Stepwise Induction of Model Trees

Donato Malerba Annalisa Appice Antonia Bellino
Michelangelo Ceci Domenico Pallotta

Dipartimento di Informatica, Università degli Studi di Bari
via Orabona 4, 70125 Bari, Italy
{malerba, appice, bellino, ceci, pallotta}@di.uniba.it

Abstract. Regression trees are tree-based models used to solve those prediction problems in which the response variable is numeric. They differ from the better-known classification or decision trees only in that they have a numeric value rather than a class label associated with the leaves. Model trees are an extension of regression trees in the sense that they associate leaves with multivariate linear models. In this paper a method for the data-driven construction of model trees is presented, namely the Stepwise Model Tree Induction (SMOTI) method. Its main characteristic is the induction of trees with two types of nodes: regression nodes, which perform only straight-line regression, and splitting nodes, which partition the sample space. In this way, the multivariate linear model associated to each leaf is efficiently built stepwise. SMOTI has been evaluated in an empirical study and compared to other model tree induction systems.

1 Introduction

Many problems encountered in practice involve the prediction of a continuous numeric attribute associated with a case. More formally, given a set of observed data ($\mathbf{x}$, y) $\in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ denotes the feature space spanned by m independent (or predictor) variables x_i (both numerical and categorical), the goal is to predict the dependent (or response) variable Y which is continuous. This problem has been approached in many ways, such as standard regression, neural nets, and regression trees [1]. A *regression tree* approximates a function y=g($\mathbf{x}$) by means of a piecewise *constant* one. *Model trees* generalize the concept of regression trees in the sense that they approximate the function above by a piecewise *linear* function, that is they associate leaves with multivariate linear models. The problem of inducing model trees from a training set has received attention both in statistics [2,9] and in machine learning. Some of the model tree induction systems developed are: M5 [8], RETIS [4], M5' [12], RegTree [5], and HTL [10,11]. All these systems perform a *top-down* induction of models trees (TDIMT) by building the tree structure through recursive partitioning of the training set and by associating leaves with models. During the construction of the tree there are three main problems to be solved: Choosing the best partition of a region of the feature space, determining the leaves of the tree and choosing a model for each leaf. Since an exhaustive exploration of all possible

F. Esposito (Ed.): AI*IA 2001, LNAI 2175, pp. 20-32, 2001.

solutions is not possible in practice, several heuristics have been proposed to solve such problems.

In this paper we present the current state of the art of research on TDIMT and, starting from the strengths and weaknesses of some approaches, we propose a new method, named Stepwise Model Tree Induction (SMOTI), which tries to match the coherence of the heuristic evaluation function, used to choose the best partition, with the type of model associated to the leaves. SMOTI constructs model trees stepwise, by adding, at each step, either a regression node or a splitting node. Regression nodes perform straight-line regression, while splitting nodes partition the sample space. The multivariate linear model associated to each leaf is obtained by composing the effect of regression nodes along the path from the root to the leaf.

The background and motivation of this work is described in the next section, while in Section 3 the method SMOTI is introduced, and its computational complexity is analyzed. Finally, in Section 4 some experimental results on eight different data sets are reported and commented on.

2. Background and Motivation

In tree-structured regression models the partitioning process is guided by a heuristic *evaluation function* that chooses the best split of observations into subgroups. In CART (Classification And Regression Trees), a well-known system for the induction of regression trees [1], the quality of the constructed tree T is measured by the mean square error $R^*(T)$: The lower the $R^*(T)$ the better. A sample estimate of the *mean square error* is:

$$R(T) = \frac{1}{N}\sum_{t\in\widetilde{T}}\sum_{x_i\in t}(y_i - \bar{y}(t))^2$$

where N is the number of training examples ($\mathbf{x}_i$, y_i), $\widetilde{T}$ is the set of leaves of the tree, and $\bar{y}(t)$ is the sample mean of the response variable, computed on the observations in the node t. In other words, $R(T)$ is the sum of the resubstitution estimates of risk $R(t)$ at the leaves of the tree. By denoting with $s^2(t)$ the sample variance of the response variable at a node t, $R(T)$ can be rewritten as:

$$R(T) = \sum_{t\in\widetilde{T}} R(t) = \sum_{t\in\widetilde{T}} \frac{N(t)}{N} s^2(t) = \sum_{t\in\widetilde{T}} p(t)s^2(t)$$

where $N(t)$ is the number of observations in the node t and $p(t)$ is the probability that a training case reaches the leaf t. When the observations in a leaf t are sub-divided into two groups, we obtain a new tree T', where t is an internal node with two children, say, t_L and t_R. Different splits generate distinct trees T', and the choice of the best split is made on the grounds of the corresponding $R(T')$. More precisely, the minimization of $R(T')$ is equivalent to minimizing $p(t_L)s^2(t_L)+ p(t_R)s^2(t_R)$, the contribution to $R(T')$ given by the split.

This heuristic criterion conceived for a regression tree learning problem has also been used in some TDIMT systems, such as HTL. In his system M5, Quinlan adopts a similar criterion, using the sample standard deviation $s(t)$ instead of the sample variance $s^2(t)$. The evaluation function used by AID [6] is *Fisher's correlation coefficient*, η^2:

$$\eta^2 = \frac{N(t_L)(\bar{y}(t_L)-\bar{y}(t))^2 + N(t_R)(\bar{y}(t_R)-\bar{y}(t))^2}{\sum_{j=1}^{N}(y_j-\bar{y}(t))^2}$$

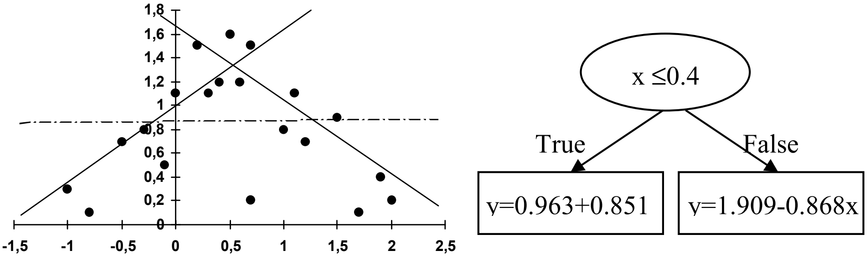
where $\bar{y}(t_L)$ ($\bar{y}(t_R)$) is the sample mean of the dependent variable computed on the set of the $N(t_L)$ ($N(t_R)$) cases falling in the left (right) child node. Briefly, the numerator is the deviance between two groups (left and right), while the denominator is the total deviance. This coefficient ranges between 0 and 1; when η^2 is close to 1, there is no variance *within* the groups. AID chooses the partition that maximizes η^2. Actually, the maximization of the deviance between two groups and the minimization of $p(t_L)s^2(t_L)+p(t_R)s^2(t_R)$ lead to the same partitioning.

The problem with these evaluation functions is that they do not take into account the models associated with the leaves of the tree. In principle, the optimal split should be chosen on the basis of the fit of each *model* to the data. In practice, many TDIMT systems choose the optimal split on the basis of the spread of observations with respect to the *sample mean*. However, a model associated to a leaf is generally more sophisticated than the sample mean. Therefore, *the evaluation function is incoherent with respect to the model tree being built.*

To illustrate the problem, let us consider the following dataset with twenty cases and only one independent variable:

x	-1.0	-0.8	-0.7	-0.5	-0.3	-0.1	0.0	0.2	0.3	0.4	0.5	0.6	0.7	1.0	1.1	1.2	1.5	1.7	1.9	2.0
y	0.3	0.1	0.2	0.7	0.8	0.5	1.1	1.5	1.1	1.2	1.6	1.2	1.5	0.8	1.1	0.7	0.9	0.1	0.4	0.2

The scatter plot of the data set is given in Figure 1a; the values of the independent variable range between -1.0 and 2.0. The best model tree is reported in Figure 1b. It is obtained by partitioning the training observations into two subgroups: X ≤ 0.4 and X> 0.4. It shows the flexibility and the power of model trees, since a simple linear regression on the whole data set would give the dashed line in Figure 1a.



a) b)

Fig. 1. A split node t with two straight-line regression models in the leaves.

Nevertheless, neither M5 nor HTL are able to find this model tree. The system Cubist (http://www.rulequest.com), a commercial version of M5, splits the data at x = -0.1 and builds the following models:

$X \leq -0.1$: $Y = 0.78 + 0.175{*}X$

$X > -0.1$: $Y = 1.143 - 0.281{*}X$

The problem illustrated above is caused by the net separation of the *splitting* stage from the *predictive* one. This separation seems to be inherited by regression tree learners, such as AID and CART, but a careful study of the heuristic criteria used in these systems shows that the evaluation functions do take into account the models built in the leaves. Indeed, the models are the sample means which play a role in the computation of $R(T)$ and η^2. However, when we try to use the same heuristic criteria for model tree induction we are rating the effectiveness of a partition with respect to different models from the ones chosen in the subsequent predictive stage.

This problem cannot potentially occur in RETIS, whose heuristic criterion is to minimize $p(t_L)s^2(t_L)+ p(t_R)s^2(t_R)$, where $s^2(t_L)$ ($s^2(t_R)$) is now computed as the mean square error with respect to the regression plane g_L (g_R) found for the left (right) child:

$$s^2(t_L) = \frac{1}{N(t_L)} \sum_{x_i \in t_L} (y_i - g_L(x_i))^2 \qquad \left(s^2(t_R) = \frac{1}{N(t_R)} \sum_{x_i \in t_R} (y_i - g_R(x_i))^2 \right)$$

In practice, for each possible partitioning the best regression planes at leaves are chosen, so that the selection of the optimal partitioning can be based on the result of the prediction stage. This corresponds to a look-ahead strategy with depth one, as in traditional top-down induction of decision/regression trees.

The weakness of the RETIS heuristic evaluation function is its high computational complexity, especially when all independent variables are continuous. For instance, in the case of N training observations, described by m independent continuous variables, the selection of the first split takes time $O(mNlogN)$ to sort all values of the m variables, plus time required to test $(N-1)m$ distinct cut points, at worst. Each test, in turn, requires the computation of two regression planes on the m independent variables. The coefficients of each linear regression function are computed according to the formula $(\mathbf{X'X})^{-1}\mathbf{X'y}$, where $\mathbf{y}$ is the $N(t)$-dimensional vector of values taken by the response variable in node t, while $\mathbf{X}$ is an $N(t)(m+1)$ matrix of observations, plus a column with only 1s [3]. The complexity of the computation of $\mathbf{X'X}$ is $N(t)(m+1)^2$, the complexity of the inversion of an $(m+1)(m+1)$ matrix is $O((m+1)^3)$, and the complexity of the computation of the product of an $(m+1)(m+1)$ matrix with an $N(t)$-dimensional vector is $N(t)(m+1)$. In general, $N(t) > m$, thus the computation of the regression function takes time $O(N(t)(m+1)^2)$. When $N(t_L)$ is small, $N(t_R)$ is almost equal to N. Thus, for at least one of the children $N(t)$ is proportional to N. To sum up, the choice of the first split takes time $O(N(N-1)m(m+1)^2)$, which is cubic in m and square in N.

In order to reduce the computational time, we could adopt a forward stepwise strategy, according to which a multiple linear regression model is built step-by-step. The proposed method is illustrated in the next section.

3. Stepwise Construction of Model Trees

To reduce the computational time a multiple linear regression model is built stepwise, according to a new forward strategy, named SMOTI, which considers *regression steps* and *splitting tests* at the same level. In this way, the development of a tree structure is not only determined by a recursive partitioning procedure, but also by some intermediate prediction functions. This means that that there are two types of nodes in the tree: regression nodes and splitting nodes. Regression nodes perform only straight-line regressions, since a multivariate linear regression model can be built stepwise by regressing Y on one single variable at a time. Regression and splitting nodes pass down observations to their children in two different ways. For a splitting node *t*, only a subgroup of the *N(t)* observations in *t* is passed to each child, and no change is made on the variables. For a regression node *t*, all the observations are passed down to its only child, but the values of the independent variables not included in the model are transformed, to remove the linear effect of those variables already included. Thus, descendants of a regression node will operate on a modified training set. Indeed, according to the statistical theory of linear regression, the incremental construction of a multiple linear regression model is made by removing the linear effect of introduced variables each time a new independent variable is added to the model [3]. For instance, let us consider the problem of building a multiple regression model with two independent variables through a sequence of straight-line regressions:

$$Y=a+bX_1 + cX_2$$

We start regressing Y on X_1, so that the model:

$$Y = a_1+b_1X_1.$$

is built. This fitted equation does not predict Y exactly. By adding the new variable X_2, the prediction might improve. Instead of starting from scratch and building a model with both X_1 and X_2, we can build a linear model for X_2 given X_1:

$$X_2 = a_2+b_2X_1$$

then compute the residuals on X_2:

$$X'_2 = X_2 - (a_2+b_2X_1)$$

and finally regress Y on X'_2 alone:

$$Y = a_3 + b_3X'_2.$$

By substituting the equation of X'_2 in the last equation we have:

$$Y = a_3 + b_3X_2 -a_2b_3-b_2b_3X_1.$$

It can be proven that this last model coincides with the first model built, that is $a=a_3-a_2b_3$, $b=-b_2b_3$ and $c=b_3$. This explains why SMOTI removes the linear effect of variables already included in the model (X_1) from variables to be selected for the next regression step (X_2).

In SMOTI the validity of either a regression step on a variable X_i or a splitting test on the same variable is based on two distinct evaluation measures, $\pi(X_i,Y)$ and $\sigma(X_i,Y)$ respectively. The variable X_i is of a continuous type in the former case, and of any type in the latter case. Both $\pi(X_i,Y)$ and $\sigma(X_i,Y)$ are mean square errors, therefore they can be actually compared to choose between three different possibilities:

1. growing the model tree by adding a regression node *t*;
2. growing the model tree by adding a splitting node *t*;
3. stop growing the tree at node *t*.

As pointed out in Section 2, the evaluation measure $\sigma(X_i,Y)$ should be coherently defined on the basis of the multivariate linear model to be associated with each leaf. In the case of SMOTI it is sufficient to consider a straight-line regression associated to each leaf t_R (t_L), since regression nodes along the path from the root to t_R (t_L) already partially define a multivariate regression model (see Figure 2).

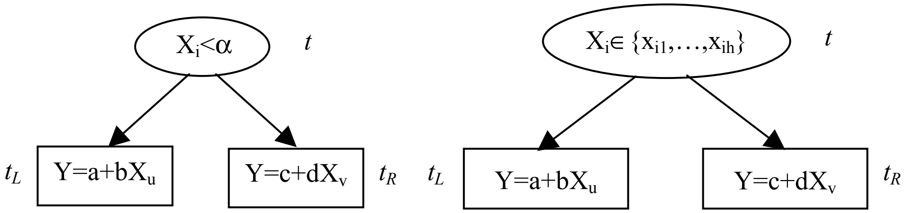


Fig. 2. A split node *t* with two straight-line regression models in the leaves. The variable is continuous in the left tree and discrete in the right tree.

If X_i is continuous and α is a threshold value for X_i then $\sigma(X_i,Y)$ is defined as:

$$\sigma(X_i,Y)=\frac{N(t_L)}{N(t)}R(t_L)+\frac{N(t_R)}{N(t)}R(t_R)$$

where $N(t)$ is the number of cases reaching t, $N(t_L)$ ($N(t_R)$) is the number of cases passed down to the left (right) child, and $R(t_L)$ ($R(t_R)$) is the resubstitution error of the left (right) child, computed as follows:

$$R(t_L)=\sqrt{\frac{1}{N(t_L)}\sum_{j=1}^{N(t_L)}(y_j-\hat{y}_j)^2}\qquad\left(R(t_R)=\sqrt{\frac{1}{N(t_R)}\sum_{j=1}^{N(t_R)}(y_j-\hat{y}_j)^2}\right).$$

The estimate:

$$\hat{y}_j=a_0+\sum_{s=1}^{m}a_s x_s$$

is computed by combining the straight-line regression associated to the leaf t_L (t_R) with all univariate regression lines associated to regression nodes along the path from the root to t_L (t_R).

If X_i is discrete, SMOTI partitions attribute values into two sets, so that binary trees are always built. Partitioning is based on the same criterion applied in CART [1, pp. 247], which reduces the search for the best subset of categories from 2^{k-1} to k-1, where k is the number of distinct values for X_i. More precisely, if $S_{X_i}=\{x_{i_1},x_{i_2},\ldots,x_{i_k}\}$ is the set of distinct values of X_i, S_{X_i} is sorted according to the average over all y_i in t. The best split is in the form: $X_i \in \{x_{i_1},\ldots,x_{i_h}\}$, h= 1, ..., k-1. For all k-1 possible splits, the evaluation measure $\sigma(X_i,Y)$ is computed as in the case of continuous variables.

The evaluation of the effectiveness of a regression step $Y=a+bX_i$ at node t cannot be naïvely based on the resubstitution error $R(t)$:

$$R(t)=\sqrt{\frac{1}{N(t)}\sum_{j=1}^{N(t)}(y_j-\hat{y}_j)^2}$$

where the estimator $\hat{y}_i$ is computed by combining the straight-line regression associated to t with all univariate regression lines associated to regression nodes along the path from the root to t. This would result in values of $\pi(X_i,Y)$ less than or equal to values of $\sigma(X_i,Y)$ for some splitting test involving X_i. Indeed, the splitting test "looks-ahead" to the best multivariate linear regressions after the split on X_i is performed, while the regression step does not. A fairer comparison would be growing the tree at a further level in order to base the computation of $\pi(X_i,Y)$ on the best multivariate linear regressions after the regression step on X_i is performed (see Figure 3).

Let t' be the child of the regression node t, and suppose that it performs a splitting test. The best splitting test in t' can be chosen on the basis of $\sigma(X_j,Y)$ for all possible variables X_j, as indicated above. Then $\pi(X_i,Y)$ can be defined as follows:

$\pi(X_i,Y) = min\ \{\ R(t),\ \sigma(X_j,Y)\ for\ all\ possible\ variables\ X_j\ \}$.

Having defined both $\pi(X_i,Y)$ and $\sigma(X_i,Y)$, the criterion for selecting the best node is fully characterized as well. A weight w (1-w) is associated to splitting (regression) nodes, so as to express the user preference for model trees with splitting tests (regression steps). Therefore, SMOTI actually compares the weighted values $w\sigma(X_i,Y)$ and (1-w)$\pi(X_i,Y)$ while selecting a node. At each step of the model tree induction process, SMOTI chooses the apparently most promising node according to a greedy strategy. A continuous variable selected for a regression step is eliminated from further consideration, so that it can appear only once in a regression node along a path from the root to a leaf.

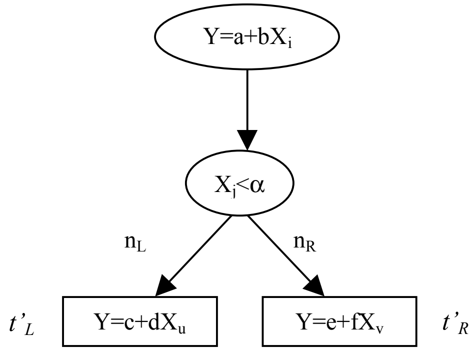


Fig. 3. Evaluation of a regression step at node t, based on the best splitting test below.

In SMOTI three different stopping criteria are implemented. The first one uses the partial F-test to evaluate the contribution of a new independent variable to the model [3]. The second stopping criterion adopted by SMOTI requires the number of cases in each node to be greater than a minimum value. The third criterion stops the induction process when all continuous variables along the path from the root to the current node are used in regression steps and there are no discrete variables in the training set.

The computational complexity of the model tree induction algorithm is highly dependent on the choice of the best splitting test or regression step for a given node. For regression steps, the worst case complexity is $O(NmlogN)$, where N is the number of examples in the training set and m is the number of independent variables. For splitting tests, the worst case complexity is $O(N+NlogN)$, where the component $NlogN$ is due to the quicksort algorithm. Therefore, the worst case complexity for the selection of any node is $O(Nm^2 logN)$, since there are m independent variables.

It is noteworthy that SMOTI is more efficient than RETIS at building model trees and defines the best partitioning of the feature space coherently with respect to the model tree being built.

4. Observations on Experimental Results

SMOTI has been implemented as a module of a knowledge discovery system and has been empirically evaluated on six datasets are taken from either the UCI Machine Learning Repository (URL: http://www.ics.uci.edu/~mlearn/MLRepository.html) or the site of the system HTL (URL: http://www.ncc.up.pt/~ltorgo/Regression/DataSets.html). They are listed in Table 1 and they all have a continuous variable to be predicted.

Table 1. Datasets used in the empirical evaluation of SMOTI.

Dataset	*No. cases*	*No. attributes*	*Continuous*	*Discrete*
Abalone	2889	8	7	1
Auto	398	8	5	3
Housing	506	14	14	0
Machine CPU	209	6	0	6
Pyrimidines	74	27	27	0
Price	159	16	15	1

Each dataset is analyzed by means of a 10-fold cross-validation, that is, the dataset is firstly divided into ten blocks of near-equal size and distribution of class values, and then, for every block, SMOTI is trained on the remaining blocks and tested on the hold-out block.

The system performance is evaluated on the basis of both the average resubstitution error and the average number of leaves. For pairwise comparison of methods, the non-parametric Wilcoxon signed rank test is used [7], since the number of folds (or "independent" trials) is relatively low and does not justify the application of parametric tests, such as the t-test. In the Wilcoxon signed rank test, the summations on both positive and negative ranks, namely W+ and W-, are used to determine the winner. In all experiments reported in this empirical study, the significance level α is set to 0.05.

4.1 Effect of Node Weighting

The first experiment aims at investigating the effect of node weighting on the predictive accuracy and complexity of the tree. A weight greater than 0.5 gives preferentiality to splitting tests, while a weight lower than 0.5 favors the selection of regression nodes. In terms of complexity of model trees, this means that the number of leaves in the tree-structure is generally higher when the weight is greater than 0.5. This intuition has been confirmed by experimental results, as can be seen in Figure 4.

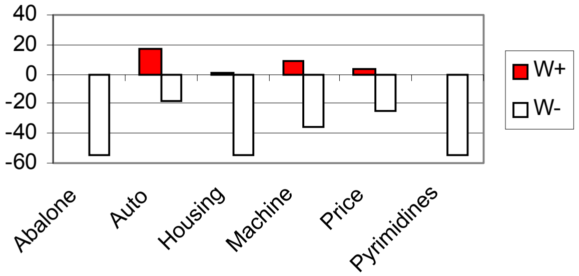


Fig. 4. Summations of positive and negative signed ranks used to compare the number of leaves in model trees built with *w*=0.5 and *w*=0.6. Differences are statistically significant for the databases Abalone, Housing, and Pyrimidine, meaning that in these cases the size of the model trees built with *w*=0.5 is significantly lower than the size of the model trees built with *w*=0.6.

It is noteworthy that, for higher values of the weight, regression nodes are often selected near the leaves of the tree, so that they can give only a local contribution to the approximation of the underlying function with a model tree. On the contrary, for lower values of the weight regression node they tend to be selected at the root, so that they give a global contribution to the approximation of the underlying function. In other words, the weight represents the trade-off between global regression models that span the whole feature space and are built using all training cases and local regression models, which fit fewer data falling in smaller portions of the feature space.

The weighting factor also affects the predictive accuracy of the induced model, as reported in Table 2. In each of the ten trials per dataset, predictive accuracy is estimated by the mean square error, computed on the corresponding validation set. Experimental results show that by increasing the weight, that is favoring the selection of splitting nodes, it is possible to obtain more accurate model trees. Moreover, we also observed that for weight values higher than 0.6 the situation does not change with respect to the case *w*=0.6, while for weight values lower than 0.5 the accuracy is lower than that observed with *w*=0.5. The conclusion is that, in almost all data sets considered, local regression steps are preferred.

Table 2. Results of the Wilcoxon signed rank test on the accuracy of the induced model. The best value is in boldface, while the statistically significant values (p≤α/2) are in italics.

Data set	**00.5 vs 0.52**			**0.5 vs 0.56**			**0.5 vs 0.6**		
	p	*W+*	*W-*	*p*	*W+*	*W-*	*p*	*W+*	*W-*
Abalone	0.083	45	**10**	*0.004*	54	***1***	*0.004*	54	***1***
Auto	0.556	34	**21**	0.492	35	**20**	1.000	28	**27**
Housing	0.492	**20**	35	0.275	39	**16**	0.432	36	**19**
Machine	0.064	46	**9**	0.064	46	**9**	0.064	46	**9**
Price	0.083	45	**10**	0.232	**40**	15	0.432	36	**19**
Pyrimidines	*0.002*	55	***0***	0.106	11	**44**	0.064	46	**9**

4.2 Comparison with Other Systems

SMOTI has also been compared to two other TDMTI systems, namely a trial version of Cubist and M5'. Since the trial version of Cubist worked only for data sets of at most 200 cases and 50 attributes, the comparison with Quinlan's system was possible only for four databases, namely Machine CPU, Price, and Pyrimidines. On the contrary, the comparison with M5' was performed on all collected data sets.

When possible, two statistics were collected for comparison purposes: the average number of leaves and the average MSE of the trees. Actually, Cubist did not report the average MSE, but it was derived from other statistics printed in the report file. On the contrary, it was impossible to derive the average number of leaves from statistics made available to the user. Experimental results are shown in Table 3.

Table 3. Tree size and predictive accuracy for three different systems: SMOTI, M5' and Cubist.

Data Sets	***SMOTI 0.5***		***SMOTI 0.6***		***M5'***		***Cubist***	
	Av. No. Leaves	***Av. MSE***	***Av. No. Leaves***	***Av. MSE***	***Av. No. Leaves***	***Av. MSE***	***Av. No. Leaves***	***Av. MSE***
Abalone	320,2	6,15	373,5	3,44	304,1	*2,62*		
Auto	31,7	7,87	31,9	7,92	24,6	*3,19*		
Housing	28,1	12,1	44,1	8,3	48,4	*3,89*		
Machine	15,6	296,05	17,1	70,11	15,2	59,69		50,07
Price	11,1	3220,51	13,6	2646,17	16,9	2150,74		2512,37
Pyrimidines	1	0,12	6,1	0,08	3,4	0,09		0,09

As pointed out before, SMOTI generates more accurate model trees when splitting tests are favored by setting the weight to 0.6. However, even in the best case, SMOTI does not perform as well as M5' and Cubist with almost all data sets. Some differences between SMOTI 0.6 and M5' are statistically significant and are reported in italics. These results, which are unfavorable to SMOTI, seem to confirm the presence of a common factor to many of the data sets used in the experiments on regression and model trees: no general behavior was noted for the underlying function to be approximated, and it can be better represented as a composition of many definite local behaviors.

4.3 Experiments on Laboratory-Sized Data Sets

In order to better understand the behavior of SMOTI, the system has been tested on laboratory-sized data sets randomly generated for seven different model trees. These model trees were automatically built for learning problems with nine independent variables (five continuous and four discrete), where continuous variables take values in the unit interval [0,1], while discrete variables take values in the set {A,B,C,D,E,F,G}. The model tree building procedure is recursively defined on the maximum depth of the tree to be generated. The choice of adding a regression or a splitting node is random and depends on a parameter $\theta \in [0,100]$: the probability of selecting a splitting node is θ%; conversely, the probability of selecting a regression node is $(100-\theta)$%. Therefore, the returned model trees have a variable number of regression/splitting nodes and leaves, while the depth of the tree is kept under control. In the experiments reported in this paper θ is fixed to 0.5 while the depth is set to 5.

Ten data points were randomly generated for each leaf, so that the size of the data set associated to a model tree depends on the number of leaves in the tree itself. Data points are generated according to the different multivariate linear models associated to the leaves. The error added to each model is distributed normally, with zero mean and variance σ^2, which is kept constant for all leaves. The value of σ^2 set for the experimentation is 0.001, which means that for almost 90% of generated data points the effect of the error is ±0.095, according to Chebyshev's inequality. The effect of the error is not marginal, given that both independent variables and their coefficients range in the unit interval.

Table 4. Results for the model tree built with parameters θ=0.5, depth=5, and σ^2=0.001.

w	*T. depth*	*I. depth*	*T. # regr. nodes*	*I. #regr. nodes*	*T. # split. nodes*	*I. # split. nodes*	*T. # leaves*	*I. # leaves*	*Av. MSE SMOTI*	*Av. MSE. M5'*
0.5	5	9	4	5	4	6	5	7	*0.24*	0.35
0.55		7		1		8		9	0.61	
0.5	5	5	4	2	6	9	7	10	*0.2*	0.36
0.55		5		1		9		10	*0.15*	
0.5	5	8	3	3	7	10	8	11	*0.19*	0.3
0.55		8		1		11		12	*0.17*	
0.5	5	10	2	5	5	9	6	10	0.53	*0.27*
0.55		6		1		9		10	0.32	
0.5	5	9	4	6	7	11	8	12	0.56	*0.24*
0.55		5		1		9		10	0.68	
0.5	5	0	4	0	0	0	1	1	*0.16*	0.29
0.55		0		0		0		1	*0.16*	
0.5	5	12	3	5	5	16	6	17	*0.15*	0.25
0.55		8		2		17		18	*0.16*	

Each dataset was analyzed by means of a 10-fold cross-validation. In order to study the effect of the weight, two different values were considered: *w*=0.5 and *w*=0.55. Experimental results are reported in Table 4. The properties of the original model trees (T. depth, T. number of regression nodes, T. number of splitting nodes, T. number of leaves) are compared to the corresponding properties of the induced tree (denoted by the initial I). The last two columns list the average mean square error reported by SMOTI and M5′. Results show that SMOTI over-partition the feature space, since the number of splitting nodes in the induced trees is always greater than the number of splitting nodes in the theoretical model tree. This is true even in the case of *w*=0.5. No similar regularity can be detected for regression nodes. Interestingly, in many cases SMOTI performs better than M5′ with respect to average MSE, even for the standard parameter setting like *w*=0.5. These results reverse negative results obtained with UCI data sets and confirm that SMOTI works quite well when both global and local behaviors are mixed up in the underlying models.

5. Conclusions

In the paper, a novel method, called stepwise model tree induction (SMOTI), has been presented. The main advantage of SMOTI is that it efficiently generates model trees with multiple regression models in the leaves. Model trees generated by SMOTI include two types of nodes: regression nodes and splitting nodes. A weight associated to the type of node allows the user to express a preference for either local regression or global regression.

Experimental results proved empirically the effect of the weight. A comparison with two other TDMTI systems has been reported for six datasets typically used to test regression tree induction algorithms. A justification of the unfavorable results for SMOTI may be the absence of a global behavior in the underlying model. An empirical comparison with M5′ on laboratory-sized data sets proved that SMOTI can induce more accurate model trees when both global and local behaviors are mixed up in the underlying model. In the future, we plan to investigate the effect of pruning model trees. To date, no study on the simplification techniques for model trees has been presented in the literature. There are several possible approaches, some based on the direct control of tree size, and others based on the extension of the set of tests considered. Both a theoretical and an empirical evaluation of these approaches in terms of accuracy and interpretability would be helpful in practical applications.

Acknowledgments

This work is part of the MURST COFIN-1999 project on “Statistical Models for Classification and Segmentation of Complex Data Structures: Methodologies, Software and Applications.” The authors thank Lynn Rudd for her help in reading the paper and Marcello Lucente for his collaboration in conducting the experiments.

References

1. Breiman L., Friedman J., Olshen R., & Stone J.: *Classification and regression tree*, Wadsworth & Brooks, 1984.
2. Ciampi A.: Generalized regression trees, *Computational Statistics and Data Analysis*, 12, pp. 57-78, 1991.
3. Draper N.R., & Smith H.: *Applied regression analysis*, John Wiley & Sons, 1982.
4. Karalic A.: Linear regression in regression tree leaves, in *Proceedings of ISSEK '92 (International School for Synthesis of Expert Knowledge)*, Bled, Slovenia, 1992.
5. Lanubile A., & Malerba D.: Induction of regression trees with RegTree, in *Book of Short Papers on Classification and Data Analysis*, Pescara, Italy, pp. 253-260, 1997.
6. Morgan J.N., & Sonquist J.A.: Problems in the analysis of survey data, and a proposal, in *American Statistical Association Journal*, pp. 415-434, 1963.
7. Orkin, M., Drogin, R.: *Vital Statistics*, McGraw Hill, New York (1990).
8. Quinlan J. R.: Learning with continuous classes, in Proceedings AI'92, Adams & Sterling (Eds.), World Scientific, pp. 343-348, 1992.
9. Siciliano R., & Mola F.: Modelling for recursive partitioning in variable selection, in *COMPSTAT '94*, Dutter R., & Grossman W. (Eds.), Physica-Verlag, pp. 172-177, 1994.
10. Torgo L.: Kernel Regression Trees, in *Poster Papers of the 9th European Conference on Machine Learning (ECML 97)*, M. van Someren, & G. Widmer (Eds.), Prague, Czech Republic, pp. 118-127, 1997.
11. Torgo L.: Functional Models for Regression Tree Leaves, in *Proceedings of the Fourteenth International Conference (ICML '97)*, D. Fisher (Ed.), Nashville, Tennessee, pp. 385-393, 1997.
12. Wang Y., & Witten I.H.: Inducing Model Trees for Continuous Classes, in *Poster Papers of the 9th European Conference on Machine Learning (ECML 97)*, M. van Someren, & G. Widmer (Eds.), Prague, Czech Republic, pp. 128-137, 1997.