hirst, King, & Sternberg 1994b), and the results of SRT. The table summarizes the test set performances in both domains as measured by the Spearman rank correlation coefficients. The Spearman rank correlation coefficient is a measure of how much the order of the test instances according to the target variable correlates with the order predicted by the induced theory. The only reason why Hirst et al. use the Spearman rank correlation coefficient instead of, say, the average error is to compare GOLEM (Muggleton & Feng 1992) (which cannot predict numbers) with other methods.[3]

For the pyrimidines, SRT performs better than other methods, but the improvement is not statistically significant. Hirst et al. emphasize that a difference in Spearman rank correlation coefficient of about 0.2 would have been required for a data set of this size. The comparatively good performance of SRT is mostly due to the detection of two outliers that cannot be recognized by other methods. These two outliers were the only ones identified in these two domains. For the triazine dataset, SRT performs quite well, but again the differences are not statistically significant.

Since the Spearman rank correlation coefficient does not measure the quantitative error of a prediction, we included several other measures as proposed by Quinlan (Quinlan 1993b). Clearly, these measures have disadvantages, too, but they represent interesting aspects of how well a theory works for a given test set. Unfortunately, we do not yet have a full comparison with other methods that are capable of predicting numbers. Tables 4 and 7 contain the cross-validation test set performances of SRT in four test domains not only in terms of the Spearman rank correlation coefficient, but also in terms of several other accuracy measures.

Furthermore, we performed experiments in the domain of finite element mesh design (for details see (Dolsak, Bratko, & Jezernik 1994)), where the background knowledge is non-determinate. Table 5 shows the results of SRT for the mesh dataset together with the results of FOSSIL (Fürnkranz 1994) and results of other methods that were directly taken from (Karalic 1995). SRT performs better than FOIL (Quinlan 1990) and mFOIL (Džeroski & Bratko 1992), but worse than the other methods. However, statistical analysis shows that only the differences between FOIL and the other algorithms are significant.

We also applied SRT to the biological problem of learning to predict the mutagenic activity of a chemical, i.e., if it is harmful to DNA. (For details see (Srinivasan *et al.* 1994) and (Srinivasan, Muggleton, & King 1995)). This domain involves non-determinate background knowledge, too. In Table 6 we compiled

[3]Despite this disadvantage of GOLEM, Hirst et al. state that GOLEM is the only method that provides understandable rules about drug-receptor interactions. SRT can be seen as a step towards integrating both capabilities.

| Measure of Accuracy | Pyr. Mean ($\sigma$) | Triaz. Mean ($\sigma$) | Mutagen. Mean ($\sigma$) |
|---|---|---|---|
| Spearman rank correlation coefficient | 0.806 (0.110) | 0.457 (0.089) | 0.683 (0.124) |
| Average error $\|E\|$ | 0.435 (0.088) | 0.514 (0.084) | 1.103 (0.121) |
| Correlation $r$ | 0.818 (0.091) | 0.457 (0.104) | 0.736 (0.089) |
| Relative Error $RE$ | 0.218 (0.170) | 0.381 (0.132) | 0.170 (0.055) |

Table 4: Performances of SRT in three domains in terms of several accuracy measures

| Struct. | FOIL | mFOIL | GOLEM | MILP | FOSSIL | FORS | SRT |
|---|---|---|---|---|---|---|---|
| A | 17 | 23 | 21 | 21 | 23 | 22 | 23 |
| B | 6 | 12 | 12 | 12 | 13 | 12 | 11 |
| C | 7 | 9 | 10 | 11 | 6 | 8 | 9 |
| D | 0 | 6 | 16 | 16 | 16 | 16 | 16 |
| E | 6 | 12 | 21 | 30 | 32 | 29 | 9 |
| $\Sigma$ | 36 | 62 | 80 | 90 | 90 | 87 | 68 |
| % | 12.9 | 22.3 | 28.8 | 32.4 | 32.4 | 31.3 | 24.4 |

Table 5: Results of several methods in the domain of finite element mesh design: numbers and percentages of correctly classified edges

results from (Karalic 1995) and (Srinivasan, Muggleton, & King 1995), and filled the result of SRT. In the table, 'S' refers to structural background knowledge, 'NS' refers to non-structural features, 'PS' refers to predefined structural features that can be utilized by propositional algorithms, and 'MDL' refers to MDL pre-pruning. (Note that the results of FORS are the best that can be found by varying three of its parameters.) In the experiments we used the 188 instances (compounds) for a 10-fold cross-validation. The accuracy concerns the problem to predict *if* a chemical is active or not. Since SRT learns a theory that predicts the *activity* (a number) instead, we had to evaluate it in a different way (by discretization) to compare the results. Summing up, the experiments showed that SRT is competitive in this domain too, although the differences between SRT and the rest are not statistically significant.

Finally, we applied SRT to a domain where we are trying to predict the half-rate of surface water aerobic aqueous biodegradation in hours (Džeroski & Kompare 1995). To simplify the learning task, we discretized this quantity and mapped it to $\{1, 2, 3, 4\}$. The background knowledge is non-determinate, and except for the molecular weight there are no "global" features available. The dataset contains 62 chemicals, and we performed 6-fold cross-validation in our tests. The results of SRT can be found in table 7. SRT is the first algorithm to be tested on the data, and the results appear to indicate that there are too few instances to find good generalizations. Again, SRT with outlier detection improves upon the result of SRT without it. Note that neither a propositional algorithm (such as CART) nor an algorithm that cannot handle non-determinate background knowledge (such as FOIL, GOLEM and DINUS/RETIS) can be applied to this problem.

To sum up the experiments, SRT turned out to be quantitatively competitive, but its main advantages are that it yields comprehensible and explicit rules for predicting numbers, even when given non-determinate background knowledge.

| Algorithm | Accuracy |
|---|---|
| Lin.Regr. + NS | 0.85 (0.03) |
| Lin.Regr. + NS + PS | 0.89 (0.02) |
| Neural Network + NS | 0.86 (0.03) |
| Neural Network + NS + PS | 0.89 (0.02) |
| CART + NS + PS | 0.88 (0.02) |
| CART + NS | 0.82 (0.03) |
| FOIL + NS + S | 0.81 (0.03) |
| Progol + NS + S | 0.88 (0.02) |
| FORS + NS + S | 0.89 (0.06) |
| FORS + NS + S + MDL | 0.84 (0.11) |
| SRT + NS + S | 0.85 (0.08) |

Table 6: Summary of accuracy of several systems in the mutagenicity domain

## Conclusion and Further Research

In this paper we presented Structural Regression Trees (SRT), a new algorithm which can be applied to learning problems concerned with the prediction of numbers from examples and relational (and non-determinate) background knowledge. SRT can be viewed as integrating the statistical method of regression trees (Breiman *et al.* 1984) into ILP. SRT can be applied to a class of problems no ILP system except FORS can handle, and learns theories that may be easier to understand than theories found by FORS (section 2). The advantages and disadvantages of SRT are basically the same as the ones of CART: regression trees have a great potential to be explanatory, but we cannot expect to achieve a very high accuracy, since we predict

| Measure of Accuracy | Biod. with Outl. Det. Mean ($\sigma$) | Biod. w/o Outl. Det. Mean ($\sigma$) |
|---|---|---|
| Spearman rank correlation coefficient | 0.463 (0.213) | 0.402 (0.232) |
| Average error $\|E\|$ | 0.744 (0.190) | 0.771 (0.210) |
| Correlation $r$ | 0.382 (0.247) | 0.364 (0.223) |
| Relative Error $RE$ | 0.363 (0.139) | 0.377 (0.141) |

Table 7: Performances of SRT with and without outlier detection in the biodegradability domain

constant values for whole regions of the instance space. As it could help to build more accurate models, one of the next steps will be to assign linear regression models to the leaves.

One of the biggest differences between SRT and FORS is that it is a tree-based and not a covering algorithm. So generally all the advantages and disadvantages known from other algorithms of these types apply (Boström 1995) (Weiss & Indurkhya 1995). However, FORS and SRT also differ in many other ways, and thus a real comparison of the search strategies employed is still to be done for relational regression.

Experiments in several real-world domains demonstrate that the approach is competitive with existing methods, indicating that its advantages (the applicability to relational regression given non-determinate background knowledge and the comprehensibility of the rules) are not at the expense of predictive accuracy.

SRT generates a series of increasingly complex trees, but currently every iteration starts from scratch. We are planning to extend the algorithm such that parts of the tree of one iteration can be reused in the next iteration.

We also plan to compare our way of coverage-based prepruning and tree selection by MDL with more traditional pruning methods à la CART (Breiman *et al.* 1984).

Besides, we addressed the problem of non-determinate literals. We adopted and generalized solutions for this problem, but they involve the tiresome task of writing a new specification of admissible literals and conjunctions for each domain. We therefore think that a more generic solution would make the application of the method easier.

## Acknowledgements

This research is sponsored by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant number P10489-MAT. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry of Science, Research, and Arts. I would like to thank Johannes Fürnkranz, Bernhard Pfahringer and Gerhard Widmer for valuable discussions. I also wish to thank Sašo Džeroski for providing the biodegradability dataset and the anonymous AAAI referees for their comments which helped to improve this paper.

## References

Bisson, G. 1992. Learning in FOL with a similarity measure. In *Proc. Tenth National Conference on Artificial Intelligence (AAAI-92)*.

Boström, H. 1995. Covering vs. Divide-and-Conquer for Top-Down Induction of logic programs. In *Proc. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1194–1200. San Mateo, CA: Morgan Kaufmann.

Breiman, L.; Friedman, J.; Olshen, R.; and Stone, C. 1984. *Classification and Regression Trees.* The Wadsworth Statistics/Probability Series. Belmont, CA: Wadsworth International Group.

Cheeseman, P. 1990. On finding the most probable model. In Shrager, J., and Langley, P., eds., *Computational Models of Discovery and Theory Formation.* Los Altos, CA: Morgan Kaufmann.

Clark, P., and Boswell, R. 1991. Rule induction with CN2: Some recent improvements. In *Proceedings of the Fifth European Working Session on Learning*, 151–161. Berlin Heidelberg New York: Springer.

Cohen, W. 1994a. Grammatically biased learning: Learning logic programs using an explicit antecedent description language. *Artificial Intelligence* 68(2).

Cohen, W. 1994b. Pac-learning nondeterminate clauses. In *Proc. Twelfth National Conference on Artificial Intelligence (AAAI-94)*.

Dolsak, B.; Bratko, I.; and Jezernik, A. 1994. Finite element mesh design: An engineering domain for ILP application. In *Proceedings of the Fourth International Workshop on Inductive Logic Programming (ILP-94)*, GMD-Studien Nr. 237, 305–320.

Džeroski, S., and Bratko, I. 1992. Handling noise in Inductive Logic Programming. In *Proceedings of the International Workshop on Inductive Logic Programming.*

Džeroski, S., and Kompare, B. 1995. Personal Communication.

Džeroski, S.; Todoroski, L.; and Urbancic, T. 1995. Handling real numbers in inductive logic programming: A step towards better behavioural clones. In Lavrac, N., and Wrobel, S., eds., *Machine Learning: ECML-95*, 283–286. Berlin Heidelberg New York: Springer.

Džeroski, S. 1995. *Numerical Constraints and Learnability in Inductive Logic Programming.* Ph.D. Dissertation, University of Ljubljana, Ljubljana, Slovenija.

Fürnkranz, J. 1994. FOSSIL: A robust relational learner. In Bergadano, F., and De Raedt, L., eds., *Machine Learning: ECML-94*, 122–137. Berlin Heidelberg New York: Springer.

Hirst, J.; King, R.; and Sternberg, M. 1994a. Quantitative structure-activity relationships by neural networks and inductive logic programming. the inhibition of dihydrofolate reductase by pyrimidines. *Journal of Computer-Aided Molecular Design* 8:405–420.

Hirst, J.; King, R.; and Sternberg, M. 1994b. Quantitative structure-activity relationships by neural networks and inductive logic programming: The inhibition of dihydrofolate reductase by triazines. *Journal of Computer-Aided Molecular Design* 8:421–432.

Karalic, A. 1992. Employing linear regression in regression tree leaves. In Neumann, B., ed., *Proc. Tenth European Conference on Artificial Intelligence (ECAI-92)*, 440–441. Chichester, UK: Wiley.

Karalic, A. 1995. *First Order Regression.* Ph.D. Dissertation, University of Ljubljana, Ljubljana, Slovenija.

Lavrac, N., and Džeroski, S. 1994. *Inductive Logic Programming.* Chichester, UK: Ellis Horwood.

Manago, M. 1989. Knowledge-intensive induction. In Segre, A., ed., *Proceedings of the Sixth International Workshop on Machine Learning*, 151–155. Morgan Kaufman.

Muggleton, S., and Feng, C. 1992. Efficient induction of logic programs. In Muggleton, S., ed., *Inductive Logic Programming.* London, U.K.: Academic Press. 281–298.

Pfahringer, B., and Kramer, S. 1995. Compression-based evaluation of partial determinations. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining.* AAAI Press.

Quinlan, J. 1990. Learning logical definitions from relations. *Machine Learning* 5:239–266.

Quinlan, J. 1992. Learning with continuous classes. In Adams, S., ed., *Proceedings AI'92*, 343–348. Singapore: World Scientific.

Quinlan, J. 1993a. *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

Quinlan, J. 1993b. A case study in machine learning. In *Proceedings ACSC-16 Sixteenth Australian Computer Science Conference.*

Rissanen, J. 1978. Modeling by shortest data description. *Automatica* 14:465–471.

Rissanen, J. 1986. Stochastic complexity and modeling. *The Annals of Statistics* 14(3):1080–1100.

Silverstein, G., and Pazzani, M. 1991. Relational cliches: Constraining constructive induction during relational learning. In Birnbaum, L., and Collins, G., eds., *Machine Learning: Proceedings of the Eighth International Workshop (ML91)*, 203–207. San Mateo, CA: Morgan Kaufmann.

Srinivasan, A.; Muggleton, S.; King, R.; and Sternberg, M. 1994. Mutagenesis: ILP experiments in a non-determinate biological domain. In *Proceedings of the Fourth International Workshop on Inductive Logic Programming (ILP-94)*, GMD-Studien Nr. 237, 217–232.

Srinivasan, A.; Muggleton, S.; and King, R. 1995. Comparing the use of background knowledge by Inductive Logic Programming systems. In *Proceedings of the 5th International Workshop on Inductive Logic Programming (ILP-95)*, 199–230. Katholieke Universiteit Leuven.

Watanabe, L., and Rendell, L. 1991. Learning structural decision trees from examples. In *Proc. Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 770–776. San Mateo, CA: Morgan Kaufmann.

Weiss, S., and Indurkhya, N. 1995. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research* 3:383–403.