Simplifying Model Trees with Regression and Splitting Nodes

Michelangelo Ceci, Annalisa Appice, Donato Malerba
{ceci, appice, malerba}@di.uniba.it

Abstract. Model trees are tree-based regression models that associate leaves with linear regression models. A new method for the stepwise induction of model trees (SMOTI) has been developed. Its main characteristic is the construction of trees with two types of nodes: regression nodes, which perform only straight-line regression, and splitting nodes, which partition the feature space. In this way, internal regression nodes contribute to the definition of multiple linear models and have a "global" effect, while straight-line regressions at leaves have only "local" effects. In this paper the problem of simplifying model trees with both regression and splitting nodes is faced. In particular two methods, named Reduced Error Pruning (REP) and Reduced Error Grafting (REG), are proposed. They are characterized by the use of an independent pruning set. The effect of the simplification on model trees induced with SMOTI is empirically investigated. Results are in favour of simplified trees in most cases.

1 Introduction

Many applications of machine learning involve the prediction of a continuous numeric attribute associated with a case. More formally, given a set of observed data $(\mathbf{x}, y) \in \mathbf{X} \times Y$, where $\mathbf{X}$ 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 multiple linear models.

Some of the model tree induction systems developed are: M5 [12], RETIS [5], M5' [16], TSIR [6], and HTL [14,15]. Almost all these systems perform a *top-down* induction of models trees (TDIMT) in two stages: the first builds the tree structure through recursive partitioning of the training set, while the second associates leaves with models. This dichotomy has an intrinsic weakness, since partitioning strategy does not consider the models that can be associated to the leaves. RETIS, which operate differently, suffers from efficiency problems [7].

SMOTI (Stepwise Model Tree Induction) is a new TDIMT method that constructs model trees *stepwise*, by adding, at each step, either a *regression node* or a *splitting node*. In this way, a multiple linear model can be associated to each node during tree-building, thus preventing problems related to two-staged induction algorithms. Efficiency is guaranteed by the stepwise construction of the multiple linear models. Moreover, SMOTI potentially solves the problem of

modeling phenomena where some variables have a global effect while others have only a local effect [8].

Similarly to other TDIMT approaches, SMOTI may generate model trees that overfit training data. Almost all TDIMT systems use some simplifying technique to determine which nodes of the tree should be taken as leaves. These techniques are generally derived from those developed for decision trees [3]. In particular, RETIS bases its pruning algorithm on Niblett and Bratko's method [9], extended later by Cestnik & Bratko [2]. M5 uses a pessimistic-error-pruning-like strategy since it compares the error estimates obtained by pruning a node or not. The error estimates are based on the training cases and corrected in order to take into account the complexity of the model in the node. Similarly, in M5' the pruning procedure makes use of an estimate, at each node, of the expected error for the test data. The estimate is the resubstitution error compensated by a factor that takes into account the number of training examples and the number of parameters in the linear model associated to the node [16]. A method à la error-based-pruning is adopted in HTL, where the upper level of a confidence interval of the resubstitution error estimate is taken as the most pessimistic estimate of the error node [15].

In this paper, the a posteriori simplification (or pruning) of model trees induced by SMOTI has been investigated. In particular, after a brief introduction to SMOTI (next Section), a framework for simplifying model trees with regression and splitting nodes is described. This framework is helpful to define two simplification methods and to investigate their theoretical properties (Sections 4 and 5). Finally, experimental results are reported and discussed in Section 6.

2 Stepwise construction of model trees

In SMOTI the top-down induction of models trees is performed by considering *regression steps* and *splitting tests* at the same level. This means that there are two types of nodes in the tree: regression nodes and splitting nodes (Figure 1). The former compute straight-line regression, while the latter partition the feature space. They 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 (left or right). No change is made on training cases. For a regression node *t*, all the observations are passed down to its only child, but the values of the independent numeric variables not included in the multiple linear model associated to *t* are transformed in order 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 model is made by removing the linear effect of introduced variables each time a new independent variable is added to the model.

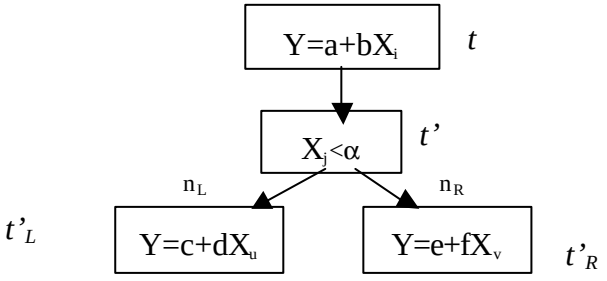


Figure 1. A model tree with both a regression node (t) and a splitting node (t')

This approach is similar to that adopted in TSIR. However, the problem with TSIR is that the multivariate model associated to the leaves cannot be correctly interpreted from a statistical point of view. More precisely, it is not possible to assert that the composition of straight-line models found along a path from the root to a leaf is equivalent to a multiple linear model associated with the leaf itself. This is due to the fact that TSIR does not remove the effect of variables in regression steps. A more detailed explanation of SMOTI and a comparison with other TDIMT methods are reported in [7].

3 A unifying framework for describing simplification methods

Pruning methods have been initially proposed to solve the overfitting problem of induced decision trees. A unifying framework for their descriptions is reported in [4]. In this paper we follow the same idea and develop two methods for the simplification of model trees with both regression nodes and splitting nodes.[1] The first method uses the classical pruning operator extended to regression nodes as well. The second method is based on a new grafting operator that replaces a splitting node with a subtree. To formally define these simplification methods, some notations are introduced. We start with the formal definition of a SMOTI tree, that is a tree with regression and splitting nodes, then we define the pruning and grafting relations, the search spaces, and finally the operators.

A (rooted) tree can be formally defined as a finite set of nodes, $N_T = \{t_0, t_1, \ldots, t_n\}$ and an associated relation $B_T \subseteq N_T \times N_T$ for which the following properties hold:

1. There exists only one node t_0 in N_T, named *root*, such that $\forall \langle t_i,t_j\rangle \in B_T: t_j \neq t_0$;
2. $\forall\, t_j \in N_T$, $t_j \neq t_0$ there exists only one node $t_i \in N_T$ such that $\langle t_i,t_j\rangle \in B_T$.

The set N_T can be partitioned into the set of internal nodes and the set of leaves $\tilde{T}$. Given a set $\mathbf{O}$ of N observations $\mathbf{O}_i$, each of which is described by m+1- dimensional feature vector, $\langle X_1,\ldots,X_m,Y\rangle$ it is possible to build a tree-structured model named *SMOTI tree*. This is a particular tree T in which:

1. each node t is associated with a subset of $\mathbf{O}$, $\mathbf{O}(t)$, possibly modified by removing the linear effect of the variables added to the model;
2. the root t_0 is associated with $\mathbf{O}$ itself;
3. every edge $\langle t_i,t_j\rangle \in B_T$ is labelled with $L_T(\langle t_i,t_j\rangle)$;

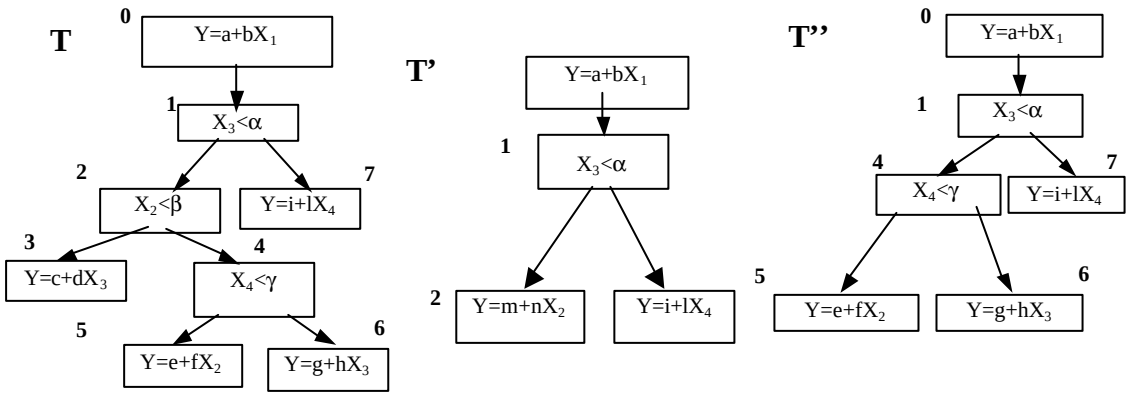


Figure 2. The model tree T' is obtained by pruning T in node 4, while T" is obtained by grafting the subtree rooted in node 4 onto the place of node 2.

[1] No simplification method was proposed in TSIR, the only other system that induces trees with two types of nodes.

$L_T(<t_i,t_j>)$ can be:

1. a straight-line regression function $Y=a+bX_k$ (*regression label*);
2. a test on a numeric variable like $X_k \leq \alpha$ ($X_k > \alpha$) (*continuous split label*);
3. a test on a discrete variable like $X_k \in \{x_{k1}, \ldots, x_{kl}\}$ ($X_k \notin \{x_{k1}, \ldots, x_{kl}\}$) (*discrete split label*).

An internal node $t_i \in N_T - \tilde{T}$ is called *regression* (*splitting*) *node* iff there exists an edge $<t_i,t_j> \in B_T$ such that $L_T(<t_i,t_j>)$ is a regression (continuous/discrete split) label. The sets of regression and splitting nodes will be denoted as R_T and S_T, respectively.

Let T denote the set of all possible model trees that SMOTI can build from $\mathbf{O}$. It is possible to define two distinct partial order relations on T, denoted $\leq_P$ and $\leq_G$, which satisfy the properties of reflexivity, antisymmetry and transitivity. Let T and T' be two model trees in T. Then $T' \leq_P T$ iff T' is obtained by T by dropping some branches. Moreover, $T' \leq_G T$ iff T' is obtained by T by dropping some split nodes and then replacing them with one of their subtrees. With reference to Figure 2, $T' \leq_P T$ and $T'' \leq_G T$, while the relations $T'' \leq_P T$ and $T' \leq_G T$ do not hold.

Hence, given a SMOTI tree T, it is possible to define two sets of trees, namely:

$$S_P(T) = \{T' \in T \mid T' \leq_P T\}$$
$$S_G(T) = \{T' \in T \mid T' \leq_G T\}$$

We observe that $S_P(T) \not\subset S_G(T)$ and $S_G(T) \not\subset S_P(T)$, since $T' \leq_P T$ does not imply $T' \leq_G T$ and viceversa.

The *pruning operator* is defined as the function:

$$\pi_T: R_T \cup S_T \rightarrow T$$

that associates each internal node t with the tree $\pi_T(t)$, which has all the nodes of T except the descendants of t. Analogously, the *grafting operator* is defined as the function:

$$\gamma_T: S_T \times N_T \rightarrow T$$

that associates each couple of internal nodes $<t,t'> \in S_T \times N_T$, with the tree $\gamma_T(<t,t'>)$, which has all nodes of T except those in the branch between t and t'. Intuitively, the pruning operator applied to a node of a tree T returns a tree $T' \leq_P T$ while the grafting operator returns a tree $T' \leq_G T$ (see Figure 2).

The problem of simplifying a model tree can be cast as a search in a state space, where states are trees in either $S_P(T)$ or $S_G(T)$, and pruning and grafting are the only operators that can be applied to move from one state to another.

In order to give a precise definition of a simplification method the goal of the search in the state space has to be defined. For this reason, a function f that estimates the goodness of a tree is introduced. It associates each tree in a generic space S(T) with a numerical value, namely:

$$f: S(T) \rightarrow \Re$$

where $\Re$ is the set of real values. The goal of the search is to find the state in S(T) with the highest f value, so that pruning can be cast as a problem of function optimization.

Finally, the way in which the state space is explored also characterizes different simplification methods, which can be formally described by a 4-tuple:

(Space, Operators, Evaluation function, Search strategy)

where the *Space* represents the search space of pruning methods, *Operators* is a set of simplification (pruning or grafting) operators, *Evaluation function* associates each tree the search space with a numerical value and the *search strategy* is the way in which the state space is explored in order to find the optimal state. This framework is used in the next sections to explain the two simplification methods.

4 Reduced Error Pruning.

This method is based on the Reduced Error Pruning (REP) proposed by Quinlan for decision trees [11]. It uses a pruning set to evaluate the goodness of the subtrees of a model tree T. The pruning set is independent of the set of observations used to build the tree T, therefore, the training set must be partitioned into a growing set used to build the tree and a pruning set used to simplify T.

Search is accomplished in the pruning state space, $(S_P(T), \{\pi_T\})$ by means of the first-better strategy, according to which we move from one state T to a state T' just generated if T' is better than T with respect to the evaluation function f. Differently from the hill-climbing search, there is no generation of all states directly reachable from T in order to select the best one. Moreover, the first better strategy differs from the well-known best first strategy in the storing of only one generated state. Obviously, in this search strategy, the order in which states are generated is of crucial importance. It depends on:

1. The traversal order: pre-order or post-order.
2. The direction of pruning: bottom-up or top-down.

In REP, the traversal is post-order and the direction is bottom-up. The evaluation function f is defined as follows:

$$f(T) = \sum_{t \in \tilde{T}} R(t)$$

where R(t) is the mean square error at leaf t. The search in the space moves from a state T_1 to a state $T_2 \in \pi_{T_1}(S_{T_1} \cup R_{T_1})$ if $f(T_1) \geq f(T_2)$. More precisely the algorithm analyzes the complete tree T and, for each internal node t, it compares the mean square error made on the pruning set when the subtree T_t is kept, with the mean square error made when T_t is pruned and the best regression function is associated to the leaf t. If the simplified tree has a better performance than the original one, it is advisable to prune Tt. This pruning operation is repeated on the simplified tree until further pruning increases the resubstitution error.

The adaptation of the techniques proposed by Quinlan to regression trees requires a reformulation of the function f, and the computation of a straight-line regression model for each node candidate to be pruned.

The following optimality theorem can be proven:

Theorem. Given a model tree T constructed on a set of observations $\mathbf{O}$ and a pruning set $\mathbf{O}'$, the REP version that determines the regression model on $\mathbf{O}$ returns the smallest tree in $S_P(T)$ with the lowest error with respect to $\mathbf{O}'$.

The specification "the REP version that determines the regression model on $\mathbf{O}$" refers to the fact that once a node t has been pruned, the model associated to t is determined on the basis of the same growing set $\mathbf{O}$. Alternatively, it could be determined on the basis of either the pruning set or the whole training set.

```
REG(tree,pruningSet)
begin
        if |pruningSet|=0 then return 0

        if the tree is a leaf then return ResubstitutionError(tree,pruningSet)
        if the root is a splitting node then
                partition pruningSet into pruningSet1 and pruningSet2
                newLeftBranch= leftBranch
                newRightBranch= rightBranch
                sxError=REG(leftBranch, pruningSet1)
                dxError=REG(rightBranch, pruningSet2)
                sxErrorGrafted=REG(newLeftBranch, pruningSet)
                dxErrorGrafted=REG (newRightBranch, pruningSet)
                if sxError+dxError<sxErrorGrafted AND
                  sxError+dxError<dxErrorGrafted then
                                return sxError+dxError
                if sxErrorGrafted>dxErrorGrafted then
                                tree= newRightBranch
                                return dxErrorGrafted
                        else
                                tree= newLeftBranch
                                return sxErrorGrafted
        if the root is a Regression Node then
                remove the effect of the regression from pruningSet into pruningSet1
                sxError=REG (leftBranch, pruningSet1)
                return sxError
end
```

Figure 3. Reduced Error Grafting algorithm.

Finally, the computational complexity of REP is linear in the number of internal nodes, since each node is visited only once to evaluate the opportunity of pruning it.

5 Reduced Error Grafting

The Reduced Error Grafting (REG) is conceptually similar to REP and uses a pruning set to evaluate the goodness of T', a subtree of T. However, the search is performed in the grafting state space, $(S_G(T), \{\gamma_T\})$, according to a first-better strategy with bottom-up post-order traversal. The evaluation function is the same defined for REP.

The search in $S_G(T)$ moves from a state T_1 to a state $T_2 \in \gamma_{T_1}(S_{T_1}, S_{T_1})$ if the inequality $f(T_1) \geq f(T_2)$ holds. More precisely the algorithm operates recursively. It analyzes the complete tree T and, for each split node t, it compares the resubstitution error made on the pruning set when the subtree T_t is kept, with the resubstitution error made on the pruning set when T_t is turned into $REG(T_{t_1})$ or $REG(T_{t_2})$, where t_1 and t_2 are children of t. Sometimes, the simplified tree has a better performance than the original one. In this case, it appears convenient to replace t with its best simplified subtree (left of right). This grafting operation is repeated on the simplified tree until the resubstitution error increases.

This method (see Figure 3) is theoretically advantaged with respect to REP, since it allows the replacement of a subtree by one of its branches. In this way, it is possible to overcome a limit of those simplification strategies that make use of the pruning operator alone. Indeed, if t is a node that should be pruned according to some criterion, while t' is a child of t that should not be pruned

according the same criterion, such simplification strategy either prunes and loses the accurate branch $T_{t'}$ or does not prune at all and keeps the inaccurate branch T_t. On the contrary, REG acts by grafting $T_{t'}$ onto the place of *t*, so saving the good sub-branch and deleting the useless node *t*. The better performance of REG have also been empirically observed, as reported in Section 6.

Similarly to REP, a theorem on the optimality of the tree returned by REG can be proven.

Theorem. Given a model tree T constructed on a set of observations $\mathbf{O}$ and a pruning set $\mathbf{O}'$, the REG version that determines the regression model on $\mathbf{O}$ returns the smallest tree in $S_G(T)$ with the lowest error with respect to $\mathbf{O}'$.

The complexity of REG is $O(N_T \log_2 N_T)$, where $|N_T|$ is the number of nodes in T.

6 Comments on experimental results

The experiment aims at investigating the effect of simplification methods on the predictive accuracy of the model trees. Reduced Error Pruning and Reduced Error Grafting were implemented as a module of KDB2000 and were empirically evaluated on the UCI data sets listed in Table 1.

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. For every trial, the training set is partitioned into growing (70%) and pruning set (30%). SMOTI is trained on the growing set, pruned on the pruning set and tested on the hold-out block (testing set).

Comparison is based on the average mean square error made on the pruning sets. For pairwise comparison of original and simplified trees the non-parametric Wilcoxon signed rank test is used [10], since the number of folds (or "independent" trials) is relatively low and it does not justify the application of parametric tests, such as t-test. In Wilcoxon signed rank test the summations on both positive and negative ranks (W+ and W-) are used to determine the winner.

Results reported in Table 2 show that pruning is generally beneficial. Moreover, the pruning method implemented in M5' pruning method outperforms both REP and REG in most data sets. However, the worst performance of REP and REG can be justified if we consider that M5' pruned a model tree, which was originally more accurate than that pruned by REP and REG because of the full use of the cases in the training set. This result is similar to that reported in [3] for decision trees. Even in that case, it was observed that methods requiring an independent pruning set are at a disadvantage. This is due to the fact that the set of pre-classified cases is limited and, if part of the set is put aside for pruning, it cannot be used to grow a more accurate tree.

Table 2. Tree predictive accuracy for the pruning of two different systems: SMOTI, M5'

Data Sets	SMOTI (w=0.55) Av. MSE	SMOTI REP Av. MSE	SMOTI REG Av. MSE	M5' Av. MSE	M5' Pruning Av. MSE
Abalone	4.483	3.91	2.59	2.62	2.12
Auto	5.22	3.7	5.8	3.19	2.787
Housing	7.522	5.09	5.8	3.89	4.241
Machine	358.929	77.001	83.15	59.69	62.031
Price	5358.15	3564.84	2290.4	2150.74	2356.95
Pyrimidines	0.1125	0.1074	0.1117	0.09	0.034

Furthermore, SMOTI simplification has been empirically evaluated on artificial data sets randomly generated for seven model trees, which where automatically built for learning problems with nine independent variables (five continuous and four discrete). 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. By varying both the number of training cases per leaf (10, 20, 30, 40, 50, 60 items) and the depth of the tree (4, 5, 6, 7, 8, 9), which are the measures of complexity of the target concept, we can study the behaviour of simplification methods and keep other factor under control. 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 $(100-\theta)$%.

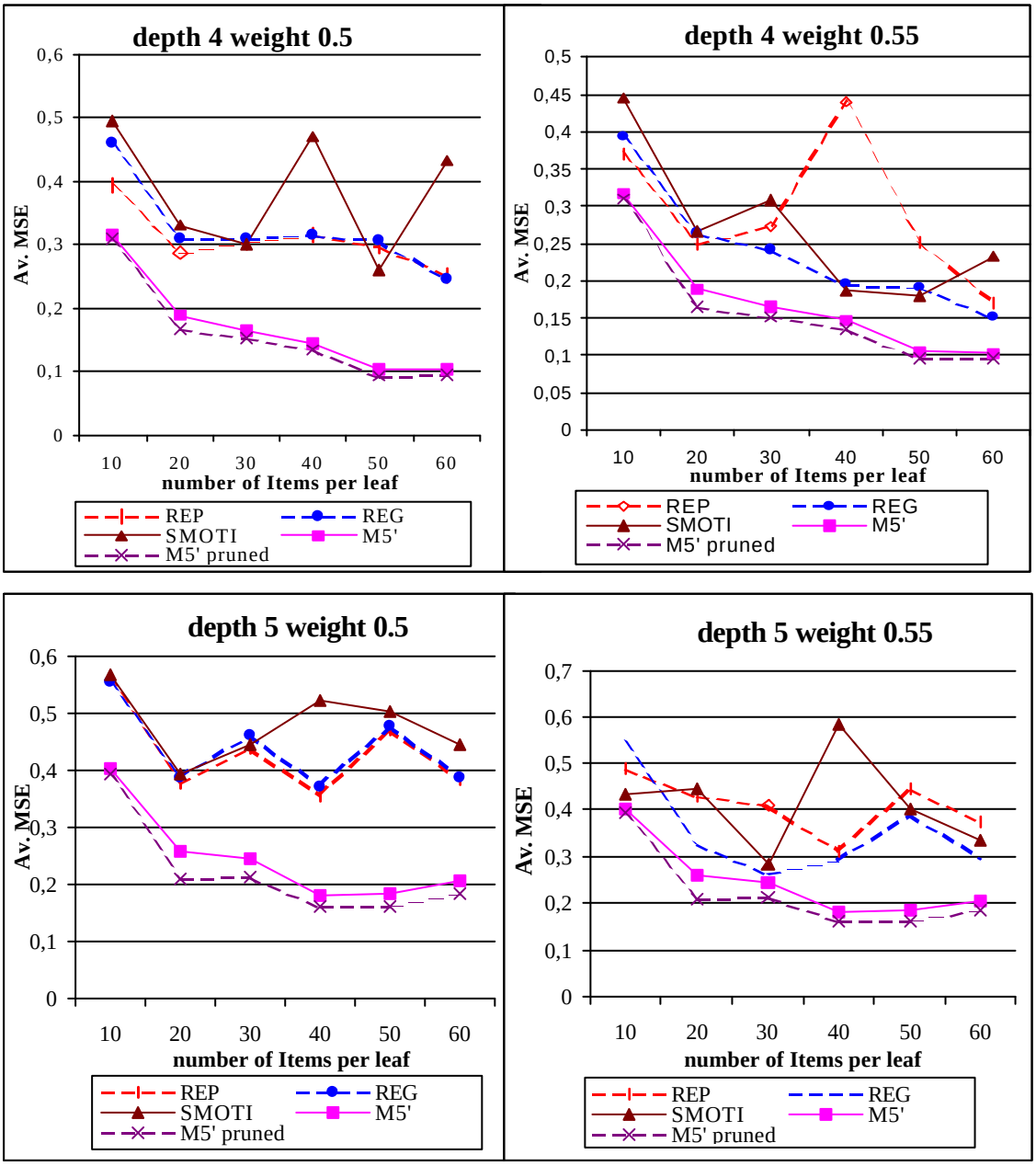
Table 3. Experimental results: REP vs REG. The better simplification method is reported. The statistically significant values ($p<\alpha/2$) are in italics.

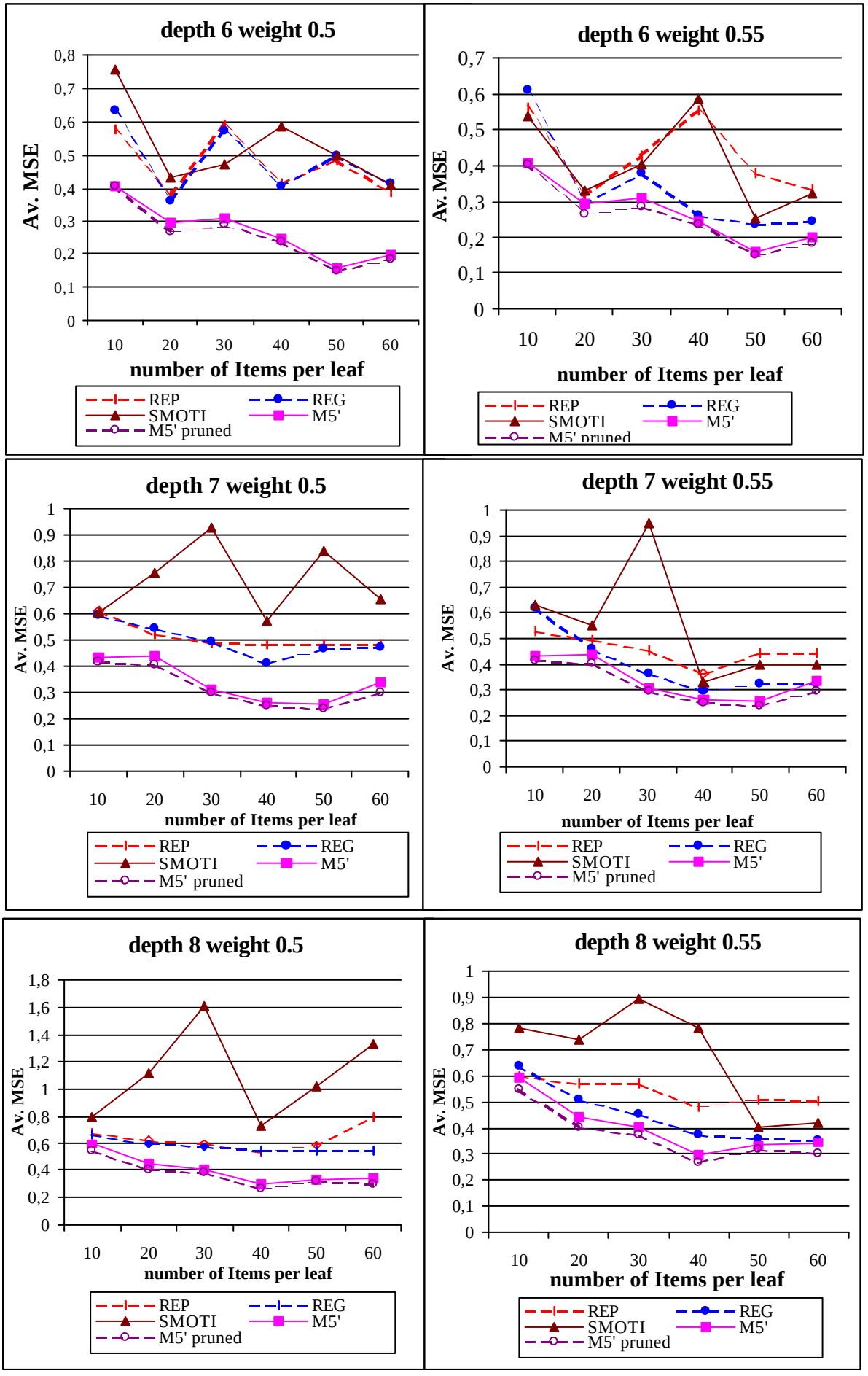
Items per leaf	Weight	Depth					
		4	5	6	7	8	9
10	0.5	REP	REG	REP	REG	REG	REP
	0.55	REP	REP	REP	REP	REP	REP
20	0.5	REP	REP	REG	REP	REG	REG
	0.55	REP	REG	REG	REP	REG	REG
30	0.5	REP	REP	-	REP	-	REG
	0.55	REG	REG	REG	REG	REG	REG
40	0.5	REP	REP	REP	REG	REG	REP
	0.55	REG	REG	REG	REG	REG	REG
50	0.5	REP	REP	REP	REG	REG	REG
	0.55	REG	REG	REG	REG	REG	REG
60	0.5	REP	REP	REP	REG	REG	REG
	0.55	REG	REG	REG	REG	REG	REG

Experimental results on the artificial data sets described above are reported in Table 3, where a statistical comparison between REP and REG is reported. It is noteworthy that when the weight is 0.5 no *statistically* significant differences ($p<0.05$) were observed between the two methods, with the only two exceptions being the cases of depth=8 and 10 items per leaf, and depth=7 and

40 items per leaf. In general, results show a better performance of REG when the depth and the number of items per leaf increase. In particular results are statistically significant in favor of REG when the weight is 0.55, and the depth and the number of items per leaf increase.

A different view of results is offered by the graphics in Figure 4, where a comparison with M5' is shown for the artificial data as well. As in the case of UCI data sets, M5' pruning strategy outperforms both REP and REG, since the model tree built by M5' on the whole training set is more accurate than the model tree built by SMOTI on 70% of the training cases. The difference in accuracy between the two starting trees is more evident in the case of w=0.5 and it decreases as the number of items per leaf increases. However, while the pruning strategy implemented in M5' is always beneficial, which means that M5' always overfits the training data, the effect of REP and REG is less evident. Indeed, the two proposed methods improve the average MSE of the final tree when both the number of items per leaf and/or the depth of the tree are fairly large.




depth 6 weight 0.5
Av. MSE
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
10 20 30 40 50 60
number of Items per leaf
REP
REG
SMOTI
M5'
M5' pruned
depth 6 weight 0.55
Av. MSE
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
10 20 30 40 50 60
number of Items per leaf
REP
REG
SMOTI
M5'
M5' pruned
depth 7 weight 0.5
Av. MSE
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
10 20 30 40 50 60
number of Items per leaf
REP
REG
SMOTI
M5'
M5' pruned
depth 7 weight 0.55
Av. MSE
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
10 20 30 40 50 60
number of Items per leaf
REP
REG
SMOTI
M5'
M5' pruned
depth 8 weight 0.5
Av. MSE
1,8
1,6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
10 20 30 40 50 60
number of Items per leaf
REP
REG
SMOTI
M5'
M5' pruned
depth 8 weight 0.55
Av. MSE
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
10 20 30 40 50 60
number of Items per leaf
REP
REG
SMOTI
M5'
M5' pruned

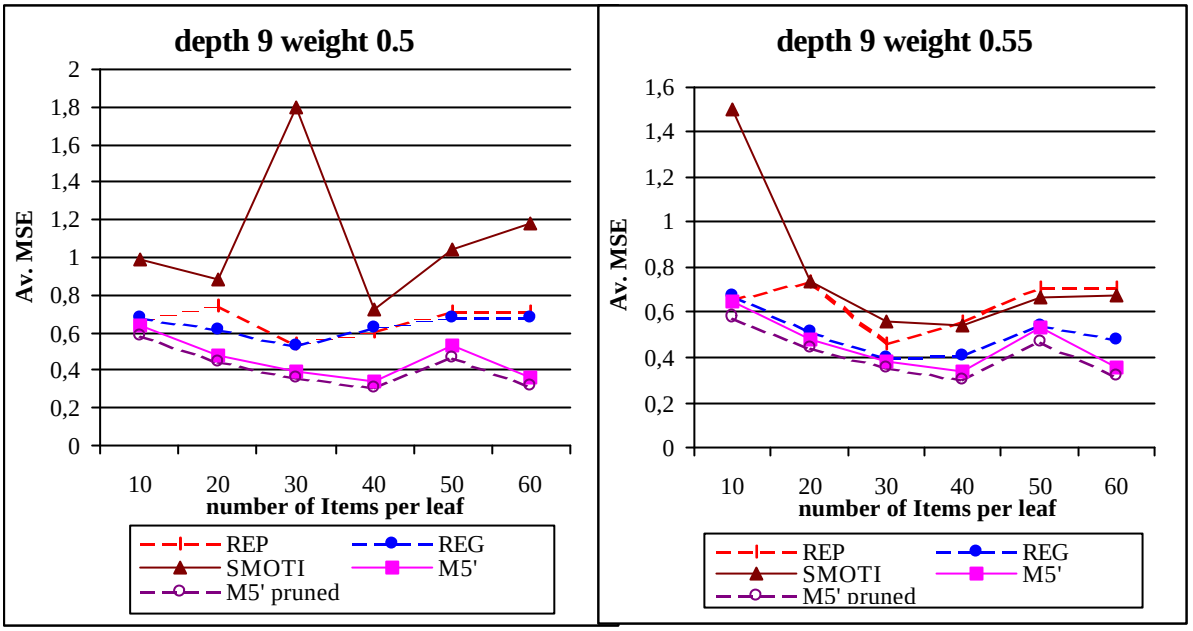


Figure 4. The comparison of average Mean Square Error for the pruning of two different systems: SMOTI (w=0.55 and w=0.5), M5' corresponding to artificial data sets built with parameters θ=0.5 and σ^2=0.001.

7 Conclusions

SMOTI is a TDIMT method, which integrates the partitioning phase and the labeling phase. Similar to many decision tree induction algorithms, SMOTI may generate model trees that overfit training data. In this paper, the *a posteriori* simplification (or pruning) of model trees has been investigated in order to solve this problem. Specifically, we developed a unifying framework for the *a posteriori* simplification of model trees with both regression nodes and splitting nodes. The framework is general enough to formulate other pruning methods. Some experimental results have been reported on the pruning methods. On UCI data, pruning was generally beneficial, while on artificial data we observed good results only under some experimental conditions. The better performance of the M5' pruning technique is explained in terms of better predictive accuracy of the model tree that M5' has to prune.

As future work, we plan to extend this comparison to other TDIMT systems (e.g. HTL and RETIS). Moreover, we intend to implement a new simplification method based on both pruning and grafting operators and to eventually extend MDL-based pruning strategies developed for regression [13] trees to the case of SMOTI trees. This extension should overcome problems we observed for small datasets since the new pruning algorithm will not require an independent pruning set.

References.

[1] Breiman L, Friedman J., Olshen R., & Stone J. Classification and regression tree, Wadsworth & Brooks, (1984).

[2] Cestnik B. and Bratko I. On estimating probabilities in tree pruning, *Proc. of the Fifth European Working Session on Learning*, Springer, (1991), 151-163.

[3] Esposito F., Malerba D., Semeraro G. A comparative analysis of methods for pruning decision trees. *IEEE Trans. PAMI*, Vol. 19, Num. 5, (1997), 476-491.

[4] Esposito F., Malerba D., Semeraro G. & Tamma V. The Effects of Pruning Methods on the Predictive Accuracy of Induced Decision Trees. *Applied Stochastic Models in Business and Industry*, Vol. 15, Num. 4, (1999), 277-299.

[5] Karalic A. Linear regression in regression tree leaves, in *Proceedings of ISSEK '92 (International School for Synthesis of Expert Knowledge)*, Bled, Slovenia, (1992).

[6] Lubinsky D. Tree Structured Interpretable Regression, in Learning from Data, Fisher D. & Lenz H.J. (Eds.), Lecture Notes in Statistics, 112, Springer, (1996), 387-398.

[7] Malerba D., Appice A., Bellino A., Ceci M. & Pallotta D. Stepwise Induction of Model Trees, in F. Esposito (Ed.), *AI*IA 2001: Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence*, 2175, Springer, Germany, (2001).

[8] Malerba, D., Appice A., Ceci M. & Monopoli, M. Trading-off Local versus Global Effects of Regression Nodes in Model Trees. *Proceeding of the 13th Int. Symposium on Methodologies for Intelligent Systems*, (2002).

[9] Niblett, T. & Bratko, I. Learning decision rules in noisy domains. In Bramer, M. A., *Research and Development in Expert Systems III*, Cambridge University Press, Cambridge, (1986), 25-34.

[10] Orkin, M. & Drogin, R. *Vital Statistics*. New York: McGraw Hill, (1990).

[11] Quinlan J.R. Simplifying decision trees. *International Journal of Man-Machine Studies*; 27, (1987), 221-234.

[12] Quinlan J. R. Learning with continuous classes, in *Proceedings AI'92*, Adams & Sterling (Eds.), World Scientific, (1992), 343-348.

[13] Robnik-Šikonja M., Kononenko I. Pruning Regression Trees with MDL. In H. Prade (Ed.), Proceedings of the 13th European Conference on Artificial Intelligence, John Wiley & Sons, Chichester, England, (1998), 455-459.

[14] 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, (1997), 118-127.

[15] Torgo L. Functional Models for Regression Tree Leaves, in *Proceedings of the Fourteenth International Conference (ICML '97)*, D. Fisher (Ed.), Nashville, Tennessee, (1997).

[16] 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, (1997), 128-137.