Backpropagation in Decision Trees for Regression

Victor Medina-Chico[1], Alberto Suárez[1], and James F. Lutsko[2]

[1] Escuela Técnica Superior de Informática
Universidad Autónoma de Madrid
Ciudad Universitaria de Cantoblanco
28049 Madrid, Spain
[2] Center for Nonlinear Phenomena and Complex Systems
Université Libre de Bruxelles
C.P. 231 Campus Plaine
B-1050 Brussels, Belgium

Abstract. A global optimization algorithm is designed to find the parameters of a CART regression tree extended with linear predictors at its leaves. In order to render the optimization mathematically feasible, the internal decisions of the CART tree are made continuous. This is accomplished by the replacement of the crisp decisions at the internal nodes of the tree with soft ones. The algorithm then adjusts the parameters of the tree in a manner similar to the backpropagation algorithm in multilayer perceptrons. With this procedure it is possible to generate regression trees optimized with a global cost function, which give a continuous representation of the unknown function, and whose architecture is automatically fixed by the data. The integration in one decision system of complementary features of symbolic and connectionist methods leads to improvements in prediction efficiency in both synthetic and real-world regression problems.

1 Introduction

The use of decision trees in regression problems has been limited by their reduced expressive capacity. For instance, CART regression trees [1] yield a rather coarse approximation of the unknown real-valued function in terms of patches of constant value. The origin of this piecewise representation is the divide-and-conquer strategy used in the generation of the decision tree: The original regression problem is decomposed into a series of simpler problems by a recursive partitioning procedure that divides the attribute space into disjoint regions. Each of these divisions is generated by a Boolean test on the attributes. The parameters of the splits are found by minimizing a cost function that measures the local prediction error. This implies that the tree is an approximation that presents finite-size jump discontinuities at the decision boundaries. Furthermore the greedy strategy used in the tree generation implies that the final predictor need not be globally optimal. There are also statistical implications in subdividing the initial space

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 348–359, 2001.

into smaller regions. Every time a split is made, the problem generally becomes simpler to solve, but the number of examples available for the learning is also reduced. In more precise terms, dividing the data can have favorable consequences for the bias of the predictor, but it generally increases its variance. The decomposition of the prediction error in terms of bias and variance is a well-known decomposition in regression [11]. The bias term may be understood as the error that remains even if we had an infinite number of observations while the variance term is a measure of the fluctuations due to the finite number of observations.

It is thus desirable to design a method to construct decision trees that are optimized with a global cost function while retaining the advantages of a symbolic knowledge representation. In this work, a backpropagation algorithm is presented that adjusts the parameters of a decision tree whose architecture is selected by a standard tree generation algorithm [21]. This algorithm is similar to the backpropagation algorithm in multilayer perceptrons. It solves the credit assignment problem by propagating the errors from the leaves of the tree up to its root, following an inverse path to that used in prediction. The advantage of the resulting predictor is that its architecture is not fixed beforehand, but is generated automatically by the data itself. The regression tree obtained is subject to the limitations of the base tree generation algorithm in finding an architecture as close as possible to the optimal one, and of the numerical optimization procedure, which may get trapped in a local minimum of the cost function.

To implement this backpropagation algorithm on the architecture of the tree needs to be reinterpreted in terms of membership functions. A classical decision tree, such as CART [1] or C4.5 [18] is characterized by Boolean membership functions. A fuzzy decision tree incorporates continuous membership functions to allow the possibility of partial memberships in the nodes of the tree. This reformulation makes it possible to employ analytic tools to adjust the tree parameters to optimize a global cost function. The starting point for the generation of fuzzy regression trees is a CART tree [1] extended with linear models at each of the leaves of the tree [22], instead of the traditional constant models used in CART. These linear models at each of the leaves have the effect of extending the expressive capabilities of the traditional CART tree and of reducing its size. Once the crisp tree is constructed, the Boolean splits are replaced by continuous sigmoidal splits that can be continuously tuned. In a manner similar to backpropagation in neural networks, the learning process involves the propagation of the tree estimates from the leaves to the root node.

There are several advantages in this approach. The representation given by a fuzzy regression tree is continuous and the parameters can be adjusted to optimize a global error function. Instead of discarding data at every decision in the tree, all training data is assigned to every node, although with different degrees of membership. Since no single tree leaf is solely responsible for the prediction, some of the model interpretability is lost. Nonetheless, the final estimator can be thought of as a hierarchical mixture of experts [14]. The optimized model is not just a smoothed version of a piecewise linear representation, and can significantly differ form the original CART tree.

The organization of the paper is as follows: In Section 2, connections with other approaches such as M5 [12], ensemble methods (bagging, boosting, PC - perturb & combine) and HME (hierarchical mixtures of experts) are outlined. In Section 3, the generation of crisp trees is reinterpreted in terms of membership functions. This interpretation leads to the design of a global optimization algorithm and to the formulation of the backpropagation algorithm for regression. The results of the experiments are presented in Section 4. Finally, Section 5 summarizes the results and conclusions of this work.

2 Relation to Previous Work

A decision tree with linear models at the leaves, M5, was proposed in Ref. [12]. In the M5 system, a fully developed regression tree giving a piecewise constant representation of the objective function is pruned by replacing the subtree attached to an inner node by a linear model whenever the latter outperforms the subtree model. The linear models are artificially restricted to using only those variables involved in the splits in the eliminated subtree. The final tree gives a piecewise linear (i.e. discontinuous) representation. No attempt is made to select the tree parameters by optimizing a global cost function.

Much of the current research on regression trees strives to overcome the shortcomings and rigidity of piecewise representations. Divide-and-conquer algorithms approach a complex task by dividing it into more elementary tasks whose solutions can be combined to yield a solution to the original problem. This leads to simple and elegant algorithms. However, one should be concerned about the statistical consequences of dividing the input space, specially when the training data available are limited. Dividing the data generally leads to an increased variance because one has more hypotheses that fit the reduced number of data points.

For unstable algorithms such as CART [1], C4.5 [18] or ID3 [19], a possible solution is to to combine different realizations of the same predictor. These different realizations are obtained by perturbing the training set in various ways. Examples are bagging [2], boosting [8], arcing [4] and output smearing [5]. All of them need some amount of instability of the base learner to small changes in the training set. The reason for this is that the predictors need to be different to reduce the variance.

The performance of ensemble methods is often quite remarkable given their simplicity. In fact, these algorithms are more immune to overfitting than one would expect considering the large amount of parameters involved. In regression, the performance of ensemble methods can be studied in terms of a decomposition of the error in terms of bias and variance [3]. The combination of the base learners leads to a reduction in the variance. In classification, the explanation is not so clear since there is not a single decomposition in terms of bias and variance [4], [15] or [16]. Besides, boosting has been argued to be more than a variance-reducing device and an explanation in terms of the margin of classification has also been proposed [20].

The main disadvantage of voting methods is their complexity. Because the amount of parameters involved and because the final hypothesis is the average of the hypotheses of the different predictors, the interpretation of a voting method is not as straightforward as that of a single predictor. The computational costs are also larger both in terms of memory and prediction speed.

The present work proposes to take advantage of combinations of different models, as well, but from a radically different perspective. Whereas ensembles of learners generally give similar weights to all the individual predictor, we propose a weighted combination the predictions given by the different units that make up a single learner. The starting point for our algorithm is a CART regression tree, extended with linear models at the leaves. The unknown function is approximated by the Piecewise Linear Model (PLM) encoded by the regression tree. The weights for combining the linear models are computed by applying a succession of fuzzy splits that replace the Boolean splits in the internal nodes of the CART tree. The resulting predictor can be thought of as a hierarchical mixture of linear predictors [14] whose architecture is fixed by the training data itself. The parameters of both the fuzzy splits and of the linear models are adjusted by minimization of a global cost function.

3 Optimization of PLM Regression Trees

In order to design a procedure to optimize the parameters of the tree structure with respect to a global cost function, one has to take into account the fact that the decisions taken in its construction are formulated in terms of hard splits. The global cost function for the tree can be written in terms of Boolean membership functions of the training examples in the leaf nodes [21]. These functions are discontinuous step functions. Therefore, their parameters cannot be tuned by standard analytical optimization routines, which require smoothness of the function and its derivatives. The solution to this problem is to replace these functions by continuous ones. This means that the decisions become smooth and that a point may simultaneously belong to different leaves, thus bringing fuzzy character into the tree structure. Following Suárez and Lutsko [21] we review the construction of a crisp regression tree in terms of membership functions. This reformulation allows the possibility of considering real-valued (fuzzy) membership functions and renders the design of a global optimization algorithm feasible. The introduction of the fuzzy character is made once the tree architecture is fixed. This is a procedure analogous to that used in the construction of a neural network. It has the advantage that selection of the architecture is an automatic process guided by the data itself and not by heuristics. There are many different algorithms to generate decision trees [1,10,19] . In this paper the base classifier is CART, designed by Breiman *et al.* [1].

The dataset from which the tree is induced consists of examples in the form of pairs $(\boldsymbol{x}_n, y_n); n = 1, \ldots, N$, where $\mathbf{x}_n = \{x_{n1}, x_{n2}, \ldots, x_{nD}\}$ is the vector of attributes (also known as predictor or independent variables) and y_n is the dependent variable to be predicted. We assume that both the dependent variable

and the attributes are ordinal variables (regression). Nominal attributes can also be handled by the decision tree. However, the possibility of fuzzifying nominal splits is not considered.

Following the usual procedure, the dataset is partitioned into a training set, with N_{train} examples, which is used to learn the model, and a statistically independent test set, with N_{test} examples, which is used to evaluate the quality of the generated model. The training set is used to build a tree (T) composed of a collection of nodes ($t_i \in T; i = 0, 1, 2...$) arranged in a hierarchical manner. The CART method is a top-down algorithm for the generation of a binary decision tree. The first node of a CART tree is the root node, t_0. By definition, all examples belong to this node. The tree is then constructed using a divide-and-conquer strategy in which the attribute space is partitioned by several Boolean tests into a set of disjoint subregions, in each of which the decision problem is simpler. Each of the tests corresponds to an internal node of the decision tree. This procedure can be repeated recursively until a stopping criterion is met. The CART prescription is to generate a maximally developed tree and then to perform cost-complexity pruning using 10-fold cross-validation.

In the original CART formulation, the prediction for the dependent variable in each region is a constant value which is equal to the average of the dependent variable in that region. A CART tree thus yields a piecewise constant model for the unknown relation between the independent and dependent variables. Piecewise Linear Model decision trees (PLM trees) differ from the original CART trees in that in each of these disjoint regions, a local linear model for the data is constructed [22]. This formulation allows for a more flexible representation of the unknown function at each of the terminal nodes. The coefficients of the attributes are then determined by minimizing the mean square error over the training set. The predictions of the leaves ($\mathbf{x}$) are linear models

$$\bar{y}_l(\mathbf{x}) = \beta_{l0} + \beta_{l1} x_1 + \ldots + \beta_{lD} x_D. \tag{1}$$

The tendency to overfitting may be partially avoided by selecting which attributes of the linear models at each of the leaf nodes have to remain. This is achieved using an algorithm designed by Jennrich [13]. According to this algorithm, a series of regressions is constructed, starting from a linear model that includes all variables. The attributes are then added or removed from the linear model according to some statistically meaningful criteria. The PLM trees generated are more flexible than traditional trees and are also smaller. The increased complexity introduced when linear models are generated at the leaves is balanced with the decrease of the number of subdivisions of the attribute space.

In a CART tree, each internal node corresponds to a test on the attributes and splits the data into two disjoint regions by means of a question. This question can be stated in terms of only one attribute (univariate splits) or of linear combinations of several attributes (multivariate splits),

$$Q_i = \boldsymbol{c}_i \cdot \boldsymbol{x} > a_i, \tag{2}$$

where the vector $\boldsymbol{c_i}$ contains the coefficients that define a variable which is a linear combination of the original variables. The parameter a_i is the threshold value of the split.

The parameters $(\boldsymbol{c}_i, a_i)$ are determined by the local optimization of a cost function, which in regression is taken to be the mean square error over the examples of the training set,

$$R_{train} = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} (y_n - \bar{y}(\boldsymbol{x}_n))^2, \tag{3}$$

where $\bar{y}(\boldsymbol{x}_n)$ is the prediction of the tree for example $\boldsymbol{x}_n$.

The Boolean test (2) can be reinterpreted in terms of membership functions. Each node t_i is characterized by an absolute membership function $\mu_i(\boldsymbol{x})$ which is equal to one for those examples that satisfy the conjunction of Boolean tests leading to the node and zero otherwise. The relative degree of membership of example $\mathbf{x}$ in node $t_{i\alpha}$, a child node of t_i, is a Boolean function that is equal to 1 if the example satisfies test on node t_i (2) and equal to zero otherwise, independent of the value of $\mu_i(\boldsymbol{x})$. For the t_{iL} left child node of t_i,

$$\mu_L^{(i)}(\boldsymbol{x}) = \theta(\boldsymbol{c}_i \cdot \boldsymbol{x} - a_i), \tag{4}$$

where $\theta(x)$ is the Heavyside step function. For the right child node,

$$\mu_R^{(i)}(\boldsymbol{x}) = \theta(a_i - \boldsymbol{c}_i \cdot \boldsymbol{x}). \tag{5}$$

The absolute degree of membership is given by

$$\mu_{i\alpha}(\boldsymbol{x}) = \mu_i(\boldsymbol{x})\mu_\alpha^{(i)}(\boldsymbol{x}), \quad \alpha = L, R, \tag{6}$$

where $\mu_i(\boldsymbol{x})$ is the absolute degree of membership for the parent node t_i, which can be recursively calculated using (6) until the root node is reached. By definition, all points belong to the root node, so that $\mu_0(\boldsymbol{x}) = 1;\ \forall \boldsymbol{x}$.

Given a vector of attributes $\boldsymbol{x}$, the value predicted by the for the dependent variable is

$$\bar{y}(\boldsymbol{x}) = \sum_{t_l \in \tilde{T}} \mu_l(\boldsymbol{x})\bar{y}_l(\boldsymbol{x}), \tag{7}$$

where $\bar{y}_l(\boldsymbol{x})$ is the linear predictor associated to the terminal node t_l given by (1). The set of terminal nodes is denoted by $\tilde{T}$.

Note that Equation (7) remains valid with real-valued degrees of membership. It is therefore possible to replace the Boolean test (2) by a fuzzy test, characterized by a real-valued membership function. A natural choice for fuzzification is to use sigmoidal functions of inverse width b_i. In this way, the membership functions (4) and (5) are replaced by

$$\mu_L^{(i)}(\boldsymbol{x}) = \frac{1}{1 + exp[-b_i(\boldsymbol{c}_i \cdot \boldsymbol{x} - a_i)]}, \qquad \mu_R^{(i)}(\boldsymbol{x}) \ = \ 1 - \mu_L^{(i)}(\boldsymbol{x}). \tag{8}$$

The splitting threshold of a standard crisp tree is broadened into a splitting band. Outside this band, examples are assigned to one of the child nodes with degree of membership very close to one. It is for the examples that fall within this band that the fuzzy character of the tree becomes important: examples are assigned to both child nodes with relevant degrees of membership. A crisp split can be seen as the limiting case of a fuzzy split when $b_i \to \infty$.

Once the fuzzy structure of the tree has been fixed, the parameters of the splits are adjusted by minimization of a global cost function. In a fuzzy regression tree, the cost function is the mean squared error of the tree predictions in the training set. The optimization problem is similar to that encountered in neural networks, where the problem is solved by the backpropagation algorithm. Here the problem is solved by a similar algorithm in which the estimations at the leaves are propagated from the leaves upwards to the root node.

It is important to notice that a fuzzy tree assigns each example to every leaf node with some degree of membership, which depends on the conjunction of tests leading to the leaf node and which can be calculated by iteration of (6) until the root node is reached. For a leaf node, the prediction for a given vector of attributes $\boldsymbol{x}$ is a linear model given by (1). For an inner node, t_i, it can be defined as the prediction of the subtree $T(t_i)$ (the tree composed of t_i as the root node and all of its descendants) and can be calculated using the recursion

$$\bar{y}_i(\boldsymbol{x}) = \mu_L^{(i)}(\boldsymbol{x})\bar{y}_{iL}(\boldsymbol{x}) + \mu_R^{(i)}(\boldsymbol{x})\bar{y}_{iR}(\boldsymbol{x}). \tag{9}$$

The value predicted by the regression tree for the dependent variable can be obtained by iterating (9) from the predictions at the leaves to the root node, in a manner similar to the backpropagation algorithm in neural networks. This algorithm fixes the structural parameters of the tree by minimizing a global cost function, thus avoiding the greedy approach of traditional crisp trees.

The optimization of the cost function (3) with respect to the parameter α_j of node t_j yields

$$\frac{\partial R_{train}(T)}{\partial \alpha_j} = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} -2(y_n - \bar{y}(\boldsymbol{x}_n)) \frac{\partial \bar{y}(\boldsymbol{x}_n)}{\partial \alpha_j} = 0. \tag{10}$$

For a leaf node t_l, the parameters α_j are the coefficients in the linear fit $\boldsymbol{\beta}_l$. Using the expression

$$\frac{\partial \bar{y}(\boldsymbol{x})}{\partial \boldsymbol{\beta}_l} = \mu_l(\boldsymbol{x})\tilde{\boldsymbol{x}}, \tag{11}$$

where $\tilde{\boldsymbol{x}} = \{1, x_1, \ldots, x_D\}$, equation (10) becomes

$$\sum_{n=1}^{N_{train}} (y_n - \bar{y}(\boldsymbol{x}_n))\mu_l(\boldsymbol{x}_n)\tilde{\boldsymbol{x}}_n = 0, \tag{12}$$

thus resulting in $D+1$ equations. For an inner node, t_i, the results are analogous to those in [21]

$$\sum_{n=1}^{N_{train}} (y_n - \bar{y}(\boldsymbol{x}_n))\mu_i(\boldsymbol{x}_n)(\bar{y}_{iL}(\boldsymbol{x}_n) - \bar{y}_{iR}(\boldsymbol{x}_n))\frac{\partial \mu_L^{(i)}(\boldsymbol{x}_n)}{\partial \boldsymbol{\xi}_i} = 0, \tag{13}$$

where $\boldsymbol{\xi}_i = \{-b_i a_i, b_i c_i\}$ are the parameters of the membership function (8), and

$$\frac{\partial \mu_L^{(i)}(\boldsymbol{x}_n)}{\partial \boldsymbol{\xi}_i} = \tilde{\mathbf{x}}\, \mu_L^{(i)}(\boldsymbol{x})\mu_R^{(i)}(\boldsymbol{x}). \tag{14}$$

The solutions to the system of equations (12) and (13) are the parameters that characterize the optimized PLM tree. In this work, the optimization problem is solved by a quasi-Newton method (the Broyden-Fletcher-Goldfarb-Shanno algorithm [17]). Because the prediction of the tree for any example is given by the combination of the predictions at each of the leaves and not by the individual predictions themselves, the estimations of the parameters of the linear models at the leaves are propagated upwards to the root node in order to obtain the values of $\bar{y}_i(\boldsymbol{x}_n)$ that are needed in the computation of (3) and its derivatives.

4 Experiments

The objective of this section is to show how the design of a global optimization algorithm leads to an improvement in the performance of a regression tree. The algorithm was tested on a variety of data sets, both synthetic and real world data sets. The Housing and Servo are real-world data sets and were obtained from the UCI repository (ftp ics.uci.edu/pub/machine-learning-databases). The synthetic data sets focus on several regression problems in the presence of noise and/or irrelevant data. These data sets have been suggested by Cherkassky and Mulier (see Table 1) [7] and by Friedman [9]. The sets suggested by Friedman are

- Friedman #1: There are ten independent predictor variables $x_1, \ldots, x_{10}$ each of which is uniformly distributed over $[0, 1]$. The response is given by

$$y = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^3 + 10x_4 + 5x_5 + N(0, 1). \tag{15}$$

- Friedman #2, #3: They are both 4-variable data sets with

$$\begin{aligned} y &= (x_1^2 + (x_2 x_3 - (1/x_2 x_4))^2)^{1/2} + \epsilon_2 \\ y &= tan^{-1}\left(\frac{x_2 x_3 - (1/x_2 x_4)}{x_1}\right) + \epsilon_3, \end{aligned} \tag{16}$$

  where the variables are uniformly distributed over the ranges

$$0 \le x_1 \le 100, \quad 20 \le (x_2/2\pi) \le 280, \quad 0 \le x_3 \le 1, \quad 1 \le x_4 \le 11. \tag{17}$$

  The noise variables ϵ_2, ϵ_3 are distributed as $N(0, \sigma_2^2), N(0, \sigma_3^2)$ with σ_2, σ_3 selected to give 3:1 signal/noise ratios.

Table 1. Data set summaries for the first experiment (from Ref. [7]).

Set	Attributes $\mathbf{x}$	Function $f(\mathbf{x})$	Range	σ_{noise}
1	$x_1 = a^2$, $x_2 = b^2$, $x_3 = \cos(a^2 + b^2)$	$a + b$	$a, b \in [0, 1]$	0.1
2	$x_1 = a^2$, $x_2 = \sin b$, $x_3 = \cos(a^2 + b^2)$	$a\ b$	$a, b \in [0, 1]$	0.1
3	$x_1 = a^2$, $x_2 = (a^2 - 0.5)^2$, $x_3 = \cos(a^2 + b^2)$	a	$a, b \in [0, 1]$	0.1
4	$x_1 = \sin 2\pi a$, $x_2 = \cos 2\pi a$	a	$a \in [0, 1]$	0.1
5	$x_1 = a$, $x_2 = \sin a$	$\cos a$	$a \in [-1, 1]$	0.05
6	$x_1 = a$, $x_2 = a^2$	$(1 - 0.5(a^2 + a^4))^{\frac{1}{2}}$	$a \in [-1, 1]$	0.05
7	$x_1 = x_2 = x_3 = a$	$(1 - a^2)^{\frac{1}{2}}$	$a \in [-1, 1]$	0.05
8	$x_1 = x_3 = a$, $x_2 = \cos a$, $x_4 = a^2$, $x_5 = a^3$	$\sin a$	$a \in [-1, 1]$	0.05
9	$x_1 = a$, $x_2 = \sin 2\pi a$	$\cos 2\pi a$	$a \in [-1, 1]$	0.05

Table 2. Data set summaries for the second experiment.

Set	Size	No. Inputs	N_{train}	N_{test}
Housing	506	13	455	51
Servo	167	4	150	17
Friedman #1	1200	10	200	1000
Friedman #2	1200	4	200	1000
Friedman #3	1200	4	200	1000

Both CART and the stepwise algorithm used to select the number of PLM variables have a tendency to prefer simple hypotheses. Following this philosophy, the global optimization algorithm only tunes the parameters corresponding to those variables that appear in the tree in the splits or in the linear models at the leaves. However, other possibilities are being considered.

Table 3 summarizes experiments with data sets with ($N_{train} = N_{test} = 300$) where the performance of the algorithm designed in this work is compared to different extensions of CART. The values reported are the root mean square errors normalized by the standard deviation of the realization of the noise. With this normalization, a perfect predictor would achieve a value of one. We observe that our algorithm gives very similar predictions than fuzzy CART, though for 7 of the data sets the predictions of our algorithm are better than the other extensions to CART. For the other two data sets (4 and 9), the results are within one standard deviation of the best result.

The decrease in size is remarkable. This is a consequence of the stepwise PLM character of the tree. The increased complexity of the predictions at the leaves is compensated with a reduction in the decision nodes, which implies a reduction in size. The size for the fuzzy versions is the same than for the non-fuzzy ones and is indicated in the third and sixth columns of Table 3.

The prediction quality is much better in the fuzzy versions of the CART algorithm. This is due to several advantages, namely:

Table 3. Results with $N_{train} = N_{test} = 300$.

Set	CART	Fuzzy CART	Size	CART+PLM	Fuzzy CART+PLM	Size
1	1.48 (0.06)	1.09 (0.03)	9.6 (1.8)	1.13 (0.03)	1.09 (0.06)	2.9 (0.3)
2	1.28 (0.07)	1.06 (0.03)	7.9 (1.1)	1.09 (0.04)	1.04 (0.01)	2.4 (0.7)
3	1.11 (0.04)	1.03 (0.03)	8.2 (0.9)	1.03 (0.02)	1.03 (0.02)	2.0 (0.0)
4	1.31 (0.13)	1.11 (0.08)	6.1 (2.0)	1.18 (0.14)	1.13 (0.09)	2.0 (0.0)
5	1.21 (0.07)	1.02 (0.02)	8.1 (1.3)	1.03 (0.02)	1.01 (0.01)	2.0 (0.0)
6	1.29 (0.08)	1.03 (0.02)	6.5 (0.7)	1.09 (0.04)	1.02 (0.01)	2.3 (0.5)
7	1.54 (0.11)	1.03 (0.02)	8.4 (1.1)	1.14 (0.05)	1.03 (0.02)	5.5 (1.3)
8	1.42 (0.11)	1.05 (0.04)	11.3 (0.9)	1.05 (0.03)	1.02 (0.01)	3.1 (1.7)
9	3.84 (0.30)	1.06 (0.03)	12.3 (0.8)	1.90 (1.25)	1.11 (0.09)	9.0 (4.0)

- The tree parameters are found through the optimization of a global cost function, in contrast to the greedy strategy used in the construction of standard decision trees.
- The functions are approximated continuously. The CART/PLM tree tries to adjust a linear model to the function in each region. There are no constraints between the boundaries of the different regions, thus yielding a discontinuous representation. This is not the case now. This also limits the tendency to overfitting.
- The notion of locality is recovered. Due to the introduction of smooth splits, data points very close to each other in the attribute space will have similar degrees of membership for each leaf.

In a second group of experiments, the performance of globally optimal CART regression trees is compared to ensemble algorithms, such as bagging [2] and smearing [5]. The data sets used in the second experiment are described briefly in Table 2, together with the training and test sizes for each of them. The numbers reported in Table 4 are averages over 10 independent runs of the algorithm and show the mean generalization error with its standard deviation between parentheses. The algorithm designed in this work is significantly better than bagging for all the data sets considered and better than smearing for 4 datasets, being second only in Friedman #1, but within one standard deviation from the best result. For the Servo dataset, the results from [5] are not shown due to unresolved discrepancies in the values [6].

The performance of Fuzzy CART trees with Linear Models at the leaves trees (Fuzzy CART+PLM) is systematically better than a Fuzzy CART tree alone (Fuzzy CART), although in some cases the results are not statistically significant. The reduced size of the trees generated and thus the interpretability of the predictions is also a factor to take into account.

Table 4. Root mean-square error estimates.

Data set	Fuzzy CART	Fuzzy CART+PLM	Bagging	Smearing
Housing	3.4 (0.3)	3.1 (0.3)	3.26	3.21
Servo	0.6 (0.2)	0.4 (0.2)	-	-
Friedman #1	2.5 (0.2)	2.30 (0.09)	2.50	2.24
Friedman #2	136 (9)	134 (7)	146	149
Friedman #3	0.15 (0.02)	0.149 (0.002)	0.158	0.153

5 Summary and Conclusions

A backpropagation algorithm to generate decision trees optimized with a global cost function has been applied to the problem of approximating an unknown real-valued function from data. The method takes as a starting point a CART regression tree extended with linear models at the leaves. In order to make optimization by analytic methods possible, the standard crisp splits of a CART tree are replaced by sigmoidal continuous splits. The predictions of the model are propagated from the leaves upward towards the root node in order to the calculate the derivatives of the global cost function used in the optimization procedure. This backpropagation algorithm makes it possible to adjust the parameters of the tree so that, given the fixed tree structure, a minimum of the cost function is reached.

The introduction of continuous splits can be seen as giving the tree a fuzzy character. In regression problems, fuzzification has the advantage of yielding a continuous approximation to the unknown function recovering the notion of locality lost in the traditional decision-tree approximation. It also enlarges the expressive capabilities of the tree incorporating in the construction of symbolic learners the robustness and flexibility of connectionist ones. However, some interpretability is lost with respect to the clarity of a single tree.

The experiments carried out both in synthetic and real-world datasets show that the algorithm designed leads to a significant improvements with respect to the performance of standard CART trees, remaining robust to noise and irrelevant attributes. Its performance is also better than methods that involve a much greater number of parameters such as bagging and smearing.

References

1. Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J.: Classification and Regression Trees. Chapman & Hall, New York (1984).
2. Breiman, L., Bagging Predictors: Machine Learning, **24** (1996) 123-140.
3. Breiman, L.: Bias, Variance and Arcing Classifiers. Technical Report 460, Statistics Department, University of California, (1996).
4. Breiman, L.: Arcing Classifiers (with Discussion). The Annals of Statistics, **24** (1998) 2350-2383.

5. Breiman, L.: Randomizing Outputs to Increase Prediction Accuracy. Machine Learning, **40** (2000) 229-242.
6. Breiman, L.: Private Communication.
7. Cherkassky, V. and Muller, F.: Statistical and Neural Network Techniques for Nonparametric Regression. In: Cheeseman, P.W. and Oldford, R.W. (eds.): Selecting Models from Data. Springer-Verlag, New York (1994) 383-392..
8. Freund, Y. and Schapire, R.E.: Experiments with a New Boosting Algorithm. In Machine Learning: Proc. 13th International Conference. Morgan-Kaufmann, San Francisco (1996) 148-156.
9. Friedman, J.H.: Multivariate Adaptative Regression Splines (with Discussion). The Annals of Statistics, **19** (1991) 1-141.
10. Gelfand, S.B., Ravishankar, C.S. and Delp, E.J.: An Iterative Growing and Pruning Algorithm for Classification Tree Design. IEEE Trans. Pattern Analysis and Machine Intelligence, **13**, 2 (1991) 163-174.
11. Geman, S., Bienenstock, E. and Doursat, R.: Neural Networks and the Bias/Variance Dilemma. Neural Computation, **4** (1992) 1-58.
12. Quinlan, J. R. Learning with continuous classes, Proceedings of the Australian Joint Conference on Artificial Intelligence (1992) 343-348.
13. Jennrich, R.E.: Stepwise Regression. In: Statistical Methods for Digital Computers. Wiley, New York (1977) 58-75.
14. Jordan, M.I. and Jacobs, R.A.: Hierarchical Mixtures of Experts and the EM algorithm. Neural Computation, **6** (1994) 181-214.
15. Kohavi, R. and Wolpert, D.H.: Bias Plus Variance Decomposition for Zero-one Loss Functions. In Machine Learning, Proc. 13th International Conference. Morgan-Kaufmann, San Francisco (1996) 275-283.
16. Kong, E.B. and Dietterich, T.G.: Error-correcting Output Coding Corrects Bias and Variance. In Proc. 12th International Conference on Machine Learning. Morgan-Kaufmann, San Francisco (1995) 313-321.
17. Press, W. Teukolsky, W.T., Vetterling, S.A. and Flannery, B.: Numerical Recipes in C: The Art of Scientific Computing. Cambridge Univ. Press, Cambridge (1993).
18. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993).
19. Quinlan, J.R.: Induction of Decision Trees. Machine Learning **1**, 1 (1986) 81-106.
20. Schapire, R.E., Freund, Y. Bartlett, P. and Lee, W.S.: Boosting the Margin: a New Explanation for the Effectiveness of Voting Methods. The Annals of Statistics **26** 5 (1998) 1651-1686.
21. Suárez, A. and Lutsko, J.F.: Globally Optimal Fuzzy Decision Trees for Classification and Regression. IEEE Trans. Pattern Analysis and Machine Intelligence **21** 12 (1999) 1297-1311.
22. Suárez, A. and Lutsko, J.F.: Automatic Induction of Piecewise Linear Models with Decision Trees. In Proc. International Conference on Artificial Intelligence, Vol 2. H.R. Arabnia ed. Las Vegas, (2000) 1025-1031.