

# Regression Tree Fuzzy Systems

Richard E. Haskell  
Computer Science and Engineering Department  
Oakland University  
Rochester, Michigan, USA 48309  
email: [haskell@oakland.edu](mailto:haskell@oakland.edu)

## Abstract

Topics: Machine Learning, Fuzzy Logic

The terminal nodes of a binary tree classifier represent discrete classes to be recognized. In this paper the classes are considered to be fuzzy sets in which a specific sample can belong to more than one class with different degrees of membership. The terminal nodes in this case will contain information about the degrees to which test samples belong to particular classes. This will allow the development of a regression tree in which a continuous output value such as the control signal of a fuzzy controller can be learned. In addition to the classes being fuzzy sets each node of the regression tree is made fuzzy by associating a membership function with the fuzzy sets  $feature\_value < threshold$  and  $feature\_value \geq threshold$ . The output value is found by dropping the input measurement vector through the tree in which it will, in general, take both paths at each node with a weighting factor determined by the node membership functions. The crisp output value (defuzzification) is a weighted sum of the class values associated with the terminal nodes. The splitting criteria for each tree node is based on the use of a fuzzy cumulative distribution function, which is a generalization of the Kolmogorov-Smirnov (K-S) distance suitable for multiple classes. The splitting of nodes is terminated when all training samples belonging to a given node have their maximum degree of membership associated with a given class. Multiple, fuzzy regression trees can be learned using different sets of data collected at different times or using different sets of features. This can help to solve the problem of missing data values. The use of multiple, fuzzy regression trees involving a large number of fuzzy rules makes the output robust and avoids the brittleness of classical classification trees.

## Introduction

Lotfi Zadeh introduced the term *fuzzy sets* in 1965 [1]. In recent years most applications of fuzzy logic have been related to fuzzy control [2–7]. Neural nets have been combined with fuzzy logic in an attempt to introduce some type of learning to the design of fuzzy controllers [8–10].

These have included the learning of membership function parameters [9] as well as trying to learn the fuzzy rules [8]. While fuzzy rules are generally in the form of easily understood *if... then* statements, neural net rules are in the form of an obscure collection of node connection weights. For this reason fuzzy logic and neural nets are generally considered to be complementary areas of research [11]. This is because it is thought that neural nets have the advantage of learning from sample data while fuzzy logic has the advantage of easily understood fuzzy rules. However, there are several disadvantages in using neural nets for learning rules. First of all, the common algorithms for learning neural nets, such as back-propagation, are inefficient and become impractical if very large quantities of training data are used. They sometimes get stuck at local minima which can lead to poor performance. Most importantly, when the learning is complete, one doesn't know what the learned rules mean. They are only in the form of a collection of weights that have no obvious physical meaning. If easily understood fuzzy rules could be learned directly without the need for neural nets then these disadvantages could be overcome. This paper describes a method of using classification and regression trees as the method of learning fuzzy rules directly. The binary trees also provide a direct method of fuzzy inference and defuzzification. The method provides automatic feature selection in that unimportant features simply do not show up in the trees. In addition to applications in fuzzy control, such fuzzy systems are well suited for all types of pattern classification problems.

In a binary tree classifier a decision is made at each non-terminal node of the tree based upon the value of one of many possible attributes or features [12, 13]. If the feature value is less than some threshold then the left branch of the tree is taken, otherwise the right branch is taken. The leaves, or terminal nodes, of the tree represent the various classes to be recognized. If the classes to be recognized are distinct, we will refer to the tree as a classification tree. If a continuous output variable is to be predicted, we will refer to the tree as a regression tree. The method developed in this paper can be used to learn both classification and regression trees.

In this paper we consider binary classification and regression trees that differ from the standard binary decision tree in the following five main respects: 1) A new splitting criteria based on a generalized K-S distance associated with a fuzzy cumulative distribution function is used to select the optimum feature and threshold at each node in the binary classification tree. 2) Each training sample can belong to more than one class with varying degrees of membership, thus more accurately modeling real-life situations. 3) The test at each non-terminal node in the tree is considered to be a fuzzy set such that a test sample being dropped through the tree can take both paths with different degrees of membership depending upon its data value for that particular feature. 4) A given test sample can end up in more than one terminal node with a classification and weighting factor associated with each terminal node. In a fuzzy regression tree a defuzzification procedure produces a crisp output value as the weighted sum of class values associated with each terminal node. 5) Multiple decision trees can be produced using different training samples or different combinations of features for the same training samples. In this case a test sample is classified by dropping its measurement vector through all of the decision trees and then collecting its weighted classifications from the terminal nodes of all of the trees. These five topics will be treated separately below followed by an example of learning fuzzy rules.

## Splitting Criteria in Binary Tree Classifiers

The main problem in designing a binary tree classifier is to determine what feature and threshold value to use at each non-terminal node based upon a set of sample training data. A related problem is deciding when to stop splitting the nodes and assigning a class label to a terminal node. The most common methods used to determine the splitting at a tree node are information theoretic methods that try, in some sense, to reduce the uncertainty of the class membership at each node by the largest amount possible [14, 15, 16]. These are often referred to as entropy minimization techniques.

An alternate approach to splitting the tree nodes uses the Kolmogorov-Smirnoff (K-S) distance between two distributions [17, 18]. While this method has proved to be useful for two classes, its generalization to the multiclass case has previously involved the generation of multiple decision trees that separate one class from all the rest. We have used a generalization of this method for the multiclass case that generates only a single decision tree by introducing a generalization of the Kolmogorov-Smirnoff distance for multiple classes [19, 20].

Consider a pattern recognition problem involving  $M$  classes. Let  $\mathbf{x}$  be the measurement vector  $x_i, i = 1, \dots, N$ . At each node of the tree one of the features  $x_k$  is selected and a threshold value  $a_k$  is chosen. To begin with assume that each sample belongs to a single class with degree of membership equal to 1. For each feature  $x_k$  the one-dimensional cumulative distribution function (cdf)  $F_i^k(a_k)$  is the probability that the measurement  $x_k$  is less than  $a_k$  given that the class is  $C_i$ . If

$$\begin{aligned} n_i &= \text{number of samples of class } C_i \text{ for which } x_k < a_k \\ N_i &= \text{total number of samples of class } C_i \end{aligned}$$

then  $F_i^k(a_k)$  can be estimated as  $n_i/N_i$ .

In the fuzzy case, each sample  $j$  can belong to any number of classes  $C_i$  to varying degrees  $\mu_{ij}^c$ . Let

$$S_i = \sum_j \mu_{ij}^c$$

be the sum of the membership functions for class  $C_i$  over all samples, and let

$$s_i = \sum_{j(x_k < a_k)} \mu_{ij}^c$$

be the sum of the membership functions for class  $C_i$  over those samples for which  $x_k < a_k$ . We can then define a *fuzzy cumulative distribution function*,  $F_i^k(a_k)$ , analogous to the probability cdf, which can be estimated as

$$F_i^k = \frac{s_i}{S_i}$$

Note that  $F_i^k$  is no longer the probability that  $x_k < a_k$  for class  $C_i$  because each sample does not belong to a single class. Rather it represents, in a sense, the degree to which all samples with  $x_k < a_k$  belong to class  $C_i$ . It reduces to the probability cdf when  $\mu_{ij}^c = 0$  or 1 for all samples.

The feature and threshold at each node in the tree are selected as follows. Let  $M$  be the number of classes and  $H^k$  be the sequence of  $F_i^k, i = 1, M$  ordered in ascending order of  $F_i^k$ . Let  $H_q^k$  be the components of the sequence  $H^k$ . That is,  $H_1^k$  is the smallest value of  $F_i^k$  and  $H_M^k$  is the largest value of  $F_i^k$ . Let

$$D_k(a_k) = \max_q |H_q^k - H_{q+1}^k|, q = 1, M-1.$$

The maximum value of  $D_k$  as a function of  $a_k$ , denoted by  $D_k^*$ , occurs at some value  $a_k^*$ . Let the maximum value of  $D_k^*$  over all features  $x_k$  be denoted by  $D^*$ . At each node the feature used is that feature which produces the

maximum value  $D^*$  and the corresponding value of  $a_k^*$  is used as the threshold. In the crisp two-class case  $D_k^*$  reduces to  $\max(a_k) | F_1^k - F_2^k |$  which is the Kolmogorov-Smirnov distance between the two distributions.

The splitting of nodes is terminated when all training samples belonging to a given node have their maximum degree of membership associated with a given class. That is, for all samples  $j$  within a given node, the maximum over  $i$  of  $\mu_{ij}^c$  occurs for the same value of  $i$ .

### Fuzzy Classes

As described in the previous section each sample  $j$  can belong to any number of classes  $C_i$  to varying degrees  $\mu_{ij}^c$ . This can more accurately model real-life situations. For example, a system to predict winners in a horse race could use past performance data for many horses in an attempt to classify a horse in a particular race as a winner or a loser. But forcing a horse to belong entirely to one of two classes, winner or loser, does not recognize the difference between a horse that wins by a nose and one that wins by 10 lengths. Similarly, a horse that loses by a nose should be given more credit than one that loses by 20 lengths. The use of fuzzy classes will allow all horses to both win and lose to some degree.

A regression tree can be produced that will predict a continuous output value. In this case fuzzy sets can be used to fuzzify the output of the training samples. For example, in the case of horse racing, instead of separating the training data into two classes, winners and losers, we could try to predict the distance by which the horse wins (positive) or loses (negative). We might define the five fuzzy sets, lose by a lot, lose by a little, photo finish, win by a little, and win by a lot. Membership functions for these fuzzy sets can then be used to determine the degree of membership of a particular training sample (horse) in each of the five classes.

### Fuzzy Nodes

In a binary classification tree an error occurs when a test sample takes the wrong branch while being classified by the decision tree. This can occur for two reasons. First the test sample is subject to various sources of noise in measuring its feature vector values. For feature values near the threshold of a particular tree node, this could lead to a value on the wrong side of the threshold. Secondly, because only a limited number of training samples were used to produce the decision tree, and these training samples were subject to their own measurement errors, the

value of the threshold used in a decision node can be slightly different from its "true" value.

This error problem can be alleviated by making the decision tree a fuzzy decision tree in which the decision at each node is a fuzzy set and each branch from the decision node is given a weight equal to the degree to which a given test sample is a member of the appropriate fuzzy set. Let the fuzzy sets  $x < a$  and  $x \_ a$  be given by the membership functions shown in Figure 1.

The left branch from a node has the weight,  $w_l$ , given by the membership function  $x < a$  associated with it, and the right branch from a node has the weight,  $w_r$ , given by the membership function  $x \_ a$  associated with it. When a test sample is being classified, both branches from the node are followed with the appropriate weight associated with each branch depending on the value of  $x$ . Note that if  $x$  is more than  $\_a$  from the threshold  $a$ , the weight value will be either 1 or 0. If  $x$  is less than  $\_a$  from the threshold  $a$ , the weight value will be between 0 and 1.

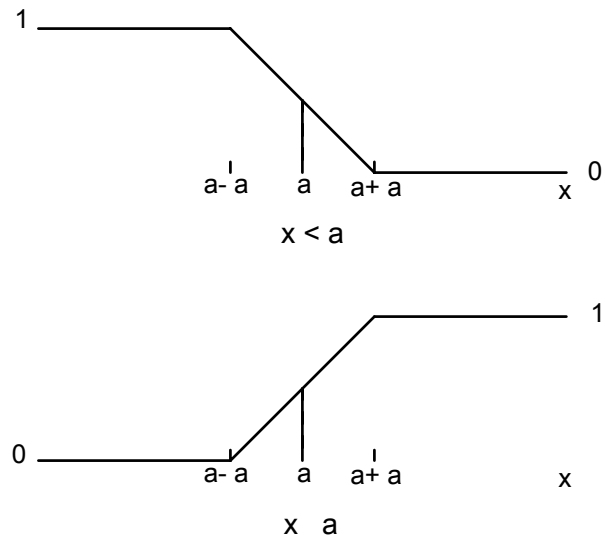


Figure 1 Fuzzy logic membership functions for  $x < a$  and  $x \_ a$

The value  $\_a$  shown in Figure 1 is determined by a single user-defined parameter,  $p$ , for all features. The parameter  $p$  is the percent of the full range of values for a given feature. For example, if feature  $x_i$  has values ranging from  $x_i^-$  to  $x_i^+$ , then  $\_a$  is given by

$$\_a = p(x_i^+ - x_i^-)/100$$

The weights for each branch in a given path of the binary tree are combined using the fuzzy logic connective AND

by taking the minimum value of all weights in a given path. The resulting weight associated with the  $k^{th}$  terminal node will be denoted as  $w_k$ .

## Defuzzification

For each terminal node  $k$ , let  $\langle \mu \rangle_k^i$  be the average degrees of membership in class  $i$  of all training samples associated with that node. Note that the largest value of  $\langle \mu \rangle_k^i$  will occur for the class label  $i$  for which all samples in the node have a maximum value of  $\mu_{ij}^c$ , inasmuch as this was the criteria for stopping the node splitting and producing a terminal node. If there are  $M$  classes then each path down the decision tree to a terminal node will represent  $M$  fuzzy rules corresponding to the  $M$  class consequents with degrees of membership  $\langle \mu \rangle_k^i$ . The weight associated with the  $k^{th}$  terminal node as the result of dropping a test sample measurement vector through the tree is  $w_k$  as discussed in the previous section. Thus, each terminal node will contribute  $w_k \langle \mu \rangle_k^i$  to the degree of membership of the test vector in class  $i$ .

One may also want to weight this degree of membership with the number of training samples that contributed to the terminal node. This would represent the idea that a terminal node resulting from 50 training samples ought to count more than a terminal node resulting from a single training sample. Let  $n_k$  be the number of training samples ending up in the  $k^{th}$  terminal node. If  $N$  is the total number of samples used to train the tree then  $q_k = n_k/N$  is a measure of the quality of node  $k$  in classifying a sample.  $q_k$  can be thought of as an a priori probability that a test sample will end up in terminal node  $k$ . We would then consider that each terminal node in the tree will contribute  $q_k w_k \langle \mu \rangle_k^i$  to the degree of membership of the test vector in class  $i$ . The total degree of membership of the test vector in class  $i$  is given by

$$\mu_i = \frac{\sum_{k=1}^K q_k w_k \langle \mu \rangle_k^i}{\sum_{k=1}^K q_k w_k} \quad (1)$$

where  $K$  is the number of terminal nodes. The test sample would be assigned to the class with the maximum value of  $\mu_i$ .

For a fuzzy regression tree used to predict a continuous output value, the  $M$  classes are defined by fuzzy set membership functions with centroids  $c_i$ . The degree of

membership of a test vector in class  $i$  is still given by Eq. (1) and the crisp output value can be found from

$$v = \frac{\sum_{i=1}^M \mu_i c_i}{\sum_{i=1}^M \mu_i} \quad (2)$$

## Multiple, Fuzzy Decision Trees

A decision tree built with large quantities of data can be very large, often containing hundreds of terminal nodes. It has been shown in [13], [20] and [21] that improved performance in classifying test data is often achieved by pruning an initial large decision tree. This is because a terminal node that contains, for example, only a single training sample may represent a sample that is not statistically significant and not representative of future test samples. Particularly if its sibling terminal node contains many training samples, it is best to prune this single-sample terminal node and direct all future test samples to the more populous sibling node. The weighting factor  $q_k$  in Eq. (1) is designed in some way to address these statistically insignificant training samples.

A disadvantage of creating large decision trees and then pruning them is that when new training samples are obtained a completely new (larger) decision tree must be learned. This is particularly inconvenient when training data are collected over a period of time or at different locations. It would be much more convenient to be able to collect training data over time, create smaller decision trees using subsets of the data as they are collected, and somehow combine the multiple decision trees to make a final classification.

Let the features be represented by a measurement vector  $x_i^j$ ,  $i = 1, \dots, P$ ,  $j = 1, \dots, N$ , where  $P$  is the total number of features measured and  $N$  is the total number of samples. The number of samples,  $N$ , will generally be a very large number. Using all of the samples at one time would lead to a very large binary decision tree. Alternatively, divide the total number of samples into  $Q$  subsets consisting of an average of  $N/Q$  samples. Then, using the algorithm described earlier, create  $Q$  binary decision trees with the  $Q$  subsets of training data. Let  $T_j$  be the  $j^{th}$  decision tree created using a set of training data  $S_j$ . Let the  $k^{th}$  terminal node of decision tree  $T_j$  be denoted by  $d_k^j$ . Let  $n_k^j$  be the number of training samples associated with node  $d_k^j$ . If  $N_j$  is the total number of samples used to train tree  $T_j$ , then  $q_k^j = n_k^j/N_j$  is a measure of the quality of node  $d_k^j$  in

classifying a sample. Let  $\langle \mu \rangle_k^{ij}$  be the average degrees of membership in class  $i$  of all training samples associated with node  $k$  in tree  $T_j$ . Let  $w_k^j$  be the weight associated with the  $k^{th}$  terminal node of tree  $T_j$  as the result of dropping a test sample measurement vector through the tree. Then if we drop a test data sample with measurement vector  $x$  through all  $Q$  decision trees, the degree of membership in class  $i$  is given by

$$\mu_i = \frac{\sum_{j=1}^Q \sum_{k=1}^{K_j} q_k^j w_k^j \langle \mu \rangle_k^{ij}}{\sum_{j=1}^Q \sum_{k=1}^{K_j} q_k^j w_k^j} \quad (3)$$

where  $K_j$  is the number of terminal nodes in tree  $T_j$ . Eq. (3) is the generalization of Eq. (1) for multiple decision trees. Eq. (2) can still be used for multiple regression trees where the values of  $\mu_i$  are determined from Eq. (3).

In addition to building multiple trees by using different sample sets, multiple trees can be created by using different sets of features on the same sample set. This technique will help to solve the missing data problem. A test sample that does not have a value for a feature in a particular tree (or path in a particular tree) may have all values needed to satisfy rules in other trees. A possible technique is to randomly select  $R$  of the  $P$  features and use these  $R$  features to build a decision tree. Multiple trees are built by selecting a new random set of  $R$  features. By this method various combinations of features are used to build the trees and sample data that have missing data values for some trees will likely have all feature values for some other trees.

## Learning Fuzzy Rules

The method of learning a binary decision tree described above has been applied to numerous applications including learning the shapes of binary images [22] and learning the rules of a fuzzy controller for a semi-active automobile suspension [23]. In this section we will illustrate how this method can find fuzzy rules by using the same ultrasound data given in the paper by Maclin et al [24] where they apply neural networks to the problem of diagnosing cancer. The data consisted of 52 cases at the Veteran's Administration Medical Center in Memphis, Tennessee. Seventeen of the cases were renal cell carcinoma (class 1), 30 cases were benign renal cysts (class 2), and 5 cases were other conditions (class 3). The 13 features associated with each of the 52 samples are listed in Table 1.

**Table 1 List of Features**

feature number	feature name	values
x1	renal mass size	14 – 107 mm
x2	patients age	22 – 79
x3	hyperechoic (0), hypoechoic (1), anechoic (2)	0 – 2
x4	solid lesion	0 = no, 1 = yes
x5	cystic lesion	0 = no, 1 = yes
x6	septated	0 = no, 1 = yes
x7	calcification	0 = no, 1 = yes
x8	no. of masses	0 = single, 1 = multiple
x9	central location	0 = no, 1 = yes
x10	peripheral location	0 = no, 1 = yes
x11	collecting system involvement	0 = no, 1 = yes
x12	metastasis	0 = no, 1 = yes
x13	retroperitoneal sign	0 = no, 1 = yes

In [24] the backpropagation algorithm was used to train a series of neural networks using 51 of the 52 samples and then testing the "held out" sample. Using a 386 computer running at 25 MHz they report [24] "In the fastest training session, the network made 2373 passes of 51 facts and checked 123,455 total facts in one hour, 14 minutes and 58 sec to learn at a tolerance of 0.005. However, pulling a different fact and using a slightly different 51 facts for training once took 12,609 passes of the 51 facts and checked 643,059 facts in 6 hr, 35 min and 27 sec to learn at a tolerance of 0.005." By contrast, using all 52 samples the classification tree shown in Figure 2 was produced using the algorithm described in this paper on a 33 MHz 486 computer in 0.275 seconds. This result is consistent with our experience in learning hundreds of classification trees, namely, that this algorithm is orders of magnitude faster than the backpropagation algorithm for learning neural networks.

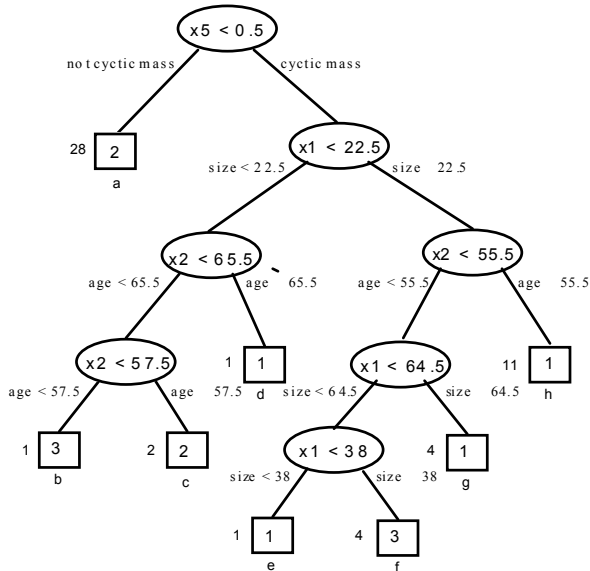


Figure 2 Binary tree learned from all 52 samples

Note that only 3 of the 13 features ( $x1$ ,  $x2$ , and  $x5$ ) are required to separate all three classes. This feature selection capability is inherent in the algorithm used to build the classification tree. The numbers in the boxes at the leaves of the tree (the terminal nodes) are the class number (1 = renal cancer, 2 = renal cyst, and 3 = other). The eight terminal nodes are labeled  $a - h$ . The numbers to the left of each leaf box are the number of samples associated with that terminal node. For example, if  $x5 < 0.5$  in the root node (meaning that the lesion is not a cystic mass), then the diagnosis is a renal cyst. This simple rule holds for 28 of the 30 renal cyst samples. The other 2 renal cyst samples have the rule

$$(x5 \_ 0.5) \leftrightarrow (x1 < 22.5) \\ \leftrightarrow (x2 < 65.5) \leftrightarrow (x2 \_ 57.5)$$

where  $\leftrightarrow$  is the logical AND operation. From Table 1, the meaning of this rule is

the lesion is cystic  
AND the size is  $< 22.5$  mm.  
AND  $57.5 \_ \text{age} < 65.5$

Using the fuzzy sets shown in Figure 1 this rule might be expressed as

the lesion is cystic  
AND the size of the renal mass is small  
AND the patient's age is moderately old

Note that 11 of the 17 cases of renal cancer end up in terminal node  $h$ . The rule for this node is

$$(x5 \_ 0.5) \leftrightarrow (x1 \_ 22.5) \\ \leftrightarrow (x2 \_ 55.5)$$

which, from Table 1 corresponds to the rule

the lesion is cystic  
AND the size is  $\_ 22.5$  mm.  
AND  $\_ \text{age} < 55.5$

Again using the fuzzy sets shown in Figure 1 this rule might be expressed as

the lesion is cystic  
AND the size of the renal mass is not small  
AND the patient's age is old

Probably only terminal nodes  $a$  and  $h$  contain enough samples to be statistically significant, although the other nodes illustrate how fuzzy rules are easily constructed. The important point is that this classification tree provides useful information about what determines whether a renal cyst is cancerous or not. In particular, a non-cystic mass is not cancerous, but a cystic one that is large enough probably is. By contrast, the neural networks learned in [24] are only "black boxes" whose predictions are rightly to be doubted without any knowledge about what the rules mean.

The classification tree shown in Figure 2 was created under the assumption (used in [24]) that each sample belongs exclusively to only one of the three classes. However, it would likely be more accurate to assign each sample to all classes with different degrees of membership as discussed earlier in this paper. In fact, a crude classification system exists for most tumors. A fuzzy regression tree created from such samples could be used to predict the degree of cancer involvement in a particular renal mass.

## Conclusion

In this paper we have described a new approach to classification and regression trees. A new splitting criteria based on a generalized K-S distance associated with a fuzzy cumulative distribution function was introduced. The idea of fuzzy classes was used to train the tree. Fuzzy sets at each tree node allow a test sample to traverse both paths with different weighting factors. A defuzzification procedure was outlined for both discrete classes and continuous output values. Finally, a method of using multiple fuzzy regression trees was described. An example related to the diagnosis of cancer illustrated how fuzzy rules can be learned from sample data.

The use of multiple fuzzy regression trees described in this paper has the following advantages: 1) Actual *if...then* rules are automatically generated that explain a particular classification and provide insight into the meaning of the data. 2) The rules can be expressed as fuzzy rules in which the number and size of the membership functions are automatically determined. 3) Generating the rules is efficient, allowing the use of large databases. 4) The method is robust and can handle the important practical problem of missing data. 5) Fuzzy regression trees can be learned from which a continuous output variable can be predicted.

## References

1. L. Zadeh, "Fuzzy Sets," *Inform. and Contr.*, Vol. 8, pp. 388-353, 1965.
2. E. H. Mamdani and S. Assilian, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," *Int. J. Man-Machine Studies*, Vol. 7, pp. 1-13, 1975.
3. R. M. Tong, "A Control Engineering Review of Fuzzy Systems," *Automatica*, Vol. 13, pp. 559-569, 1977.
4. J. Maiers and Y. S. Sherif, "Applications of Fuzzy Set Theory," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 15, pp. 175-189, Jan./Feb. 1985.
5. R. Sutton and D. R. Towill, "An introduction to the use of fuzzy sets in the implementation of control algorithms," *J. of the Institution of Electronic and Radio Engineers*, Vol. 55, pp. 357-367, 1985.
6. C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Parts I & II," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 20, Part I: pp. 404-418, Part II: pp. 419-435, March/April, 1990.
7. T. Munakata and Y. Jani, "Fuzzy Systems: An Overview," *Communications of the ACM*, Vol. 37, No. 3, pp. 69-76, March 1994.
8. B. Kosko, *Neural Networks and Fuzzy Systems - A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1992.
9. H. R. Berenji and P. Khedkar, "Learning and Tuning Fuzzy Logic Controllers Through Reinforcements," *IEEE Trans. on Neural Networks*, Vol. 3, pp. 724-740, 1992.
10. D. Nauck, F. Laswonn and R. Kruse, "Combining Neural Networks and Fuzzy Controllers, in *Fuzzy Logic in Artificial Intelligence*, E. P. Klement and W. Slany (Eds.), Proc. 8th Austrian Artificial Intelligence Conference, FLAI'93, Linz, Austria, June 1993, Springer-Verlag, Berlin, pp. 35-46, 1993.
11. L. A. Zadeh, "Fuzzy Logic, Neural Networks, and Soft Computing," *Communications of the ACM*, Vol. 37, No. 3, pp. 77-84, March 1994.
12. G. R. Dattatreya and L. N. Kanal, "Decision Trees in Pattern Recognition," *Progress in Pattern Recognition 2*, L. N. Kanal and A. Rosenfeld (Editors), Elsevier Science Publishers B. V. (North-Holland), 1985.
13. L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, Wadsworth & Brooks/Cole, Monterey, CA, 1984.
14. I. K. Sethi and G. P. R. Sarvarayudu, "Hierarchical Classifier Design Using Mutual Information," *IEEE Trans. on Pattern Anal. and Machine Intell.*, Vol. PAMI-4, pp. 441-445, 1982.
15. J. R. Quinlan, "Learning Efficient Classification Procedures and their Application to Chess End Games," in *Machine Learning, An Artificial Intelligence Approach*, R. S. Michalski, et. al. Eds., Tioga Publishing Co., Palo Alto, CA pp. 463-482, 1983.
16. J. L. Talmon, "A multiclass nonparametric partitioning algorithm," *Pattern Recognition Letters*, vol. 4, pp. 31-38, 1986.
17. J. H. Friedman, "A Recursive Partitioning Decision Rule for Nonparametric Classification," *IEEE Trans. on Computers*, Vol C-26, pp. 404-408, April 1977.
18. E. M. Rounds, "A Combined Nonparametric Approach to Feature Selection and Binary Decision Tree Design," *Proc. 1979 IEEE Computer Society Conf. on Pattern Recognition and Image Processing*, pp. 38-43, 1979.
19. R. E. Haskell, G. Castelino and B. Mirshab, "Computer Learning Using Binary Tree Classifiers," *Proc. 1988 Rochester Forth Conference, Programming Environments*, Rochester, NY, pp. 77-78, June 14-18, 1988.
20. R. E. Haskell and A. Noui-Mehidi, "Design of Hierarchical Classifiers," *First Great Lakes Computer Science Conference*, Western Michigan Univ., Oct. 18-20, 1989.
21. J. R. Quinlan, "Decision Trees as Probabilistic Classifiers," *Proc. Fourth Int. Workshop on Machine Learning*, U. of Cal, Irvine, pp. 31-37, June 22-25, 1987.
22. R. E. Haskell and B. Mirshab, "Learning Shape Features Using a Binary Tree Classifier," *3rd International Conference on CAD/CAM Robotics & Factories of the Future*, Southfield, MI, August 14-17, 1988.
23. D. M. Briggs, K. C. Cheok, N. Huang and R. E. Haskell, "A Heuristic Binary-Tree-Based Fuzzy Logic Controller for Semi-Active Suspensions," *Robotics and Manufacturing, Recent Trends in Research, Education, and Applications*, Vol. 4, p. 637-642, ASME Press, New York, 1992. *Proc. Fourth*

International Symposium on Robotics and Manufacturing (ISRAM '92), Nov. 11-13, Sante Fe, New Mexico.

24. P. S. Maclin, J. Dempsey, J. Brooks and J. Rand, "Using Neural Networks to Diagnose Cancer," *Journal of Medical Systems*, Vol. 15, pp. 11-19, 1991.