

# Neuro-Fuzzy Classification and Regression Trees

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

## Abstract

Keywords: Machine Learning, Decision Trees, Fuzzy Logic, Neural Nets

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. Large decision trees are typically pruned to provide better classification accuracy when used with test data. A stock market prediction example is used to show that making a large fuzzy tree is an attractive alternative to pruning. Fuzzy classification and regression trees can be considered to be a fuzzy neural network in which the structure of the network is learned rather than the weights. Such neuro-fuzzy classification and regression trees should lend themselves to efficient implementation in a VLSI chip in which each test sample can propagate through all paths simultaneously.

## Introduction

Lotfi Zadeh introduced the term *fuzzy sets* in 1965 [1]. Neural nets have been combined with fuzzy logic in an attempt to introduce some type of learning to the design of fuzzy controllers [2–3]. 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 [4]. This paper describes a method of using classification and regression trees as the method of learning fuzzy rules directly. The binary trees are also a direct method of fuzzy inference and defuzzification providing an alternative to both neural networks and conventional fuzzy inference systems. The method provides automatic feature selection in the sense that unimportant features simply do not show up in the trees. The resulting classification or regression tree can be considered to be a neural network in which the weights are not learned during training but rather it is the structure of the network that is learned. The weights associated with each link in the tree (network) are a function of the fuzziness of the tree and the values of the weights depend only on the values of the test data presented to the network. The resulting neuro-fuzzy regression tree can be implemented in a VLSI chip. In addition to applications in fuzzy control, such neuro-fuzzy regression trees are well suited for all types of pattern classification problems. An example of using fuzzy regression trees in conjunction with a stock market investment strategy will be described to illustrate the advantages of fuzzifying the trees rather than pruning them.

## Binary Tree Classifiers

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 [5, 6]. 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.

### Node spitting criteria

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 [7, 8, 9]. 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 [10, 11]. 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 [12, 13, 14].

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)  $F_{ik}(a_k)$  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,  $H1k$  is the smallest value of  $Fik$  and  $H Mk$  is the largest value of  $Fik$ . Let

$$Dk(ak) = \max_q |Hqk - Hq+1k|, q = 1, M-1.$$

The maximum value of  $Dk$  as a function of  $ak$ , denoted by  $Dk^*$ , occurs at some value  $ak^*$ . Let the maximum value of  $Dk^*$  over all features  $xk$  be denoted by  $D^*$ . At each node the feature used is that feature which produces the maximum value  $D^*$  and the corresponding value of  $ak^*$  is used as the threshold.  $Dk^*$  reduces to  $\max(ak) |F1k - F2k|$  in the crisp two-class case 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  $\mu_{ijc}$  occurs for the same value of  $i$ .

## 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. We will let the fuzzy sets  $x < a$  and  $x \geq a$  be given by the membership functions shown in Figure 1.

The left branch from a node has the weight,  $wl$ , given by the membership function  $x < a$  associated with it, and the right branch from a node has the weight,  $wr$ , given by the membership function  $x \geq a$  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  $\Delta a$  from the threshold  $a$ , the weight value will be either 1 or 0. If  $x$  is less than  $\Delta a$  from the threshold  $a$ , the weight value will be between 0 and 1.

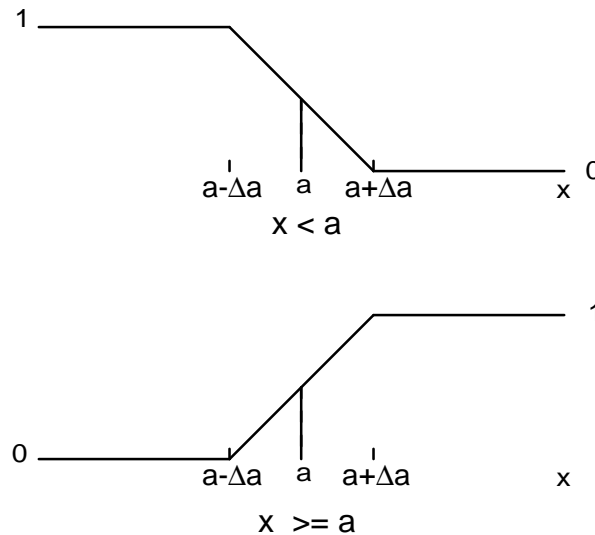


Figure 1 Fuzzy logic membership functions for  $x < a$  and  $x \geq a$

The value  $\Delta a$  shown in Figure 1 is determined by a single user-defined parameter,  $p$ , called the *fuzzy percent*. The parameter  $p$  is the percent of the full range of values for a given feature within the subspace associated with a particular node. For example, if feature  $x_i$  has values ranging from  $x_{ij-}$  to  $x_{ij+}$  in node  $j$ , then  $\Delta a$  is given by

$$\Delta a = p(x_{ij+} - x_{ij-})/100$$

The fuzzy percent,  $p$ , is a measure of the fuzziness of the resulting tree. Increasing this fuzzy percent tends to improve the performance of real-world problems as will be shown in the stock market example at the end of this paper.

## Defuzzification of class values

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  $w_k$ . For each terminal node  $k$ , let  $\langle \mu \rangle_{ki}$  be the average degrees of membership in class  $i$  of all training samples associated with that node. Note that the largest value of  $\langle \mu \rangle_{ki}$  will occur for the class label  $i$  for which all samples in the node have a maximum value of  $\mu_{ijc}$ , 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  $\langle \mu \rangle_{ki}$ . 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  $w_k \langle \mu \rangle_{ki}$  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  $q_k w_k \langle \mu \rangle_{ki}$  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

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

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

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

A decision tree built with large quantities of data can be very large, often containing hundreds of terminal nodes. It has been shown in [6], [13] and [15] 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  $qk$  in Eq. (1) is in some way designed to address these statistically insignificant training samples.

For several years I have used a technique for pruning crisp classification trees which always seemed to improve the performance when applied to test data [13]. The technique is simply to prune any terminal node that contains only  $n$  training samples, where the prune number,  $n$ , is typically 1 to 5, depending on the size of the tree. This has the effect of eliminating weak terminal nodes in favor of their stronger siblings whose class prediction is based on a larger number of training samples.

While this method seems to work it is a rather drastic technique in the sense of possibly throwing away important information in the pruned nodes. More recent experiments have shown that making the tree fuzzy when presented with test data using the technique illustrated in Figure 1 generally has a more beneficial effect than pruning the trees. These two methods will be directly compared in a stock market prediction problem described at the end of this paper.

## Neuro-Fuzzy Classification Networks

An example of a crisp classification tree in which each sample is characterized by two features,  $x_1$  and  $x_2$ , and belongs to one of three distinct classes is shown in Figure 2. The classification rule can be stated as

```

IF  $x_1 < a$ 
THEN IF  $x_2 < b$ 
      THEN class 2
      ELSE class 1
ELSE class 3

```

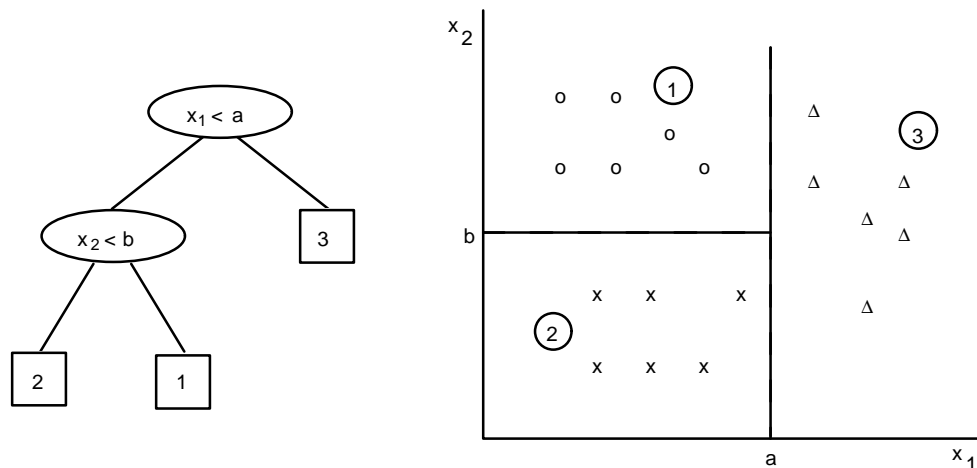


Figure 2 Example of a 3-class classification tree

Now assume that each sample can belong to all classes to varying degrees. The same tree might result, but in this case the samples in each terminal node will have a degree of membership in all classes characterized by  $\langle \mu \rangle_{ki}$ , the average degree of membership in class  $i$  for all samples in terminal node  $k$ . If the tree is made fuzzy using the fuzzy sets in Figure 1 then a test sample dropped through the tree will, in general, end up in more than one terminal node and Eq. 1 is used to find the degree of membership in each class. If the classes are fuzzy numbers representing a continuous output then Eq. 2 is used to predict a crisp output.

The fuzzy classification tree resulting from Figure 2 can be represented as the neuro-fuzzy network shown in Figure 3. Each neuron in this network represents one of the nodes in the classification tree and has two inputs and two outputs. One input is always one of the network inputs  $x_i$  and the other input is a weight value coming, in general, from another neuron. Each neuron is a fuzzy neuron of the type shown in Figure 4 where the output weights depend only on the value of the input  $x$  and the threshold  $a$ .

Note that in the neuro-fuzzy classification tree shown in Figure 3 it is the structure of the tree that is learned from the training data rather than the weights. The values of the weights vary from test sample to test sample and are determined only when a test sample is dropped through the tree. The neuro-fuzzy classification tree shown in Figures 3 and 4 should lend itself to efficient implementation in an FPGA or VLSI chip [16]. This could be useful for large trees in which each test sample can propagate through all paths simultaneously.

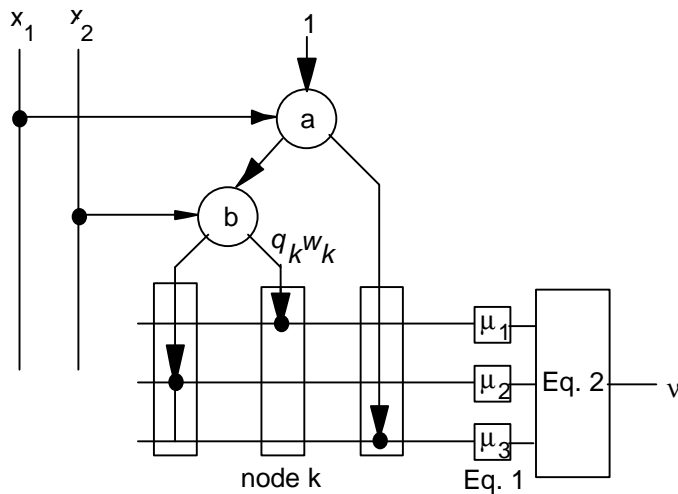


Figure 3 A neuro-fuzzy classification tree

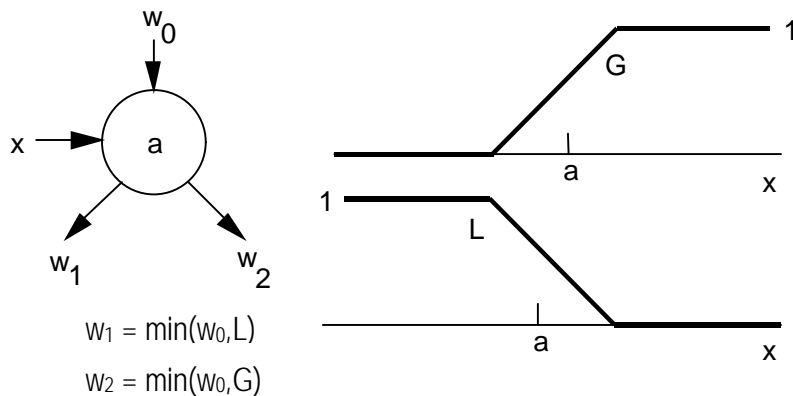


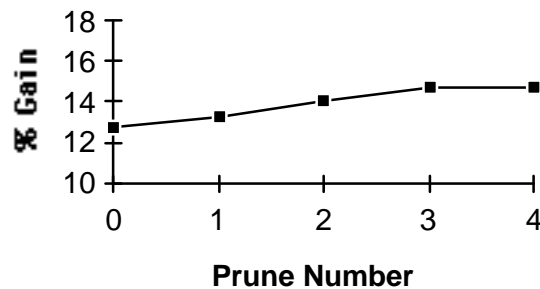
Figure 4 A classification tree fuzzy neuron

## Example: Stock Sector Predictions

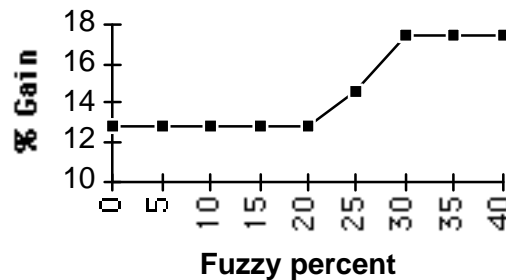
As an example a series of experiments were conducted with the goal of determining which of 31 Fidelity select stock funds one should be invested in during any particular quarter to maximize overall portfolio return. These select funds each invest in a particular market sector such as computers, energy, autos, financial services, health, retail, etc. It is not uncommon for the fund that has the best return in one quarter to be near the bottom of the list the next quarter. Thus, the goal was to predict the ranking of the 31 funds 13 weeks into the future. The resulting percent gain was taken to be the average return one would earn by investing equally in the top 7 from the predicted list.

A separate regression tree was created for each of the 31 funds and used to predict that fund's 13-week percent gain. A 13-week exponential moving average was applied to the original weekly net asset value data. Seven features were used to train each tree. These included the four 13-week percent gains at the current time,  $t_0$ , as well as at times  $t_0 - 4$ ,  $t_0 - 13$ , and  $t_0 - 26$ . The other three features were the changes in 13-week percent gains from the current time to 4, 13 and 26 weeks ago. These seven features were used to predict the percent gain at time  $t_0 + 13$ .

Four years of weekly training data were used for the years 1991 – 1994. The resulting regression trees were then used to predict the percent gain during the first 13 weeks of 1995 for each fund and the funds were then ranked. If one invested in all 31 select stocks during these first 13 weeks of 1995 the average total return would have been 10.9%. (The corresponding gain in the S&P 500 was 8.6%.) Investing in the top 7 funds predicted by the 31 regression trees would produce the predicted percent gains shown in Figure 5. Note in Figure 5a that increasing the prune number from 0 to 3 increases the percent gain from 12.8% to 14.3%. On the other hand by making the trees 30% fuzzy the percent gain increases from 12.8% to 17.5%. This is over twice the gain of the S&P 500 for the same 13-week period.



(a)



(b)

Figure 5 13-week percent gain vs. prune number and fuzzy percent

## Conclusion

This paper has shown that classification and regression trees can be made fuzzy in two ways. First each sample can belong to different classes to varying degrees. Secondly, the splitting criteria at each node in the tree,  $xi < a$ , is considered to be a fuzzy set. Experience has shown that introducing these fuzzy constructs improves the performance of classification and regression trees. The fuzzy classification tree can be considered to be a fuzzy neural network in which it is the structure of the network that is learned rather than the weights. The weights in the network depend only on the value of the input to the network. The neuro-fuzzy classification tree should lend itself to efficient implementation in a VLSI chip [16]. Such neuro-fuzzy classification and regression trees have the advantages of both fuzzy systems and neural networks without the associated disadvantages.

## References

1. L. Zadeh, "Fuzzy Sets," *Inform. and Contr.*, Vol. 8, pp. 388-353, 1965.
2. B. Kosko, *Neural Networks and Fuzzy Systems - A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1992.
3. 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.
4. L. A. Zadeh, "Fuzzy Logic, Neural Networks, and Soft Computing," *Communications of the ACM*, Vol. 37, No. 3, pp. 77-84, March 1994.
5. 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.
6. L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, Wadsworth & Brooks/Cole, Monterey, CA, 1984.
7. 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.
8. 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.
9. J. L. Talmon, "A multiclass nonparametric partitioning algorithm," *Pattern Recognition Letters*, vol. 4, pp. 31-38, 1986.
10. J. H. Friedman, "A Recursive Partitioning Decision Rule for Nonparametric Classification," *IEEE Trans. on Computers*, Vol C-26, pp. 404-408, April 1977.
11. 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.
12. 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.
13. R. E. Haskell and A. Noui-Mehidi, "Design of Hierarchical Classifiers." In N. A. Sherwani, E. de Donker, and J. A. Kapenga, editors, *Computing in the 90's: The First Great Lakes Computer Science Conference Proceedings*, pp. 118-124, Berlin, 1991. Springer-Verlag. Conference held at Western Michigan Univ., Kalamazoo, MI, Oct. 18-20, 1989.
14. R. E. Haskell, "Regression Tree Fuzzy Systems," *Proc. ICSC Symposium on Soft Computing, Fuzzy Logic, Artificial Neural Networks and Genetic Algorithms*, University of Reading, Whiteknights, Reading, England, pp. B.1-B.6, March 26 - 28, 1996.
15. 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.
16. V. Beiu, "How to Build VLSI-Efficient Neural Chips," *Proc. Int. ICSC Symposium on Engineering of Intelligent Systems - EIS'98*, Vol 2, Neural Networks, Tenerife, Spain, pp. 66-75, Feb. 11-13, 1998.