Globally Optimal Fuzzy Decision Trees for Classification and Regression

Alberto Suárez and James F. Lutsko

Abstract—A fuzzy decision tree is constructed by allowing the possibility of partial membership of a point in the nodes that make up the tree structure. This extension of its expressive capabilities transforms the decision tree into a powerful functional approximant that incorporates features of connectionist methods, while remaining easily interpretable. Fuzzification is achieved by superimposing a fuzzy structure over the skeleton of a CART decision tree. A training rule for fuzzy trees, similar to backpropagation in neural networks, is designed. This rule corresponds to a global optimization algorithm that fixes the parameters of the fuzzy splits. The method developed for the automatic generation of fuzzy decision trees is applied to both classification and regression problems. In regression problems, it is seen that the continuity constraint imposed by the function representation of the fuzzy tree leads to substantial improvements in the quality of the regression and limits the tendency to overfitting. In classification, fuzzification provides a means of uncovering the structure of the probability distribution for the classification errors in attribute space. This allows the identification of regions for which the error rate of the tree is significantly lower than the average error rate, sometimes even below the Bayes misclassification rate.

Index Terms—Automatic learning, decision trees, fuzzy set theory, global optimization, backpropagation, nonparametric regression, classification.

1 Introduction

The use of fuzzy set theory and fuzzy membership functions [1], [2] in decision trees for classification dates back to early work by Chang and Pavlidis [3] and by Zadeh (in [4], where a decision tree is referred to as a branching questionnaire). These papers are devoted to the design of efficient algorithms that extract a class prediction for an input from a given fuzzy decision tree. In more recent work [5], [6], the focus has shifted to the the problem of automatic induction from a set of training data of the fuzzy classification tree itself. This task is of practical relevance because a fuzzy decision trees can be used in the automatic design of a knowledge-based representation containing linguistic variables and fuzzy rules, replacing the currently used heuristic fuzzy-rule search methods [7], [8], [9], [10].

The present research is inspired by the role that fuzzification has played in the problem of clustering, which may be viewed as a form of unsupervised learning: The classical, crisp clustering problem is to assemble a set of points into a number of groups (clusters) so as to minimize some cost function. This is essentially a combinatoric problem. Fuzzifying this problem by allowing a point's membership in a cluster to be a real-valued variable transforms the crisp, combinatoric problem into an analytic problem which can be attacked with all the machinery of analytic calculus.

- *A. Suárez is with the Escuela Técnica Superior de Informática, Universidad Autónoma de Madrid, Ctra. de Colmenar Viejo, Km. 15, E-28049 Madrid, Spain. E-mail: alberto.suarez@ii.uam.es.*
- *J.F. Lutsko is with i2 Technologies, Airway Park, Lozenberg 23, B-1932 Sint-Stevens-Woluwe, Belgium.*

Manuscript received 29 July 1998; revised 12 Oct. 1999.
Recommended for acceptance by I. Sethi.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107867.

In the area of supervised learning, a strong dichotomy exists between symbolic approaches, such as the generation of decision trees, and subsymbolic approaches, most notably, connectionist methods like neural networks [11]. Both types of model have their advantages and drawbacks: Decision trees are, by their nature, readily interpretable and well-suited to classification problems. They are a less natural model for regression and suffer from a well-known sensitivity to the data used in their construction. A further advantage of decision trees is that there exist a number of efficient algorithms that are able to find near-optimal architectures for the tree. Connectionist models are generally analytic so that powerful algorithms are available for determining their parameters, they are robust, and they are naturally suited for regression or binary classification problems. However, they are usually difficult to interpret and are thus less useful when explanation, rather than prediction, is the goal. Understanding the relation between these two types of models is one of the motivations that guides this investigation of fuzzy decision trees.

In the present work, we extend the existing procedures for the generation of decision trees in order to allow the possibility of partial memberships in the nodes of the decision tree. This reformulation of the tree construction algorithm in terms of fuzzy degrees of membership makes it possible to employ analytic tools in the construction of decision trees that are globally optimal. In particular, it allows the design of an elegant algorithm, analogous to backpropagation in a neural network, that determines the parameters of the membership functions in a global manner. The starting point for the generation of fuzzy decision trees is a CART tree [12], whose crisp (Boolean) splits are replaced by fuzzy sigmoidal splits. The parameters of the fuzzy splits are then determined by the global

optimization of a cost function. The learning procedure involves the propagation of the tree estimates from the leaves of the tree to the root node. Since this backpropagation takes place directly on the structure of tree itself, it maintains the knowledge representation provided by the decision tree, which is of fundamental value when we are interested not only in performance, but also in articulating a representation that is intelligible.

Some of the elements of the present work have already been addressed separately in the literature: Soft decisions have been introduced in the work of Schuermann and Doster [13], Quinlan [14], Jordan and Jacobs [15], and Sethi [16]. In [17], Wang and Suen use a fixed fuzzification scheme for the splits in order to improve the efficiency of a decision tree classifier. They then use a global training algorithm in order to generate extended terminal nodes that improve the performance of the classifier. Note, however, that, in this work, the parameters of the fuzzy splits are not modified during the global training. The use of CART in the design of fuzzy inference systems is discussed in [18].

Previous work on fuzzy decision trees (see, for instance, Janickow [6]), is focused on extensing the capabilities of the decision tree so that it is able to perform approximate reasoning with linguistic variables. In contrast to this goal, the objective of the present research is to use fuzzy set theory in the construction of decision trees in order to incorporate into a symbolic method the flexibility characteristic of connectionist approaches.

There have also been some efforts directed toward the integration of symbolic and subsymbolic approaches. Of particular relevance are the work initiated by Sethi on entropy nets [19], [16], [20], and the tree-structured hierarchical mixture of experts proposed by Jordan and Jacobs [15]. In Sethi's work, [19], [16], [20], a mapping between a decision tree and a perceptron with two hidden layers is proposed. (In fact, it is also possible to construct a mapping from the decision tree to a single hidden layer perceptron, as shown by Park in [21].) Making use of this correspondence, the knowledge representation articulated by the decision tree is translated into the architecture of a neural network whose connections can then be retrained by a backpropagation algorithm. This procedure has the advantage over the usual connectionist methods that it provides an automatic procedure to determine the architecture of the network. Furthermore, the initial values of the network weights are also fixed by the tree-generation algorithm. This initialization speeds the training of the network and, in some cases, prevents the backpropagation algorithm from getting trapped in local minima [21]. In more recent work, Sethi and Yoo [20] exploit this mapping in order to generate decision trees whose hierarchy of tests is determined in a global fashion by a neural learning algorithm. Their starting point is a fully developed binary tree of a fixed depth whose internal nodes are initially empty. The leaves of the tree are labeled before the learning process starts. The decision tree is then converted into a multilayer network whose weights are determined by a learning scheme that uses backpropagation of the errors on the auxiliary network. This scheme also involves a softened form of competition between the output units corresponding to the terminal nodes of the tree in such a way that the tree decision hierarchy is established in the course of the training. Thus, the multivariate splits corresponding to the inner nodes of the tree are determined simultaneously in a global fashion. The size and the effective architecture of the final network can be controlled only in an indirect fashion by tuning two parameters: the original depth of the tree, and a growth factor, which limits the number of wins that a particular output layer unit of the network may accumulate. The a priori limitation in the depth of the decision tree may hinder the performance in problems where deep, but not very complex, trees are the optimal architecture. In the present work, the global optimization problem is solved by designing a backpropagation algorithm on the tree itself without having to resort to an auxiliary neural network. This has the advantage that it is not necessary to implement any sort of competitive learning to enforce the decision hierarchy. The initial values of the fuzzy splits are given by those of the crisp tree, which have been obtained by a series of local optimizations presumably leading to a nearly optimal solution to the inductive learning problem. By using the skeleton of the CART tree, we take advantage of the well-studied heuristics developed to extract the tree architecture directly from the training data and also avoid limiting a priori the final size, depth, or complexity of the tree.

In the tree-structured hierarchical mixture of experts proposed by Jordan and Jacobs [15], a neural learning algorithm is also used to determine the parameters of the classifier, but little guide is given as to how to the architecture of the mixture should be chosen.

The organization of the paper is: In Section 2, the generation of crisp decision trees is recast in terms of membership functions. This reformulation permits the fuzzification of the crisp trees, which is described in Section 3. Sections 4 and 5 are devoted to the formulation of the backpropagation algorithm for regression and classification, respectively. The results of experiments to demonstrate the performance of the globally optimal fuzzy decision trees are also presented in these sections. Finally, Section 6 summarizes the results and conclusions of this work.

2 Construction of a Crisp Decision Tree

The objective of the present research is to take advantage of extensions of the concept of "membership of a point $\mathbf{x}$ to a set A" beyond the Boolean paradigm (either $\mathbf{x} \in A$ or $\mathbf{x} \ni A$) in order to improve the automatic construction of binary trees designed to solve regression and classification problems.

Traditional set theory considers only crisp sets: The degree of membership of a point $\mathbf{x}$ to a set A is given by the Boolean variable $\mu_A(\mathbf{x})$, which can take only the values 1, if $\mathbf{x} \in A$, and 0, if $\mathbf{x} \ni A$. Fuzzy set theory [1], [2] expands the concept of "belonging to a set" by introducing the possibility of partial membership of a point to the set in question. In particular, the degree of membership of a point $\mathbf{x}$ to a fuzzy set A is allowed to take any real value between 0 and 1, i.e., $\mu_A(\mathbf{x}) \in [0, 1]$. By eliminating the Boolean constraint for the set memberships, we improve

the knowledge representation capabilities of the decision tree, incorporating into its construction the flexibility that is characteristic of neural networks. This should in turn lead to an enhanced performance in terms of robustness and quality of results of the trees used to tackle regression and classification problems.

There are various approaches to the construction Boolean decision trees [12], [22], [23]. There also exist many possible strategies to incorporate fuzziness in the architecture of decision trees. We choose to reduce the arbitrariness in constructing fuzzy decision trees by demanding that several properties of crisp decision trees are recovered as the memberships become Boolean. In order to elucidate how this requirement can be met, as well as to fix notation, we briefly review the construction of crisp decision trees. We follow the method CART (Classification And Regression Trees) developed by Breiman et al. [12].

For the sake of concreteness, we assume that there are D predictor variables (or independent variables) and that they are continuous, i.e., $\mathbf{x} \equiv \left\{x^{(d)}\right\}_{d=1}^{D} \in S \subset \mathcal{R}^D$, where S has finite support. The inclusion of nominal variables presents no difficulties except that we do not consider the possibility of fuzzifying nominal splits. The dependent variable, whose value is the objective of the prediction, can be either nominal (classification) or real-valued (regression).

We have at our disposal a training set consisting of N_{train} labeled examples $\{(\mathbf{x}_n, y_n); n = 1, \ldots, N_{train}\}$, where both the predictor variables (or attributes) and the dependent variable have been recorded (in a database, for example). The training set is used to build a tree (T), composed of a collection of nodes $(t_i \in T; i = 0, 1, 2, \ldots)$ arranged in a hierarchical manner. CART uses a top-down approach to build a binary tree. The first node of a CART tree T is the root node, denoted by t_0. By definition, all examples are assigned to this node. The tree is constructed by a divide-and-conquer strategy in which the attribute space is partitioned by a hierarchy of Boolean tests into a set nonoverlapping regions, in each of which the decision problem is simpler. Each of the tests in this hierarchy corresponds to an internal node of the decision tree.

Assume that the tree has been grown up to a given configuration and that, at this stage, node t_i is a terminal node of the tree. This node is characterized by a membership function $\mu_i(\mathbf{x})$ which is equal to 1 for those examples $\mathbf{x}$ that satisfy the conjunction of Boolean tests leading to t_i, and 0, otherwise. The number of examples from the training set in node t_i is

$$N_i = \sum_{n=1}^{N_{train}} \mu_i(\mathbf{x}_n). \tag{1}$$

In a regression problem, node t_i gives a prediction of the value of the dependent variable equal to the average value of this variable y for the elements of the training set assigned to t_i

$$\bar{y}_i = \frac{1}{N_i} \sum_{n=1}^{N_{train}} \mu_i(\mathbf{x}_n) y_n. \tag{2}$$

In classification problems, a class label is associated to node t_i according to a majority rule:

$$\bar{y}_i = argmax_k \left\{ N_i^{(k)} \right\}, \tag{3}$$

where

$$N_i^{(k)} = \sum_{n=1}^{N_{train}} \mu_i(\mathbf{x}_n) \delta(y_n, k) \tag{4}$$

is the number of elements of class k from the training set assigned by the tree to node t_i.

In order to obtain a better prediction for the dependent variable, we can split the data in t_i into two subsets by means of a question Q_i. For continuous attributes, the space of questions that is explored is given by the (D+1)-parameter family:

$$Q_i \equiv \mathbf{c}_i \cdot \mathbf{x} > a_i, \tag{5}$$

where $\mathbf{c}_i \cdot \mathbf{x} = \sum_{d=1}^{D} c_i^{(d)} x^{(d)}$. The vector $\mathbf{c}_i \equiv \left\{ c_i^{(d)} \right\}_{d=1}^{D} \in \mathcal{R}^D$ contains the coefficients that define a new composite variable that is the linear combination of the primitive variables. The parameter $a_i \in \mathcal{R}$ is the threshold value of the split. It defines a hyperplane $\mathbf{c}_i \cdot \mathbf{x} = a_i$, perpendicular to the axis of the composite variable $\mathbf{c}_i \cdot \mathbf{x}$, along which the segmentation of the space of attributes is made. Relation (5) splits the data into two disjoint sets, each of which is assigned to one the child nodes of the node t_i. In this manner, node t_i becomes an internal node, with two child nodes associated to it.

The Boolean test (5) can be expressed in terms of the relative degree of membership for the left child node

$$\mu_L^{(i)}(\mathbf{x}) = \theta(\mathbf{c}_i \cdot \mathbf{x} - a_i), \tag{6}$$

where the Heaviside step function is defined as

$$\theta(z) = \begin{cases} 1 & \text{if} \quad z > 0 \\ 1/2 & \text{if} \quad z = 0 \\ 0 & \text{if} \quad z < 0. \end{cases} \tag{7}$$

The case $z = 0$ does not occur in practice and the degrees of membership are Boolean variables. For the right node,

$$\mu_R^{(i)}(\mathbf{x}) = 1 - \theta(\mathbf{c}_i \cdot \mathbf{x} - a_i) = \theta(a_i - \mathbf{c}_i \cdot \mathbf{x}). \tag{8}$$

The absolute degree of membership is given by

$$\mu_{i\alpha}(\mathbf{x}) = \mu_i(\mathbf{x}) \mu_\alpha^{(i)}(\mathbf{x}), \qquad \alpha = L, R, \tag{9}$$

where $\mu_i(\mathbf{x})$ is the absolute degree of membership for the parent node, t_i, which can be calculated by recursion of (9), until the root node is reached. All points belong to the root node and, therefore,

$$\mu_0(\mathbf{x}) = 1.0, \ \ \forall\, \mathbf{x}. \tag{10}$$

This recursion corresponds to the succession of splits produced by the answers to the corresponding series of tests connecting the root node to node t_i. Note the relative memberships of a pair of child nodes add up to unity

$$\mu_R^{(i)}(\mathbf{x}) + \mu_L^{(i)}(\mathbf{x}) = 1, \tag{11}$$

and that their absolute memberships add up to the absolute membership of the parent node

$$\mu_{iR}(\mathbf{x}) + \mu_{iL}(\mathbf{x}) = \mu_i(\mathbf{x}). \tag{12}$$

The values of parameters $(\mathbf{c}_i, a_i)$ defining the question Q_i (5) are determined by the local optimization of a cost function. In regression, one chooses the parameters for which the reduction mean square error of the tree over the training set is largest. In classification, the CART method selects the split that leads to the largest decrease in the impurity of the tree (Gini criterion). Other local optimization criteria may also used (e.g., information gain, information-gain ratio [24]).

The set of leaf or terminal nodes $\{t_l\}$ is denoted by $\tilde{T}$, and its cardinality (i.e., the number of terminal nodes) by $|\tilde{T}|$. In terms of this parameter, the total number of nodes of the tree is $2|\tilde{T}| - 1$ and the number of inner nodes is $|\tilde{T}| - 1$. This set $\tilde{T}$ is used for the actual predictions: Suppose that we intend to classify an example characterized by the vector of attributes $\mathbf{x}_{test}$. We evaluate the succession of tests from the root node, following a path that is determined by the results of those tests at each of the internal nodes. Eventually this path leads us to one of the terminal nodes, say t_l. Equation (9) can be used recursively to find the degree of membership of $\mathbf{x}_{test}$ to the leaf node t_l.

In a regression problem, the value of y assigned by the tree to a given vector of attributes, $\mathbf{x}_{test}$, is

$$\bar{y}(\mathbf{x}_{test}) = \sum_{t_l \in \tilde{T}} \mu_l(\mathbf{x}_{test}) \bar{y}_l, \tag{13}$$

where $\bar{y}_l$ is given by (2) and, by construction, only one of the $\{\mu_l(\mathbf{x}_{test}); t_l \in \tilde{T}\}$ is equal to unity, the rest being zeros. The error rate of the tree on the training set is

$$R_{train}(T) = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} (y_n - \bar{y}(\mathbf{x}_n))^2. \tag{14}$$

For classification problems, the class label assigned to $\mathbf{x}_{test}$ is

$$\bar{y}(\mathbf{x}_{test}) = \bar{y}_l, \quad \text{if} \quad \mu_l(\mathbf{x}_{test}) = 1. \tag{15}$$

Assuming equal cost for all misclassified examples, the classification error as estimated from the training set is equal to the ratio of points in the training set misclassified by the tree

$$R_{train}(T) = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} (1 - \delta(\bar{y}(x_n), y_n)). \tag{16}$$

Unless we establish stopping criteria, the growing process of the decision tree may continue until $R_{train}(T) = 0$. There is, however, an optimum-sized tree beyond which, in spite of the fact that $R_{train}(T)$ continues to decrease monotonically, the quality of the prediction deteriorates (i.e., the true error rate, or an unbiased estimate, such as the one given by the error rate evaluated on a test set, increases). In order to avoid over-fitting the tree to the data in the training set, and to obtain the optimally-sized tree, several strategies have been proposed. The most successful ones involve overgrowing the tree and then pruning, or eliminating, the branches that contain nonsignificant splits. There are a wide variety of pruning methods [25], from simple ones that use an independent pruning set to estimate the true error rate (Reduced Error Pruning) to more sophisticated methods using Cross Validation [12].

3 Fuzzification of a Crisp Regression Tree

The introduction of membership functions in (1)-(16) sets the basis for fuzzification, provided that we allow the degrees of membership to assume nonintegral values in the range $[0, 1]$. One possible approach to the automatic generation of fuzzy decision trees consists of allowing fuzzy splits to compete with crisp splits at each node: The split chosen would be the one that yields the best local improvement of the corresponding cost function, irrespective of whether it is a crisp or a fuzzy split. The formulas derived in the previous section are valid also with fuzzy splits and could, in principle, be used to generate fuzzy decision trees. However, empirically, we have observed that fuzzification at this stage, considering only sigmoidal fuzzy splits (a restricted class of fuzzy splits that will be discussed in the course of this section), is not likely to improve the quality of the regression or the classification in a significant manner. We believe that this behavior can be accounted for by the local nature of the optimizations carried out in the process of generating the decision tree: Fuzzy rules are, by their nature, only meaningful taken as a whole, globally. It is natural that a local method developed for constructing disjoint rules fails to find good, mutually dependent fuzzy rule sets. Instead of this direct approach, we have found it preferable to introduce the fuzzy character once the tree architecture has been fixed by means of a standard decision tree generator (e.g., CART). This procedure is analogous to first fixing the architecture and then finding the parameters in a neural network, with the advantage that, here, the architecture is automatically generated by the data.

In order to fuzzify the crisp decision tree, we make the observation that a crisp test can be thought of as the limiting case of a fuzzy test. In particular, consider the crisp split associated to the inner node t_i, which is given by the question $\mathbf{c}_i \cdot \mathbf{x} > a_i$. This test results in the splitting of the data in t_i into two disjoint subsets characterized by the relative membership functions (6), (8). We can think of replacing the Heaviside step-functions in these membership functions with a sigmoidal function of inverse width b_i

$$\begin{aligned} \mu_L^{(i)}(\mathbf{x}) &= \frac{1}{1 + \exp[-b_i(\mathbf{c}_i \cdot \mathbf{x} - a_i)]}, \\ \mu_R^{(i)}(\mathbf{x}) &= \frac{1}{1 + \exp[b_i(\mathbf{c}_i \cdot \mathbf{x} - a_i)]} = 1 - \mu_L^{(i)}(\mathbf{x}). \end{aligned} \tag{17}$$

Equations (17) correspond to a fuzzy sigmoidal split on the composite variable $\mathbf{c}_i \cdot \mathbf{x}$ centered at a_i, with an inverse width given by b_i. The splitting threshold is thus broadened into a splitting band. Outside this band, the examples from the data are assigned to one of the child nodes with a degree of membership close to unity (i.e., they behave almost as if the split were crisp). Examples that fall within the splitting

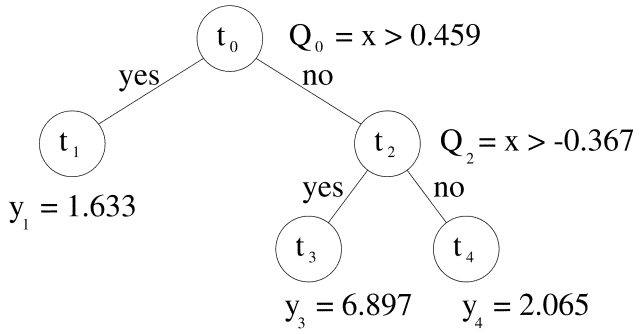

Fig. 1. Crisp decision tree for the regression of a Lorentzian generated by CART with the restriction that there be a minimum of 20 points in the terminal nodes.

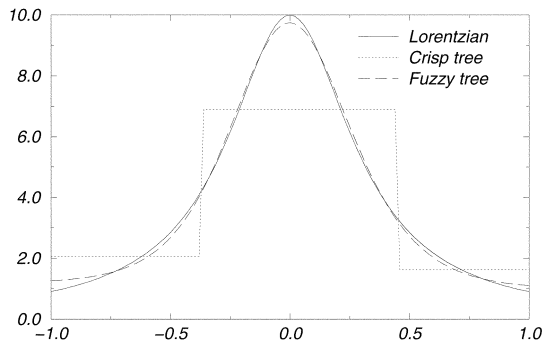

Fig. 2. Regression curves proposed by crisp (dotted line) and fuzzy (dashed line) trees for the regression of a Lorentzian (full line).

zone are assigned with significant degrees of membership to both child nodes. The inverse of the parameter b_i measures the width of this region. A crisp split can be thought of as the limit of a sigmoidal fuzzy split where the parameter b_i is very large. At the other extreme is a split where $b_i = 0$. Such a split assigns all examples in the parent node to both child nodes with equal degrees of membership. The parent node is thus replicated into two equivalent nodes and the global effect is tantamount to having no split at all.

In order to illustrate the idea, consider the regression problem of fitting the Lorentzian

$$y = \frac{1}{0.1 + x^2}, \tag{18}$$

from 100 points (x, y), uniformly distributed at random in the interval $[-1, 1]$, by means of a decision tree. The crisp tree generated by CART is depicted in Fig. 1. Based upon this tree architecture, we generate a fuzzy tree by replacing the crisp splits by fuzzy splits whose parameters are summarized in Table 1. These parameters are determined with the help of a global optimization algorithm that is discussed later on in this paper. Fig. 2 shows the regression curves proposed by the crisp tree (dotted line) and by the fuzzy one (dashed line). The crisp tree divides the interval $[-1, 1]$ into three segments. For each of these segments, the prediction for the value of y is equal to the average of the dependent variable in that segment. It is clear that the constraints used in the generation of the crisp tree are too rigid and that they limit the representation capacity of the tree. Allowing the use of fuzzy tests for the splits leads to a marked improvement in the quality of the regression. This improvement can be ascribed to the augmented expressive power and flexibility of the fuzzy decision tree.

Once we have established the manner in which the fuzzy character is introduced in the structure of the decision tree, we are left with the problem of specifying a procedure to determine the set of parameters defining the tree in an efficient manner. This objective is accomplished by a backpropagation algorithm that solves the problem of optimizing the corresponding cost function for the tree as a whole. The details of the implementation of this backpropagation algorithm are different for regression and for classification problems. Hence, these two cases will be treated separately.

4 Regression

The objective of this section is to design an algorithm to estimate the parameters of a fuzzy regression tree by the minimization of the mean square error, as estimated on the training set (14). According to the fuzzification procedure

TABLE 1
Parameters of Regression Tree for Fitting a Lorentzian

Parameters	Crisp tree	Fuzzy tree
$Q_0 \equiv (a_0, b_0)$	(0.459, ∞)	(-0.017, 5.599)
$Q_2 \equiv (a_2, b_2)$	(-0.367, ∞)	(-0.019, 6.267)
d_1	1.633	0.991
d_3	6.897	35.472
d_4	2.065	1.194

discussed in the previous section, the crisp split at the inner node of a CART tree is replaced by a fuzzy split, defined by the relative membership functions

$$\mu_L^{(i)}(\mathbf{x}) = \frac{1}{1+\exp\{-\boldsymbol{\xi}_i \cdot \tilde{\mathbf{x}}\}} \qquad \mu_R^{(i)}(\mathbf{x}) = \frac{1}{1+\exp\{\boldsymbol{\xi}_i \cdot \tilde{\mathbf{x}}\}}. \tag{19}$$

with

$$\tilde{\mathbf{x}} = \left\{1, x^{(1)}, x^{(2)}, \ldots, x^{(D)}\right\}, \quad \boldsymbol{\xi}_i = \{-b_i a_i, b_i \mathbf{c}_i\}. \tag{20}$$

In contrast to the crisp case, a fuzzy decision tree assigns each example to several leaf nodes with different degrees of membership. Given a vector of attributes $\mathbf{x}$, the value predicted for the dependent variable is a combination of the predictions at the leaves

$$\bar{y}(\mathbf{x}) = \sum_{t_l \in \tilde{T}} \mu_l(\mathbf{x}) d_l, \tag{21}$$

where $\mu_l(\mathbf{x})$ is the membership of the point to the terminal node t_l, which yields a prediction d_l for the dependent variable.

The optimization problem is reminiscent of that encountered in a neural network with hidden layers, where one has to deal with the credit assignment problem (i.e., in what measure does a given node contribute to the global error). For neural networks, the problem is solved by propagating errors backward in the network (the backpropagation algorithm) [26]. Similarly, for a fuzzy decision tree, the problem can be solved by a global optimization algorithm in which the estimations at the leaves are propagated upward in the tree to the root node.

Let $\bar{y}_i(\mathbf{x}_n)$ denote the partial estimate of y at any node t_i for the point $\mathbf{x}_n$. For a leaf node $t_l \in \tilde{T}$, this quantity is some number d_l. In the case of a crisp tree, the value of d_l is equal to $\bar{y}_l$, which is computed from the training set through (2). For a fuzzy tree, we do not make any a priori assumptions about d_l, although, as a result of the global optimization, it turns out to be a kind of average over the training data assigned to the leaf. For an internal node, we define $\bar{y}_i(\mathbf{x}_n)$ as the prediction of the subtree $T(t_i)$ (i.e., the subtree of T composed of t_i as the root node, and the descendent nodes of t_i). It can be computed by the recursive relation

$$\bar{y}_i(\mathbf{x}_n) = \mu_L^{(i)}(\mathbf{x}_n)\bar{y}_{iL}(\mathbf{x}_n) + \mu_R^{(i)}(\mathbf{x}_n)\bar{y}_{iR}(\mathbf{x}_n), \tag{22}$$

in terms of the partial estimates of its children nodes $\bar{y}_{iL}(\mathbf{x}_n)$, $\bar{y}_{iR}(\mathbf{x}_n)$. The relative degrees of membership $\mu_L^{(i)}(\mathbf{x}_n)$, $\mu_R^{(i)}(\mathbf{x}_n)$ are given by (19). Note that, for an inner node, $\bar{y}_i(\mathbf{x}_n)$ is different from the quantity $\bar{y}_i$ defined by (2).

The basis of this backpropagation algorithm is the observation that the value predicted by the full regression tree for the dependent variable, given the set of attributes $\mathbf{x}_n$, can be obtained by iterating (22) from the predictions at the leaves $\bar{y}_l(\mathbf{x}_n) \equiv d_l$, upward, until the root node is reached

$$\bar{y}(\mathbf{x}_n) = \bar{y}_{root}(\mathbf{x}_n). \tag{23}$$

The optimization of the error rate (14) with respect to the parameter α_j of node t_j yields the equation

$$\frac{\partial R_{train}(T)}{\partial \alpha_j} \equiv \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} (-2(y_n - \bar{y}(\mathbf{x}_n))) \frac{\partial \bar{y}(\mathbf{x}_n)}{\partial \alpha_j} = 0. \tag{24}$$

For a leaf node t_l, $\alpha_l = d_l$, using

$$\frac{\partial \bar{y}(\mathbf{x}_n)}{\partial d_l} = \mu_l(\mathbf{x}_n), \tag{25}$$

we get

$$\sum_{n=1}^{N_{train}} (y_n - \bar{y}(\mathbf{x}_n))\mu_l(\mathbf{x}_n) = 0 \qquad t_l \in \tilde{T}. \tag{26}$$

The optimization equation for the parameters at the inner node t_i is

$$\sum_{n=1}^{N_{train}} (y_n - \bar{y}(\mathbf{x}_n))\mu_i(\mathbf{x}_n)(\bar{y}_{iL}(\mathbf{x}_n) - \bar{y}_{iR}(\mathbf{x}_n)) \frac{\partial \mu_L^{(i)}(\mathbf{x}_n)}{\partial \boldsymbol{\xi}_i} = 0 \tag{27}$$

with

$$\frac{\partial \mu_L^{(i)}(\mathbf{x})}{\partial \boldsymbol{\xi}_i} = \tilde{\mathbf{x}}\mu_L^{(i)}(\mathbf{x})\mu_R^{(i)}(\mathbf{x}). \tag{28}$$

Equation (27) has been derived from (24) using the fact that for an inner node t_i, $\alpha_i = \boldsymbol{\xi}_i$ and, thus,

$$\frac{\partial \bar{y}(\mathbf{x}_n)}{\partial \boldsymbol{\xi}_i} = \mu_i(\mathbf{x}_n)(\bar{y}_{iL}(\mathbf{x}_n) - \bar{y}_{iR}(\mathbf{x}_n)) \frac{\partial \mu_L^{(i)}(\mathbf{x}_n)}{\partial \boldsymbol{\xi}_i}. \tag{29}$$

The solutions of the system of equations (26)-(27),

$$\left(\boldsymbol{\xi}_i^*, \{d_l^*\}_{l=1}^{|\tilde{T}|}\right),$$

are the parameters that characterize the globally optimal fuzzy tree. In the set of experiments presented in this work, the optimization problem is solved by a quasi-Newton method (the Broyden-Fletcher-Goldfarb-Shanno optimization algorithm [27]). Backpropagation is used in order to obtain, from the estimations at the leaves, d_l, the values of $\bar{y}_i(\mathbf{x}_n)$ which are needed in the computation of the error function (14) and its derivatives with respect to the parameters characterizing the fuzzy tree.

4.1 Interpretation of the Prediction Given by a Fuzzy Tree

As indicated in the previous section, the solution to the regression problem by a fuzzy decision tree involves estimations made at all the leaf nodes of the tree: The decision path obtained by the successive application of the crisp tests at the inner nodes of a crisp decision tree becomes a decision pattern by the successive application of fuzzy splits. This decision pattern is, in fact, a bundle of alternative decision paths that branch out from the root node and end at a leaf of the fuzzy decision tree. One may view the value of the dependent variable produced by each of the different leaves as an estimate given by a single prediction unit. The final prediction is then made by combining these estimates: For a given vector of attributes,

the value of the dependent variable is equal to the weighted average of the values given by each of the leaves; the weight of a given leaf in the average is the degree of membership of the example to the leaf in question. In the terminology of fuzzy sets, this defuzzification scheme is known as the center-of-gravity method [10]. Although other defuzzification procedures could be used, we have found that this particular one exhibits a good performance in most regression problems. This processing is substantially different from the regression given by the tree-structured hierarchical mixture of experts proposed by Jordan and Jacobs [15], where each of the experts produces an individually meaningful prediction. Regression in a fuzzy decision tree is a global process involving the estimates given by each of the terminal nodes, each of which need not be a significant prediction when considered individually (see, for instance, the leaf estimations in the problem of fitting a Lorentzian).

There are several advantages derived from the use of fuzzy splits instead of Boolean ones for a regression problem. First, the expressive capability of the fuzzy tree is enlarged with respect to a crisp one. In particular, the representation is no longer restricted to a set of piece-wise constant approximations. Second, fuzzification imposes a continuity constraint at the boundaries of the splits, which acts as a mechanism to limit the degree of overfitting of the decision tree. Thanks to this feature, we also recover the notion of locality, which is absent in crisp decision trees: Two examples that are close to each other in the space of attributes are treated in a similar fashion; therefore, the values of the dependent variable predicted by the regression tree are also similar, in agreement with the assumed properties of smoothness and continuity of the multivariate function we intend to approximate. Furthermore, this feature implies that all decision patterns can be transformed one into another in a smooth fashion, allowing the design of a global optimization algorithm for the fuzzy decision tree. This is in sharp contrast to the counterintuitive behavior of crisp decision trees, where two examples on opposite sides of a crisp split, no matter how close they may be, are handled by separate branches of the tree and can, therefore, be assigned very different values for the dependent variable. This separation into disjoint sets also implies that information present in examples assigned to a given subtree is never used in the generation of a separate subtree.

4.2 Experiments

The objective of this section is to show how fuzzification, together with backpropagation optimization, leads to a systematic improvement in the performance of a regression tree. As discussed in the introduction, the crisp tree is generated by the CART algorithm. In regression problems, we have not detected significant differences between fuzzifying a CART tree with only univariate splits or with univariate and multivariate splits. Unless otherwise stated, we take as the starting point for fuzzification a CART tree with multivariate splits. The CART tree is then fuzzified by replacing the crisp splits by sigmoidal fuzzy splits of nonzero width. The parameters of the tree, which include both the coefficients of the linear combinations that appear at the inner leaves and the leaf predictions, are then determined by minimization of the error function (14), using the backpropagation described in the previous section. Given the large dimensionality of the optimization problem, it is possible that the algorithm becomes trapped in one of the local minima of hypersurface. Consequently, the starting values for the optimization parameters should be carefully selected. The center of the fuzzy split is initially set to be equal to the threshold of the corresponding crisp split. The choice for the initial widths of the splits is less straightforward

$$\frac{1}{b_i^{init}} = 2\min\left[\max_{Train}(\mathbf{c}_i\cdot\mathbf{x_n}-a_i),\ \max_{Train}(a_i-\mathbf{c}_i\cdot\mathbf{x_n})\right]/f^{(i)}. \tag{30}$$

The factor in the numerator corresponds to scaling the width of the split to the range of the splitting variable in that node. The factor in the denominator of (30), $f^{(i)}$, appears so that splits that lead to a larger improvement of the quality of the regression in the crisp tree are initialized to be crisper than those that appear less significant:

$$f^{(i)} = \sqrt{\frac{N_i R(t_i)}{N_{iL}R(t_{iL})+N_{iR}R(t_{iR})}} - 1, \tag{31}$$

where the error rate for a node is

$$R(t_i) = \frac{1}{N_i}\sum_{n=1}^{N_{train}}\mu_i(\mathbf{x}_n)(y_n-\bar{y}_i)^2. \tag{32}$$

If the crisp split on node t_i leads to a perfect regression, it remains crisp and if it leads to no improvement, it is made to be maximally fuzzy (i.e., as if no split were present).

In a first set of experiments, we evaluate the performance of the decision trees in a series of noiseless regression problems. We first consider the problem of fitting a two-dimensional Gaussian

$$f(x,y) = exp\{-2(x^2+y^2))\} \tag{33}$$

from 250 training examples uniformly distributed at random in the square $x\in[-1,1]$, $y\in[-1,1]$. The original function is displayed in Fig. 3a. The approximations given by the crisp CART tree (univariate splits) and by the globally optimal fuzzy tree with the same architecture as the crisp tree are plotted in Fig. 3b and Fig. 3c, respectively. A second example, which exhibits a more complex structure, is the function

$$f(x,y) = sin(r)/r; \quad r=\sqrt{x^2+y^2} \tag{34}$$

from $1,000$ training examples uniformly distributed at random in the square $x\in[-1,1]$, $y\in[-1,1]$. Fig. 4a displays the original function, Fig. 4b is the approximation given by the CART crisp tree (with univariate splits). Fig. 4c presents the results of the globally optimal fuzzy regression tree.

In both of these examples, the crisp decision tree gives a piecewise constant representation of the function, which is qualitatively correct, in that it correctly captures the salient features of the objective functions. Nonetheless, the approximation is rather coarse. Fuzzification, together with global

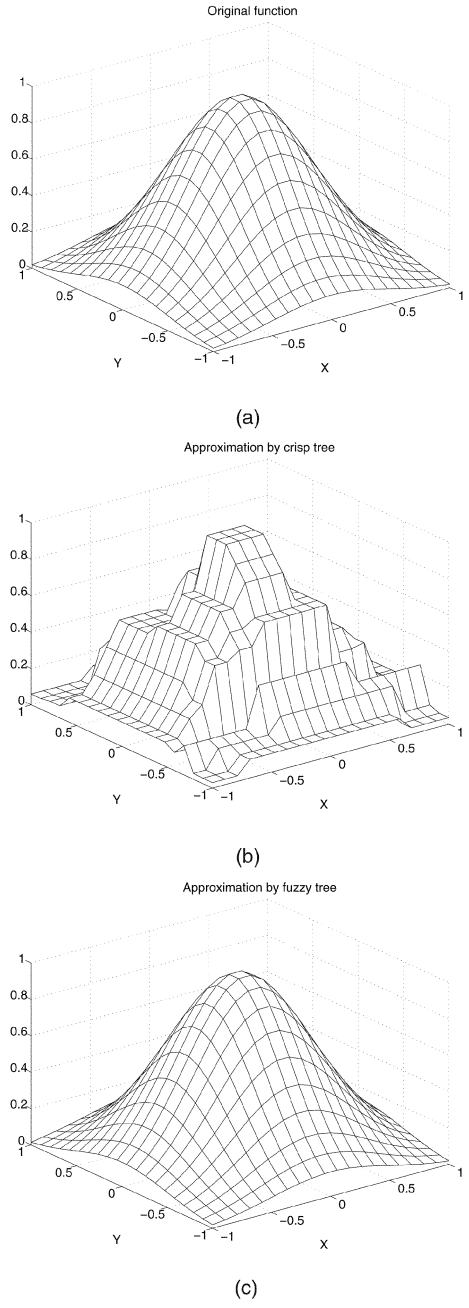

Fig. 3. Approximations given by a regression tree to a two-dimensional Gaussian: (a) original function, (b) crisp CART tree, and (c) fuzzy tree.

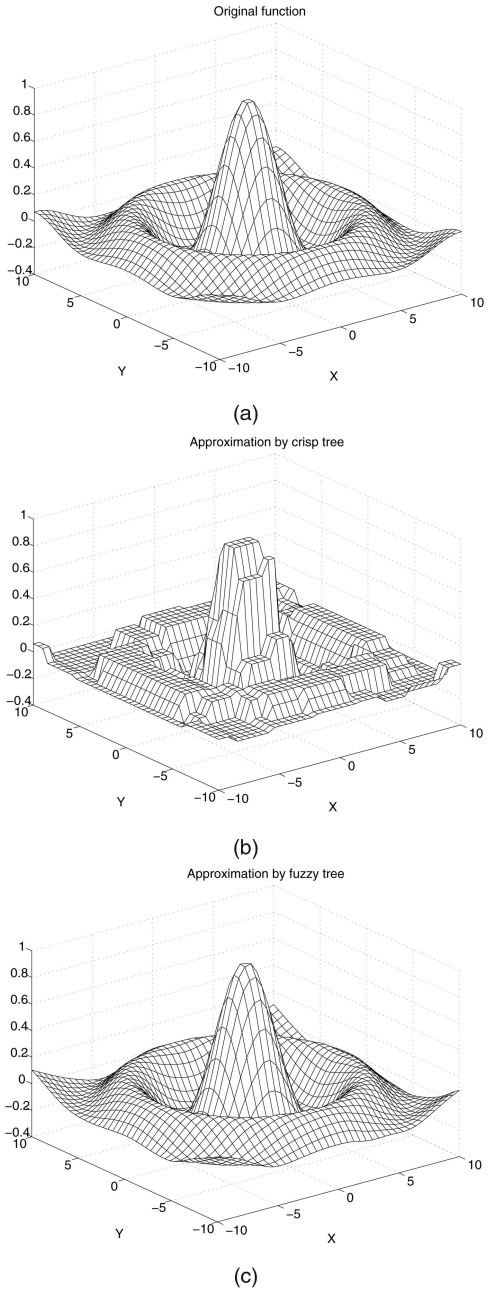

Fig. 4. Approximations given by a regression tree to the function (34): (a) original function, (b) crisp CART tree, and (c) fuzzy tree.

optimization, provides a continuous representation with the flexibility necessary to reproduce the finer details.

A second set of experiments focuses on a series of regression problems in the presence of noise, which are summarized in Table 2. They are taken from [28], where they are used to compare various methods for nonparametric regression: Multivariate Adaptive Regression Splines (MARS) [29], Constrained Topological Mapping (CTM), Generalized Memory-Based Learning (GMBL), and k-nearest neighbors [28], [30]. The sets are constructed in the following fashion: a and b are random variables with a uniform distribution within a range indicated in the fourth column of Table 2. The formulas used to calculate the predictor variables are indicated in the second column of the aforementioned table. Data is assumed to be noisy. The value of the dependent value y is obtained by adding a noise component to the formula in the third column:

$$y = f(\mathbf{x}) + noise.$$

The noise term is a random variable with a Gaussian distribution centered around 0 and with a standard deviation σ tabulated in the fifth column of Table 2.

Tables 3 and 4 compile the results for 10 realizations of each of the sets studied under the experimental conditions reported in [28], [30]. Experiments are carried out with

TABLE 2
Sets for Regression (from [28])

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

training sets of 100 samples. The quality of the regression is measured in terms of the root mean square (RMS) error of the regression in a test set of 100 samples, normalized by either by the standard deviation of the realization of the noise (Tables 3 and 5) or by the standard deviation on the dependent variable, y (Table 4). Table 3 summarizes the results for the 10 realizations of the data. The numbers reported are the averages over the different realizations followed by the standard deviations between parentheses. In the second column, the average and standard deviation of the crisp regression tree size are tabulated. The third and fourth columns give the value of the normalized RMS (NRMS), which is obtained by dividing the root mean square error by the standard deviation of the noise for the crisp and for the globally optimal fuzzy regression tree, respectively. A value NRMS close to 1.0 indicates that the regression is exact up to the noise level. Inspection of Table 3 leads to the conclusion that the quality of the regression by the fuzzified tree is significantly better that of the corresponding crisp tree. Table 4 presents, in parallel, the results reported in [28] and those obtained in this work. The second column of this table displays the two best results of the RMS normalized by $\sigma(y)$, the standard deviation of y, for a single realization of the data, as reported in [28]. The third column reports the best result for a single realization in this investigation. The fourth column gives the average and standard deviation over 10 realizations of the data. Unfortunately, these two sets of results cannot be directly compared, given that, in [28], the values of the normalized RMS are for a single realization of the data [31] and that, in [28], the smoothing parameters of the various methods were varied to achieve the best results. Nevertheless, the results reported in [28] are all, except for set 6 (where the fuzzy tree results are clearly inferior), within about one standard deviation of the averages in the fourth column (this work). This indicates that the fuzzification of a basic CART tree yields a regression tree whose performance is comparable with sophisticated regression schemes, such as MARS, CTM, or GMBL.

The previous set of experiments was made with a rather small training set. We expect that, with a larger amount of training data, the global optimization algorithm should exhibit a better performance. In Table 5, we present the results for 10 realizations of the nine sets with $N_{train} = N_{test} = 300$. Under abundant training data, such as this case, we observe indeed that the results of the globally optimal fuzzy tree significantly improve the error given by the crisp CART tree. In all cases, the normalized error rates are close to one, which indicates that the regression is close to being optimal.

5 Classification

Similarly to regression, the starting point for the construction of a globally optimal fuzzy classification tree is a crisp CART classification tree. The Boolean tests at the internal nodes of the decision tree are replaced by sigmoidal fuzzy splits with membership functions

$$\mu_L^{(i)}(\mathbf{x}) = \frac{1}{1 + \exp\left\{ b_i \left(\frac{1}{|\mathbf{c}_i|} \mathbf{c}_i \cdot \mathbf{x} - \tilde{a}_i \right) \right\}}, \quad \mu_R^{(i)}(\mathbf{x}) = 1 - \mu_L^{(i)}(\mathbf{x}), \tag{35}$$

where b_i is a parameter controlling the inverse width of the fuzzy split and the remaining parameters are

$$|\mathbf{c}_i|^2 = \mathbf{c}_i \cdot \mathbf{c}_i, \quad \tilde{a}_i = a_i / |\mathbf{c}_i|. \tag{36}$$

The normalization constant $|\mathbf{c}_i|$ is introduced so that the inverse width of the split depends only on b_i. The expressions derived for crisp trees in terms of membership functions are still valid in the case of fuzzy trees: Node t_i is characterized by the set of quantities $\{N_i^{(j)}, j = 1, \ldots, J\}$, the number of examples from the training set in that node of class j, which are given by (4). A class label is assigned to t_i by the majority rule (3).

TABLE 3
Summary of Results for the Regression Problems ($N_{train} = 100$, $N_{test} = 100$)

Set	Size	RMS/σ(noise) Crisp tree	RMS/σ(noise) Fuzzy tree
1	5.3 (0.8)	1.73 (0.14)	1.15 (0.09)
2	4.1 (0.8)	1.54 (0.18)	1.08 (0.04)
3	5.1 (0.7)	1.26 (0.10)	1.10 (0.07)
4	4.5 (1.1)	1.52 (0.25)	1.39 (0.35)
5	4.3 (0.6)	1.63 (0.31)	1.06 (0.03)
6	4.3 (0.6)	1.63 (0.31)	1.34 (0.26)
7	5.5 (1.1)	2.18 (0.38)	1.09 (0.05)
8	6.5 (0.7)	1.98 (0.19)	1.14 (0.08)
9	6.0 (1.0)	6.39 (0.68)	1.14 (0.05)

Classification takes place at the terminal nodes: Suppose we want to classify the example characterized by the vector of attributes $\mathbf{x}$ by means of a fuzzy tree. We apply the series of fuzzy tests until we determine the degree of membership of this example to each of the leaves. Thus, for each leaf t_l, the degree of membership is $\mu_l(\mathbf{x})$ and the classification label $\bar{y}_l$ is determined by (3). In order to arrive at a single prediction, we apply the defuzzification scheme

$$Class(\mathbf{x}) = argmax_j \left\{ \sum_{l \in \tilde{T}} \mu_l(\mathbf{x}) \delta(j, \bar{y}_l) \right\}, \tag{37}$$

which has been chosen among several possible defuzzification rules because of the consistency of the classification results obtained with this prescription.

5.1 Backpropagation for Classification

The objective is to find the values of the parameters $\{b_i, \mathbf{c}_i, \tilde{a}_i\}$ that optimize a global quality function. From the different possible alternatives, we choose to optimize the global impurity of the tree

$$I = \sum_{l \in \tilde{T}} \frac{N_l}{N_{train}} \left(1 - \sum_{k=1}^{K} \left(\frac{N_l^{(k)}}{N_l} \right)^2 \right), \tag{38}$$

TABLE 4
Comparison with Earlier Work

Set	RMS/σ(y) [Ref.[28]] Single realization	RMS/σ(y) [This work] Best single realization	RMS/σ(y) average (σ)
1	0.2598, 0.2750	0.2526	0.276 (0.022)
2	0.4696, 0.4881	0.4082	0.453 (0.027)
3	0.3634, 0.3698	0.3370	0.358 (0.028)
4	0.3606, 0.4058	0.3502	0.448 (0.103)
5	0.3367, 0.3494	0.3312	0.363 (0.026)
6	0.2254, 0.2339	0.2756	0.318 (0.040)
7	0.2662, 0.2680	0.2086	0.240 (0.021)
8	0.1081, 0.1089	0.1036	0.108 (0.007)
9	0.0916, 0.1031	0.0741	0.083 (0.009)

TABLE 5
Summary of Results for the Regression Problems ($N_{train} = 300$, $N_{test} = 300$)

Set	Size	RMS/σ(noise) Crisp tree	RMS/σ(noise) Fuzzy tree
1	9.8 (1.5)	1.441 (0.052)	1.089 (0.031)
2	6.8 (1.0)	1.291 (0.068)	1.063 (0.034)
3	8.5 (1.3)	1.092 (0.042)	1.034 (0.029)
4	6.2 (2.3)	1.304 (0.137)	1.105 (0.082)
5	8.1 (1.2)	1.207 (0.067)	1.023 (0.017)
6	6.4 (0.7)	1.309 (0.080)	1.032 (0.018)
7	8.4 (1.0)	1.536 (0.106)	1.028 (0.015)
8	11.4 (0.8)	1.412 (0.086)	1.050 (0.035)
9	12.3 (0.8)	3.861 (0.292)	1.061 (0.032)

with the usual definitions for $N_l^{(j)}$, the population of leaf node t_l in class j.

The impurity criterion presents the advantage that it is consistent with the criterion used to construct the tree and that it does not require that the class label of a leaf node be fixed by the crisp tree. However, empirically, we find that it has the drawback that, in almost every case, the fuzzy tree has a higher impurity than the crisp one. In order to avoid the fuzzy tree collapsing into a crisp tree, the parameter b_i is kept fixed in the course of the optimization. In order to determine the value of the inverse width parameter, we perform a series of preliminary optimizations of the tree using only $2/3$ of the training data, with different values of the b_i

$$b_i^{init} = 2^n / f^{(i)}; n = -4, -3, \ldots, 3, 4, \tag{39}$$

The factor $f^{(i)}$ in the denominator (39) corresponds to scaling the width of the split to the range of the splitting variable in that node:

$$f^{(i)} = min\left[\max_{Train} (\mathbf{c}_i \cdot \mathbf{x_n} - a_i),\ \max_{Train} (a_i - \mathbf{c}_i \cdot \mathbf{x_n}) \right]. \tag{40}$$

We select the value of b_i that leads to the lowest misclassification error as estimated the remaining $1/3$ of the training data. Once the value of b_i is chosen by this procedure, we perform a final optimization with all the training data. The optimal value of the parameters is obtained from the solution of the system of equations

$$\frac{\partial I}{\partial \alpha} \equiv \sum_{l \in \tilde{T}} \sum_{n=1}^{N_{train}} \frac{\partial \mu_l(\mathbf{x}_n)}{\partial \alpha} \Psi_l(y_n) = 0, \tag{41}$$

where

$$\Psi_l(y_n) = -\sum_{k=1}^{K} \frac{N_l^{(k)}}{N_l^2} \left[2N_l\ \delta(y_n, k) - N_l^{(k)} \right], \tag{42}$$

α stands for each of the parameters characterizing the fuzzy decision tree.

We can define at each node t_i the quantity $\Psi_i(\mathbf{x}_n, y_n)$

$$\Psi_i(\mathbf{x}_n, y_n) = \mu_L^{(i)}(\mathbf{x}_n)\Psi_{iL}(\mathbf{x}_n, y_n) + \mu_R^{(i)}(\mathbf{x}_n)\Psi_{iR}(\mathbf{x}_n, y_n), \tag{43}$$

which is obtained by backward recursion from the leaves of the tree. With the help of these auxiliary quantities, (41) can be reformulated as

$$\frac{\partial I}{\partial \alpha_i} \equiv \mu_i(\mathbf{x}_n)(\Psi_{iL}(\mathbf{x}_n, y_n) - \Psi_{iR}(\mathbf{x}_n, y_n)) \frac{\partial \mu_L^{(i)}(\mathbf{x}_n)}{\partial \alpha_i} = 0, \tag{44}$$

where α_i stands for either $\tilde{a}_i$ or $\mathbf{c}_i$ and

$$\begin{aligned} \frac{\partial \mu_L^{(i)}(\mathbf{x}_n)}{\partial \tilde{a}_i} &= b_i \mu_L^{(i)}(\mathbf{x}_n)\mu_R^{(i)}(\mathbf{x}_n) \\ \frac{\partial \mu_L^{(i)}(\mathbf{x}_n)}{\partial \mathbf{c}_i} &= -b_i \frac{1}{|\mathbf{c}_i|} \left(\mathbf{x} - \frac{\mathbf{c}_i \cdot \mathbf{x}}{|\mathbf{c}_i|^2} \mathbf{c}_i \right) \mu_L^{(i)}(\mathbf{x}_n)\mu_R^{(i)}(\mathbf{x}_n). \end{aligned} \tag{45}$$

In practice, we optimize the cost function (38) using a variable metric method [27], where the auxiliary quantities $\Psi_i(\mathbf{x}_n, y_n)$ at the inner nodes are calculated by back-propagation of the values $\Psi_l(y_n)$ at the leaves.

5.2 Classification Experiments

In order to test the fuzzification method proposed, we apply it to a collection of standard classification problems from the UC Irvine Machine Learning Database Repository and to the waveform recognition problem proposed by Breiman et al. [12]. The waveform recognition problem is a synthetic set with a known Bayes misclassification rate (around 0.14), where the number of instances that can be generated is unlimited. The rest of the datasets allow us to assess the performance of fuzzy classification trees in real-world

TABLE 6
Summary of Results for the Classification Problems (First Architecture)

SET	N_{train}	N_{test}	Size	Error Crisp	Error Fuzzy	w-l
WAVEFORM	300	5000	7.1 (2.1)	0.309 (0.026)	0.175 (0.020)	10-0
DIABETES	500	268	4.0 (1.4)	0.257 (0.020)	0.252 (0.013)	6-4
GERMAN	600	400	5.7 (1.7)	0.267 (0.025)	0.266 (0.018)	3-7
SONAR	120	88	4.2 (2.1)	0.301 (0.040)	0.251 (0.050)	8-2
CANCER-W	500	199	5.9 (1.9)	0.056 (0.016)	0.042 (0.013)	8-1
HEART-C	150	153	4.6 (2.8)	0.256 (0.031)	0.224 (0.043)	8-2

TABLE 7
Summary of Results for the Classification Problem (Second Architecture)

SET	N_{train}	N_{test}	Size	Error Crisp	Error Fuzzy	w-l
WAVEFORM	300	5000	3.5 (0.7)	0.223 (0.012)	0.181 (0.014)	10-0
DIABETES	500	268	4.1 (1.7)	0.252 (0.018)	0.257 (0.020)	4-5
GERMAN	600	400	3.8 (1.0)	0.264 (0.031)	0.262 (0.031)	4-5
SONAR	120	88	2.2 (0.4)	0.264 (0.046)	0.291 (0.066)	4-5
CANCER-W	500	199	2.0 (0.0)	0.036 (0.013)	0.041 (0.011)	3-5
HEART-C	150	153	5.1 (1.7)	0.265 (0.025)	0.260 (0.038)	5-3

examples. These are: diabetes, German (numeric), sonar (aspect-angle independent), breast-cancer (Wisconsin), heart (Cleveland).

Tables 6 and 7 present the results for classification in the data sets investigated for two different tree architectures: In Table 6 (first architecture), the architecture of the fuzzy classification tree is given by a CART tree with only univariate splits. In Table 7 (second architecture), the architecture of the fuzzy classification tree is fixed by a CART tree allowing the possibility of multivariate splits ($\beta = 0.1$). The second and third columns display the number of points used for training and testing in each of the sets. The remaining columns summarize the results of 10 different random partitions of the data into training and testing sets. The results are given as averages over these 10 different partitions, with the standard deviation reported between parentheses. The size of the tree is reported in the fourth column of this table and its classification error in the fifth column. The sixth column shows the results for the globally optimal fuzzy decision tree. Finally, the last column gives the number of cases in which the fuzzy tree performs better (w) or worse (l) than the crisp one. Ties are omitted.

We see that, in the waveform dataset, there is a very significant improvement of the classification performance for both architectures. In this example, it is clear that the presence of multivariate splits, whose coefficients are determined by a global optimization, markedly improves on the error rate, bringing it closer to the Bayes limit (estimated to be around 0.14). For the remaining sets, the improvement of performance is only apparent in three of the datasets (sonar, cancer-w, heart-c) for the first architecture. In fact, there seems to be a slight deterioration of performance in the second architecture (Table 7), which

TABLE 8
Classification Error per Quartile for Examples Rank-Ordered According to Their Fuzzy Entropy (First Architecture)

SET	Error Fuzzy	Error 1st Quartile	Error 2nd Quartile	Error 3rd Quartile	Error 4th Quartile
WAVEFORM	0.175 (0.020)	0.044 (0.014)	0.103 (0.029)	0.214 (0.044)	0.336 (0.057)
DIABETES	0.252 (0.013)	0.072 (0.048)	0.194 (0.061)	0.303 (0.051)	0.439 (0.045)
GERMAN	0.266 (0.018)	0.114 (0.052)	0.199 (0.047)	0.307 (0.040)	0.444 (0.051)
SONAR	0.251 (0.050)	0.164 (0.077)	0.191 (0.141)	0.259 (0.110)	0.391 (0.108)
CANCER-W	0.042 (0.013)	0.002 (0.006)	0.000 (0.000)	0.006 (0.010)	0.158 (0.056)
HEART-C	0.224 (0.043)	0.084 (0.069)	0.158 (0.077)	0.274 (0.076)	0.374 (0.062)

may indicate that the size of the tree generated by the CART algorithm including multivariate splits is smaller than optimal. It is also likely that, in these datasets, the crisp tree performance is already close to being optimal. The best overall results seem to be given by globally optimal fuzzy decision tree with the architecture fixed by a crisp CART with univariate splits.

We conclude that the fuzzification of the classification tree followed by optimization in general does not significantly modify the overall quality of the classification. Nonetheless, there is a different aspect where fuzzification of the classification trees is advantageous: With a fuzzification scheme based on allowing sigmoidal fuzzy splits, the hyperplanes defining the crisp splits become splitting volumes, where examples belonging to different classes could in principle coexist. Instances within these regions are, in general, more difficult to classify. In this manner, fuzzification introduces a natural metric in the space of attributes that measures the distance of that point to the relevant splits. Indirectly, this metric also quantifies the intrinsic difficulty of classifying points in a given region of the space of attributes. The picture is no longer one in which the probability of making a classification error is uniformly distributed in attribute space. By fuzzifying a classification tree, we obtain not only an average error rate, but also the structure of the error probability distribution in attribute space for a given decision tree, as is demonstrated by the experiments presented in the following section.

5.3 Fuzzy Entropy and Accuracy of the Classification

The classification of a given instance in a fuzzy tree is made jointly by the different leaves (the terminal nodes $t_l \in \tilde{T}$) of the decision tree. The extent of this effect can be quantified by defining the fuzzy entropy of an example characterized by the vector of attributes $\mathbf{x}$

$$S_{Fuzz}(\mathbf{x}) = -\sum_{k=1}^{K} \mu^{(k)}(\mathbf{x})(1-\mu^{(k)}(\mathbf{x})), \tag{46}$$

with the definition

$$\mu^{(k)}(\mathbf{x}) = \sum_{l\in\tilde{T}} \mu_l(\mathbf{x})\delta(\bar{y}_l, k). \tag{47}$$

The experiments performed on different datasets show that there is a strong correlation between how fuzzy a classification of an instance is (as measured, for instance, by its fuzzy entropy) and the likelihood of its being misclassified. This correlation should be expected, given the fact that examples with a high fuzzy entropy are in a region (in the relevant variable) where one of the splits is made. It seems natural that regions close to a split (where points of two or more different classes are close to, and possibly, mixed with, each other), are more difficult to classify.

Thus, the fuzzy entropy of a point in attribute space introduces a natural measure of the proximity of a given instance in the space of attributes to the relevant splits. Proximity to a split indicates that the example is located in a region where classes coexist. This in turn implies that it is more difficult for the tree to classify those instances accurately. By rank-ordering the examples according to their fuzzy entropy, we are able to uncover the structure of the probability of misclassification in attribute space. The results for the six datasets studied are reported in Table 8 (first architecture) and Table 9 (second architecture). The first column is the average error rate for the fuzzy classification tree. The remaining columns give the error percentages for each of the quartiles of the test data, where the examples have been ordered by their fuzzy entropy. These results show that the error rate is not uniform in attribute space: Points with lower fuzzy entropy have a much lower classification rate than points with higher entropy. In the waveform data, it is striking to see how the error rate in the first and second quartiles are well below the

TABLE 9
Classification Error per Quartile for Examples Rank-Ordered According to Their Fuzzy Entropy (Second Architecture)

SET	Error Fuzzy	Error 1st Quartile	Error 2nd Quartile	Error 3rd Quartile	Error 4th Quartile
WAVEFORM	0.181 (0.014)	0.084 (0.040)	0.100 (0.021)	0.180 (0.021)	0.357 (0.026)
DIABETES	0.257 (0.020)	0.090 (0.054)	0.193 (0.036)	0.296 (0.056)	0.449 (0.069)
GERMAN	0.262 (0.031)	0.156 (0.067)	0.171 (0.046)	0.285 (0.051)	0.436 (0.062)
SONAR	0.291 (0.066)	0.168 (0.064)	0.277 (0.110)	0.282 (0.093)	0.436 (0.112)
CANCER-W	0.041 (0.011)	0.002 (0.006)	0.012 (0.017)	0.004 (0.008)	0.144 (0.051)
HEART-C	0.260 (0.038)	0.153 (0.078)	0.189 (0.074)	0.263 (0.089)	0.431 (0.118)

Bayes error rate (which is only an average rate). Once more it is seen that the results are better for the fuzzy tree generated from the CART tree with only univariate splits (Table 8).

Similar measures of the distance to the relevant split have been introduced for other classifiers, such as nearest neighbors or linear classifiers. The measure proposed here, based on the fuzzy entropy, has the advantage that it does not depend on the metric structure of the original attribute space or on the linear separability of the problem at hand. Furthermore, it is readily available since it can be calculated concurrently with the classification of the example.

6 Summary and Conclusions

A fuzzy decision tree has been generated by replacing the Boolean tests in a standard CART decision tree by fuzzy sigmoidal splits. The result of these splits is a function giving the degree of membership of an example in a node of the decision tree. In contrast with a crisp CART tree, where a single leaf is responsible for the prediction, the classification or regression is made jointly by all the terminal nodes of the decision tree using the full set of membership values to the terminal nodes of the tree, which are computed by evaluating the hierarchy of fuzzy tests on the internal nodes linking the root to the leaves. The structure of the models allows us to construct an elegant backpropagation algorithm. Furthermore, the use of the crisp tree as a starting point allows the models to scale favorably with the number of attributes, in contrast to methods that partition the space of (nominal) variables maximally [5], [6] or artificially restrict the size of the tree or auxiliary network [20].

In regression problems, the crisp CART tree gives a piece-wise constant representation of the unknown objective function, which has the disadvantage of being discontinuous. Fuzzification of the regression tree yields a more flexible representation. Furthermore, the functional representation is continuous. This continuity constraint acts as an efficient mechanism that limits the amount of overfitting to the training data. The experiments carried out show that the fuzzification followed by backpropagation optimization of the parameters of the tree leads to a marked improvement with respect to the performance of a CART tree. Furthermore, the efficiency of the fuzzy regression tree does not deteriorate in the presence of noise. The results are generally better when abundant training data is available to carry out the training.

For classification problems, the results, as far as classification performance is concerned, are less clear. In the waveform dataset, where there is large room for improvement over the CART results, the globally optimal fuzzy classification tree gives an error rate which is significantly lower than CART and fairly close to the Bayes limit. In other datasets, the performance of the fuzzy tree is comparable to that of the crisp CART tree. However, in classification problems, the fuzzy tree has an additional advantage over the crisp one: It is possible to give a measure of the proximity of a given point in the space of attributes to a decision boundary and, therefore, identify those points whose classification is more prone to errors. This measure is evaluated simultaneously with the classification of the example and is independent of the metric of the original attribute space. Thus, fuzzy classification trees have some of the characteristics of nearest-neighbor models, in which such proximity measures arise naturally.

Besides extending the expressive capacity of the decision tree, fuzzification incorporates into a symbolic method the flexibility and robustness of a subsymbolic one. In particular, it allows the design of a backpropagation algorithm that fixes the parameters of the tree by the global optimization of a cost function. While we have concentrated in this paper on the global optimization of the models given a static dataset, the backpropagation method can be used,

just as in standard connectionist methods, to train and/or correct a model in real-time.

ACKNOWLEDGMENTS

Part of this work has been developed by the authors in the Expert Systems Applications and Development Group at the Katholieke Universiteit Leuven (Belgium), with financial support from the JOULE program of the European Commission (project EXLIBRIS). Alberto Suárez acknowledges support from CICYT (Spain) project TIC98-0247-C02-02.

REFERENCES

[1] L. Zadeh, "Fuzzy Sets," *Information and Control,* vol. 8, pp. 338-353, 1965.
[2] L. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 3, pp. 28-44, 1973.
[3] R.L.P. Chang and T. Pavlidis, "Fuzzy Decision Tree Algorithms," *IEEE Trans. Systems, Man, and Cybernetics,* no. 1, pp. 28-35, 1977.
[4] L.A. Zadeh, "Fuzzy Sets and Their Application to Classification and Clustering," *Classification and Clustering,* J. van Ryzin, ed. New York: Academic Press, 1977.
[5] P.E. Maher and D.C. St. Clair, "Uncertain Reasoning in an ID3 Machine Learning Framework," *Proc. Second IEEE Int'l Conf. Fuzzy Systems,* pp. 7-12, 1993.
[6] C.Z. Janickow, "Fuzzy Decision Trees: Issues and Methods," *IEEE Trans. Systems, Man, and Cybernetics B: Cybernetics,* vol. 28, no. 1, pp. 1-14, 1998.
[7] A. Kandel, *Fuzzy Techniques in Pattern Recognition.* New York: Wiley-Interscience, 1982.
[8] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Application to Modeling and Control," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 15, pp. 116-132, 1985.
[9] *Fuzzy Model Identification: Selected Approaches,* H. Hellendoorn and H.D. Driankov, eds. Berlin: Springer, 1997.
[10] N.K. Kasabov, *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering.* Cambridge, Mass.: MIT Press, 1996.
[11] J. Shavlik, "Learning by Symbolic and Neural Methods," *The Handbook of Brain Theory and Neural Networks,* M.A. Arbib ed., pp. 533-537, MIT Press, 1995.
[12] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees.* New York: Chapman & Hall, 1984.
[13] J.S. Schuermann and W. Doster, "A Decision Theroretic Approach to Hierarchical Classifier Design," *Pattern Recognition,* vol. 17, no. 3, pp. 359-369, 1984.
[14] J.R. Quinlan, "Decision Trees as Probabilistic Classifiers," *Proc. Fourth Int'l Workshop Machine Learning,* pp. 31-37, Irvine, Calif., 1987.
[15] M.I. Jordan and R.A. Jacobs, "Hierarchical Mixtures of Experts and the EM Algorithm," *Neural Computation,* vol. 6, pp. 181-214, 1994.
[16] I.K. Sethi, "Neural Implementation of Tree Classifiers," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 25, no. 8, pp. 1,243-1,249, 1995.
[17] Q.R. Wang and C.Y. Suen, "Large Tree Classifier with Heuristic Search and Global Training," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 9, no. 1, pp. 91-102, Jan. 1987.
[18] J. Jang, "Structure Determination in Fuzzy Modeling: A Fuzzy CART Approach," *Proc. IEEE Conf. Fuzzy Systems,* pp. 480-485, 1994.
[19] I.K. Sethi, "Entropy Nets: From Decision Trees to Neural Networks," *Proc. IEEE,* vol. 78, no. 10, pp. 1,605-1,613, 1990.
[20] I.K. Sethi and J.H. Yoo, "Structure-Driven Induction of Decision Tree Classifiers through Neural Learning," *Pattern Recognition,* vol. 30, no. 11, pp. 1,893-1,904, 1997.
[21] Y. Park, "A Comparison of Neural Net Classifiers and Linear Tree Classifiers: Their Similarities and Differences," *Pattern Recognition,* vol. 27, no. 11, pp. 1,493-1,503, 1994.
[22] J.R. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, no. 1, pp. 81-106, 1986.
[23] S.B. Gelfand, C.S. Ravishankar, and E.J. Delp, "An Iterative Growing and Pruning Algorithm for Classification Tree Design," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 2, pp. 163-174, Feb. 1991.
[24] J.R. Quinlan, *C4. 5: Programs for Machine Learning.* San Mateo, Calif.: Morgan Kauffmann, 1993.
[25] F. Esposito, D. Malerba, and G. Semeraro, "A Comparative Analysis of Methods for Pruning Decision Trees," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 5, pp. 476-491, 1997.
[26] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation.* New York: Addison-Wesley, 1991.
[27] W. Press, W.T. Teukolsky, S.A. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge, U.K.: Cambridge Univ. Press, 1993.
[28] V. Cherkassky and F. Mulier, "Statistical and Neural Network Techniques for Nonparametric Regression," *Selecting Models from Data,* P. Cheeseman and R.W. Oldford, eds., pp. 383-392, New York: Springer-Verlag, 1994.
[29] J.H. Friedman, "Multi-Variate Adaptive Regression Splines," *Annals of Statistics,* vol. 19, pp. 1-141, 1991.
[30] V. Cherkassky, D. Gehring, and F. Mulier, "Comparison of Adaptive Methods for Function Estimation from Samples," *IEEE Trans. Neural Networks,* vol. 7, no. 4, pp. 969-984, 1996.
[31] V. Cherkassky, private communication.

Alberto Suárez received the degree of Licenciado in chemistry from the Universidad Autónoma de Madrid, Spain, in 1988, and the PhD degree in physical chemistry from the Massachusetts Institute of Technology, Cambridge, Massachusetts, in 1993. After holding postdoctoral positions at Stanford University, Stanford, California, the Université Libre de Bruxelles, Belgium, and the Katholieke Universiteit Leuven, Belgium, he is currently a professor in the Computer Science Department of the Universidad Autónoma de Madrid. He has worked on relaxation theory in condensed media, stochastic and thermodynamic theories of nonequilibrium systems, lattice-gas automata, and decision tree induction. His current research interests include machine learning, probabilistic automata, and information processing in the presence of noise.

James F. Lutsko received the BS degree in physics and the BA degree in mathematics from the University of Florida in 1982 and the PhD in physics from the same institute in 1986. He was a postdoctoral fellow at Argonne National Laboratory from 1987-1989, where he worked in the field of computational materials science, studying the elastic properties and high-temperature behavior of grain boundaries, nano-crystals, and free surfaces. From 1989-1991, he was a research assistant in the Physics Department at the Free University of Brussels, during which time he worked on density functional theory and the multifractal behavior of flows in porous media. From 1991-1997, he was research group leader in the Department of Chemical Engineering at the Katholieke Universiteit Leuven, Belgium, where he divided his time between research in nonequilibrium statistical mechanics and the application of artificial intelligence techniques to problems in chemical engineering. The latter included work on decision tree induction, genetic algorithms, neural networks, and expert systems. Since 1997, he has worked as a senior developer at i2 Techologies, Belgium.