Some Neural Net Realizations of Fuzzy Reasoning

Kuhu Pal and Nikhil R. Pal*
Machine Intelligence Unit, Indian Statistical Institute, 203 B. T. Road, Calcutta 700 035, India

James Keller
Computer Engineering and Computer Science Department, 217 Engineering Building West, University of Missouri — Columbia, Columbia, Missouri 65211

In this paper we analyze the neural network implementation of fuzzy logic proposed by Keller et al. [*Fuzzy Sets Syst*., **45**, 1–12 (1992)], derive a learning algorithm for obtaining an optimal α for the net, and, for a special case, we show how one can directly (avoiding training) compute the optimal α. We address how training data can be generated for such a system. Effectiveness of the optimal α is then established through numerical examples. In this regard, several indices for performance evaluation are discussed. Finally, we propose a new architecture and demonstrate its effectiveness with numerical examples. © 1998 John Wiley & Sons, Inc.

1. INTRODUCTION

Fuzzy logic exhibits a variety of exotic applications, ranging from process control through medical diagnosis, securities trading, robot arm control, etc.[1–6] Its most popular area of application is in control engineering. Neural networks (NN), a biologically motivated computational paradigm with learning ability, have also been used in many applications.[7] Apart from the learning ability of a NN, it has inherent robustness and parallelism. Fuzzy logic, on the other hand, has the capability of modeling vagueness, handling uncertainty, and supporting human-type reasoning. Integration of these two soft computational paradigms (often known as neuro-fuzzy computing) is, therefore, expected to result in more intelligent systems.[8,9]

During the last few years, extensive research has been going on in the integration of fuzzy system with neural networks. The aim of such work is to combine the expert knowledge or operators' experience of fuzzy systems with the computational capabilities of neural network in an efficient manner to solve

*Author to whom correspondence should be addressed.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 13, 859–886 (1998)
 CCC 0884-8173/98/090859-28

complex problems.[8–23] Integration of fuzzy logic and NNs often is done in two ways—a fuzzy system implemented in a neural architecture and a NN equipped with the capability of handling fuzzy information. Several attempts have been made in both directions. Of course, there are several hybrid systems that may not be categorized strictly in either of these two classes. The fusion of fuzzy logic and neural networks focuses on the process by which individual merits can be combined and by which analogies between them can be superposed.

Keller et al.[10] proposed a neural implementation of fuzzy logic. In this note, we analyze that system and derive learning rules for finding good parameters for the network. For a special case, we show how the optimal parameter can be computed and we demonstrate the method with some examples. We also discuss several indices for performance evaluation. Finally, we propose a new architecture that exhibits better characteristics than the network in Ref. 10.

The paper is organized as follows. Section 2 briefly discusses the basics of fuzzy logic and neural networks. Section 3 presents the network proposed by Keller et al. Section 4 analyzes the net and derives an algorithm for tuning the network parameters. It also derives the optimal value of the parameter for a special case of the network. Section 5 addresses the issue of performance evaluation, while Section 6 presents the results with optimal α. In Section 7, we introduce a new architecture and present numerical examples with the new architecture. Finally, the paper is concluded in Section 8.

2. BASICS OF FUZZY LOGIC AND NEURAL NETWORKS

2.1. Fuzzy Logic

Fuzzy sets were introduced in 1965 by Zadeh[24] as a new way to represent vagueness in everyday life. Fuzzy sets are generalizations of crisp sets and have greater flexibility to capture faithfully various aspects of incompleteness or imperfection in information, and can be used to model human reasoning/thinking processes.

Let $A = \{\mu_A(u)/u,\ u \in U\}$ be a fuzzy set defined on X and let $R = \sum_{U\times V} \mu_R(u,v)/(u,v)$ be a fuzzy relation representing a rule if x is A then y is B, defined on $X \times Y$, i.e., $R = A \times B$. Here $\times$ is the Cartesian product implemented through some T-norm and U and V are the respective universes of X and Y. Now given x is A' the conclusion B' of y is B' can be obtained by the composition of A' and R. The composition of A' and R results in a fuzzy set B' defined on y as

$$B' = A' \circ R = \mathrm{Proj}(\mathrm{Ce}(A') \cap R) \quad \text{on } y \tag{1}$$

Here Ce and Proj are the cylindrical extension and projection operators, respectively.[25] If the intersection is performed with the maximum operation and projection with the minimum operation, then

$$\mu'_B(y) = \max_x \min\{\mu'_A(x), \mu_R(x,y)\} \tag{2}$$

This is known as *max-min* composition. Similarly, the *max-prod* composition is defined as

$$\mu'_B(y) = \max_x \{\mu'_A(x) \cdot \mu_R(x, y)\} \tag{3}$$

Normally, $B' = R \circ A \neq B$, where $\circ$ is a composition operator implemented through a pair of T-norm and S-norm. This is a very undesirable property.

If there are n rules, then the composite relation representing the rule base for the system is

$$R = \bigcup_{i=1}^{n} R_i = \bigcup_{i=1}^{n} A_i \times B_i \tag{4}$$

Given a set of rules the composition can be done with respect to each rule separately, and then the different conclusion B'_i can be aggregated to get a resultant conclusion B'. On the other hand, the relations representing different rules can be aggregated first as in Eq. 4 and then the composition operator can be directly applied to R.

Suppose we have a rule if x is A then y is B. Now given x is A', we want to derive a conclusion y is B', such that the disagreement between A and A' is reflected between B and B'. Moreover, given x is A, we want to have a conclusion $y = B'$ as close as possible to B.

2.2. Neural Networks

Neural networks,[7] like fuzzy logic systems, are excellent at developing human-made systems that can perform types of information processing similar to what our brain does. The concept of artificial neural networks was inspired by biological neural networks, but the heart of this emerging technology is rooted in different disciplines. Biological neurons are believed to be the structural constituents of the brain and they are much slower than silicon logic gates. However, inferencing in biological NNs is faster than the fastest computer available today. The brain compensates for the relatively slower operation by a really large number of neurons with massive interconnections between them. Biological neural networks enjoy the following characteristics:

- They are nonlinear devices, highly parallel, robust, and fault tolerant.
- They have a built-in capability to adapt their synaptic weights to changes in the surrounding environment.
- They can handle easily imprecise, fuzzy, noisy, and probabilistic information.
- They can generalize from known tasks or examples to unknown ones.

Artificial NN is an attempt to mimic some or all of these characteristics. This soft computational paradigm is different from a programmed instruction sequence. Here information is stored in the synaptic connections. A neuron is an elementary processor with primitive types of operations, like summing the weighted inputs coming to it and then amplifying or thresholding the sum. The

computational neuron model proposed by McCulloch and Pitts is a simple binary threshold unit. The ith neuron computes the weighted sum of all its inputs from other units and outputs a binary value, 0 or 1, depending on whether this weighted sum is greater than equal or less than a threshold θ_i. Thus,

$$x_i(t+1) = f\left(\sum_{ij} w_{ij} x_j(t) - \theta_i \right)$$

where

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

If the synaptic weight $w_{ij} > 0$, then it is called an excitatory connection; if $w_{ij} < 0$, it is viewed as an inhibitory connection. A simple generalization of the McCulloch–Pitts neuron by replacing the threshold function f with a more general nonlinear function enhances the power of the networks built from such neurons. Although the development of neural networks was inspired by the model of the brain, its purpose is not just to mimic a biological neural net, but to use principles from the nervous system to solve complex problems in an efficient manner.

Thus we see that fuzzy logic can model vagueness and can perform human-style reasoning, while NNs compute in the style of the brain, having robustness, adaptability, and generalization ability. Therefore, integration of the two paradigms into a single system is expected to result in a more powerful computational system.

3. NN IMPLEMENTATION OF FUZZY LOGIC

Keller et al.[10] proposed a novel neural network architecture for computation of fuzzy logic inferences. Each single rule of the form

$$\text{if } x_1 \text{ is } A_1, x_2 \text{ is } A_2, \ldots, \text{ and } x_n \text{ is } A_n \text{ then } y \text{ is } B$$

is implemented using the basic neural structure shown in Figure 1. It is a four-layer network. Suppose A_i is described by a possibility distribution with n_i components. Then the input layer will have n groups of nodes, where the ith group has n_i nodes representing a fuzzy set for the ith antecedent variable. The input layer receives the fuzzy sets $A'_1, A'_2, \ldots, A'_n$ that characterize the possibility distribution of the facts if x_1 is A'_1, x_2 is $A'_2, \ldots,$ x_n is A'_n.

Let the fuzzy set A_i be characterized by the membership values $A_i = \{a_{ij}, j = 1, 2, \ldots, n_i\}$. Without loss of generality, we assume that $n_i = p \ \forall i = 1, 2, \ldots, n$. The second layer of the net has n nodes and the ith node in layer 2 is connected to ith group of p nodes in layer 1, which corresponds to the antecedent clause x_i *is* A_i.

Let w_{ij} be the connection weight between the ith node of layer 2 {i.e., the node corresponding to the ith antecedent clause) and the jth node of the ith

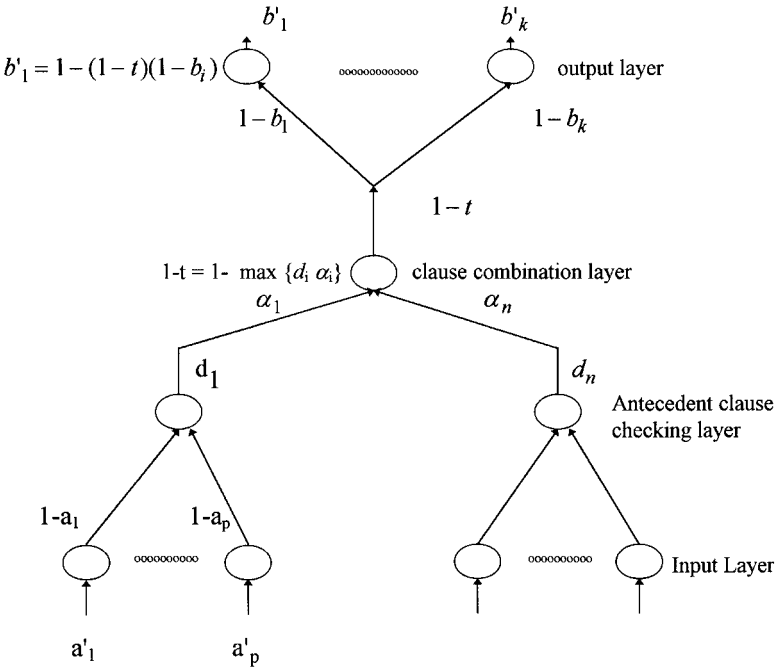

Figure 1. The neural network architecture of Keller et al.[10] for fuzzy logic inference.

group of nodes in layer 1. With a view to generating a measure of disagreement between the input possibility distribution (x_i *is* A'_i) and the antecedent clause (x_i *is* A_i), Keller et al. set the connection weight $w_{ij} = 1 - a_{ij}$, i.e., the complement of A. Then they computed a local measure of similarity between A and A' through a measure of disagreement between the complement set A^c and A'. Thus, the kth node of the second layer computes the level of disagreement between the kth antecedent clause and the kth input fuzzy set.

Keller et al. suggested the following measure of disagreement:

$$d_k = \max_j (w_{kj} * a'_{kj}) = \max_j \left((1 - a_{kj}) * a'_{kj} \right)$$

Here $*$ denotes an operator, either product or minimum. When $*$ is product and min we get the following measures of disagreement:

$$d_k^1 = \max_j \left((1 - a_{kj}) . a'_{kj} \right)$$

and

$$d_k^2 = \max_j \left\{ \min \left((1 - a_{kj}), a'_{kj} \right) \right\}$$

They also used another measure of disagreement: $d_k^3 = \max(|a_{kj} - a'_{kj}|)$, where $a'_{kj} = \mu_{A'_k}(x_k)$.

Using these operators, the disagreement values for all nodes (each node corresponds to an antecedent clause) are combined to obtain an overall level of disagreement between the antecedent clauses and the input data. This disagreement value provides an inhibiting signal for the firing of the rule. The connecting links between layers 2 and 3 are represented by α_i, where α_i is chosen subjectively. The weight α_i corresponds to the importance of the various antecedent clauses, which are supplied subjectively.

The third layer, known as clause combination layer, has exactly one node corresponding to each antecedent clause, which combines the disagreement values produced by the nodes in layer 2. The combination node computes $1 - t = 1 - \max_i\{\alpha_i \cdot d_i\}$. The output layer has m nodes, each corresponding to the m components of the output fuzzy set B'. The connection weight between the combination node and the ith node of the output layer is set as $u_i = 1 - b_i$. The activation function of each output node is given by

$$b_i' = 1 - u_i(1 - t) = b_i + t - b_i t$$

$(1 - t)$ can be interpreted as the level of agreement between the antecedent clause and the input fuzzy sets.

4. OPTIMAL CHOICE OF α

This network has several interesting properties.[10] For example, if $t = 0$ (total agreement), then the computed conclusion is exactly *y is B*. On the other hand, if $t = 1$ (total disagreement), then the output is all 1, i.e., *y is UNKNOWN*. However, except for d^3, $t \neq 0$ even when $A' = A$.

The method of computation of the disagreement is such that even two widely different input fuzzy sets may result in the same value of disagreement and hence the same output. Let us illustrate this with an example. For simplicity, we consider a rule with one antecedent clause and one consequent clause, if *x is LOW then y is HIGH*. Note that in Ref. 10, the same rule was used.

We use the same membership functions as in Ref. 10, but with more quantization levels. Table I shows the membership functions LOW (LO) and HIGH (HI). Table I also includes the other membership functions used by Keller et al. In addition to these, we have generated two distorted versions of LOW, named L1 and L2. The input fuzzy sets for LOW (LO), VERY LOW (VL), MORE OR LESS LOW (ML), NOT LOW (NL), and HIGH (HI) are shown in Figure 2(a). Figure 2(b) depicts the input fuzzy sets LOW (LO) and L1 and L2—the distorted versions of LOW (LO). It is evident from Table I that LO is more similar to L1 than L2, and we expect a higher disagreement value for L2 than that for L1. But d^1, d^2, and d^3 all show the same disagreement value and hence the same output fuzzy set B', for both L1 and L2. This happens because of the specific choice of d^1, d^2, d^3 used; this is not a desirable property. In Ref. 10, $\alpha_i = 1$ was used and the authors mentioned that α_i could be learned but no results or guidelines were provided.

Table I. Different fuzzy sets used.

Lab	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
LO	1.0	0.83	0.67	0.50	0.33	0.17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HI	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.0
ML	1.0	0.91	0.82	0.71	0.57	0.42	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
VL	1.0	0.70	0.45	0.25	0.11	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NL	0.0	0.17	0.33	0.50	0.67	0.83	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
L1	1.0	0.83	0.67	0.75	0.33	0.17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
L2	1.0	0.92	0.83	0.75	0.50	0.25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

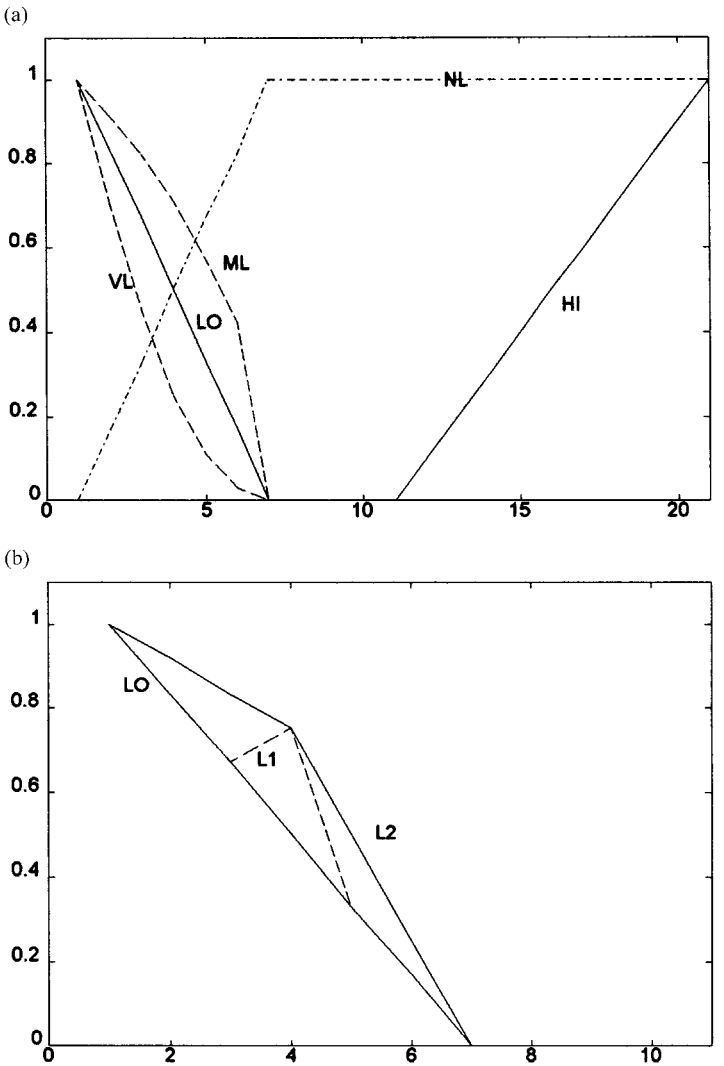

Figure 2. Fuzzy sets used for learning of: *if x is LOW then y is HIGH* (a) five input sets —LO, ML, NL, VL, and HI and (b) three input sets—LO, L1, and L2.

Next we address the issue of α tuning and direct computation of the optimal α for a special case.

4.1. Tuning of α

Tuning α raises an important issue—what would be the training data? Suppose we want to learn *if x is LOW then y is HIGH*. As a first choice, what comes to our mind is to use the data corresponding to the pair (LOW, HIGH).

Let X be the vector containing the membership values corresponding to LOW and Y be the same for HIGH. Now when X is given as an input to the net, suppose the network produces an output vector Y'. We can now learn α, so that $\|Y - Y'\|^2$ is minimum.

This choice is not a good one because the net may learn the relation *if x is LOW then y is HIGH* quite well, but the net may be so much biased to this relation that it may fail to realize the right kind of generalization capability. Under such a situation, the reasoning network might start behaving like an ordinary MLP, i.e., approximate the input–output mapping but lose its reasoning ability. We generate the training data using the same concept as Keller and Tahani.[15]

Our objective is not just to learn the relation *if x is A then y is B*. The net should learn in such a manner that when the input is A', the net should produce a B', so that the similarity/dissimilarity between A and A' is reflected between B and B'. Therefore, the training set should contain such (A', B'). For example, in the case of *if x is LOW then y is HIGH*, we can generate training data using the following case:

if x is VERY LOW then y is VERY HIGH
if x is MORE OR LESS LOW then y is MORE OR LESS HIGH
if x is NOT LOW then y is UNKNOWN

Thus, to make the net learn *if x is LOW then y is HIGH*, we train it using the four sets of training vectors (LOW, HIGH), (VERY LOW, VERY HIGH), (MORE OR LESS LOW, MORE OR LESS HIGH), and (NOT LOW, UNKNOWN). These four are very natural choices for (LOW, HIGH) relation. Some other distorted cases like (LOW′, HIGH′) may also be added to the set. One can, of course, argue against the use of (VERY LOW, VERY HIGH) as this may cause the system to make a conclusion that is more specific than HIGH, but this is not important in the present case. Our intention is to show that you can always have some reasonable and consistent training data. Moreover, since $b_i' \geq b_i$,[10] the net will never make a more specific conclusion than B.

We emphasize here that the above set of four vectors is needed just to learn the relation *if x is LOW then y is HIGH*. One might feel that the net is also training for the relation *if x is VERY LOW then y is VERY HIGH* or *if x is MORE OR LESS LOW then y is MORE OR LESS HIGH*, but this is not correct, because to learn the relation (VERY LOW, VERY HIGH), the training set may contain (LOW, HIGH), (VERY LOW, VERY HIGH), and (MORE OR LESS LOW, MORE OR LESS HIGH) and must contain (NOT VERY LOW, UNKNOWN).

In this context we mention that other trainable network structures have also been used to perform fuzzy logic inference. In Refs. 14–16, standard multilayer perceptrons were used to learn rule families for applications of fuzzy logic inference. The most desirable aspect of these structures is their ability to generalize the inference process from training data. The properties of fuzzy aggregation networks were studied[17,20] with regard to both their ability (accuracy) to perform inference and their generalization capabilities. Some compar-

isons of techniques can be found in Refs. 19 and 21. The other side of MLP for fuzzy logic inferencing is that the logical structure of fuzzy reasoning is lost and it behaves like a black-box-type function approximator, which usually picks up one of several possible generalizations (each corresponding to a local minimum of the error function that drives the learning process). Usually such a system provides good generalizations,[19,21] but it can settle to a very bad (undesirable) generalization too. However, in the present case, even with a tuned value of α, the fixed weight network preserves the logical structure of fuzzy inferencing and hence the chance of a very bad generalization is drastically reduced. Thus with a tuned value of α, we can get some generalizability like an MLP, yet retain the understandability of the fixed weight net.

Now we derive the learning algorithm for α. Suppose we want to learn *if x is A then Y is B*. Let the training set be $T = \{(A_i, B_i), i = 1, 2, N\}$, where $A_1 = A$ and $B_1 = B$, and (A_i, B_i), $i = 2, \ldots, N$, are $(N - 1)$ pairs of fuzzy sets that are semantically consistent with (A, B). When A_i is an input to the system, the desired output is B_i. Let the output produced by the net be denoted by O_i, $i = 1, 2, \ldots, N$. Suppose each of B_i and O_i is represented by a p-component membership vector. Then we can learn α by minimizing

$$E = \sum_{i=1}^{N} e_i = \sum_{i=1}^{N} \left(\|B_i - O_i\|\right)^2 \tag{5a}$$

or

$$E = \sum_{i=1}^{N} \sum_{j=1}^{p} \left(\beta_{ij} - O_{ij}\right)^2 \tag{5b}$$

where $B_i = (\beta_{i1}, \beta_{i2}, \ldots, \beta_{ip})$ is the desired output vector corresponding to the training data vector A_i, and $O_i = (O_{i1}, O_{i2}, \ldots, O_{ip})$ is the computed output vector for A_i.

We attempt to minimize E by gradient descent on the sample error function e_i. Ignoring the subscript i, for the sake of notational simplicity, we update α at the tth step as

$$\alpha_t = \alpha_{t-1} - \eta * \frac{\partial e}{\partial \alpha_{t-1}}$$

where η is the learning coefficient. Here,

$$\frac{\partial e}{\partial \alpha} = \frac{\partial \sum_{j=1}^{p} \left(\beta_j - O_j\right)^2}{\partial \alpha} = \frac{\partial \sum_{j=1}^{p} \left(\beta_j - b_j - \alpha d + \alpha b_j d\right)^2}{\partial \alpha} \tag{6}$$

The update equation thus becomes

$$\alpha_t = \alpha_{t-1} + \eta d \sum_{j=1}^{p} \left(\beta_j - O_j\right)\left(1 - b_j\right) \tag{7}$$

For a batch version of the learning algorithm, the update equation will be

$$\alpha_t = \alpha_{t-1} + \eta \sum_{i=1}^{N} d_i \sum_{j=1}^{p} (\beta_{ij} - O_{ij})(1 - b_{ij}) \tag{8}$$

A straightforward extension of the learning rule for the case with more than one antecedent clause is not possible because of the max operator used in the computation of t in layer 3.

To get around this problem, we propose to use a *soft-max* operator (SM) instead of max. Two possible choices for SM are

$$\mathrm{SM1}(x_1, \ldots, x_n) = \frac{\sum_{i=1}^{n} x_i \exp(-sx_i)}{\sum_{i=1}^{n} \exp(-sx_i)} \tag{9}$$

and

$$\mathrm{SM2}(x_1, \ldots, x_n) = \frac{\{\sum_{i=1}^{n} x_i\}^{1/s}}{n^{1/s}} \tag{10}$$

Note that

$$\lim_{s \to -\infty} \mathrm{SM1}(x_1, \ldots, x_n) = \max\{x_1, \ldots, x_n\}$$

and

$$\lim_{s \to \infty} \mathrm{SM2}(x_1, \ldots, x_n) = \max\{x_1, \ldots, x_n\}$$

Therefore, choosing a reasonably large (negative or positive depending on the case) value for s, practically we can realize the max operator and yet we can use calculus to derive the learning rules. Denoting $\gamma_i = \alpha_i d_i$ we can compute

$$t = \mathrm{SM1}(\gamma_1, \ldots, \gamma_n)$$

or

$$t = \mathrm{SM2}(\gamma_1, \ldots, \gamma_n)$$

Subsequently, the learning rules can be derived. For an application system there could be many rules, and in that case, for each rule one such net can be trained. However, for the case with a single antecedent clause, we can directly calculate the *optimal* value of α, without learning as

$$E = \sum_{i=1}^{N} \sum_{j=1}^{p} (\beta_{ij} - b_j - \alpha d_i + b_j \alpha d_i)^2 \tag{11}$$

Now

$$\frac{\partial E}{\partial \alpha} = 0 \quad \Rightarrow \quad 2\sum_{i=1}^{N}\sum_{j=1}^{p}\left\{(\beta_{ij} - b_j)(b_j - 1)d_i - \alpha d_i^2(1 - b_j)(b_j - 1)\right\} = 0 \tag{12}$$

$$\Rightarrow \quad \sum_{i=1}^{N} d_i \sum_{j=1}^{p} (\beta_{ij} - b_j)(b_j - 1) = \alpha \sum_{i=1}^{N} d_i^2 \sum_{j=1}^{p} (1 - b_j)(b_j - 1) \tag{13}$$

$$\Rightarrow \quad \alpha = \frac{\sum_{i=1}^{N} d_i \sum_{j=1}^{p} (\beta_{ij} - b_j)(b_j - 1)}{\sum_{i=1}^{N} d_i^2 \sum_{j=1}^{p} (1 - b_j)(b_j - 1)} \tag{14}$$

5. EVALUATION OF THE NETWORK

In order to evaluate the performance of net, in Ref. 10 two indices Avg (average distance) and Max (maximum distance) as defined next were used.

Average Distance (Avg)

Let $B = \{b_i\}$ be the vector representing the desired (target) output and let $B' = \{b_i'\}$ be the output produced by the net. Then

$$\text{Avg} = \frac{\sum_{i=1}^{p} |b_j - b_j'|}{p}$$

Maximum Distance (Max)

Max is defined as

$$\max = \max_j \left\{|b_j - b_j'|\right\}$$

Note that Keller et al. computed Max and Avg with respect to B, when the network was set for the relation if *x is A then y is B*. In the present investigation, we propose to compute them as distances from the target fuzzy set in the training data. In other words, for the training data (A_i, B_i), we compute Avg and Max using the pair (B_i, B_i'), where B_i' is the conclusion suggested by the net when A_i was the input.

In addition to Avg and Max, we also use a measure of fuzziness for evaluation of the effectiveness of the network.

Measures of Fuzziness

A measure of fuzziness gives an idea about the average ambiguity in a fuzzy set. It quantifies the average ambiguity in making a decision whether an element

belongs to the set or not. The fuzziness of a crisp set using any measure should be zero, as there is no ambiguity about whether an element belongs to the set or not. If a set is maximally ambiguous [$\mu_A(x) = 0.5\ \forall x$], then its fuzziness should be maximum. When a membership value approaches either 0 or 1, ambiguity about whether an element belongs to the set decreases. More formally, a measure of fuzziness for a discrete fuzzy set is a mapping H: $P(X) \to R^+$ that quantifies the degree of fuzziness present in A; $P(X)$ is the power set of X.

According to Ebanks,[26] a measure of fuzziness should satisfy the following five axioms for $A, B \in P(X)$:

AXIOM P1: SHARPNESS. $H(A) = 0 \Leftrightarrow \mu_A(x) = 0$ *or* $1\ \forall x \in X$.

AXIOM P2: MAXIMALITY. $H(A)$ *is maximum* $\Leftrightarrow \mu_A(x) = 0.5\ \forall x \in X$.

AXIOM P3: RESOLUTION. $H(A) \geq H(A^*)$, *where* A^* *is a sharpened version of* A.

AXIOM P4: SYMMETRY. $H(A) = H(A^c)$, *where* $\mu_{A^c}(x) = 1 - \mu_A(x)\ \forall x \in X$.

AXIOM P5: VALUATION. $H(A \cup B) + H(A \cap B) = H(A) + H(B)$.

Ebanks proposed another requirement called *generalized additivity*, but according to Pal and Bezdek[27] Axioms P1–P5 are sufficient requirements for measures of fuzziness. Pal and Bezdek[27] proposed two new classes of fuzziness measure: the *multiplicative* class and the *additive* class.

A measure of fuzziness under the multiplicative class is defined as follows: Let f: $[0, 1] \to \Re^+$ be concave increasing in $[0, 1]$. Define $\hat{g}(t) = f(t)f(1 - t)$ and $g(t) = \hat{g}(t) - \text{Min}_{0 \leq t \leq 1}\{\hat{g}(t)\}$. Then $H_*(A) = K\sum_{i=1}^{n} g(\mu_i)$, $K \in \Re^+$, satisfies P1–P5.

As an example, define $f(t) = te^{1-t}$. Then under the multiplicative class we get

$$H_*(A) = H_{*\,\text{QE}}(A) = K\sum_{i=1}^{n}\{\mu_i(1 - \mu_i)\} \tag{15}$$

the quadratic entropy of a fuzzy set.[27]

Measures of fuzziness under the additive class are defined in the following way. Let f: $[0, 1] \to \Re^+$ and be concave. Define $\hat{g}(t) = f(t) + f(1 - t)$ and $g(t) = \hat{g}(t) - \text{Min}_{0 \leq t \leq 1}\{\hat{g}(t)\}$. Then $H_+(A) = K\sum_{i=1}^{n} g(\mu_i)$, $K \in \Re^+$, satisfies P1–P5 and is a measure of fuzziness.

We now illustrate the additive class with the same $f(t)$ that we used for the multiplicative class, i.e., $f(t) = te^{1-t}$. Then $H_+(A) = \Sigma\{\mu_i e^{1-\mu_i} + (1 - \mu_i)e^{\mu_i}\}$ is a measure of fuzziness[27] (subject to adjustment of constants).

In order to compare the desired output with the output suggested by the network, in addition to the average distance and maximum distance, we propose to use a measure of fuzziness. If the (Avg or Max) distance between two fuzzy sets is small and the measures of fuzziness for them also are comparable, then we can infer, possibly with more confidence, that the fuzzy characteristics of the desired output fuzzy set are preserved in the computed fuzzy set. Comparison of only distance may not be enough, as there can be two widely different fuzzy sets B' and B'' such that the distances between B and B', and B and B'' are comparable. Similarly, only fuzziness may not be adequate to compare two fuzzy sets. In the present investigation, as an illustration, we have used only the quadratic fuzzy entropy, i.e., Eq. 15.

6. RESULT WITH OPTIMAL α

In order to learn the relation (LOW, HIGH) in Table I, we used (LOW, HIGH), (VERY LOW, VERY HIGH), (MORE OR LESS LOW, MORE OR LESS HIGH), and (NOT LOW, UNKNOWN) to learn α and compute the optimal α. Table II shows the α obtained from the batch learning (Eq. 8) and the computed optimal α (Eq. 14). We have experimented with α produced by both the batch version of the learning algorithm (Eq. 8) and the optimal α (Eq. 14). Since α values produced by Eqs. 8 and 14 are very close (practically the same), in our subsequent discussion we report results obtained with the optimal α.

Table III(a) depicts the output fuzzy sets produced (with d^1) by the original method of Keller et al.[10] with $\alpha = 1$, while Table III(b) shows the output obtained from the same method using the optimal value α as computed by Eq. 14.

Table IV presents the different performance indices d, Avg, Max, and the difference in fuzziness between the desired and the computed fuzzy sets both for $\alpha = 1$ and the optimal value of α. The total square error with $\alpha = 1$ is 1.103, while the same with the optimal α ($\alpha = 0.8254$) is 0.96. Clearly, the optimal α gives better overall performance. Table IV reveals that, except for NOT LOW (NL), in all cases optimal α produces better agreement (in term of all three performance measures) between the desired and computed outputs.

Table II. α obtained by batch algorithm Eq. 8 and optimal α by Eq. 14.

d^k Used	α by Batch Learning	Optimal α
d^1	0.8265	0.8254
d^2	0.605	0.6036
d^3	0.8927	0.8926
d^4	0.9430	0.9423

Table III. Output fuzzy sets.

Lab	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(a) Obtained by $\alpha = 1$ with d^1																					
LO	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.32	0.40	0.48	0.55	0.62	0.70	0.77	0.85	0.92	1.0
ML	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.45	0.51	0.57	0.63	0.69	0.75	0.82	0.88	0.94	1.0
VL	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.23	0.32	0.40	0.49	0.57	0.66	0.74	0.83	0.91	1.0
NL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
L1	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.44	0.50	0.56	0.62	0.69	0.75	0.81	0.88	0.94	1.0
L2	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.44	0.50	0.56	0.62	0.69	0.75	0.81	0.88	0.94	1.0
(b) Obtained with the optimal α with d^1																					
LO	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.29	0.37	0.44	0.52	0.60	0.68	0.76	0.84	0.92	1.0
ML	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.39	0.45	0.52	0.59	0.66	0.73	0.80	0.86	0.93	1.0
VL	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.21	0.30	0.39	0.47	0.56	0.65	0.74	0.82	0.91	1.0
NL	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.83	0.84	0.86	0.88	0.90	0.91	0.93	0.95	0.97	0.98	1.0
L1	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.38	0.45	0.52	0.59	0.65	0.72	0.79	0.86	0.93	1.0
L2	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.38	0.45	0.52	0.59	0.65	0.72	0.79	0.86	0.93	1.0

Table IV. Performance evaluation for d^1.

	$\alpha = 1$				With Optimal $\alpha = 0.8254$			
Input	d	Avg	Max	Ent. Diff.	d	Avg	Max	Ent. Diff.
LO	0.25	0.18	0.25	0.42	0.25	0.15	0.21	0.37
ML	0.38	0.22	0.38	0.51	0.38	0.18	0.32	0.48
VL	0.15	0.19	0.33	0.35	0.15	0.17	0.31	0.31
NL	1.00	0.00	0.00	0.00	1.00	0.13	0.17	0.44
L1	0.38	0.28	0.38	0.50	0.38	0.23	0.31	0.47
L2	0.38	0.28	0.38	0.50	0.38	0.23	0.31	0.47
	Average square error = 1.10				Average square error = 0.96			

Table V shows the results obtained by d^2, while Table VI presents the results with d^3. The values of different indices are reported in Tables VII and VIII.

The output fuzzy sets for all inputs are shown in Figures 3, 4, and 5, respectively, for d^1, d^2, and d^3. Figure 3(b) reveals that in the case of d^1 with optimal α, the conclusion for NL is practically unknown. For d^2 and d^3, as shown by Figures 4 and 5, the conclusions for VL, L1, and L2 are the same. This is not a desirable characteristic. It is clear from Tables III, V, and VI and from these figures that the outputs of L1 and L2 are the same for all cases. Tables IV, V, and VIII show that for d^1, d^2, and d^3, the disagreement value d is the same for both L1 and L2. Thus the distorted versions L1 and L2 produce the same output, although the inputs are significantly different. This is evident from Tables III, V, and VI.

7. A MODIFIED SCHEME

7.1. The New Architecture

A new network structure that makes the original network of Keller et al. more powerful is shown in Figure 6. We modify only that part of the network that computes the disagreement value d_i between the input and the antecedent. In Ref. 10, d_is were computed using the Max-Min or Max-Prod or Sup-norm operation. As a result, as illustrated in Tables III, V, and VI, several widely different inputs might result in the same output fuzzy set. In order to eliminate this undesirable characteristic, we use a different method, based conceptually on the same idea of disagreement used by Keller et al. In Ref. 10 the dissimilarity between the antecedent (A) and the input fuzzy set (A') was computed by a measure of similarity between A^c and A'. In the present case, we use the same concept, but our measure of disagreement is

$$d = \frac{\langle A^c, A' \rangle}{\|A^c\| \, \|A'\|} = d^4 \quad \text{(say)} \tag{16}$$

Table V. Output fuzzy sets.

Lab	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(a) Obtained by $\alpha = 1$ with d^2																					
LO	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.0
ML	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.62	0.66	0.70	0.74	0.79	0.83	0.87	0.91	0.96	1.0
VL	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.40	0.46	0.53	0.60	0.66	0.73	0.80	0.87	0.93	1.0
NL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
L1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.0
L2	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.0
(b) Obtained with the optimal α with d^2																					
LO	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.37	0.44	0.51	0.58	0.65	0.72	0.79	0.86	0.93	1.0
ML	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.41	0.48	0.54	0.61	0.67	0.74	0.80	0.87	0.93	1.0
VL	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.28	0.36	0.44	0.52	0.60	0.68	0.76	0.84	0.92	1.0
NL	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.64	0.68	0.72	0.76	0.80	0.84	0.88	0.92	0.96	1.0
L1	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.37	0.44	0.51	0.58	0.65	0.72	0.79	0.86	0.93	1.0
L2	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.37	0.44	0.51	0.58	0.65	0.72	0.79	0.86	0.93	1.0

Table VI. Output fuzzy sets.

Lab	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(a) Obtained by $\alpha = 1$ with iid[3]																					
LO	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.0
ML	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.32	0.40	0.47	0.55	0.62	0.70	0.77	0.85	0.92	1.0
VL	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.32	0.40	0.48	0.55	0.62	0.70	0.77	0.85	0.92	1.0
NL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
L1	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.32	0.40	0.48	0.55	0.62	0.70	0.77	0.85	0.92	1.0
L2	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.32	0.40	0.48	0.55	0.62	0.70	0.77	0.85	0.92	1.0
(b) Obtained by optimal α with d^3																					
LO	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.0
ML	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.30	0.37	0.45	0.53	0.61	0.69	0.77	0.84	0.92	1.0
VL	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.30	0.38	0.46	0.53	0.61	0.69	0.77	0.84	0.92	1.0
NL	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.90	0.91	0.92	0.94	0.95	0.96	0.97	0.98	0.99	1.0
L1	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.30	0.38	0.46	0.53	0.61	0.69	0.77	0.84	0.92	1.0
L2	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.30	0.38	0.46	0.53	0.61	0.69	0.77	0.84	0.92	1.0

Table VII. Performance evaluation for d^2.

	$\alpha = 1$				With Optimal $\alpha = 0.6036$			
Input	d	Avg	Max	Ent. Diff.	d	Avg	Max	Ent. Diff.
LO	0.50	0.37	0.50	0.50	0.50	0.22	0.30	0.46
ML	0.57	0.35	0.57	0.47	0.57	0.20	0.35	0.50
VL	0.33	0.32	0.44	0.54	0.33	0.23	0.36	0.42
NL	1.00	0.00	0.00	0.00	1.00	0.29	0.40	0.76
L1	0.50	0.37	0.50	0.50	0.50	0.22	0.30	0.46
L2	0.50	0.37	0.50	0.50	0.50	0.22	0.30	0.46
	Average square error = 1.66				Average square error = 1.19			

where A^c and A' are vectors of membership values denoting the corresponding fuzzy sets, $\langle \cdot \rangle$ denotes the dot product, and $\| \cdot \|$ represents the Euclidean norm. To implement Eq. 16 we propose a four-layer network. The first layer (layer 1) has p neurous that take $A' = (a'_1, \ldots, a'_p)$ as input. The second layer has one node, while the third and fourth layers have p nodes each. All nodes in layer 1 are connected to the second layer node. Moreover, every node of layer 1 is connected to the corresponding node of layer 3. All third layer nodes are connected to the second layer node. In addition, all third layer nodes are connected to a single node in layer 4 that finally computes the disagreement value.

Thus each node in layer 1 has two outgoing links and the node in the second layer has p incoming links and p outgoing links, while each node in the third layer has two incoming links and one outgoing link. Note that the subnet just described is needed to compute the disagreement value for one atomic antecedent clause. If the antecedent part of the rule has n such atomic antecedent clause, then n such subgroups—one for each clause—will be needed (see Fig. 6).

Let $W_{ij}^{l,m}$ be the connection between node i in layer l and node j in layer m. The different connection weights are $w_{ij}^{1,2} = 1$, $w_{ii}^{1,3} = 1$, $w_{1j}^{2,3} = 1$, and $w_{i1}^{3,4} = (1 - a)/\|A^c\|$. We denote the output from the ith node of layer l to the jth node of layer m by $o_{ij}^{l,m}$, i.e., the input received by the jth node of layer m by

Table VIII. Performance evaluation for d^3.

	$\alpha = 1$				With Optimal $\alpha = 0.8926$			
Input	d	Avg	Max	Ent. Diff.	d	Avg	Max	Ent. Diff.
LO	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ML	0.24	0.15	0.24	0.42	0.24	0.14	0.22	0.39
VL	0.25	0.26	0.39	0.48	0.25	0.24	0.37	0.45
NL	1.00	0.00	0.00	0.00	1.00	0.08	0.11	0.29
L1	0.25	0.18	0.25	0.42	0.25	0.16	0.22	0.39
L2	0.25	0.18	0.25	0.42	0.25	0.16	0.22	0.39
	Average square error = 0.82				Average square error = 0.76			

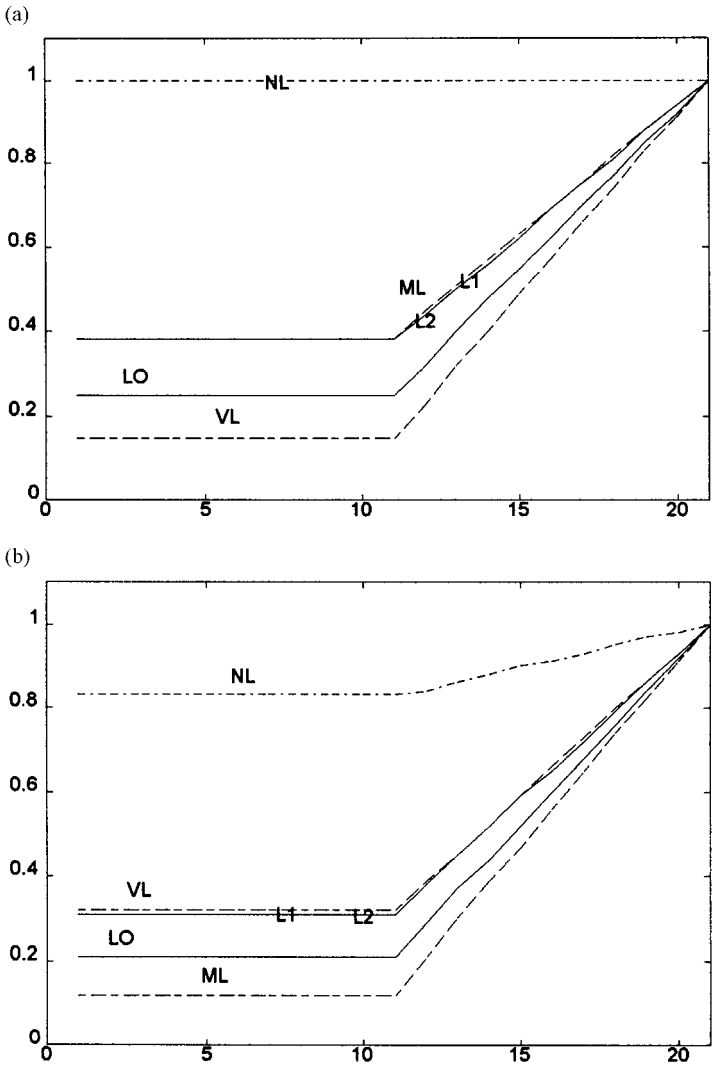

Figure 3. Output fuzzy sets for the rule *if x is LOW then y is HIGH* with d^1 (a) using $\alpha = 1$ and (b) using optimal α.

the ith node of layer m. The activation functions of the neurons in different layers are different. The nodes in layer 1 simply transfer the signal unattenuated, i.e., the activation function is $f(x) = x$. The transfer function of the second layer node is

$$f\left(o_{i1}^{1,2}, i = 1, \ldots, p\right) = \sqrt{\sum_{i=1}^{p} \left(w_{i1}^{1,2} \cdot o_{i1}^{1,2}\right)^2}$$

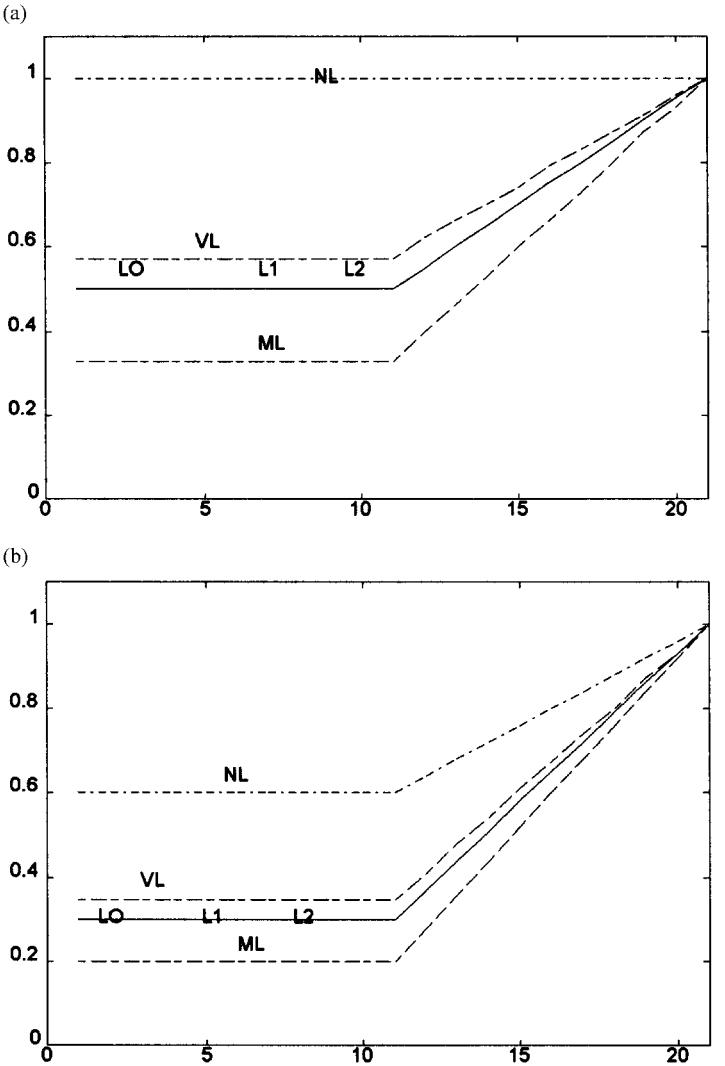

Figure 4. Output fuzzy sets for the rule *if x is LOW then y is HIGH* with d^2 (a) using $\alpha = 1$ and (b) using optimal α.

A third layer node computes

$$f\left(o_{ii}^{1,3}, o_{1i}^{2,3}\right) = \frac{w_{ii}^{1,3} \cdot o_{ii}^{1,3}}{w_{1i}^{2,3} \cdot o_{1i}^{2,3}} = o_{i1}^{3,4}$$

The activation function of the fourth layer neuron is given by

$$f\left(o_{i1}^{3,4}, i = 1, \ldots, p\right) = \sum_{i=1}^{p} w_{i1}^{3,4} \cdot o_{i1}^{3,4}$$

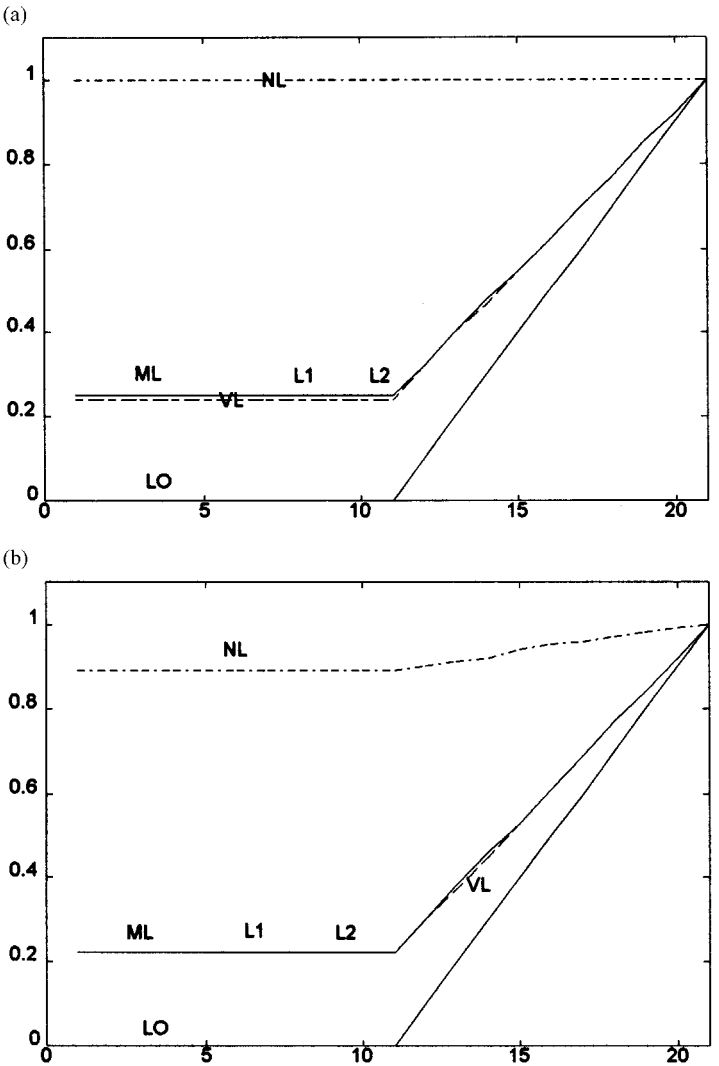

Figure 5. Output fuzzy sets for the rule *if x is LOW then y is HIGH* with d^3 (a) using $\alpha = 1$ and (b) using optimal α.

It is easy to see that the output of the fourth layer node can be obtained from Eq. 16 and the entire operation is easily implementable in a neural paradigm.

The architecture of and the operation in the fifth and sixth layers are, respectively, identical to those of layers 3 and 4 of the original network proposed in Ref. 10. Keller et al. proved some results about their network. Here we prove some such results for the modified network.

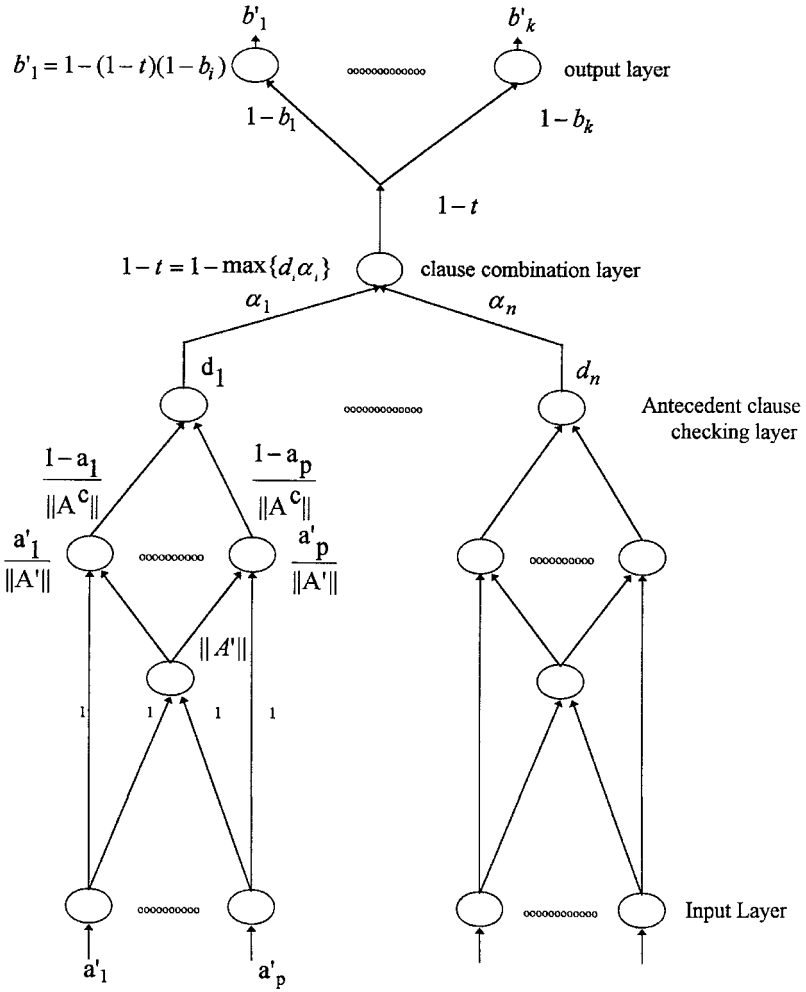

Figure 6. Modified neural network architecture for fuzzy logic inference.

THEOREM 1. *Consider the net with a single antecedent clause. Suppose A is a crisp subset of its universe of discourse. The proposed network produces the standard modus ponens results, i.e., if the input x is A is such that $A' = A$, then the network result is y is B.*

Proof. Let A be a crisp subset of its domain of discourse and $A' = A$. Then

$$d^4 = \sum_{i=1}^{p} \frac{(1 - a_i)a_i}{\|A^c\| \, \|A\|} = \frac{1}{\|A^c\| \, \|A\|} \sum_{i=1}^{p} (1 - a_i)a_i$$

Now since A is crisp, $a_i \in \{0, 1\}$. Hence, $(1 - a_i)(a_i) = 0 \ \forall i = 1, \ldots, p$. This results in $d^4 = 0$. Therefore,

$$b'_i = b_i + \alpha \cdot 0 - b_i \cdot \alpha \cdot 0 \quad \text{or} \quad b'_i = b_i \qquad \forall i = 1, 2, \ldots, p$$

Thus $B = B'$ and hence the proof. ■

Note that there is no restriction on B—it could be either crisp or fuzzy.

COROLLARY 1. *For a multiclause network, suppose $A_1, A_2, \ldots, A_n$ are crisp subsets of their respective domains of discourse. Suppose the input x_1 is $A'_1, \ldots, x_n$ is A'_n be such that $A'_i = A_i$, $\forall i = 1, 2, \ldots, n$. Then the result is y is B.*

Proof. For each clause the disagreement node at layer 4 produces $d_i^4 = 0$ $\forall i = 1, 2, \ldots, n$ resulting in $t = 0$ at the clause combination node in layer 5. Hence the result follows. ■

THEOREM 2. *Consider the network with $\alpha = 1$. Suppose the input x is A' is such that $A' = A^c$. Then the network produces y is UNKNOWN; i.e., the possibility distribution of y is identically equal to* 1.

Proof. We have

$$d^4 = \sum_{i=1}^{p} \frac{(1 - a_i)a'_i}{\|A^c\| \, \|A'\|} = \sum_{i=1}^{p} \frac{(1 - a_i)(1 - a_i)}{\|A^c\| \, \|A^c\|} = 1$$

Now since $\alpha = 1$ and $t = 1$ at the clause combination node,

$$b'_i = b_i + \alpha \cdot 1 - b_i \cdot \alpha \cdot 1 \quad \text{or} \quad b'_i = 1 \qquad \forall i = 1, 2, \ldots, p \text{ since } \alpha = 1$$

COROLLARY 2. *Consider the net with multiple clauses and with $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 1$. Now suppose the input fuzzy sets A'_i, $i = 1, \ldots, n$, are such that $A'_i = A_i^c$ for some i. Then the network produces the result y is UNKNOWN.*

Proof. In the given situation at least one node, say k, in the fourth layer will produce $d_k^4 = 1$. Therefore, $\max_i\{d_i, \alpha_i\} = 1$. Thus the clause combination node produces $t = 1$. Hence the proof. ■

These properties show some desirable characteristics of the proposed network.

7.2. Results with New Architecture

We implemented the network described in Figure 6 for the same relation *if x is LOW then y is HIGH*. Table IX shows the output fuzzy sets obtained using

Table IX. Output fuzzy sets.

Lab	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(a) Obtained by $\alpha = 1$ with d^4																					
LO	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.24	0.32	0.41	0.49	0.58	0.66	0.75	0.83	0.92	1.0
ML	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.18	0.27	0.36	0.45	0.55	0.64	0.73	0.82	0.91	1.0
VL	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.28	0.36	0.44	0.52	0.60	0.68	0.76	0.84	0.92	1.0
NL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
L1	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.24	0.33	0.41	0.50	0.58	0.66	0.75	0.83	0.92	1.0
L2	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.26	0.34	0.43	0.51	0.59	0.67	0.75	0.84	0.92	1.0
(b) Obtained by optimal α with d^4																					
LO	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.23	0.31	0.40	0.49	0.57	0.66	0.74	0.83	0.91	1.0
ML	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.18	0.27	0.36	0.45	0.54	0.63	0.73	0.82	0.91	1.0
VL	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.27	0.35	0.43	0.51	0.59	0.67	0.76	0.84	0.92	1.0
NL	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.95	0.95	0.96	0.97	0.97	0.98	0.98	0.99	0.99	1.0
L1	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.24	0.32	0.41	0.49	0.58	0.66	0.75	0.83	0.92	1.0
L2	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.25	0.33	0.42	0.50	0.58	0.67	0.75	0.83	0.92	1.0

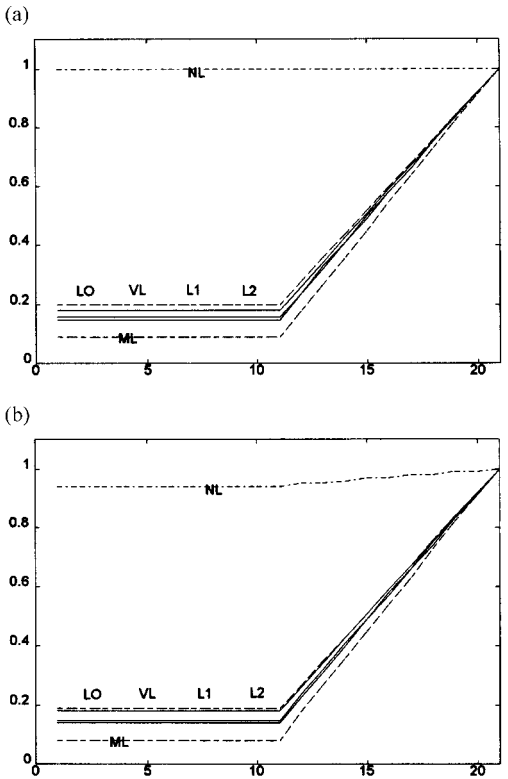

Figure 7. Output fuzzy sets for the rule *if x is LOW then y is HIGH* with d^4 (a) using $\alpha = 1$ and (b) using optimal α.

the new architecture. Figures 7(a) and (b) present a graphical representation of the output fuzzy sets obtained for $\alpha = 1$ and the optimal α, respectively.

As described, we find that the output fuzzy sets obtained for L1 and L2 are different. We make the following observations from Table IX and Figure 7: output fuzzy sets produced for LO, L1, and L2 are different for both optimal α and $\alpha = 1$. For both choices of α, the output fuzzy set corresponding to LOW (LO) is closer to the HIGH (HI) than the output fuzzy sets for L1 and L2. Table IX also reveals that the conclusions (output fuzzy sets) for L1 and L2 are less specific than that for LOW and, again, the conclusion for L2 is less specific than that for L1. This is reasonable as L1 and L2 are distorted versions of LOW and L2 has more distortion than L1. Moreover, the optimal α results in better conclusions. For example, with LOW (LO) as input, optimal α results in a conclusion [row corresponding to LO in Table IX(b)] closer to HIGH (HI in Table I) than that observed with $\alpha = 1$ [Table IX(a)]. This is also revealed by different performance evaluation indices in Table X. Observe that the input fuzzy set VERY LOW (VL) is more specific than LOW (LO), but the conclusion computed by the net with VL as input is less specific than HIGH (HI). This is a

Table X. Performance evaluation for d^4.

	$\alpha = 1$				With Optimal $\alpha = 0.9423$			
Input	d	Avg	Max	Ent. Diff.	d	Avg	Max	Ent. Diff.
LO	0.15	0.11	0.15	0.29	0.15	0.10	0.14	0.28
ML	0.09	0.15	0.30	0.25	0.09	0.14	0.29	0.24
VL	0.20	0.14	0.20	0.37	0.20	0.13	0.19	0.35
NL	1.00	0.00	0.00	0.00	1.00	0.04	0.06	0.16
L1	0.16	0.12	0.16	0.30	0.16	0.11	0.15	0.29
L2	0.18	0.13	0.18	0.33	0.18	0.12	0.17	0.32
	Average square error = 0.61				Average square error = 0.59			

desirable property, because we cannot make a conclusion more precise than HIGH (HI) from a rule *if x is LOW then y is HIGH*.

Comparison of Table IX with Tables III, V, and VI shows that the modified net results in better conclusions for L1 and L2. For example, the possibility distribution that resulted for L1 and L2 by the modified net is closer to HIGH (HI in Table I) than those produced by the original network of Keller et al. with d^1, d^2, and d^3. However, for d^3, the original network of Keller et al.[10] produces exactly the conclusion HIGH (HI), when LOW (LO) in Table I is given as input.

8. CONCLUSIONS

We have addressed the issue of selecting an optimal value of α for the network proposed by Keller et al. for a neural implementation of fuzzy logic. We have provided a learning algorithm for α and also have computed the optimal value of α for some special cases. We also proposed a modified architecture for neural implementation of fuzzy inferencing. Effectiveness of the modified net is established through numerical examples. The use of the proposed schemes in developing various application systems is currently under investigation and will be reported in the future.

The work of Kuhu Pal was supported by the Council of Scientific and Industrial Research of India. The research of Nikhil R. Pal was partially supported by the Department of Science and Technology of the Government of India, Grant No. III 5(21)/96-ET.

References

1. M. Sugeno, Ed., *Industrial Applications of Fuzzy Control*, North-Holland, Amsterdam, 1985.
2. C.C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller—Parts I and II, *IEEE Trans. Syst., Man, Cybernetics*, **20**, 404–435 (1990).
3. L.A. Zadeh, "Fuzzy logic," *IEEE Comput. Mag.*, **April**, 83–93 (1988).
4. K.L. Self, "Fuzzy logic design," *IEEE Spectrum*, **27**, 42–44 (1990).
5. E.M. Scharf and N.J. Nandie, *The Application of a Fuzzy Controller to the Control of a multi-degree-freedom Robot arm in Industrial Application of Fuzzy Control*, M. Sugeno, Ed., North-Holland, Amsterdam, 1985.

6. I.H. Suh and T.W. Kim, "Fuzzy membership function based neural networks with applications to the visual servoing of robot manipulators," *IEEE Trans. Fuzzy Syst.*, **2**, 203–220 (1994).
7. S. Haykin, *Neural Networks—A Comprehensive Foundation*, Macmillan, New York, 1994.
8. S.K. Pal and N.R. Pal, Soft computing: Goal, tools and feasibility, *J. IETE*, **42**, 195–204 (1996) (special issue on neural networks).
9. M.M. Gupta and D.H. Rao, "On the principles of fuzzy neural networks," *Fuzzy Sets and Syst.*, **61**, 1–18 (1994).
10. J.M. Keller, R.R. Yager, and H. Tahani, "Neural network implementation of fuzzy logic," *Fuzzy Sets and Syst.*, **45**, 1–12 (1992).
11. H. Takagi and I. Hayashi, "NN-driven fuzzy reasoning," *Int. J. Approx. Reasoning*, **5**, 191–212 (1991).
12. J.J. Shann and H.C. Fu, "A fuzzy neural network for rule acquiring on fuzzy control systems," *Fuzzy Sets and Syst.*, **71**, 345–457 (1995).
13. J. Nie and D. Linkens, *Fuzzy-Neural Control*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
14. J. Nie and D. Linkens, "Fuzzy reasoning implemented by neural networks, "*Proc. Int. Joint Conf. on Neural Networks*, Baltimore, MD, June 1992, Vol. II, pp. 702–706.
15. J. Keller and H. Tahani, "Backpropagation neural networks for fuzzy logic," *Inform. Sci.*, **45**, 1–12 (1992).
16. J. Keller and H. Tahani, "Implementation of conjunctive and disjunctive fuzzy logic rules with neural networks, "*Int. J. Approx. Reasoning*, **6**, 221–240 (1992) (special issue on fuzzy logic and neural networks for control).
17. J. Keller, R. Krishnapuram, and F. Rhee, "Evidence aggregation networks for fuzzy logic inference," *IEEE Trans. Neural Networks*, **3**, 761–769 (1992).
18. J. Keller and R. Yager, "Fuzzy logic inference neural networks," *Proc. SPIE Symp. on Intelligent Robots and Computer Vision VIII*, Philadelphia, PA, Nov. 1989, pp. 582–591.
19. J. Keller, "Experiments in neural network architectures for fuzzy logic," *Proc. Second Joint Tech. Workshop on Neural Networks and Fuzzy Logic*, Johnson Space Center, Houston, 1990, pp. 201–216.
20. J. Keller, Y. Hayashi, and Z. Chen, "Interpretation of nodes in networks for fuzzy logic, "*Proc. Second IEEE Int. Cong. on Fuzzy Systems*, San Francisco, CA, 1993, pp. 1203–1207.
21. J. Keller, "The inference process in fuzzy logic through artificial neural network structures, "*Proc. First European Cong. on Fuzzy and Intelligent Technologies*, Aachen, Germany, 1993, pp. 195–201.
22. Y. Hayashi, J.J. Buckley, and E. Czogala, "Direct fuzzification of neural network and fuzzified delta rule," *Proc. 2nd Int. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan, 1992, pp. 73–76.
23. H. Ishibuchi, R. Fujioka, and H. Tanaka, "Neural networks that learn from fuzzy if–then rules," *IEEE Trans. Fuzzy Syst.*, **1**, 85–97 (1993).
24. L.A. Zadeh, "Fuzzy sets," *Inform. Control*, **8**, 338–353 (1965).
25. G.J. Klir and T.A. Floger, *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
26. B.R. Ebanks, "On measures of fuzziness and their representations," *J. Math. Anal. Appl.*, **94**, 24–37 (1983).
27. N.R. Pal and J.C. Bezdek, "Measuring fuzzy uncertainty," *IEEE Trans. Fuzzy Syst.*, **2**, 107–118 (1994).