Generating Rule Sets from Model Trees

Geoffrey Holmes, Mark Hall and Eibe Frank

Department of Computer Science
University of Waikato, New Zealand
{geoff,mhall,eibe}@cs.waikato.ac.nz
Ph. +64 7 838-4405

Abstract. Model trees—decision trees with linear models at the leaf nodes—have recently emerged as an accurate method for numeric prediction that produces understandable models. However, it is known that decision lists—ordered sets of If-Then rules—have the potential to be more compact and therefore more understandable than their tree counterparts.

We present an algorithm for inducing simple, accurate decision lists from model trees. Model trees are built repeatedly and the best rule is selected at each iteration. This method produces rule sets that are as accurate but smaller than the model tree constructed from the entire dataset. Experimental results for various heuristics which attempt to find a compromise between rule accuracy and rule coverage are reported. We show that our method produces comparably accurate and smaller rule sets than the commercial state-of-the-art rule learning system Cubist.

1 Introduction

Recent work in knowledge discovery on time series data (DLRS98), indicates that the scope of application of machine learning algorithms has gone beyond the relatively "straightforward" classification of nominal attributes in data. These applications are important to business, medicine, engineering and the social sciences, particularly in areas concerned with understanding data from sensors (KP98).

Of equal importance, particularly for business applications is the prediction, and consequent interpretation, of numeric values. For example, the 1998 KDD-Cup concentrated on predicting whether or not someone would donate to a charity. It is arguable that the charity would like to know both the amount someone is likely to donate and the factors which determine this donation from historical data so that they can produce a more effective marketing campaign.

Predicting numeric values usually involves complicated regression formulae. However, in machine learning it is important to present results that can be easily interpreted. Decision lists presented in the If-Then rule format are one of the most popular description languages used in machine learning. They have the potential to be more compact and more predictive than their tree counterparts (WI95). In any application, the desired outcome is a small descriptive model which has strong predictive capability. It has to be small to be interpretable and understandable, and it has to be accurate so that generalization capabilities can be attributed to the model.

In this paper we present a procedure for generating rules from model trees (Qui92), based on the basic strategy of the PART algorithm (FW98), that produces accurate and compact rule sets. Section 2 discusses the motivation for PART and alternative approaches to continuous class prediction. Section 3 describes the adaptation of PART to model trees. Section 4 presents an experimental evaluation on standard datasets. We compare the accuracy and size of the rule sets of our procedure with model trees and the rule-based regression learner Cubist[1], the commercial successor of M5 (Qui92). Section 5 concludes with a discussion of the results and areas for further research on this problem.

2 Related Work

Rule learning for classification systems normally operates in two-stages. Rules are induced initially and then refined at a later stage using a complex global optimization procedure. This is usually accomplished in one of

[1] A test version of Cubist is available from http://www.rulequest.com

two ways; either by generating a decision tree, mapping the tree to a rule set and then refining the rule set based on boundary considerations of the coverage achieved by each rule, or by employing the separate-and-conquer paradigm. As with decision trees this strategy usually employs a rule optimization stage.

Frank and Witten (1998) combined these two approaches in an algorithm called PART (for partial decision trees) in order to circumvent problems that can arise with both these techniques. Rules induced from decision trees are computationally expensive and this expense can grow alarmingly in the presence of noise (Coh95), while separate-and-conquer methods suffer from a form of overpruning called "hasty generalization" (FW98).

PART works by building a rule and removing its cover, as in the separate-and-conquer technique, repeatedly until all the instances are covered. The rule construction stage differs from standard separate-and-conquer methods because a partial pruned decision tree is built for a set of instances, the leaf with the largest coverage is made into a rule, and the tree is discarded. The pruned decision tree helps to avoid the overpruning problem of methods that immediately prune an individual rule after construction. Also, the expensive rule optimization stages associated with decision tree rule learning are not performed. Results on standard data sets show smaller rule sizes with no loss in accuracy when compared with the decision tree learner C4.5 (Qui93) and greater accuracy when compared with the separate-and-conquer rule learner RIPPER (Coh95). In this paper we adapt the basic procedure of PART to continuous class prediction to examine whether similar results can be obtained, namely smaller rule sets with no loss in accuracy.

Although the literature is light in the area of rule-based continuous class prediction, a taxonomy can be found. A first split can be made on whether a technique generates interpretable results. Those that do not include neural networks, and various statistical approaches at dealing with non-linear regression, such as MARS (Fre91) and projection pursuit (FS81). Those that do produce readable output are further split on whether or not they are based on the two major paradigms for rule generation—rule sets represented as regression or model trees, and the separate-and-conquer rule-learning approach. Examples from the regression tree family include: CART (BOS84), RETIS (Kar92) and M5 (Qui92). Separate-and-conquer methods include a system that maps a regression problem into a classification problem (WI95), and a propositional learning system (Tor95).

3 Generating Rules From Model Trees

Model trees (Qui92) are a technique for dealing with continuous class problems that provide a structural representation of the data and a piecewise linear fit of the class. They have a conventional decision tree structure but use linear functions at the leaves instead of discrete class labels. The first implementation of model trees, M5, was rather abstractly defined in (Qui92) and the idea was reconstructed and improved in a system called M5′ (WW97). Like conventional decision tree learners, M5′ builds a tree by splitting the data based on the values of predictive attributes. Instead of selecting attributes by an information theoretic metric, M5′ chooses attributes that minimise intra-subset variation in the class values of instances that go down each branch.

After constructing a tree, M5′ computes a linear model for each node; the tree is then pruned back from the leaves, so long as the expected estimated error decreases. The expected error for each node is calculated by averaging the absolute difference between the predicted value and the actual class value of each training example that reaches the node. To compensate for an optimistic expected error from the training data, this average is multiplied by a factor that takes into account the number of training examples that reach the node and the number of parameters in the model that represent the class value at that node.

This process of tree construction can lead to sharp discontinuities occurring between adjacent linear models at the leaves of the pruned tree. A procedure called smoothing is used to compensate for these differences. The smoothing procedure computes a prediction using the leaf model, and then passes that value along the path back to the root, smoothing it at each node by combining it with the value predicted by the linear model for that node.

3.1 Rule Generation

The method for generating rules from model trees, which we call M5′Rules, is straightforward and works as follows: a tree learner (in this case model trees) is applied to the full training dataset and a pruned tree is learned. Next, the best leaf (according to some heuristic) is made into a rule and the tree is discarded. All instances covered by the rule are removed from the dataset. The process is applied recursively to the remaining instances and terminates when all instances are covered by one or more rules. This is the basic separate-and-conquer strategy for learning rules; however, instead of building a single rule, as it is done usually, we build a full model tree at each stage, and make its "best" leaf into a rule. This avoids potential for over-pruning called hasty generalization (FW98). In contrast to PART, which employs the same strategy for categorical prediction, M5′Rules builds full trees instead of partially explored trees. Building partial trees leads to greater computational efficiency, and does not affect the size and accuracy of the resulting rules.

This paper concentrates on generating rules using unsmoothed linear models. Because the tree from which a rule is generated is discarded at each stage, smoothing for rules would have to be done as a post processing stage after the full set of rules has been produced. This process is more complicated than smoothing model trees—it would involve determining the boundaries between rules and then installing linear models to smooth over them.

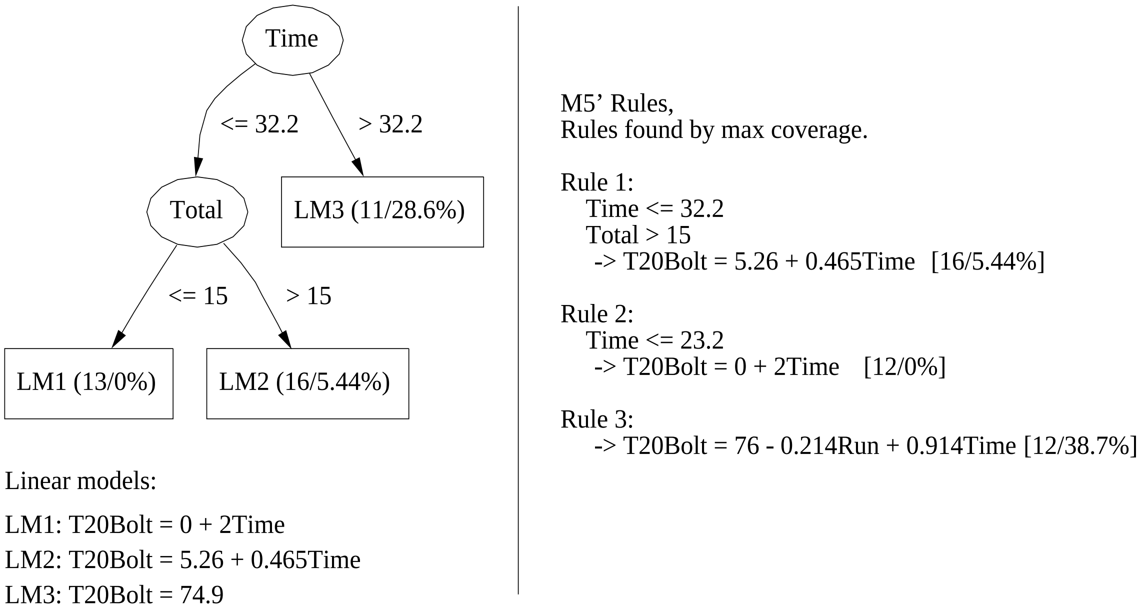

Fig. 1. Model tree and rules for the bolts dataset.

3.2 Rule Selection Heuristics

So far we have described a general approach to extracting rules from trees, applicable to either classification or regression. It remains to determine, at each stage, which leaf in the tree is the best candidate for addition to the rule set. The most obvious approach (FW98) is to choose the leaf which covers the most examples. Figure 1 shows a tree produced by M5′ and the rules generated by M5′Rules using the coverage heuristic for the dataset *bolts* (Sta99). The values at the leaves of the tree and on the consequent of the rules are the coverage and percent root mean squared error respectively for instances that reach those leaves (satisfy those rules).

Note that the first rule will always map directly to one branch of the tree, however, subsequent rules often do not. In Figure 1, Rule 1 and LM2 are identical as are Rule 2 and LM1, however, Rule 3 and LM3 are very different.

We have experimented with three other heuristics, designed to identify accurate rules and to trade off accuracy against coverage. These measures are similar to those used in the separate-and-conquer procedure when evaluating the specialization of one rule from another (Tor95).

The first of these calculates the percent root mean squared error as shown in Equation 1:

$$\%\ \mathrm{RMS} = \frac{\sqrt{\sum_{i=1}^{N_r}(Y_i - y_i)^2/N_r}}{\sqrt{\sum_{i=1}^{N}(Y_i - \overline{Y})^2/N}}, \tag{1}$$

Table 1. Continuous class datasets used in the experiments

Dataset	Instances	Missing	Numeric	Nominal
values (%)	attributes	attributes		
auto93	93	0.7	16	6
autoHorse	205	1.1	17	8
autoMpg	398	0.2	4	3
autoPrice	159	0.0	15	0
baskball	96	0.0	4	0
bodyfat	252	0.0	14	0
breastTumor	286	0.3	1	8
cholesterol	303	0.1	6	7
cleveland	303	0.1	6	7
cloud	108	0.0	4	2
cpu	209	0.0	6	1
echoMonths	131	7.5	6	3
elusage	55	0.0	1	1
fishcatch	158	6.9	5	2
housing	506	0.0	12	1
hungarian	294	19.0	6	7
lowbwt	189	0.0	2	7
mbagrade	61	0.0	1	1
meta	528	4.3	19	2
pbc	418	15.6	10	8
pharynx	195	0.1	1	10
pollution	60	0.0	15	0
pwLinear	200	0.0	10	0
quake	2178	0.0	3	0
sensory	576	0.0	0	11
servo	167	0.0	0	4
sleep	62	2.4	7	0
strike	625	0.0	5	1
veteran	137	0.0	3	4
vineyard	52	0.0	3	0

where Y_i is the actual class value for example i, y_i is the class value predicted by the linear model at a leaf, N_r is the number of examples covered by leaf, $\overline{Y}$ is the mean of the class values, and N is the total number of examples. In this case, small values of % RMS (less than 1) indicate that the model at a leaf is doing better than simply predicting the mean of the class values.

One potential problem with percent root mean squared error is that it may favour accuracy at the expense of coverage. Equations 2 and 3 show two heuristic measures designed to trade off accuracy against coverage. The first, simply normalises the mean absolute error at a leaf using the number of examples it covers; the second, multiplies the correlation between the predicted and actual class values for instances at a leaf by the number of instances that reach the leaf.

$$\text{MAE / Cover} = \frac{\sum_{i=1}^{N_r} |Y_i - y_i|}{2N_r}, \tag{2}$$

$$\text{CC} \times \text{Cover} = \frac{\sum_{i=1}^{N_r} Y_i y_i}{N_r \sigma_Y \sigma_y} \times N_r. \tag{3}$$

In Equation 3, Y_i and y_i are the actual value and predicted value for instance i expressed as deviations from their respective means.

4 Experimental Results

In order to evaluate the performance of M5′Rules on a diverse set of machine learning problems, experiments were performed using thirty continuous class datasets. The datasets and their properties are listed in Table 1, and can be obtained from the authors upon request. Nineteen of these datasets were used by Kilpatrick and Cameron-Jones (KCJ98), six are from the StatLib repository (Sta99), and the remaining five were collected by Simonoff (Sim96).

Table 2. Experimental results: comparing M5′Rules with M5′. The values are mean absolute error averaged over ten ten-fold cross-validation runs. Results are marked with a ∘ if they show a significant improvement over M5′ unsmoothed, and with a • if they show a significant degradation. Results marked with a √ show where M5′ Rules has produced significantly fewer rules than M5′; those marked with a × show where M5′ Rules has produced significantly more rules than M5′.

Dataset	M5′ Unsmoothed	M5′R % RMS		M5′R MAE/Cover		M5′R CC×Cover		M5′R Cover	
auto93	3.66±0.2	3.66±0.2		3.66±0.2		3.66±0.2		3.66±0.2	
autoHorse	8.97±0.5	9.44±0.5	√	9.36±0.5	√	9.40±0.5	•√	9.32±0.5	√
autoMpg	2.08±0.0	2.10±0.1	√	2.08±0.1	√	2.08±0.0	√	2.08±0.0	√
autoPrice	1522.96±53.2	1636.90±96.6	•√	1655.50±109.9	•√	1650.81±129.0	√	1637.44±124.7	•√
baskball	0.07±0.0	0.07±0.0		0.07±0.0		0.07±0.0		0.07±0.0	
bodyfat	0.37±0.1	0.40±0.0	×	0.38±0.1	×	0.37±0.1		0.36±0.1	
breastTumor	8.06±0.1	8.06±0.1		8.06±0.1		8.06±0.1		8.06±0.1	
cholesterol	40.98±1.4	40.91±1.4		40.99±1.4		40.77±1.4		40.98±1.4	
cleveland	0.66±0.0	0.65±0.0		0.66±0.0		0.66±0.0		0.66±0.0	
cloud	0.29±0.0	0.28±0.0		0.28±0.0		0.29±0.0		0.29±0.0	
cpu	13.40±1.2	13.31±1.3		13.33±1.3		13.18±1.5		13.27±1.4	
echoMonths	8.90±0.1	8.90±0.1		8.90±0.1		8.90±0.1		8.90±0.1	
elusage	9.57±0.6	9.57±0.6		9.57±0.6		9.57±0.6		9.57±0.6	
fishcatch	38.70±1.6	39.47±1.5	√	41.55±1.5	•√	38.53±1.9	√	38.61±1.8	√
housing	2.75±0.2	2.64±0.1	√	2.71±0.2	√	2.71±0.1	√	2.77±0.1	√
hungarian	0.28±0.0	0.28±0.0	√	0.28±0.0	√	0.28±0.0		0.28±0.0	
lowbwt	370.93±6.7	370.93±6.7		370.93±6.7		370.93±6.7		370.57±6.4	
mbagrade	0.23±0.0	0.23±0.0		0.23±0.0		0.23±0.0		0.23±0.0	
meta	115.73±13.3	123.82±24.5	√	135.33±22.8	√	131.29±12.8	√	127.72±25.1	√
pbc	716.13±12.8	715.67±12.2		716.13±12.8		716.13±12.8		716.13±12.8	
pharynx	352.85±5.8	352.66±6.1		351.82±7.5		352.76±7.9		353.24±5.9	
pollution	35.15±2.0	35.15±2.0		35.03±2.1		34.99±2.1		35.03±2.1	
pwLinear	1.15±0.0	1.15±0.0		1.15±0.0		1.15±0.0		1.15±0.0	
quake	0.15±0.0	0.15±0.0	√	0.15±0.0		0.15±0.0	•√	0.15±0.0	
sensory	0.58±0.0	0.58±0.0	√	0.58±0.0	√	0.59±0.0	√	0.58±0.0	√
servo	0.31±0.0	0.32±0.0	√	0.32±0.0	√	0.32±0.0	√	0.32±0.0	√
sleep	2.56±0.1	2.56±0.1		2.56±0.1		2.56±0.1		2.56±0.1	
strike	215.87±7.1	231.12±9.7	•√	220.14±4.9	√	222.95±6.4	√	214.91±7.4	√
veteran	92.06±4.3	90.48±4.8		90.49±4.8		90.91±4.5		91.52±4.7	
vineyard	2.48±0.1	2.51±0.2	√	2.51±0.1	√	2.43±0.1	√	2.51±0.1	√

∘,•(√ ,×) statistically significant improvement or degradation

As well as M5′Rules using each of the rule-selection heuristics described above, M5′ (with unsmoothed linear models) and the commercial regression rule learning system Cubist were run on all the datasets. Default parameter settings were used for all algorithms. The mean absolute error, averaged over ten ten-fold cross-validation runs and the standard deviations of these ten error estimates were calculated for each algorithm-dataset combination. The same folds were used for each algorithm.

Table 2 compares the results for M5′Rules with those for M5′ unsmoothed. Results for M5′Rules are marked with a ∘ if they show a significant improvement over the corresponding results for M5′, and with a • if they show a significant degradation. Results marked with a √ show where M5′Rules has produced significantly fewer rules than M5′; those marked with a × show where M5′Rules has produced significantly more rules than M5′. Results are considered "significant" if the difference is statistically significant at the 1% level according to a paired two-sided t-test, each pair of data points consisting of the estimates obtained in one ten-fold cross-validation run for the two learning algorithms being compared.

From Table 2 it can be seen that all four heuristic methods for choosing rules give results that are rarely significantly worse than M5′. In fact, choosing rules simply by coverage gives an excellent result—accuracy on only one dataset is significantly degraded. Each of the remaining three heuristics degrade accuracy on two datasets.

As well as accuracy, the size of the rule set is important because it has a strong influence on comprehensibility. Correlation times coverage and plain coverage never result in a larger rule set than M5′. These two heuristics reduce the size of the rule set on eleven, and ten datasets respectively. Both percent root mean squared error and mean absolute error over cover increase the size of the rule set on one dataset, while decreasing size on twelve and eleven datasets respectively.

Table 3 compares accuracy for M5′Rules with those for Cubist. Table 4 and Table 5 compare the average number of rules produced and average number of conditions per rule set respectively. The results for both

Table 3. Experimental results: comparing accuracy of M5′Rules with Cubist. The values are mean absolute error averaged over ten ten-fold cross-validation runs. Results are marked with a ∘ if they show a significant improvement over Cubist, and with a • if they show a significant degradation (the precision of the results shown in the table is such that some appear identical but in fact are significantly different, e.g. baskball).

Dataset	Cubist	M5′R % RMS		M5′R MAE/Cover		M5′R CC×Cover		M5′R Cover	
auto93	4.07±0.2	3.66±0.2	∘	3.66±0.2	∘	3.66±0.2	∘	3.66±0.2	∘
autoHorse	9.27±0.5	9.44±0.5		9.36±0.5		9.40±0.5		9.32±0.5	
autoMpg	2.24±0.1	2.10±0.1	∘	2.08±0.1	∘	2.08±0.0	∘	2.08±0.0	∘
autoPrice	1639.12±63.8	1636.90±96.6		1655.50±109.9		1650.81±129.0		1637.4±124.74	
baskball	0.07±0.0	0.07±0.0	∘	0.07±0.0	∘	0.07±0.0	∘	0.07±0.0	∘
bodyfat	0.33±0.0	0.40±0.0	•	0.38±0.1		0.37±0.1		0.36±0.1	
breastTumor	8.97±0.1	8.06±0.1	∘	8.06±0.1	∘	8.06±0.1	∘	8.06±0.1	∘
cholesterol	43.02±1.5	40.91±1.4	∘	40.99±1.4		40.77±1.4	∘	40.98±1.4	
cleveland	0.65±0.0	0.65±0.0		0.66±0.0		0.66±0.0		0.66±0.0	
cloud	0.26±0.0	0.28±0.0	•	0.28±0.0	•	0.29±0.0	•	0.29±0.0	•
cpu	10.96±1.1	13.31±1.3	•	13.33±1.3	•	13.18±1.5	•	13.27±1.4	•
echoMonths	9.41±0.2	8.90±0.1	∘	8.90±0.1	∘	8.90±0.1	∘	8.90±0.1	∘
elusage	7.59±0.2	9.57±0.6	•	9.57±0.6	•	9.57±0.6	•	9.57±0.6	•
fishcatch	41.66±0.8	39.47±1.5	∘	41.55±1.5		38.53±1.9	∘	38.61±1.8	∘
housing	2.37±0.1	2.64±0.1	•	2.71±0.2	•	2.71±0.1	•	2.77±0.1	•
hungarian	0.23±0.0	0.28±0.0	•	0.28±0.0	•	0.28±0.0	•	0.28±0.0	•
lowbwt	340.29±7.2	370.93±6.7	•	370.93±6.7	•	370.93±6.7	•	370.57±6.4	•
mbagrade	0.23±0.0	0.23±0.0		0.23±0.0		0.23±0.0		0.23±0.0	
meta	107.26±9.8	123.82±24.5		135.33±22.8	•	131.29±12.8	•	127.72±25.1	•
pbc	774.76±16.3	715.67±12.2	∘	716.13±12.8	∘	716.13±12.8	∘	716.13±12.8	∘
pharynx	448.93±2.7	352.66±6.1	∘	351.82±7.5	∘	352.76±7.9	∘	353.24±5.9	∘
pollution	34.68±2.4	35.15±2.0		35.03±2.1		34.99±2.1		35.03±2.1	
pwLinear	1.14±0.0	1.15±0.0		1.15±0.0		1.15±0.0		1.15±0.0	
quake	0.15±0.0	0.15±0.0		0.15±0.0		0.15±0.0		0.15±0.0	
sensory	0.61±0.0	0.58±0.0	∘	0.58±0.0	∘	0.59±0.0	∘	0.58±0.0	∘
servo	0.38±0.0	0.32±0.0	∘	0.32±0.0	∘	0.32±0.0	∘	0.32±0.0	∘
sleep	2.84±0.2	2.56±0.1	∘	2.56±0.1	∘	2.56±0.1	∘	2.56±0.1	∘
strike	201.31±5.0	231.12±9.7	•	220.14±4.9	•	222.95±6.4	•	214.91±7.4	•
veteran	88.76±5.5	90.48±4.8		90.49±4.8		90.91±4.5		91.52±4.7	
vineyard	2.28±0.1	2.51±0.2	•	2.51±0.1	•	2.43±0.1	•	2.51±0.1	•

∘,• statistically significant improvement or degradation

accuracy and number of rules—as well as Table 2—are summarised for quick comparison in Table 6. Each entry in Table 6 has two values: the first indicates the number of datasets for which the method associated with its column is significantly more accurate than the method associated with its row; the second (in braces) indicates the number of datasets for which the method associated with its column produces significantly smaller rule sets than the method associated with its row.

From the first row and the first column of Table 6 it can be noted that all four versions of M5′Rules—as well as (perhaps surprisingly) M5′—outperform Cubist on more datasets than they are outperformed by Cubist. % RMS and CC × Cover are more accurate than Cubist on twelve datasets, Cover on eleven datasets and MAE / Cover on ten datasets. By comparison, Cubist does better than all four M5′Rules variants on nine datasets, eight of which are the same for all variants. When rule set sizes are compared, it can be seen that M5′Rules produces smaller rule sets than Cubist more often than not. % RMS and CC × Cover produce smaller rule sets than Cubist on twenty-three datasets, and MAE / Cover and Cover produce smaller ones on twenty-two datasets. Cubist, on the other hand, produces smaller rule sets than all variants of M5′Rules on only six datasets. From Table 4, it can be seen that in many cases M5′Rules produces far fewer rules than Cubist. For example, on the sensory dataset Cubist produces just over forty-five rules, while M5′Rules is more accurate with approximately four rules. Furthermore, from Table 5 it can be seen that M5′Rules generates fewer conditions per rule set than Cubist—it is significantly better on twenty-four and worse on at most five. For some datasets (sensory, pbc, breastTumor, cholesterol) the differences are dramatic.

5 Conclusion

We have presented an algorithm for generating rules for numeric prediction by applying the separate-and-conquer technique to generate a sequence of model trees, reading one rule off each of the trees. The algorithm

Table 4. Experimental results: number of rules produced by M5′Rules compared with number of rules produced by Cubist

Dataset	Cubist	M5′R % RMS	M5′R MAE/Cover	M5′R CC×Cover	M5′R Cover
auto93	2.92±0.2	1.08±0.1 ○	1.08±0.1 ○	1.08±0.1 ○	1.08±0.1 ○
autoHorse	5.31±0.3	2.12±0.6 ○	2.20±0.6 ○	2.83±0.7 ○	2.79±0.5 ○
autoMpg	6.21±0.5	3.41±0.4 ○	3.87±0.4 ○	3.94±0.4 ○	3.91±0.4 ○
autoPrice	3.28±0.2	4.88±0.5 ●	4.70±0.4 ●	4.75±0.5 ●	4.24±0.4 ●
baskball	5.17±0.2	1.00±0.0 ○	1.00±0.0 ○	1.00±0.0 ○	1.00±0.0 ○
bodyfat	1.38±0.2	3.97±0.6 ●	3.73±0.4 ●	3.58±0.4 ●	3.51±0.4 ●
breastTumor	22.19±0.6	1.06±0.1 ○	1.06±0.1 ○	1.06±0.1 ○	1.06±0.1 ○
cholesterol	18.63±0.8	2.33±0.4 ○	2.45±0.5 ○	2.08±0.4 ○	2.46±0.5 ○
cleveland	8.27±0.8	1.07±0.1 ○	1.06±0.1 ○	1.16±0.3 ○	1.18±0.3 ○
cloud	1.09±0.1	2.62±0.5 ●	2.55±0.4 ●	2.60±0.4 ●	2.63±0.4 ●
cpu	2.00±0.0	2.74±0.2 ●	2.72±0.2 ●	2.70±0.2 ●	2.71±0.2 ●
echoMonths	6.24±0.4	1.00±0.0 ○	1.00±0.0 ○	1.00±0.0 ○	1.00±0.0 ○
elusage	2.00±0.0	1.62±0.2 ○	1.62±0.2 ○	1.62±0.2 ○	1.62±0.2 ○
fishcatch	2.00±0.0	3.47±0.3 ●	2.87±0.4 ●	3.63±0.3 ●	3.63±0.3 ●
housing	6.90±0.4	9.61±1.6 ●	8.44±0.8 ●	8.32±0.8 ●	8.57±0.7 ●
hungarian	9.15±0.5	1.56±0.2 ○	1.56±0.2 ○	1.65±0.2 ○	1.69±0.3 ○
lowbwt	6.45±0.2	1.05±0.1 ○	1.05±0.1 ○	1.05±0.1 ○	1.04±0.1 ○
mbagrade	3.57±0.1	1.00±0.0 ○	1.00±0.0 ○	1.00±0.0 ○	1.00±0.0 ○
meta	12.90±0.3	5.40±0.4 ○	5.00±0.4 ○	5.40±0.6 ○	4.66±0.5 ○
pbc	21.63±0.9	1.63±0.1 ○	1.64±0.1 ○	1.65±0.1 ○	1.64±0.1 ○
pharynx	7.96±0.1	2.07±0.5 ○	2.20±0.5 ○	2.16±0.4 ○	2.18±0.4 ○
pollution	1.53±0.2	1.22±0.1 ○	1.19±0.1 ○	1.20±0.1 ○	1.21±0.1 ○
pwLinear	2.00±0.0	2.00±0.0	2.00±0.0	2.00±0.0	2.00±0.0
quake	4.53±0.9	2.45±0.3 ○	3.61±0.4	1.98±0.2 ○	3.66±0.4
sensory	45.31±1.1	4.19±0.5 ○	4.04±0.4 ○	3.97±0.3 ○	4.13±0.5 ○
servo	5.77±0.1	5.05±0.3 ○	5.20±0.4 ○	4.18±0.3 ○	4.09±0.3 ○
sleep	2.17±0.2	1.00±0.0 ○	1.00±0.0 ○	1.00±0.0 ○	1.00±0.0 ○
strike	16.65±1.5	4.68±0.9 ○	4.78±0.9 ○	4.95±1.0 ○	4.86±1.1 ○
veteran	6.48±0.6	1.26±0.3 ○	1.27±0.3 ○	1.29±0.2 ○	1.36±0.3 ○
vineyard	2.77±0.1	2.27±0.2 ○	2.07±0.2 ○	2.18±0.2 ○	2.07±0.2 ○

○,● statistically significant improvement or degradation

is straightforward to implement and relatively insensitive to the heuristic used to select competing rules from the tree at each iteration. M5′Rules using the coverage heuristic is significantly worse, in terms of accuracy, on only one (autoPrice) of the thirty bench mark datasets when compared with M5′. In terms of compactness, M5′Rules never produces larger rule sets and produces smaller sets on ten datasets. When compared to the commercial system Cubist, M5′Rules outperforms it on size and is comparable on accuracy. When based on the number of leaves it is more than three times more likely to produce significantly fewer rules. When the number of conditions per rule are used to estimate rule size M5′Rules is eight times more likely to produce rules with fewer conditions than Cubist.

Published results with smoothed trees (WW97) indicate that the smoothing procedure substantially increases the accuracy of predictions. Smoothing cannot be applied to rules in the same way as for trees because the tree containing the relevant adjacent models is discarded at each iteration of the rule generation process. It seems more likely that improvements to M5′Rules will have to be made as a post-processing optimization stage. This is unfortunate because generation of accurate rule sets without global optimization is a compelling aspect of the basic PART procedure, on which M5′Rules is based. However, smoothing usually increases the complexity of the linear models at the leaf nodes, making the resulting predictor more difficult to analyze.

6 Acknowledgements

We would like to thank Gordon Paynter for suggesting the problem and Yong Wang for helpful discussions on model trees.

Table 5. Experimental results: average number of conditions per rule set produced by M5′Rules compared with average number of conditions per rule set produced by Cubist

Dataset	Cubist	M5′R % RMS	M5′R MAE/Cover	M5′R CC×Cover	M5′R Cover
auto93	4.37±0.6	0.08±0.1 ∘	0.08±0.1 ∘	0.09±0.1 ∘	0.09±0.1 ∘
autoHorse	11.60±1.1	4.47±2.2 ∘	3.89±1.5 ∘	3.96±2.2 ∘	3.63±1.3 ∘
autoMpg	14.13±1.6	5.17±1.1 ∘	3.50±0.4 ∘	3.53±0.5 ∘	3.49±0.5 ∘
autoPrice	5.63±0.6	8.61±1.7 •	7.64±1.4 •	7.19±1.0 •	5.98±0.7
baskball	11.75±0.7	0.00±0.0 ∘	0.00±0.0 ∘	0.00±0.0 ∘	0.00±0.0 ∘
bodyfat	0.76±0.5	6.86±1.7 •	4.07±0.8 •	3.00±0.6 •	3.08±0.6 •
breastTumor	81.32±3.9	0.06±0.1 ∘	0.06±0.1 ∘	0.06±0.1 ∘	0.06±0.1 ∘
cholesterol	81.86±4.7	1.78±0.8 ∘	1.48±0.6 ∘	1.58±0.6 ∘	1.47±0.5 ∘
cleveland	27.37±4.3	0.19±0.3 ∘	0.15±0.3 ∘	0.24±0.4 ∘	0.21±0.4 ∘
cloud	0.18±0.1	2.23±0.7 •	1.59±0.4 •	1.79±0.6 •	1.76±0.6 •
cpu	2.00±0.0	1.91±0.2	1.86±0.3	1.82±0.2	1.82±0.2
echoMonths	16.53±1.7	0.00±0.0 ∘	0.00±0.0 ∘	0.00±0.0 ∘	0.00±0.0 ∘
elusage	2.00±0.0	0.64±0.2 ∘	0.63±0.2 ∘	0.62±0.2 ∘	0.62±0.2 ∘
fishcatch	2.00±0.0	4.43±1.1 •	3.66±0.9 •	3.40±0.6 •	3.40±0.6 •
housing	18.28±1.6	30.44±6.9 •	18.28±2.7	16.30±2.6	15.73±2.1
hungarian	33.04±3.3	0.88±0.3 ∘	0.79±0.3 ∘	0.81±0.3 ∘	0.84±0.4 ∘
lowbwt	21.19±0.9	0.06±0.1 ∘	0.05±0.1 ∘	0.06±0.1 ∘	0.06±0.1 ∘
mbagrade	6.61±0.4	0.00±0.0 ∘	0.00±0.0 ∘	0.00±0.0 ∘	0.00±0.0 ∘
meta	25.87±0.6	8.81±1.0 ∘	7.34±0.7 ∘	10.53±1.8 ∘	6.49±0.9 ∘
pbc	100.86±5.6	0.65±0.1 ∘	0.65±0.1 ∘	0.65±0.1 ∘	0.65±0.1 ∘
pharynx	7.96±0.1	2.70±1.5 ∘	1.86±0.9 ∘	1.60±0.6 ∘	1.79±0.7 ∘
pollution	1.18±0.5	0.28±0.2 ∘	0.25±0.1 ∘	0.25±0.1 ∘	0.25±0.1 ∘
pwLinear	2.00±0.0	1.00±0.0 ∘	1.00±0.0 ∘	1.00±0.0 ∘	1.00±0.0 ∘
quake	9.64±3.0	3.04±0.5 ∘	2.66±0.4 ∘	3.62±0.9 ∘	2.69±0.5 ∘
sensory	218.80±6.8	8.88±1.6 ∘	5.43±1.1 ∘	5.70±0.9 ∘	5.40±1.2 ∘
servo	13.33±0.2	7.42±0.9 ∘	7.15±0.9 ∘	5.19±0.6 ∘	5.05±0.5 ∘
sleep	2.38±0.8	0.00±0.0 ∘	0.00±0.0 ∘	0.00±0.0 ∘	0.00±0.0 ∘
strike	46.55±5.2	9.27±2.9 ∘	4.98±1.2 ∘	7.40±2.1 ∘	5.25±1.6 ∘
veteran	19.63±2.5	0.50±0.5 ∘	0.42±0.3 ∘	0.41±0.3 ∘	0.41±0.4 ∘
vineyard	4.13±0.4	1.90±0.2 ∘	1.56±0.2 ∘	1.53±0.2 ∘	1.56±0.2 ∘

∘,• statistically significant improvement or degradation

Table 6. Results of paired t-tests ($p = 0.01$): number indicates how often method in column significantly outperforms method in row; number in braces indicates how often method in column produces significantly fewer rules than method in row.

	Cubist	M5′	% RMS	MAE / Cover	CC × Cover	Cover
Cubist	-	12 {20}	12 {23}	10 {22}	12 {23}	11 {22}
M5′	8 {6}	-	0 {12}	0 {11}	0 {11}	0 {10}
% RMS	9 {6}	2 {1}	-	1 {2}	0 {4}	1 {5}
MAE / Cover	9 {6}	2 {1}	1 {2}	-	1 {4}	1 {2}
CC × Cover	9 {6}	2 {0}	1 {2}	1 {1}	-	1 {2}
Cover	9 {6}	1 {0}	0 {3}	0 {1}	0 {2}	-

Bibliography

[BOS84]L. Breiman, J. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Monterrey, Ca, 1984.

[Coh95]W. W. Cohen. Fast effective rule induction. In *Proc. of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[DLRS98]G. Das, K. I. Lin, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Proc. of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 16–22. AAAI Press, 1998.

[Fre91]J. Freidman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–141, 1991.

[FS81]J. Freidman and W. Stuetzle. Projection pursuit regression. *J. American Statistics Association*, 76:817–823, 1981.

[FW98]E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proc. of the Fifteenth International Conference on Machine Learning*, pages 144–151. Morgan Kaufmann, 1998.

[Kar92]A. Karalic. Employing linear regression in regression tree leaves. In *Proc. of the Tenth European Conference on Artificial Intelligence*, Vienna, Austria, 1992.

[KCJ98]D. Kilpatrick and M. Cameron-Jones. Numeric prediction using instance-based learning with encoding length selection. In Nikola Kasabov, Robert Kozma, Kitty Ko, Robert O'Shea, George Coghill, and Tom Gedeon, editors, *Progress in Connectionist-Based Information Systems*, volume 2, pages 984–987. Springer-Verlag, 1998.

[KP98]E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Proc. of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 239–243. AAAI Press, 1998.

[Qui92]J. R. Quinlan. Learning with continuous classes. In *Proc. of the Fifth Australian Joint Conference on Artificial Intelligence*, pages 343–348, World Scientific, Singapore, 1992.

[Qui93]J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA., 1993.

[Sim96]J. Simonoff. *Smoothing Methods in Statistics.* Springer-Verlag, New York, 1996.

[Sta99]StatLib. Department of Statistics, Carnegie Mellon University, 1999. http://lib.stat.cmu.edu.

[Tor95]L. Torgo. Data fitting with rule-based regression. In J. Zizka and P. Brazdil, editors, *Proc. of the Workshop on Artificial Intelligence Techniques (AIT'95)*, Brno, Czech Republic, 1995.

[WI95]S. Weiss and N. Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403, 1995.

[WW97]Y. Wang and I. H. Witten. Induction of model trees for predicting continuous classes. In *Proc. of the poster papers of the European Conference on Machine Learning*, pages 128–137, Prague, Czech Republic, 1997.