

Constraint Based Induction of Multi-Objective Regression Trees

Jan Struyf¹ and Sašo Džeroski²

¹Katholieke Universiteit Leuven, Dept. of Computer Science
Celestijnenlaan 200A, B-3001 Leuven, Belgium

Jan.Struyf@cs.kuleuven.be

²Jozef Stefan Institute, Dept. of Knowledge Technologies
Jamova 39, 1000 Ljubljana, Slovenia
Saso.Dzeroski@ijs.si

Abstract. Constrained based inductive systems are a key component of inductive databases and responsible for building the models that satisfy the constraints in the inductive queries. In this paper, we propose a constraint based system for building multi-objective regression trees. A multi-objective regression tree is a decision tree capable of predicting several numeric variables at once. We focus on size and accuracy constraints. By either specifying maximum size or minimum accuracy, the user can trade-off size (and thus interpretability) for accuracy. Our approach is to first build a large tree based on the training data and to prune it in a second step to satisfy the user constraints. This has the advantage that the tree can be stored in the inductive database and used for answering inductive queries with different constraints. Besides size and accuracy constraints, we also briefly discuss syntactic constraints. We evaluate our system on a number of real world data sets and measure the size versus accuracy trade-off.

1 Introduction

The idea behind inductive databases [13, 7] is to tightly integrate databases with data mining. An inductive database not only stores data, but also models that have been obtained by running mining algorithms on the data. By means of a query language, the end user can retrieve particular models. For example, the user could query the system for a decision tree that is smaller than 20 nodes, has an accuracy above 80%, and with a particular attribute in the top node. If the database does not include a model satisfying the constraints, then an induction algorithm is called to construct it.

In this paper we propose a constraint based induction algorithm for multi-objective regression trees (MORTs). MORTs are regression trees [6] capable of predicting several numeric variables at once [2]. This has two main advantages over building a separate regression tree for each target: (1) a single MORT is usually much smaller than the total size of the individual trees for all variables, and (2) a MORT explicitates dependencies between the different target variables.

The approach that we propose is to first build a large tree based on the training data and then to prune it in a second step to satisfy the user constraints. This has the advantage that the tree can be stored in the inductive database and used for answering inductive queries with different constraints. The pruning algorithm that we propose is an extension to MORTs of the pruning algorithm for classification trees developed by Garofalakis et al. [12], which in turn is based on earlier work by Bohanec and Bratko [3] and Almuallim [1]. It is a dynamic programming algorithm that searches for a subtree of the given tree that satisfies the size and accuracy constraints. It can either minimize tree size and return the smallest tree satisfying a minimum accuracy constraint or maximize accuracy and return the most accurate tree satisfying a maximum size constraint.

After extending the pruning algorithm to MORTs, we present an extensive empirical evaluation measuring the size versus accuracy trade-off of MORTs on several real world data sets. Our evaluation shows (1) that the accuracy of MORTs is close to that of a set of single-objective regression trees of the same size, and (2) that in many cases tree size can be reduced significantly (thereby increasing interpretability) at the expense of only a small accuracy loss.

The rest of this paper is organized as follows. In Section 2, we briefly discuss MORTs and their induction algorithm. Section 3 reviews the pruning algorithm by Garofalakis et al. and Section 4 extends it to MORTs. Accuracy and syntactic constraints are discussed in Section 5. The empirical evaluation follows in Section 6. Future work is discussed in Section 7 and Section 8 states the main conclusions.

2 Multi-Objective Regression Trees (MORTs)

MORTs are regression trees [6] capable of predicting several numeric target variables at once. An example of a MORT is depicted in Fig. 1. Each leaf stores a vector with as components the predictions for the different target variables.

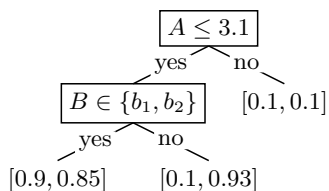


Fig. 1. A MORT predicting two numeric variables.

MORTs have been introduced as a special instance of predictive clustering trees [2]. In this framework, a tree is viewed as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. MORTs are constructed with a standard top-down induction algorithm similar to that of CART [6] or

```

procedure ComputeError( $N, k$ )
1: if Tree[ $N, k$ ].computed
2:   return Tree[ $N, k$ ].error
3: Tree[ $N, k$ ]. $k_1 := -1$ 
4: Tree[ $N, k$ ].error := leaf_error( $N$ )
5: if  $k \geq 3$  and  $N$  is no leaf
6:   for  $k_1 := 1$  to  $k - 2$ 
7:      $k_2 := k - k_1 - 1$ 
8:      $e :=$  ComputeError( $N_1, k_1$ )
9:       + ComputeError( $N_2, k_2$ )
10:    if  $e <$  Tree[ $N, k$ ].error
11:      Tree[ $N, k$ ].error :=  $e$ 
12:      Tree[ $N, k$ ]. $k_1 := k_1$ 
13: Tree[ $N, k$ ].computed := true
14: return Tree[ $N, k$ ].error

procedure PruneToSizeK( $N, k$ )
1: ComputeError( $N, k$ )
2: PruneRecursive( $N, k$ )

procedure PruneRecursive( $N, k$ )
1: if  $N$  is a leaf
2:   return
3: if  $k < 3$  or Tree[ $N, k$ ]. $k_1 = -1$ 
4:   remove children of  $N$ 
5: else
6:    $k_1 :=$  Tree[ $N, k$ ]. $k_1$ 
7:    $k_2 := k - k_1 - 1$ 
8:   PruneRecursive( $N_1, k_1$ )
9:   PruneRecursive( $N_2, k_2$ )

```

Fig. 2. The constraint-based decision tree pruning algorithm.

C4.5 [15]. The heuristic used in this algorithm for selecting the attribute tests in the internal nodes is intra-cluster variation summed over the subsets induced by the test. Intra-cluster variation is defined as $N \cdot \sum_{t=1}^T \text{Var}[y_t]$, with N the number of examples in the cluster, T the number of target variables, and $\text{Var}[y_t]$ the variance of target variable t in the cluster. Minimizing intra-cluster variation results in homogeneous leaves, which in turn results in accurate predictions (the predicted vector in a leaf is the vector mean of the target vectors of the training examples belonging to it). More details about MORTs can be found in [2].

3 Constraint-Based Decision Tree Pruning

Fig. 2 defines the pruning method proposed by Garofalakis et al. [12] for computing for a given maximum tree size k a subtree of the given tree (rooted at node N) with maximum accuracy (minimum error). First ComputeError is called to find out which nodes are to be included in the solution and then PruneRecursive is called to remove the other nodes.

ComputeError employs dynamic programming to compute in Tree[N, k].error the error of the minimum-error subtree rooted at N containing at most k nodes. This subtree is either the tree in which N is pruned to a leaf or a tree in which N has two children (we only consider binary trees) N_1 and N_2 such that N_1 (N_2) is a minimum error subtree of size at most k_1 (k_2) and $k_1 + k_2 = k - 1$. The algorithm computes the minimum over these possibilities in the for-loop starting on line 6. The possibility that N is pruned to a leaf is taken into account by initializing the error to leaf_error(N) in line 4. The flag Tree[N, k].computed is used to avoid repeated computation of the same information.

After ComputeError completes, Tree[N, k]. k_1 stores the maximum size of the left subtree in the minimum-error subtree of at most k nodes rooted at N . Note that if Tree[N, k]. $k_1 = -1$, then this subtree consists of only the node

N . PruneRecursive is called next to prune nodes that do not belong to the minimum-error subtree.

The time and space complexity of the algorithm(s) are both $\mathcal{O}(nk)$ with n the size of the tree and k the maximum tree size parameter [12].

4 Size Constraints for MORTs

The pruning algorithm discussed in the previous section was originally developed in a classification setting with as error measure the number of misclassified examples or the minimum description length cost. It is however not difficult to see that the algorithm can be used in combination with any error measure that is *additive*, i.e., a measure for which it holds that if a data set is partitioned into a number of subsets, the error computed on the whole set is equal to the sum of the errors computed on the subsets in the partition.

Definition 1 (Additive error measure). *An error measure f is additive iff for any data set D and for any partition of D into subsets D_i it holds that $f(D) = \sum_i f(D_i)$.*

The additivity property of the error measure is used in lines 8-9 of the ComputeError algorithm.

Examples of error measures in the multi-objective regression setting that satisfy the additivity property are squared error and absolute error.

Definition 2 (Squared and absolute error). *Given a data set with N examples and T targets, squared error is defined as $SE = \sum_{i=1}^N \sum_{t=1}^T (y_{t,i} - y_{t,i}^p)^2$ and absolute error as $AE = \sum_{i=1}^N \sum_{t=1}^T |y_{t,i} - y_{t,i}^p|$, with $y_{t,i}$ the actual and $y_{t,i}^p$ the predicted value for target variable t of example i .*

Obviously, the pruning algorithm can also be used to minimize these error measures by just plugging them in at line 4. Note that minimizing squared error implicitly also minimizes error measures that are a monotonically increasing function of the former, such as mean squared error (MSE) and root mean squared error (RMSE). The same holds for absolute error and mean absolute error (MAE)¹. Therefore, the pruning algorithm can be trivially extended to all these error measures. In the empirical evaluation (Section 6), we will use the pruning algorithm in combination with squared error (Definition 2).

We end this section with a number of remarks.

- To obtain good results, it is required that the heuristic used for building the tree is “compatible” with the error measure, i.e., the heuristic should be designed to optimize the same error measure as is used in the pruning algorithm. In our case, one might say that this requirement holds because the intra-cluster variation heuristic locally optimizes squared error. Locally

¹ Or for any error measure based on a Minkowski distance $d(x, y) = (\sum |x_k - y_k|^p)^{\frac{1}{p}}$.

optimizing the error measure is however not always the best choice, e.g., in the context of classification trees, one should use information gain as heuristic and not accuracy [6].

- Some error measures that are used in regression tasks, such as Pearson correlation, are neither additive nor a monotonically increasing function of an additive measure. These error measures cannot be minimized with the pruning algorithm of Fig. 2.
- Garofalakis et al. [12] also propose a method for pushing the constraints into the tree building phase. While this makes tree building more efficient, it has the disadvantage that the resulting tree is specific to the constraints in the given query and that it cannot be used anymore for answering queries with other constraints.

Pushing constraints in the case of MORTs is more difficult than in the case of classification trees. The reason is that the constraint pushing algorithm requires the computation of a lower bound on the error of a partially built tree. To our knowledge, such a lower bound has not yet been defined for regression trees or MORTs.

- In this paper, we focus on MORTs, but a similar approach is also possible for predictive clustering trees [2] in general, as long as the error measure has the additivity property. For example, one could consider multi-objective classification trees (MOCTs) with as error measure the number of misclassified examples summed over the different target variables. For multi-objective trees with both numeric and nominal target variables one could define an additive error measure as the (weighted) sum of the measure on the nominal variables and that on the numeric variables.

5 Maximum Error and Syntactic Constraints

The pruning algorithm can be used to find a subtree with minimum error given a maximum size constraint. The same algorithm can also be used for solving the following, dual problem: given a maximum error constraint, find the smallest tree that satisfies this constraint. To accomplish this, one constructs a sequence of pruned trees using the algorithm of Fig. 2 for increasing values of the size constraint k , i.e., $k_1 = 1, k_2 = 3, \dots, k_m = 2m - 1$, until a tree that satisfies the maximum error constraint is found. The resulting tree is the smallest tree having an error less than the maximum error constraint. (Computing the sequence of trees is computationally cheap because the pruning algorithm does not access the data; $\text{leaf_error}(N)$ can be computed and stored for each node before running the pruning algorithm. Moreover, the $\text{Tree}[N, k]$ values of small trees can be reused when constructing larger trees.)

In the multi-objective regression setting, one approach is to specify the maximum error summed over all target variables. Another approach is to specify a bound for each individual target variable. The latter can be useful if an application demands that some target variables are predicted more accurately than others.

Besides size and error constraints, syntactic constraints are also important in practice. Although they are not the focus of this paper, we discuss them briefly. Syntactic constraints can be used as follows in the context of decision trees. Suppose that a domain expert knows which attributes are important for a given application. A syntactic constraint can then be used to mine for a decision tree with such an attribute in the top node. Although other trees with different attributes in the top node might be equally accurate, the one with the attribute selected by the expert will probably be more easy to interpret.

CLUS, the system that we will use in the empirical evaluation, supports this type of syntactic constraints. The idea is that the user can specify a partial tree (a subtree including the root node) in the inductive query. The induction algorithm is then initialized with this partial tree and the regular top-down induction method is used to complete it.

The ability to use syntactic (partial tree) constraints allows for a greater involvement of the user in the construction of the decision tree and a greater user influence on the final result. Some domain knowledge of the user can be taken into account in this way.

6 Empirical Evaluation

The goal of our empirical evaluation is two-fold. First we would like to evaluate the size versus error trade-off that is possible by using the size constraints in real world applications. Second, we compare single-objective and multi-objective regression. The hypothesis that we test is that a single multi-objective tree of size s is equally accurate as a set of single-objective trees, one for each target variable, each one of the same size s . Having one single small multi-objective model that is equally accurate is advantageous because it is easier to interpret than a set of trees. Moreover, it can explicitly represent dependencies between the different targets. E.g., the tree in Fig. 1 shows that $A > 3.1$ has a negative influence on both targets, while $(A \leq 3.1) \wedge (B \notin \{b_1, b_2\})$ has a negative influence on the first target, but a positive effect on the second.

6.1 Experimental Setup

The size, error and syntactic constraints have been implemented in CLUS². CLUS is a system for building clustering trees [2] in general and MORTs in particular.

The data sets that we use are listed, together with their properties, in Table 1. Most data sets are of ecological nature. Each data set represents a multi-objective regression problem and the number of target variables T varies from 2 to 39. A detailed description of the data sets can be found in the references included in Table 1.

For each data set, we run CLUS in single-objective mode for each target variable and in multi-objective mode. We use 10-fold cross-validation to estimate

² CLUS is available from the authors upon request.

Table 1. Data set properties: domain, number of instances (N), number of input attributes (Attr), and number of target attributes ($|T|$).

	Domain	Task	N	Attr	$ T $
E_1	Sigma real [8]		817	4	2
E_2	Sigma simulated [11]		10368	11	2
E_3	Soil quality 1 [9]	Acari/Coll./Biodiv.	1944	139	3
E_4		Acari groups	"	"	4
E_5		Coll. groups	"	"	5
E_6		Coll. species	"	"	39
E_7	Soil quality 2 [14]		393	48	3
E_8	Water quality [10]	Plants	1060	16	7
E_9		Animals	"	"	7
E_{10}		Chemical	1060	836	16

the performance of the resulting trees. For each run, we build one large tree, store it, and then generate subtrees of this tree using the pruning algorithm discussed in Section 3 for different values of the size constraint k . We set the pruning algorithm to minimize squared error on the training set. (I.e., we follow the approach proposed in [12]. Note that the algorithm can also be used in combination with a separate validation set.)

We also include results obtained with the M5' system from the Weka toolkit [17]. Note that M5' only supports single-objective regression.

6.2 Results

Fig. 3 and Fig. 4 present the results. For each experiment, the mean squared error (MSE) and the average squared Pearson correlation \bar{r}^2 (averaged over the T target variables) is reported. For most data sets, the results for the multi-objective tree are close to these of the set of single-objective trees (SORTs), especially for large tree sizes. Most results are slightly in favor of the SORTs. Hence, the increased interpretability offered by MORTs comes at the price of a small increase in error. One exception is Soil quality 2, where MORTs perform a little better than SORTs. This effect can be explained by the fact that the target variables are highly correlated in this data set.

The largest performance difference is obtained on Soil quality 1, Collembola species. Here SORTs perform clearly better than MORTs. But the number of target variables (39) is also high. Note that this also implies that the total size of the SORTs is 39 times the size of the MORT. To investigate this effect further, we have plotted the results with total model size on the horizontal axis in Fig. 5 and Fig. 6. These results show that for a given total size, the error obtained with a MORT is in 6 out of 10 data sets clearly smaller than that of the set of SORTs. (For the other 4, the measured error is similar.)

Observe that the error curves are typically flat for a large size-interval. Therefore, tree size can in most cases be kept small without losing much accuracy.

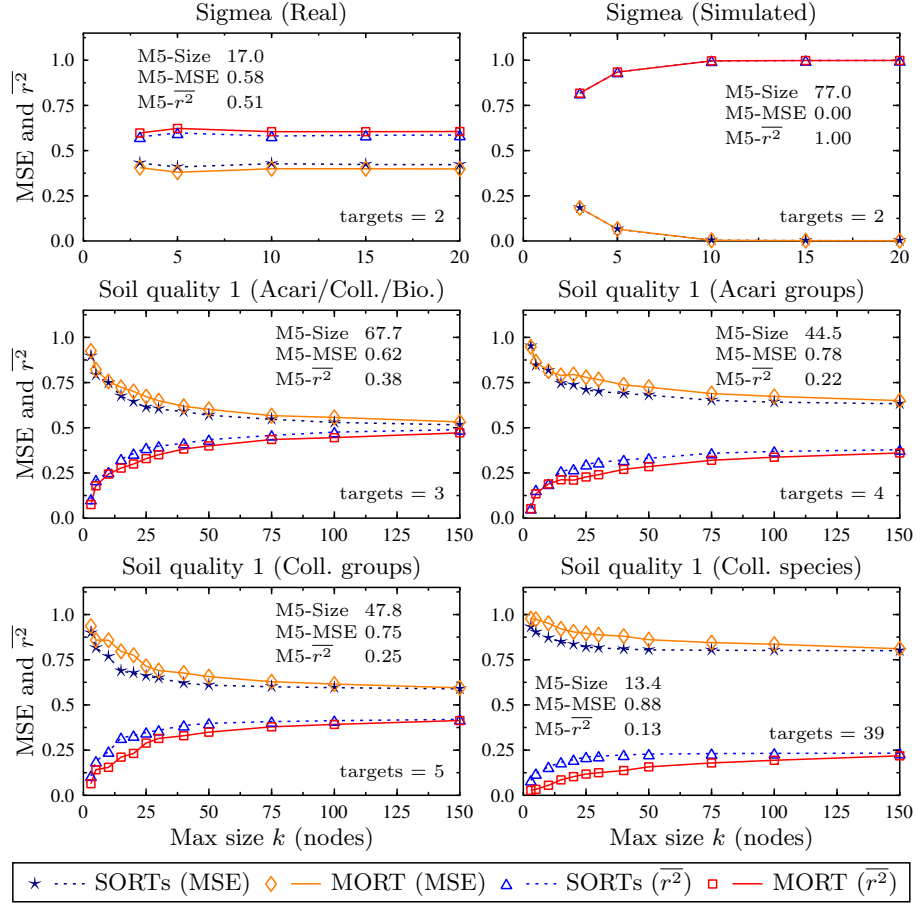


Fig. 3. Comparing the MSE and average squared correlation $\overline{r^2}$ of SORTs and MORTs for different values of the size constraint k .

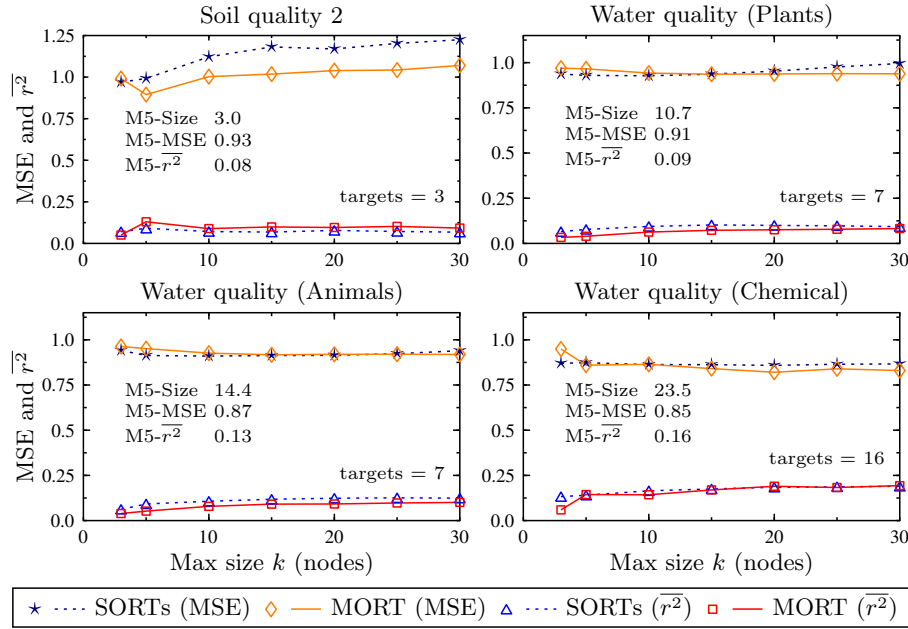


Fig. 4. Comparing the MSE and average squared correlation $\overline{r^2}$ of SORTs and MORTs for different values of the size constraint k .

Based on graphs as in Fig. 3 and Fig. 4 the domain expert can easily select a tree that has a good trade-off between size (and thus interpretability) and accuracy.

It is interesting to note that no overfitting occurs, except in Soil quality 2. I.e., for most data sets, the error decreases with tree size to a constant value and does not increase again for larger trees.

The graphs also include results for M5' in regression tree mode. Accuracy-wise, the results of M5' are close to the results obtained with CLUS. The size of the M5' trees is always situated in the flat part of the error curve. For some data sets M5' generates trees that are rather large. The most extreme case is Sigmea Simulated where it generates a tree with 77 nodes. By setting a size constraint, unnecessarily large trees can be easily avoided.

To summarize, MORTs together with size constraints are a good choice if interpretability is important and a small loss in accuracy can be tolerated. If accuracy is more important, then a larger MORT might still be preferable over a set of SORTs, of which the total size will be even larger.

7 Further Work

As we already noted in Section 4, the size and error constraints can also be extended to multi-objective classification and multi-objective prediction with

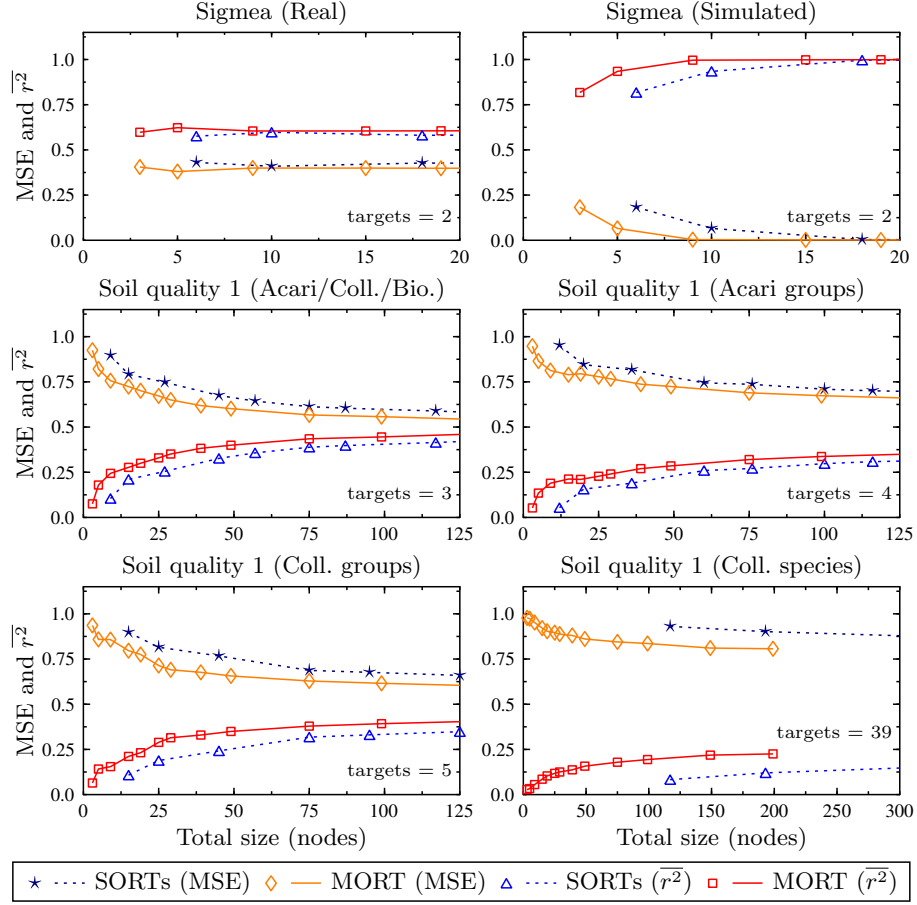


Fig. 5. MSE and average squared correlation $\overline{r^2}$ versus total model size.

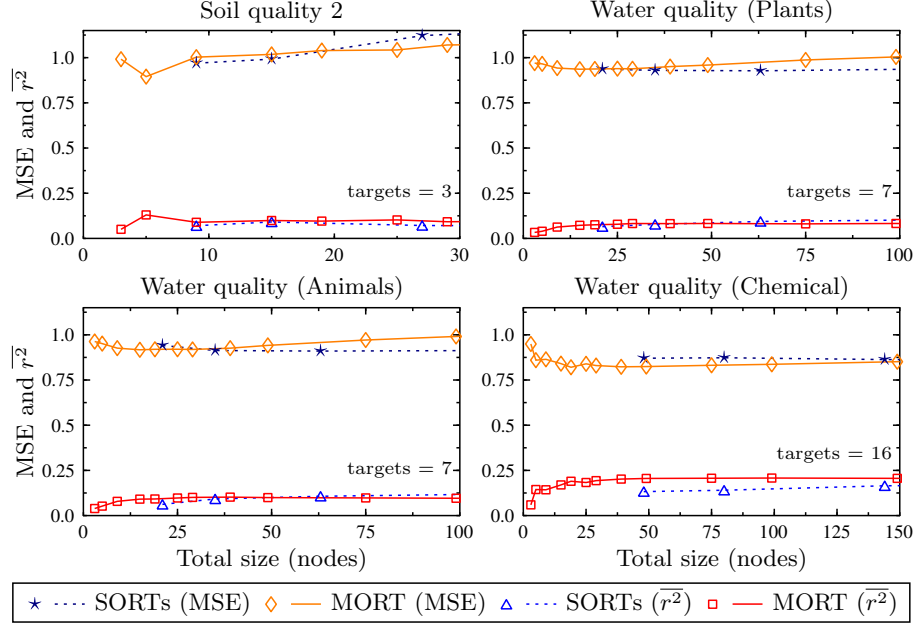


Fig. 6. MSE and average squared correlation $\overline{r^2}$ versus total model size.

both nominal and numeric targets. We would like to experimentally evaluate these settings as well.

We are also planning to compare the MORT and SORT settings in more detail. Currently, we have compared the extreme case where each target attribute is predicted by a single regression tree to the other extreme case where all target attributes are predicted by one single MORT. An in-between approach could be to partition the target attributes into subsets and construct a MORT for each subset. Target attributes that depend in a similar way on the input attributes should be combined in the same subset. Clearly, constructing sets of MORTs for all possible partitions and picking the partition that minimizes a given trade-off between error and complexity would be too expensive. Therefore, we would like to investigate heuristic approaches. One method could be to construct the partition by clustering the attributes using correlation as similarity measure, i.e., to put highly correlated attributes in the same cluster. Note that we already observed in the experiments in Section 6 that MORTs perform good if the target attributes are highly correlated.

We would also like to investigate the effect of several parameters of the input data on the relative performance of a MORT compared to a set of SORTs, such as training set size or the effect of noise.

Furthermore, we would like to investigate the use of MORTs in ensemble methods. This has already been explored to some extent by Sain and Carmack [16],

who propose a boosting approach with MORTs. Similarly, MORTs could be used for bagging [4] and in random forests [5].

8 Conclusion

In this paper, we have proposed a system for constrained based induction of multi-objective regression trees (MORTs). It supports size, error and syntactic constraints and works in two steps. In a first step, a large tree is built that satisfies the syntactic constraints. This tree is stored in the inductive database and used in a second step to generate trees that satisfy particular size or error constraints. To accomplish this, we have extended the pruning algorithm introduced by Garofalakis et al. to MORTs. Two modes of operation are supported: (1) given a maximum size constraint, return a subtree with the smallest error, and (2) given a maximum error constraint, return the smallest subtree that satisfies this constraint.

While we have focused on MORTs, the pruning algorithm can also be extended to predictive clustering trees in general. E.g., it can also be used for multi-objective classification and multi-objective prediction with both nominal and numeric targets.

In an empirical evaluation, we have tested our approach on a number of real world data sets. Our evaluation shows (1) that the accuracy of MORTs is close to that of a set of single-objective regression trees, each of the same size, and (2) that in many cases tree size can be reduced significantly (thereby increasing interpretability) at the expense of only a small accuracy loss. MORTs together with size constraints are thus a good choice if interpretability is important and a small loss in accuracy can be tolerated. Moreover, if we consider total size instead of average tree size, we observe that for a given total size, the error obtained with a MORT is smaller than or similar to that of a set of SORTs.

Acknowledgments

The authors are grateful to Hendrik Blockeel who provided valuable comments on the text and the empirical evaluation. Jan Struyf is a postdoctoral fellow of the Fund for Scientific Research of Flanders (FWO-Vlaanderen).

References

1. H. Almuallim. An efficient algorithm for optimal pruning of decision trees. *Artificial Intelligence*, 83(2):347–362, 1996.
2. H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In *Proceedings of the 15th International Conference on Machine Learning*, pages 55–63, 1998.
3. M. Bohanec and I. Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15(3):223–250, 1994.
4. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

5. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
6. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.
7. L. De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, 2002.
8. D. Demšar, M. Debeljak, C. Lavigne, and S. Džeroski. Modelling pollen dispersal of genetically modified oilseed rape within the field, 2005. Abstract presented at The Annual Meeting of the Ecological Society of America, Montreal, Canada, 7-12 August 2005.
9. D. Demšar, S. Džeroski, P. Henning Krogh, T. Larsen, and J. Struyf. Using multiobjective classification to model communities of soil microarthropods. *Ecological Modelling*, 2005. To appear.
10. S. Džeroski, D. Demšar, and J. Grbović. Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, 13(1):7–17, 2000.
11. S. Džeroski, N. Colbach, and A. Messean. Analysing the effect of field characteristics on gene flow between oilseed rape varieties and volunteers with regression trees, 2005. Submitted to the The Second International Conference on Co-existence between GM and non-GM based agricultural supply chains (GMCC-05). Montpellier, France, 14-15 November 2005.
12. M. Garofalakis, D. Hyun, R. Rastogi, and K. Shim. Building decision trees with constraints. *Data Mining and Knowledge Discovery*, 7(2):187–214, 2003.
13. T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.
14. C. Kampichler, S. Džeroski, and R. Wieland. The application of machine learning techniques to the analysis of soil ecological data bases: Relationships between habitat features and collembola community characteristics. *Soil Biology and Biochemistry*, 32:197–209, 2000.
15. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann, 1993.
16. R. S. Sain and P. S. Carmack. Boosting multi-objective regression trees. *Computing Science and Statistics*, 34:232–241, 2002.
17. I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005. 2nd Edition.