Simplification Methods for Model Trees with Regression and Splitting Nodes

Michelangelo Ceci, Annalisa Apice, and Donato Malerba

Dipartimento di Informatica, Università degli Studi
via Orabona, 4, 70126 Bari, Italy
{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

In the classical regression setting, data is generated IID from an unknown distribution P on some domain $\mathbf{X}$ and labeled according to an unknown function g with range Y. The domain $\mathbf{X}$ is spanned by m independent (or predictor) random variables x_i (both numerical and categorical), while Y is a subset of $\Re$, that is, the dependent (or response) variable y is continuous. A learning algorithm receives a training sample $S=\{(\mathbf{x}, y) \in \mathbf{X} \times Y \mid y=g(\mathbf{x}) \}$ and attempts to return a function f close to g on the domain $\mathbf{X}$. Closeness of f to g can be measured in many ways, for instance, by means of the expected square error.

Statisticians have traditionally approached this problem by means of standard (non-) linear regression techniques. More recently, trees have begun to play an important role in statistical model building, especially in the context of regression problems. Methods for inducing or learning *regression trees* have been proposed by Breiman *et al.* [1]. A regression tree approximates the function g by means of a piecewise *constant* function, that is it associates a constant to each leaf. A generalization of regression trees is represented by *model trees* which approximate the function g 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' [17], TSIR [6], and HTL [15,16]. Almost all these systems perform a *top-down* induc-

P. Perner and A. Rosenfeld (Eds.): MLDM 2003, LNAI 2734, pp. 20–34, 2003.

tion 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 [7]. RETIS, which operate differently, suffers from problems of efficiency and collinearity. 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. The stepwise construction of the multiple linear models provides a solution to problems of efficiency and collinearity by selecting only a subset of variables [18]. 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 techniques 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 [17]. 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 [16].

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 (Secs. 4 and 5). Finally, experimental results are reported and discussed in Sect. 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 (Fig. 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 both the dependent and 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,

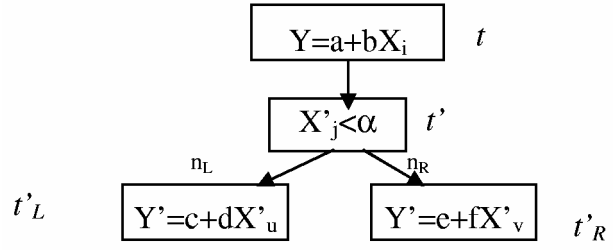

Fig. 1. A model tree with both a regression node (t) and a splitting note (t')

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. For instance, let us consider the problem of building a multiple regression model with two independent variables through a sequence of straight-line regressions:

$$\hat{Y} = a + bX_1 + cX_2$$

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

$$\hat{Y} = a_1 + b_1X_1$$

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:

$$\hat{X}_2 = a_2 + b_2X_1$$

then compute the residuals on X_2 and Y:

$$X'_2 = X_2 - (a_2 + b_2X_1)$$
$$Y' = Y - (a_1 + b_1X_1)$$

and finally regress Y' on X'_2 alone:

$$\hat{Y}' = a_3 + b_3X'_2.$$

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

$$\widehat{Y - (a_1 + b_1X_1)} = a_3 + b_3(X_2 - (a_2 + b_2X_1))$$

Since $\widehat{Y - (a_1 + b_1X_1)} = \hat{Y} - (a_1 + b_1X_1)$ we have :

$$\hat{Y} = (a_3 + a_1 - a_2b_3) + (b_1 - b_2b_3)X_1 + b_3X_2.$$

It can be proven that this last model coincides with the first model built, that is $a = a_3 + a_1 - a_2b_3$, $b = b_1 - b_2b_3$ and $c = b_3$. Therefore, when the first regression line of Y on X_1 is built we pass down both the residuals of Y and *the residuals of the regression of* X_2 *on* X_1. This means we remove the linear effect of the variables already included in the model (X_1) from both the response variable (Y) and those variables to be selected for the next regression step (X_2).

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 exactly 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 $\mathcal{O}$ of N observations $\mathcal{O}_i$, 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 $\mathcal{O}$, $\mathcal{O}(t)$, possibly modified by removing the linear effect of the variables added to the model;
2. the root t_0 is associated with $\mathcal{O}$ itself;
3. every edge $\langle t_i, t_j \rangle \in B_T$ is labelled with $L_T(\langle t_i, t_j \rangle)$;

$L_T(\langle t_i, t_j \rangle)$ can be:

a) a straight-line regression function $Y = a + bX_k$ (*regression label*)[2];
b) a test on a numeric variable like $X_k \leq \alpha$ ($X_k > \alpha$) (*continuous split label*);
c) 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 $\langle t_i, t_j \rangle \in B_T$ such that $L_T(\langle t_i, t_j \rangle)$ is a regression (continuous/discrete split) label.

---

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

[2] For the sake of simplicity, only original variables are reported in straight-line regression functions and in continuous splits.

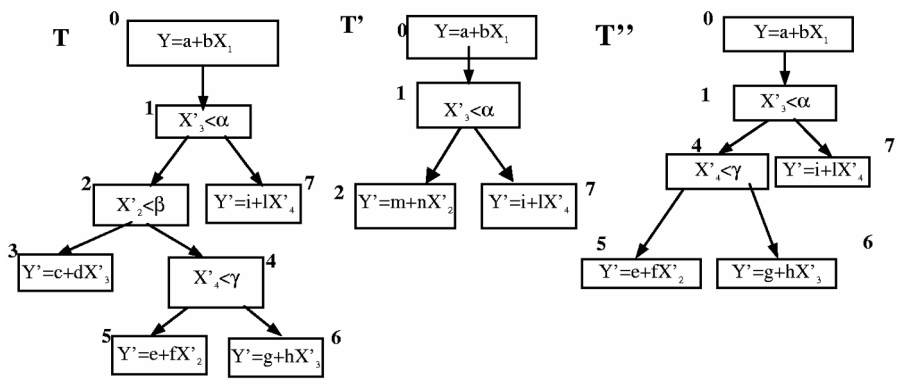

Fig. 2. The model tree T' is obtained by pruning T in node 2, while T'' is obtained by grafting the subtree rooted in node 4 onto the place of node 2

If p is a path from the root t_0 of a tree T to a leaf t_i of T,

$$p = <t_0,t_1>,<t_1,t_2>,\ldots,<t_{i-1},t_i>$$

then the label associated to p, $L_T(p)$, is the sequence of labels:

$$L_T(p)=L_T(<t_0,t_1>), L_T(<t_1,t_2>),..., L_T(<t_{i-1},t_i>).$$

Let T denote the set of all possible model trees that SMOTI can build from $\mathcal{O}$. 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 for each path p' from the root of T' to a leaf in T', there exists a path p from the root of T to a leaf of T such that $L_T(p')$ is a prefix of $L_T(p)$. Moreover, T' $\leq_G$ T, iff for each path p' from the root of T' to a leaf in T', there exists a path p from the root of T to a leaf of T, such that $L_T(p')$ is obtained from $L_T(p)$ by dropping some continuous/discrete split labels in the sequence.

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:

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

where R_T and S_T denote the sets of regression and splitting nodes, respectively. The operator 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 γ_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 the 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 in 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 (Figure 3).

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:

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

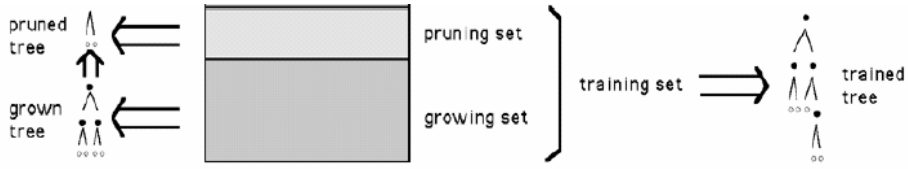

Fig. 3. The original data set can be split into two subsets: the growing set and the pruning set. The union of the growing and pruning set is called the training set. Trees learned from the growing/training set are called grown/trained trees, respectively. Pruning trees can be obtained by pruning either grown trees or trained trees. In the former case, a pruning set is used

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 T_t. This pruning operation is repeated on the simplified tree until further pruning increases the resubstitution error.

As already observed in [14], Quinlan's description of REP for decision trees does not specify how to choose the class associated to the pruned nodes. Following the majority class criterion, three alternatives are possible: the class is determined on the basis of the growing set, the pruning set or the training set. Analogously, in the case of model trees, the straight-line regression to be associated to a pruned node can be determined on the basis of one of the three sets: growing, pruning or training.

The following optimality theorem can be proven:

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

The specification "the REP version that determines the regression model on $\mathcal{O}$" 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 $\mathcal{O}$. Alternatively, it could be determined on the basis of either the pruning set or the whole training set.

Proof . We prove the theorem by induction on the depth of T.

Base Case. Let T be a root tree $\{t_0\}$. Then T is the only tree in $S_P(T)$ and REP returns T, since no pruning operation is possible.

Inductive Step: The proof is based on the additive property of the resubstitution error for model trees, according to which a local optimization on each branch T_{t_i} of t_0 leads to a global optimization on T. We assume the inductive hypothesis to be true for all model trees T' of depth *d* and we prove the theorem for the tree T of depth *d*+1, *d*≥0. Since T has a depth greater than 0, there are two possibilities:

a. T is the tree rooted in t_0, a regression node with a child t_1, and the subtree T_{t_1} has depth *d*.
b. T is the tree rooted in t_0, a splitting node with two children t_1 and t_2, and both subtrees T_{t_1}, T_{t_2} have maximum depth *d*.

Case a. REP, which follows the bottom-up direction when pruning, first prunes T_{t_1} and then checks whether T should be pruned in t_0. For the inductive hypothesis, REP finds the optimally pruned tree $T^*_{t_1}$ for the tree rooted in t_1. Let T' be the tree rooted in t_0, whose subtree is $T^*_{t_1}$. Then according to the definition of f, $f(T') = f(T^*_{t_1})$, since T' and $T^*_{t_1}$ have the same leaves. Moreover, for any tree $T'' \in S_P(T)$ of depth greater than 0 we have $f(T') \le f(T'')$, since $f(T') = \sum_{t \in \tilde{T}^*_{t_1}} R(t) \le \sum_{t \in \tilde{T}''_{t_1}} R(t) = f(T'')$.

Therefore, if $f(\{t_0\}) \le f(T')$ then REP prunes T in t_0, and the returned tree is the best subtree of T, since $R(t_0) = f(\{t_0\}) \le f(T') \le f(T'')$ for any tree $T'' \in S_P(T)$ of depth greater than 0. On the contrary, if $f(\{t_0\}) > f(T')$ then REP does not prune T in t_0 and the returned tree is T', which is the smallest tree in $S_P(T)$ with the lowest error with respect to $\mathcal{O}$.

Case b. Analogously, in the case of the splitting node, REP follows the bottom-up direction so that it first prunes T_{t_1} and T_{t_2} and then checks whether T should be pruned in t_0. For the inductive hypothesis REP finds the optimally pruned tree $T^*_{t_1}$ for the tree rooted in t_1 and $T^*_{t_2}$ for the tree rooted in t_2. Let T' be the tree rooted in t_0, whose subtrees are $T^*_{t_1}$ and $T^*_{t_2}$. Then $f(T') = f(T^*_{t_1}) + f(T^*_{t_2})$, since the leaves of T' are leaves of either $T^*_{t_1}$ or $T^*_{t_2}$. Moreover, for any tree $T'' \in S_P(T)$ of depth greater than 0, we have $f(T') \le f(T'')$ since $f(T') = \sum_{t \in \tilde{T}^*_{t_1}} R(t) + \sum_{t \in \tilde{T}^*_{t_2}} R(t) \le \sum_{t \in \tilde{T}''_{t_1}} R(t) + \sum_{t \in \tilde{T}''_{t_2}} R(t) = f(T'')$.

Therefore, if $f(\{t_0\}) \le f(T')$ then REP prunes T in t_0, and the returned tree is the best subtree of T, since $R(t_0) = f(\{t_0\}) \le f(T') \le f(T'')$ for any tree $T'' \in S_P(T)$ of depth greater than 0. On the contrary, if $f(\{t_0\}) > f(T')$ then REP does not prune T in t_0 and the returned tree is T', which is the smallest tree in $S_P(T)$ with the lowest error with respect to $\mathcal{O}$.

■

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 $f(T_1) \ge f(T_2)$ 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.

```
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
```

Fig. 4. Reduced error grafting algorithm

This method (see Fig. 4) is theoretically favored 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.

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 $\mathcal{O}$ and a pruning set $\mathcal{O}$, the REG version that determines the regression model on $\mathcal{O}$ returns the smallest tree in $S_G(T)$ with the lowest error with respect to $\mathcal{O}$.

Proof. We prove the theorem by induction on the depth of T.

Base Case. Let T be a root tree $\{t_0\}$. Then T is the only tree in $S_G(T)$ and REG returns, T since no grafting operation is possible.

Inductive Step: we assume the inductive hypothesis to be true for all model trees T' of depth *d* and we prove the theorem for the tree T of depth *d*+1, d≥0. Since T has a depth greater than 0, there are two possibilities:

a. T is the tree rooted in t_0, a regression node with a child t_1, and the subtree T_{t_1} has depth *d*.
b. T is the tree rooted in t_0, a splitting node with two children, t_1 and t_2, and both subtrees T_{t_1} and T_{t_2} have maximum depth *d*.

Case a. REG, which follows the bottom-up direction first simplifies T_{t_1}. For the inductive hypothesis REG finds the optimally simplified tree $T^*_{\tilde{t}_1}$ for the tree T_{t_1}.[3]

Let T' be the tree rooted in t_0, whose subtree is $T^*_{\tilde{t}_1}$. Then, according to the definition of *f*, $f(T') = f(T^*_{\tilde{t}_1})$, since T' and $T^*_{\tilde{t}_1}$ have the same leaves. Moreover, for any tree $T'' \in S_G(T)$ of depth greater than 0 and rooted in t_0 with child *t*", we have $f(T') \le f(T'')$, since $f(T') = \sum_{t \in \tilde{T}^*_{t_1}} R(t) \le \sum_{t \in \tilde{T}''_{t_1}} R(t) = f(T'')$.

Therefore, REG returns T', which is the smallest tree in $S_G(T)$ with the lowest error with respect to $\mathcal{O}$.

Case b. In the case of a splitting node, REG follows the bottom-up direction, simplifies T_{t_1} and T_{t_2} with respect to $\mathcal{O}$ and then checks whether one of the simplified subtrees should be grafted in t_0. We denote the set of examples which fall in T_{t_1} (T_{t_2}) as $\mathcal{O}_1'$ ($\mathcal{O}_2'$). Let $T^*_{\tilde{t}_1}$ ($T^*_{\tilde{t}_2}$) be the subtree rooted in t_1 (t_2), returned by REG when pruned with respect to $\mathcal{O}_1'$ ($\mathcal{O}_2'$). For the inductive hypothesis, $T^*_{\tilde{t}_1}$ ($T^*_{\tilde{t}_2}$) is the optimally grafted subtree rooted in t_1 (t_2) with respect to $\mathcal{O}_1'$ ($\mathcal{O}_2'$), respectively, since the depth of $T^*_{\tilde{t}_1}$ ($T^*_{\tilde{t}_2}$) is not greater than *d*. Analogously, let $T^*_{\hat{t}_1}$ ($T^*_{\hat{t}_2}$) be the subtree rooted in t_1 (t_2), returned by REG when pruned with respect to $\mathcal{O}$. For the inductive hypothesis, $T^*_{\hat{t}_1}$ ($T^*_{\hat{t}_2}$) is the optimally grafted subtree rooted in t_1 (t_2) with respect to $\mathcal{O}'$.

[3] Note that the grafting operation applied to the tree T_{t_1} may not produce a tree rooted in t_1, so we denote it as $T^*_{\tilde{t}_1}$.

Let T' be the tree rooted in t_0 whose subtrees are $T^*_{t_1}$ and $T^*_{t_2}$. Then, for any tree $T'' \in S_G(T)$ of depth greater than 0 and rooted in t_0 with children t'_1 and t'_2, we have $f(T') \leq f(T'')$, since $f(T') = \sum_{t \in \tilde{T}^*_{t_1}} R(t) + \sum_{t \in \tilde{T}^*_{t_2}} R(t) \leq \sum_{t \in \tilde{T}''_{t'_1}} R(t) + \sum_{t \in \tilde{T}''_{t'_2}} R(t) = f(T'')$.

Therefore, if $f(T^*_{\hat{t}_1}) \leq f(T')$ and $f(T^*_{\hat{t}_1}) \leq f(T^*_{\hat{t}_2})$, where $f(T^*_{\hat{t}_1})$ and $f(T^*_{\hat{t}_2})$ are computed with respect to $\mathcal{O}'$, then REG replaces T' with $T^*_{\hat{t}_1}$, which is the best subtree of T, since

- $R(T^*_{\hat{t}_1}) = f(T^*_{\hat{t}_1}) \leq f(T') \leq f(T'')$ for any tree $T'' \in S_G(T)$ rooted in t_0 with depth greater than 0.
- $R(T^*_{\hat{t}_1}) = f(T^*_{\hat{t}_1}) \leq f(T'')$ for any tree $T'' \in S_G(T)$ not rooted in t_0 (both $T^*_{\hat{t}_1}$ and T'' have maximum depth d and the inductive hypothesis holds on $\mathcal{O}$).

Otherwise, if $f(T^*_{\hat{t}_2}) \leq f(T')$ and $f(T^*_{\hat{t}_2}) \leq f(T^*_{\hat{t}_1})$, REG replaces T' with $T^*_{\hat{t}_2}$, which is the best subtree of T, since

- $R(T^*_{\hat{t}_2}) = f(T^*_{\hat{t}_2}) \leq f(T') \leq f(T'')$ for any tree $T'' \in S_G(T)$ rooted in t_0 with depth greater than 0.
- $R(T^*_{\hat{t}_2}) = f(T^*_{\hat{t}_2}) \leq f(T'')$ for any tree $T'' \in S_G(T)$ not rooted in t_0 (both $T^*_{\hat{t}_2}$ and T'' have maximum depth d and the inductive hypothesis holds on $\mathcal{O}$).

Finally, if both $f(T^*_{\hat{t}_1}) > f(T')$ and $f(T^*_{\hat{t}_2}) > f(T')$ then REG does not simplify T in t_0 and the returned tree is T'. Obviously, T' is better than any tree $T'' \in S_G(T)$ rooted in t_0. Moreover, T' is better than any tree $T'' \in S_G(T)$ not rooted in t_0, since either $T'' \in S_G(T_{t_1})$ and $f(T'') \geq f(T^*_{\hat{t}_1}) > f(T')$ or $T'' \in S_G(T_{t_2})$ and $f(T'') \geq f(T^*_{\hat{t}_2}) > f(T')$. Therefore, the returned tree T' is the smallest tree in $S_G(T)$ with the lowest error with respect to $\mathcal{O}$.

■

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

(http://www.di.uniba.it/~malerba/software/kdb2000/) and has been empirically evaluated on ten datasets, taken from either the UCI Machine Learning Repository (http://www.ics.uci.edu/~mlearn/MLRepository.html), the site of the system HTL (http://www.ncc.up.pt/~ltorgo/Regression/DataSets.html) or the site of WEKA (http://www.cs.waikato.ac.nz/ml/weka/index.html). They are listed in Table 1 and have a continuous variable to be predicted. They have been used as benchmarks in related studies on regression trees and model trees.

Table 1. Ten datasets used in the empirical evaluation of SMOTI

Dataset	No. Cases	No. Attributes	Continuous	Discrete	Goal
Abalone	4177	10	9	1	Predicting the age of abalone from physical measurements
Auto-Mpg	392	8	5	3	Predicting the city-cycle fuel consumption
Auto-Price	159	27	17	10	Predicting auto price
Bank8FM	4499	9	9	0	Predicting the fraction of bank customers who leave the bank because of full queues
Cleveland	297	14	7	7	Predicting the heart disease in a patient.
Housing	506	14	14	0	Predicting housing values in areas of Boston
Machine CPU	209	10	8	2	Predicting relative CPU performance
Pyrimidines	74	28	28	0	Predicting the activity (QSARs) from the descriptive structural attributes
Triazines	74	61	61	0	Predicting the structure (QSARs) from the descriptive structural attributes
Wisconsin Cancer	186	33	33	0	Predicting the time to recur for a breast cancer case

Each dataset is analyzed by means of a 10-fold cross-validation. For every trial, the data set is partitioned so that 90% of cases are left in the training set and 10% are put aside for the independent test set. The training set is, in turn, partitioned into growing (70%) and pruning set (30%). SMOTI is trained on the growing set, pruned on the pruning set and tested on the test set. Comparison is based on the average mean square error (*Avg.MSE*) made on the test sets:

$$Avg.MSE = \frac{\sum_{i=1..k} R(T_i)}{k}$$

where k is the number of folds and $R(T_i)$ is the resubstitution error of the *i-th* cross-validated tree computed on the corresponding testing set.

Experimental results are listed in Table 2. The table reports the average MSE of (un-pruned/pruned) SMOTI trees built on training/growing set. For comparison purposes, results obtained on the M5', which is implemented in the public available system WEKA, are reported as well. Results show that pruning is generally beneficial since REP and REG decrease the Avg.MSE of SMOTI trees built on the growing set in nine out of ten data sets. Moreover, the pruning method implemented in M5' pruning method outperforms both REP and REG in most data sets. However, the poorer 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.

A different view of results is offered in Table 3, which reports a percentage of the avg. MSE made by pruned trees on the test sets with respect to the avg. MSE made by un-pruned trees on the same testing sets. The table emphasizes the gain of the use of pruning. In particular, pruning is beneficial when the value is less than 100%, while it is not when the value is grather than 100%. Results reported confirm that pruning is beneficial for nine out of ten datasets. Moreover, the absolute difference of Avg. MSE for REP and REG is below 5% in seven datasets. Finally, it is worthwhile to notice that the gain of REP and REG is better than the corresponding gain of M5' pruning method in six datasets. This induces to hypothesize that the better absolute performances of M5' are mainly due to the fact that the tree to be pruned is more accurate because of the full use of training cases.

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

	SMOTI un-pruned		SMOTI pruned		M5'	
	Avg trained MSE	*Avg* grown MSE	*Avg* REP MSE	*Avg* REG MSE	*Avg* not pruned MSE	*Avg* Pruned MSE
Abalone	2,5364	6,7244	2,1851	2,1797	2,77242	2,12669
AutoMpg	3,14938	4,48666	3,56337	3,74361	3,20106	2,83555
Auto Price	2246,03	2481,74	2746,32	2890,42	2358,81	2390,12
Bank8FM	0,0383	0,0427	0,0358	0,0342	0,04099	0,03198
Cleveland	1,3160	1,5215	0,9148	0,9349	1,24963	0,90286
Housing	3,58	5,7176	4,0803	3,9120	4,27927	3,8159
Machine CPU	55,3148	71,6991	70,9533	69,1451	57,3527	58,3412
Pyrimidines	0,10566	0,18727	0,10343	0,13527	0,09279	0,08640
Triazines	0,2017	0,1820	0,1559	0,2290	0,15503	0,1318
Wisconsin Cancer	51,4138	72,3762	33,4649	37,4556	45,4064	34,3972

Table 3. Average percentage of the MSE for pruned trees w.r.t. the MSE of un-pruned trees. MSE is computed on the testing set. Best values are in bold

Data Set	REP/un-pruned SMOTI on growing set	REG/un-pruned SMOTI on growing set	Pruned M5' /un-pruned M5'
Abalone	32.49%	**32.42%**	76.71%
AutoMpg	**79.42%**	83.44%	88.58%
Auto Price	110.66%	116.47%	**101.33%**
Bank8FM	83.84%	80.09%	**78.02%**
Cleveland	**60.12%**	61.44%	72.25%
Housing	71.36%	**68.42%**	89.17%
Machine CPU	98.95%	**96.43%**	101.72%
Pyrimidines	**55.23%**	72.23%	93.11%
Triazines	85.65%	125.82%	**85.02%**
Wisconsin Cancer	**46.23%**	51.75%	75.75%

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. Two methods, named REP and REG, have been defined on the basis of this framework, which is general enough to formulate other pruning methods. Some experimental results have been reported on the pruning methods and show that pruning is improves the average mean square error in most of datasets. Moreover, the comparison with another well-known TDIMT method, namely M5', which uses the training data both for growing and for pruning the tree, induces to hypothesize that putting aside some data for pruning can lead to worse results.

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.

Acknowledgments. The work presented in this paper is in partial fulfillment of the research objectives set by the MIUR COFIN-2001 project on "Methods for the extraction, validation and representation of statistical information in a decision context".

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. Tapio Elomaa, Matti Kääriäinen (2001). An Analysis of Reduced Error Pruning. *Journal of Artificial Intelligence Research*, 15, pp. 163–187.
15. 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.
16. Torgo L. Functional Models for Regression Tree Leaves, in *Proceedings of the Fourteenth International Conference (ICML '97)*, D. Fisher (Ed.), Nashville, Tennessee, (1997).
17. 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.
18. Weisberg S. *Applied regression analysis, 2nd edn*. New York: Wiley, 1985.