ELSEVIER

Fuzzy Sets and Systems 66 (1994) 1-13

FUZZY sets and systems

Invited Review

Fuzzy neural networks: A survey

James J. Buckley[a,*], Yoichi Hayashi[b]

[a] *Mathematics Department, University of Alabama at Birmingham, Birmingham, AL 35294, USA*
[b] *Department of Computer and Information Sciences, Ibaraki University, Hitachi-shi, Ibaraki 316 Japan*

Received February 1994

Abstract

In this paper a fuzzy neural network will be a layered, feedforward, neural net that has fuzzy signals and/or fuzzy weights. We survey recent results on learning algorithms and applications for fuzzy neural networks.

Key words: Neural networks; Learning algorithms; Regression; Fuzzy controller; Fuzzy expert systems; Hierarchical analysis; Fuzzy equations; Universal approximator

1. Introduction

In this section we will first describe what we mean by a neural net, hybrid neural net, fuzzy neural net, and hybrid fuzzy neural net. Then we introduce notation that will be used in the rest of the paper.

Consider the three layered, feedforward, neural net shown in Fig. 1. For simplicity we have assumed only two input neurons, one hidden layer, and one output neuron. We begin by having the signals and weights of all real numbers.

All neurons have a transfer function f which translates input to output. Usually the input neurons have $y = f(x) = x$ (no change in input) and all the other neurons have the sigmoidal function $y = f(x) = (1 + e^{-x})^{-1}$. However, the transfer function, in general, can be any mapping f from the real numbers into the real numbers. Also, we will usually not use a bias term in the sigmoidal function.

In this section we will have $f(x) = x$ in the two input neurons and the sigmoidal function in all other neurons. If the input signals are x_1 and x_2 (see Fig. 1), then the output from neuron $\#1$ ($\#2$) in the input layer is x_1 (x_2). The input to neuron $\#k$ in the hidden layer is

$$I_k = x_1 w_{1k} + x_2 w_{2k}, \quad 1 \leqslant k \leqslant K. \tag{1}$$

The output from hidden neuron $\#k$ will be

$$z_k = f(I_k), \quad 1 \leqslant k \leqslant K, \tag{2}$$

for sigmoidal f. It follows that the input to the output neuron is

$$I_0 = z_1 v_1 + \cdots + z_K v_K, \tag{3}$$

and its output is

$$y = f(I_0) \tag{4}$$

* Corresponding author.

SSDI 0165-0114(94)00103-E

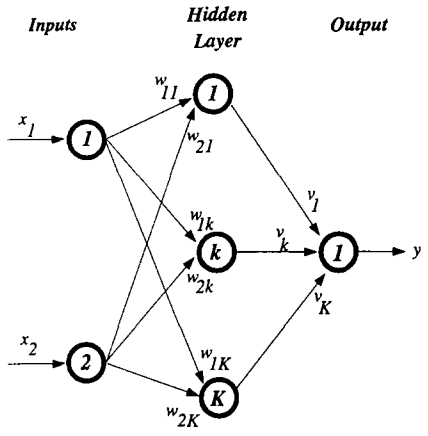

Fig. 1. Neural network.

for sigmoidal f. We will have sigmoidal f in the output neuron when we require y to be in $[0,1]$, otherwise we omit f and have $y = I_0$.

What we have just described is what we call a regular neural net (NN). If we employ other operations like a t-norm, or a t-conorm, to combine the incoming data to a neuron (Eqs. (1) and (3)) we obtain what we call a hybrid neural net (HNN). HNNs also process real number signals and have real number weights. There are many applications of HNNs (see [7,8,26,29,32]). We will be concerned with fuzzifying both NNs and HNNs in this paper.

A regular fuzzy neural net (FNN) is a NN with fuzzy signals and/or fuzzy weights. We differentiate between different types of FNNs as follows: (1) FNN_1 has real number input signals but fuzzy weights; (2) FNN_2 has fuzzy set input signals and real number weights; and (3) FNN_3 has both fuzzy set input signals and fuzzy weights. We will be predominately interested in FNN_3 in this paper.

Let us now describe in more detail the internal computations of a FNN_3. We place a bar over a symbol if it represents a fuzzy set and all our fuzzy sets will be fuzzy subsets of the real numbers. In a FNN_3 the inputs $\bar{X}_1, \bar{X}_2$, the weights $\bar{W}_{ik}, \bar{V}_k$, and the output $\bar{Y}$ will all be fuzzy. The architecture of FNN_3 will be the same as in Fig. 1. The output from input neuron $\#1$ ($\#2$) is $\bar{X}_1$ ($\bar{X}_2$). So, the input to hidden neuron $\#k$ is

$$\bar{I}_k = \bar{X}_1 \bar{W}_{1k} + \bar{X}_2 \bar{W}_{2k}, \quad 1 \leqslant k \leqslant K, \tag{5}$$

where we use standard fuzzy arithmetic to compute $\bar{I}_k$. The output from the kth hidden neuron will be

$$\bar{Z}_k = f(\bar{I}_k), \quad 1 \leqslant k \leqslant K, \tag{6}$$

for sigmoidal f, where the extension principle is used to obtain $\bar{Z}_k$. It follows that the input to the output neuron is

$$\bar{I}_0 = \bar{Z}_1 \bar{V}_1 + \cdots + \bar{Z}_K \bar{V}_K \tag{7}$$

and the final output will be

$$\bar{Y} = f(\bar{I}_0), \tag{8}$$

using regular fuzzy arithmetic in Eq. (7) and the extension principle in Eq. (8). When we do not need $\bar{Y}$ to be a fuzzy subset of $[0,1]$ we omit f in Eq. (8) and set $\bar{Y} = \bar{I}_0$.

What we have just described is a regular FNN_3 where standard fuzzy arithmetic (add, multiply) is used to compute the output. In a hybrid fuzzy neural net (HFNN) we may combine the fuzzy signals and weights using other operations besides addition and multiplication to obtain $\bar{I}_k$ and $\bar{I}_0$. HFNNs are of recent research interest (see [4,7,9,13]) and we shall discuss them again in the following sections.

In Section 2 we briefly survey the literature on FNN_i, $i = 1,2,3$, and HFNNs. Section 3 is concerned with learning algorithms for FNN_i, $i = 1,2,3$, and applications of fuzzy neural nets are outlined in Section 4. Section 5 contains a brief review and our conclusions.

We will now introduce notation to be employed in the rest of the paper. If $\bar{F}$ is a fuzzy set, then $\bar{F}(x)$ represents the membership function for $\bar{F}$ evaluated at x. The α-cut of a fuzzy set $\bar{F}$ is defined as

$$\bar{F}[\alpha] = \{x \mid \bar{F}(x) \geqslant \alpha\} \quad \text{for } 0 < \alpha \leqslant 1. \tag{9}$$

The $\alpha = 0$ cut of $\bar{F}$ is defined separately to be the closure of the union of all the $\bar{F}[\alpha]$, $0 < \alpha \leqslant 1$. $\bar{F}[0]$ is also called the support of $\bar{F}$. A triangular fuzzy number $\bar{N}$ is specified by three numbers $a < b < c$ where: (1) $\bar{N}(x) = 0$ for $x \leqslant a, x \geqslant c$ and

$\bar{N}(b) = 1$; and (2) the graph of $\bar{N}(x)$ is a straight line segment from $(a,0)$ to $(b,1)$ and from $(b,1)$ to $(c,0)$. We write $\bar{N} = (a/b/c)$. A triangular shaped fuzzy number is similar to a triangular fuzzy number except that the graph of $\bar{N}(x)$ need not be straight line segments on $[a,b]$ or on $[b,c]$. A fuzzy set is discrete when the membership function is positive at only a finite number of values of x. We say a triangular fuzzy number $\bar{N} = (a/b/c)$ is non-negative when $a \geq 0$. If $\bar{N}$ and $\bar{M}$ are two fuzzy sets, then we write $\bar{N} \leq \bar{M}$ when $\bar{N}(x) \leq \bar{M}(x)$ for all x.

2. Literature

Probably the first papers to introduce fuzzy sets into neural nets were [45,46] where the authors generalized the McCulloch–Pitts model by using intermediate values between zero and one. A survey paper [53] in 1990 discussed the fusion of neural nets and fuzzy logic, however very little research on fuzzy neural nets was done by then with the exception of [44] and Yamakawa's fuzzy neuron. In [44] the authors proposed adding fuzzy membership functions to the perceptron.

Yamakawa's initial fuzzy neuron is discussed in [56–58] and his new fuzzy neuron in [59]. His initial fuzzy neuron was of type FNN_1. The new fuzzy neuron has, instead of a single weight on each incoming arc to a neuron, a set of fixed fuzzy sets and real number weights. A learning algorithm is applied to the weights. Learning algorithms are discussed in more detail in Section 3. See also [50] for a discussion about improving Yamakawa's initial fuzzy neuron.

Similar to Yamakawa's new fuzzy neuron is the fuzzy neural net presented in [47,55]. They also have a collection of weights, and fuzzy sets, attached to each incoming arc to a neuron. Learning is applied to both the weights and the fuzzy set. The authors in [48,49] also use a FNN_1, present a learning algorithm, and suggest applications to fuzzy control.

Next, let us consider fuzzy neural nets of type FNN_2. In [51], they have fuzzy signals, real number weights, and a fuzzy threshold within the neurons. The authors in [36,37,39–41] have a FNN_2 with learning on the real weights performed by α-cuts on the signals, and they generalize to a FNN_3 in [38,41].

Gupta [21–25] has presented various models of a fuzzy neuron. These models include FNN_1, FNN_2 and FNN_3 but no learning algorithms were presented in these papers.

In a series of papers [4, 7, 9–11, 13, 27, 28, 30, 31, 33–35] the authors discuss fuzzy neural nets with emphasis on learning algorithms and applications of FNN_3.

The need for hybrid FNNs is twofold: applications and the fact that regular FNN_3s are not universal approximators. Applications using hybrid fuzzy neural nets [4,13] are discussed in Section 4 and the result that hybrid FNN_3s can be universal approximators [7,9] is discussed also in Section 4.

We will survey many of the results in the papers referenced above in the following two sections.

3. Learning

In this section we will survey learning algorithms for FNN_i, $i = 1, 2$, or 3, that have been studied in the literature. We concentrate mostly on FNN_3. No learning algorithms have been presented for HFNNs.

3.1. Fuzzy backpropagation

In [4, 11, 13, 28, 31, 34, 35] the authors developed a fuzzy backpropagation algorithm for FNN_3. Let the training set be $(\bar{X}_l, \bar{T}_l)$, $\bar{X}_l = (\bar{X}_{l1}, \bar{X}_{l2})$ for inputs and $\bar{T}_l$ desired output, $1 \leq l \leq L$. Given input $\bar{X}_l$ let the actual output be $\bar{Y}_l$. The authors assumed that $\bar{X}_{lj}$, $\bar{W}_{ik}$, and $\bar{V}_k$ are all triangular fuzzy numbers with $\bar{X}_{lj}$ in $[0,1]$ and the weights are in $[-1,1]$. Also, $\bar{T}_l$ and $\bar{Y}_l$ will be triangular shaped fuzzy numbers in $[0,1]$.

The error measure they adopted was

$$\bar{E} = \frac{1}{2}\sum_{l=1}^{L} (\bar{T}_l - \bar{Y}_l)^2, \tag{10}$$

which is to be minimized. However, $\bar{E}$ will not be zero, due to fuzzy arithmetic, even when $\bar{Y}_l = \bar{T}_l$, all

l. Therefore, they required a special stopping rule for the iterations.

Let $\bar{T}_l[0] = [t_{l1}, t_{l2}]$. If $\bar{Y}_l = \bar{T}_l$ all l, then $\bar{E}[0] = [-\lambda, \lambda]$, where

$$\lambda = \frac{1}{2}\sum_{l=1}^{L}(t_{l2} - t_{l1})^2. \tag{11}$$

Let $\varepsilon > 0$ denote some acceptable deviation from the value of $\bar{E}$ when $\bar{Y}_l = \bar{T}_l$ all l. Then the stopping rule they adopted was to end the iterations on the values of the weights when $\bar{E}$ is inside the set

$$\Omega = [-\lambda - \varepsilon, \lambda + \varepsilon] \times [0, 1]. \tag{12}$$

They now directly fuzzified the standard delta rule in backpropagation to update the values of the weights.

It is interesting to note that this procedure fails to converge to a correct set of weights. What they found [11] was that there are values of the weights that make $\bar{E}$ inside Ω but do not make $\bar{Y}_l$ close to $\bar{T}_l$, all l. That is, they have the wrong stopping rule and the algorithm converges to the wrong weights. The algorithm has been corrected but no new results have been reported.

3.2. Backpropagation on α-cuts

Also in [11] (see also [31, 34, 35]), the authors discuss a backpropagation algorithm for the individual α-cuts of the weights in a FNN_3. Let $\bar{E}[\alpha] = [e_1(\alpha), e_2(\alpha)]$, $\bar{W}_{ik}[\alpha] = [w_{ik1}(\alpha), w_{ik2}(\alpha)]$, and $\bar{V}_k[\alpha] = [v_{k1}(\alpha), v_{k2}(\alpha)]$, $0 \leqslant \alpha \leqslant 1$. They developed a backpropagation algorithm to update the values of $w_{ik1}(\alpha), w_{ik2}(\alpha), v_{k1}(\alpha)$, and $v_{k2}(\alpha)$, based on the partial derivates of $e_1(\alpha)$ and $e_2(\alpha)$ with respect to $w_{ik1}(\alpha), \ldots, v_{k2}(\alpha)$, for different values of α in $[0, 1]$. This method can also fail because when you put the α-cuts of a weight back together you may not get a fuzzy set. The problem is that the backpropagation algorithm independently updates the α-cuts of the weights. No constraint is built in to insure that the new weights are in fact fuzzy sets. You could obtain $\bar{W}_{11}[0] = [0, 1]$, but $\bar{W}_{11}[0.5] = [-1, 0]$.

3.3. α-cut based backpropagation

In a series of papers [36–41] these authors developed backpropagation based learning algorithms for: (1) real signals and interval weights and biases; (2) FNN_2; and (3) FNN_3. Let us now review their learning method for FNN_3 [38, 41].

They assumed that: (1) the inputs $\bar{X}_{lj}$ are non-negative fuzzy numbers; (2) the weights and bias terms are symmetric triangular fuzzy numbers; and (3) $\bar{T}_l$ is a fuzzy number. Let $\bar{T}_l[\alpha] = [t_{l1}(\alpha), t_{l2}(\alpha)]$, $\bar{Y}_l[\alpha] = [y_{l1}(\alpha), y_{l2}(\alpha)]$, $\bar{W}_{ik}[0] = [w_{ik1}, w_{ik2}]$, and $\bar{V}_k[0] = [v_{k1}, v_{k2}]$. Also let $\alpha_s \in [0, 1]$, $1 \leqslant s \leqslant S$, so that $0 \leqslant \alpha_1 < \alpha_2 < \cdots < \alpha_S \leqslant 1$. The error measure to be minimized is

$$E = \frac{1}{2}\sum_{l=1}^{L}\sum_{s=1}^{S}\alpha_s(E_{ls1} + E_{ls2}), \tag{13}$$

where

$$E_{ls1} = (t_{l1}(\alpha_s) - y_{l1}(\alpha_s))^2, \tag{14}$$

$$E_{ls2} = (t_{l2}(\alpha_s) - y_{l2}(\alpha_s))^2. \tag{15}$$

They developed a standard backpropagation algorithm for the supports of the weights and bias terms. Since these fuzzy sets are symmetric triangular fuzzy numbers you know the whole fuzzy set if you know their support. For example, one can write down a formula for $\partial E/\partial v_{k1}, \ldots, \partial E/\partial w_{ik2}$ similar to the delta rule in standard backpropagation and this is used to compute the new supports of the weights. A small example is given in [38, 41] showing good results for this learning method. However, this procedure gets more and more complicated (the partials of E with respect to the weights get more involved) if the weights are other types of fuzzy numbers and/or the inputs are not non-negative. This method is not applicable when the inputs and/or the weights become more general fuzzy sets.

3.4. Random search

Suppose that the inputs and $\bar{T}_l$ are fuzzy numbers and the weights are triangular fuzzy numbers. Let $\bar{W}_{ik} = (w_{ik1}/w'_{ik}/w_{ik2})$, $\bar{V}_k = (v_{k1}/v'_k/v_{k2})$. We adopt some error measure to be minimized. Assume we wish to minimize E in Eq. (13). In this method

we randomly generate $w_{ik1}, w'_{ik}, w_{ik2}, v_{k1}, v'_k, v_{k2}$, all i,k, to minimize E. For a reasonably large FNN_3 this procedure will be too time consuming so let us now look at a directed random search method called genetic algorithms.

3.5. Genetic algorithms

Genetic algorithms [16, 20, 52] are finding more and more applications in fuzzy systems. Recently they have been applied to fuzzy optimization [6, 12]. In [10] the authors use genetic algorithms, to train a FNN_3. We will now review this recent research on learning. The type of genetic algorithm used will depend on the kinds of fuzzy sets used for input and weights, and the error measure to be minimized.

3.5.1

Suppose $\bar{X}_{lj}, \bar{W}_{ik}$, and $\bar{V}_k$ are all triangular fuzzy numbers with $\bar{T}_l$ a triangular shaped fuzzy number. Define $\bar{T}_l[0] = [t_{l1}, t_{l2}]$ and $\bar{Y}_l[0] = [y_{l1}, y_{l2}]$. The authors wish to minimize E where $E = \max\{E_1, E_2\}$ and

$$E_1 = \frac{1}{2}\sum_{l=1}^{L}(t_{l1} - y_{l1})^2, \tag{16}$$

$$E_2 = \frac{1}{2}\sum_{l=1}^{L}(t_{l2} - y_{l2})^2. \tag{17}$$

A (regular) genetic algorithm is designed to manage the supports of the weights in order to minimize E. Since E is based on the supports of $\bar{Y}_l$ and $\bar{T}_l$ the algorithm only needs to keep track of the supports of the weights. An example in [10] shows that this type of learning algorithm can be used to train a FNN_3.

However, this learning method may fail because it only ensures that the support of $\bar{Y}_l$ is close to the support of $\bar{T}_l$, all l. It may happen that the algorithm terminates with E very small but $\bar{Y}_l(x)$ differs significantly from $\bar{T}_l(x)$ for some x in the intersection of their supports. Then we must use an error measure that takes into account the whole shape of $\bar{Y}_l(x)$ and the whole shape of $\bar{T}_l(x)$, all l. These new error measures are discussed in the rest of this subsection.

3.5.2

The inputs and weights are assumed to be triangular fuzzy numbers with $\bar{T}_l$ a triangular shaped fuzzy number. The error measure to be minimized will be based on α-cuts of $\bar{Y}_l$ and $\bar{T}_l$ similar to Eq. (13). Let

$$E = \frac{1}{2}\sum_{l=1}^{L}\sum_{s=1}^{S}(E_{ls1} + E_{ls2}), \tag{18}$$

with E_{ls1} (E_{ls2}) specified by Eq. (14) (Eq. (15)). The (regular) genetic algorithm will be similar to the one in 3.5.1 except that it searches for the supports of the weights, and where the membership function will be one, to minimize E in Eq. (18). Since the input and weights are triangular fuzzy numbers all that one needs to know is the support of the weights, and where membership equals one, to compute E employing interval arithmetic.

3.5.3

Now suppose that triangular fuzzy weights are not sufficient to make $\bar{Y}_l$ approximately equal to $\bar{T}_l$ for all l. So let the weights be triangular shaped fuzzy numbers with inputs triangular fuzzy numbers and $\bar{T}_l$ a triangular shaped fuzzy number. The error measure is the same as the E defined in Eq. (18). The genetic algorithm is the same as in 3.5.2 except for one major change: it keeps track of certain α-cuts of the weights. That is, they store for each weight its α_s, $1 \leqslant s \leqslant S$, cuts. Knowing the α_s-cuts of the weights and inputs one can compute the α_s-cuts of the output using interval arithmetic and hence obtain E.

3.5.4

Now the inputs, weights, and $\bar{T}_l$ are arbitrary discrete fuzzy subsets of $[-M, M]$ for some $M > 0$. Suppose the fuzzy sets $\bar{Y}_l$ and $\bar{T}_l$ can have their membership functions positive only at x_p in $[-M, M]$ where $-M = x_0 < x_1 < \cdots < x_p = M$. Then the error measure will be based on membership values so let

$$E = \frac{1}{2}\sum_{l=1}^{L}\sum_{p=0}^{P}(\bar{Y}_l(x_p) - \bar{T}_l(x_p))^2, \tag{19}$$

which is to be minimized. That is, the authors in [10] designed a fuzzy genetic algorithm to find the

weights to drive E to zero. A population member H in a regular genetic algorithm looks like

$$H = (h_1, h_2, \ldots, h_m), \tag{20}$$

where each h_i is zero or one. Binary notation (zeros and ones) is used to specify real numbers in a regular genetic algorithm. In a fuzzy genetic algorithm H is defined as in Eq. (20) but each h_i is a real number in [0, 1]. In a fuzzy genetic algorithm the h_i give membership values in fuzzy sets. Since the elements in H are membership values of fuzzy sets it was called a fuzzy genetic algorithm.

3.6. Fuzzy chaos

Let $\mathcal{F}$ denote all the fuzzy numbers in some interval $[-M, M]$, $M > 0$. Now suppose that F is a fuzzy chaotic mapping from $\mathcal{F}$ into $\mathcal{F}$ [18, 19] and let $\bar{N}_{i+1} = F(\bar{N}_i)$, $i = 0, 1, 2, \ldots$ with $\bar{N}_0$ initially chosen in $\mathcal{F}$. Since F is chaotic the sequence $\bar{N}_i$ appears to be a random sequence of fuzzy numbers in $[-M, M]$. This method could be used, as an alternative to pure random search, to train a FNN$_3$. Fuzzy chaos has been applied to solve a fuzzy optimization problem [5, 14].

Assume that we have selected some error measure E, possible one of those defined in Sections 3.3 or 3.5, to be minimized. We will employ a fuzzy chaotic mapping F as the basis of a search for the fuzzy number weights to make E close to zero. Initially pick values for the weights $\bar{W}_{ik,0}$ and $\bar{V}_{k,0}$ in $\mathcal{F}$. Now generate a sequence of weights $\bar{W}_{ik,u+1} = F(\bar{W}_{ik,u})$, $\bar{V}_{k,u+1} = F(\bar{V}_{k,u})$, $u = 0, 1, 2, \ldots$ looking for those values in $\mathcal{F}$ that produce a value of E close to zero. This would be an interesting topic for future research.

3.7. Other learning methods

In [48, 49] the authors have: (1) real number signals; (2) monotone increasing membership functions for the fuzzy weights; and (3) a special fuzzy error measure. They employ a special learning algorithm, inspired by the standard backpropagation learning algorithm, so that the FNN$_1$ can learn the fuzzy weights.

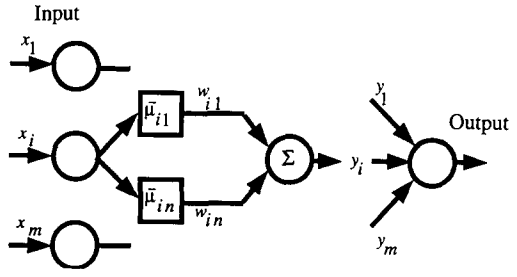

Fig. 2. Yamakawa's fuzzy neuron.

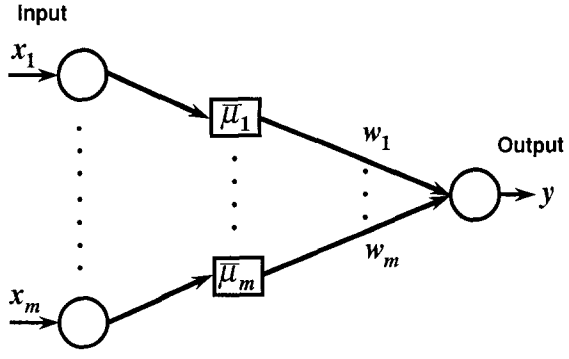

Fig. 3. Fuzzy neuron.

Yamakawa's new fuzzy neuron [59] has a learning algorithm for the weights. The input to a fuzzy neuron is shown in Fig. 2. The collection $\bar{\mu}_{ij}$ and w_{ij}, $1 \leqslant j \leqslant n$, exist for each input x_i, $1 \leqslant i \leqslant m$. The $\bar{\mu}_{ij}$ are fixed fuzzy sets (triangular fuzzy numbers) and the w_{ij} are real number weights. Given x_i, only two neighboring $\bar{\mu}_{ij}$ can be not zero, say $\bar{\mu}_{ik}(x_i)$ and $\bar{\mu}_{i,k+1}(x_i)$ are not zero. In Fig. 2,

$$y_i = \bar{\mu}_{ik}(x_i)w_{ik} + \bar{\mu}_{i,k+1}(x_i)w_{i,k+1} \tag{21}$$

and the input to the neuron is y_i, $1 \leqslant i \leqslant m$, with output $y_1 + \cdots + y_m$. The learning algorithm for the w_{ij} is based on a heuristic rule which produces an update formula for the weights similar to that obtained in backpropagation in standard NNs.

The FNN in [47, 55] is similar to Yamakawa's fuzzy neuron. This fuzzy neural net is based on the fuzzy neuron shown in Fig. 3. The output y is

computed as

$$y = \frac{\sum_{i=1}^{m} w_i \bar{\mu}_i(x_i)}{\sum_{i=1}^{m} w_i} \tag{22}$$

The $\bar{\mu}_i$ are assumed to be trapezoidal fuzzy numbers. A FNN is made up of a network of fuzzy neurons shown in Fig. 3. The authors claim to have a learning algorithm both for the weights (w_i) and the trapezoidal fuzzy numbers $\bar{\mu}_i$.

4. Applications

In this section we will survey some possible applications of fuzzy neural nets that have been discussed in the literature.

4.1. Fuzzy regression

In systems identification we have an unknown process Q, a set of inputs $\bar{X}_l = (\bar{X}_{l1}, \bar{X}_{l2})$ and outputs $\bar{T}_l$, $1 \leqslant l \leqslant L$, and we would like to identify a Q so that

$$Q(\bar{X}_{l1}, \bar{X}_{l2}) = \bar{T}_l, \quad 1 \leqslant l \leqslant L. \tag{23}$$

In fuzzy regression we try for Q a linear, quadratic, exponential, ... function. In this section let us use a quadratic. So, for Q we will attempt to substitute.

$$\bar{Y} = \bar{A}\bar{X}_1^2 + \bar{B}\bar{X}_1\bar{X}_2 + \bar{C}\bar{X}_2^2 \tag{24}$$

for triangular fuzzy numbers $\bar{A}, \bar{B}, \bar{C}$. Fuzzy regression is becoming an active area of research [1, 17, 42, 43, 54].

We may now use a FNN_3 two ways: (1) computationally equivalent to the quadratic (Eq. (24)); or (2) learn the weights so that the quadratic approximates the unknown process Q.

First let us build a FNN_3 equivalent to the quadratic. In fact, this can be done for any fuzzy polynomial. The network is shown in Fig. 4. Nodes 1 and 3 square their input, nodes 2 and 4 have output equal to input, nodes 5 and 7 add their inputs to produce output, and node 6 multiplies the inputs. All arcs have fixed weight equal to one except those showing $\bar{A}, \bar{B}, \bar{C}$. In the second and third layer we multiply the signal times the weight.

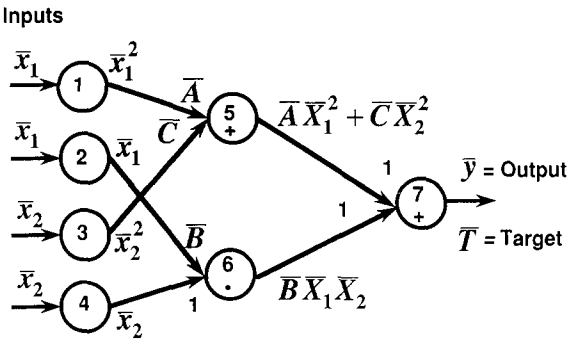

Fig. 4. Fitting a quadratic by a fuzzy neural net.

Therefore, the final output $\bar{Y}$ is given by Eq. (24). So, if we know $\bar{A}, \bar{B}$ and $\bar{C}$ the FNN_3 in Fig. 4 can compute the value of the quadratic.

Now suppose that we do not know $\bar{A}, \bar{B}, \bar{C}$ and we would like to find their values so that the output from the quadratic $\bar{Y}_l$, when the inputs are $\bar{X}_{l1} = \bar{X}_1$, $\bar{X}_{l2} = \bar{X}_2$, is approximately $\bar{T}_l$, all l. We want the FNN_3 in Fig. 4 to learn its weights $(\bar{A}, \bar{B}, \bar{C})$ for the training data $(\bar{X}_l, \bar{T}_l)$, $1 \leqslant l \leqslant L$. A specialized backpropagation learning algorithm for this FNN_3 is presented in [4, 13, 30, 33]. No numerical results have been reported using this algorithm.

We cannot expect a FNN_3 to be able to learn the values of its weights for any training data. It was shown in [9, 27] that a regular FNN_3 is not a universal approximator. Let F be a continuous mapping from $\mathscr{F} \times \mathscr{F}$ into $\mathscr{F}$ and suppose F produced the training data. That is, $\bar{T}_l = F(\bar{X}_{l1}, \bar{X}_{l2})$, $1 \leqslant l \leqslant L$. In general we cannot expect to train a regular FNN_3 to learn this data unless F is monotone [9]. F is monotone if and only if given $\bar{X}'_1 \leqslant \bar{X}_1$ and $\bar{X}'_2 \leqslant \bar{X}_2$, all in $\mathscr{F}$, then $F(\bar{X}'_1, \bar{X}'_2) \leqslant F(\bar{X}_1, \bar{X}_2)$. The fact that hybrid fuzzy neural nets can be universal approximators is discussed at the end of this section.

4.2. Fuzzy controller

We will show how to model an elementary fuzzy controller using a $HFNN_1$ (see also [4, 13, 30, 33]). The fuzzy controller is specified by the rule table given in Table 1 and the definition of the fuzzy

Table 1
Fuzzy control rules in the industrial process (rule number given in the right-hand corner)

e	Δe				
	$\bar{G}_1$	$\bar{G}_2$	$\bar{G}_3$	$\bar{G}_4$	$\bar{G}_5$
$\bar{F}_1$		$\bar{A}_1$ 1			
$\bar{F}_2$		$\bar{A}_1$ 2		$\bar{A}_2$ 3	
$\bar{F}_3$	$\bar{A}_2$ 4		$\bar{A}_3$ 5		$\bar{A}_4$ 6
$\bar{F}_4$		$\bar{A}_4$ 7		$\bar{A}_5$ 8	
$\bar{F}_5$				$\bar{A}_5$ 9	

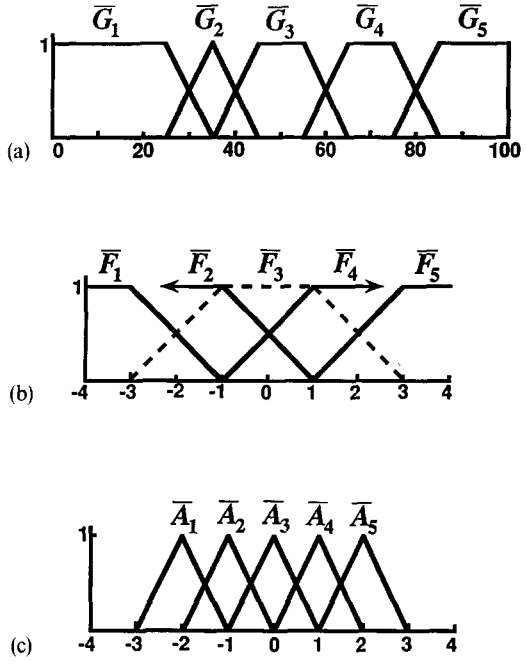

Fig. 5. Fuzzy numbers in the fuzzy controller rules: (a) for error; (b) for change in error; (c) for output.

numbers is presented in Fig. 5. This fuzzy controller accepts singleton input e = error and Δe = change in error. Given values for e and Δe, the nine rules are evaluated as follows: $\Delta_1 = \min(\bar{F}_1(e), G_2(\Delta e))$, $\ldots, \Delta_9 = \min(\bar{F}_5(e), \bar{G}_4(\Delta e))$. Then, since we have nine rules but only five control actions, we maximize the Δ_i corresponding to the same action $\bar{A}_k$, as follows: $\varepsilon_1 = \max(\Delta_1, \Delta_2)$, $\varepsilon_2 = \max(\Delta_3, \Delta_4)$, $\varepsilon_3 = \Delta_5$, $\varepsilon_4 = \max(\Delta_6, \Delta_7)$, $\varepsilon_5 = \max(\Delta_8, \Delta_9)$. Then each ε_k is assigned to its $\bar{A}_k$, $1 \leqslant k \leqslant 5$. To defuzzify the result we first compute $\bar{A} = \bigcup(\varepsilon_k \bar{A}_k)$, where union is taken as maximum, find δ which is equal to the center of gravity of $\bar{A}$, and then δ is the defuzzified output from the controller.

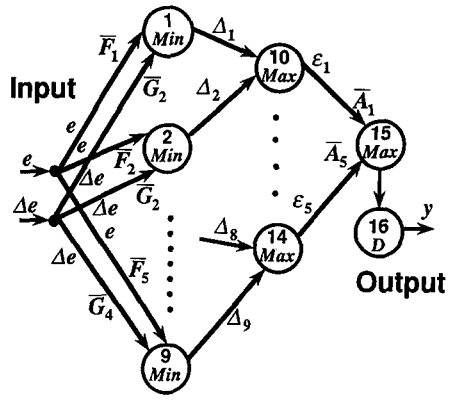

Fig. 6. Fuzzy controller as a fuzzy neural net.

This fuzzy controller, modeled as a (hybrid) FNN_1, is shown in Fig. 6. In this net the input to node 1 is $\bar{F}_1(e)$ and $\bar{G}_2(\Delta e)$ as in rule number one, and the output is $\Delta_1 = \min\{\bar{F}_1(e), \bar{G}_2(\Delta e)\}$. Nodes 1–9 take the *min* of its inputs so they act like evaluating the nine control rules. Also, the interaction of signal e and fuzzy weight $\bar{F}_1$, for neuron 1, is to evaluate $\bar{F}_1$ at e (same for neurons 1–9). Neurons 10–14 take the *max* of their inputs, the weights for these neurons are all equal to one. Neuron 15 takes the union (*max*) of its fuzzy input which is $\varepsilon_i \bar{A}_i$ (multiplication), $1 \leqslant i \leqslant 5$. The last neuron 16 has the weight one and acts as the defuzzifier.

How is this fuzzy neural net to be used? It can be used in place of the fuzzy controller, or it is to learn the fuzzy control rules (the weights $\bar{F}_i, \bar{G}_i$ and $\bar{A}_i$) given some training data. Because of the use of max and min this net will require a special learning algorithm [4, 13, 30, 33].

4.3. Fuzzy expert system

A FNN_3 is ideal for modeling a fuzzy expert system [4, 30, 31, 33–35]. Suppose we are given

a fuzzy expert system with one block of rules

$$\mathcal{R}_i\text{: If } X = \bar{A}_i \text{ and } Y = \bar{B}_i, \text{ then } Z = \bar{C}_i, \quad 1 \leqslant i \leqslant n. \tag{25}$$

Given some data $X = \bar{A}'$ and $Y = \bar{B}'$, the system is to come up with its final conclusion $Z = \bar{C}'$. In this paper it does not matter the exact details (generalized modus ponens, etc.) on how it gets $\bar{C}'$, we will only assume that the rules are evaluated separately and their results are combined to obtain $\bar{C}'$.

A fuzzy neural network of the fuzzy expert system is shown in Fig. 7. The two input nodes have their output equal to their input. We input the information on X and Y and the nodes 1–n represent the n rules. Consider node 1 which is to model $\mathcal{R}_1$. Given $X = \bar{A}'$ and $Y = \bar{B}'$ when $\mathcal{R}_1$ is evaluated suppose the conclusion is $Z = \bar{C}'_1$. Then, once the net is trained the output from node 1 should be approximately $\bar{C}'_1$. All the rules are evaluated producing separate conclusions $Z = \bar{C}'_i, 1 \leqslant i \leqslant n$, when the system is presented with $X = \bar{A}'$, $Y = \bar{B}'$. The fuzzy expert system now combines all the $\bar{C}'_i$, $1 \leqslant i \leqslant n$, into one final conclusion $Z = \bar{C}'$. The weights $\bar{V}_i$ and the output node are to model forming $\bar{C}'$ from $\bar{C}'_i$, $1 \leqslant i \leqslant n$.

Now suppose we have some training data $X = \bar{A}'_j$ and $Y = \bar{B}'_j$ for inputs and $Z = \bar{E}'_j$ for final conclusion, $1 \leqslant j \leqslant K$. That is, we run the fuzzy expert system for $X = \bar{A}'_j$, $Y = \bar{B}'_j$ and it produces $Z = \bar{E}'_j$, all j. This then becomes the training set for FNN_3. One of the learning algorithms discussed in the third section might be used to train this fuzzy neural net. Once the FNN_3 in Fig. 7 has been trained its generalization property will allow it to approximately operate as a fuzzy expert system.

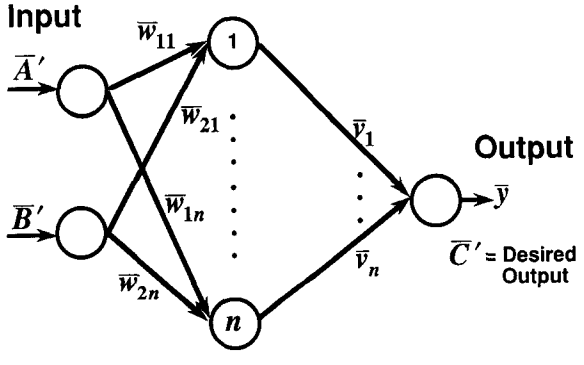

Fig. 7. Fuzzy expert system as a fuzzy neural network.

4.4. Fuzzy hierarchical analysis

This application shows a fuzzy neural network for fuzzy hierarchical analysis [2]. Here we are concerned with ranking a set of alternatives across a collection of criteria. To simplify notation, etc. let us assume we have three alternatives a_1, a_2, a_3 and two criteria c_1, c_2. One first compares a_1 to a_2, a_1 to a_3, a_2 to a_3 for c_1 and then for c_2. You also need to compare c_1 and c_2. We are to rank the alternatives not the criteria. However, in hierarchical analysis we must pairwise compare all the criteria, as discussed below, in order to obtain the final ranking of the alternatives.

Let us consider the pairwise comparison of a_1, a_2, a_3 for criterion c_1. A person (expert, judge) is asked to supply ratios for each pairwise comparison a_1 to a_2, a_1 to a_3, and a_2 to a_3. If the person considers a_1 more important than a_3, then the ratio might be $\frac{3}{1}$, or $\frac{5}{1}$, or $\frac{7}{1}$. The numbers in the ratio are usually taken from the set $\{1, 2, \ldots, 9\}$. The ratio indicates the strength with which a_1 dominates a_3. If the ratio for a_1 to a_3 is $\frac{5}{1}$, then the ratio for a_3 to a_1 is $\frac{1}{5}$. Of course, the ratio for a_i to a_i is one for all i.

In fuzzy hierarchical analysis you are allowed to use fuzzy ratios, or crisp ratios, in all comparisons. A fuzzy ratio is the fuzzification of i/j where $i, j \in \{1, 2, \ldots, 9\}$. The exact value of the fuzzy ratio does not matter in this paper so we assume that these are all non-negative fuzzy sets.

For c_1 assume that when we compare a_1 to a_2 we get $\bar{A}_1$, a_1 to a_3 is $\bar{A}_2$, and a_2 versus a_3 is $\bar{A}_3$. If you consider a_1 more important than a_2, for c_1, then we would expect that the support of $\bar{A}_1$ lies in $(1, M)$ for some $M > 1$. Similarly, if we believe that a_2 is more important, then the support of $\bar{A}_1$ lies in $(0, 1)$. Employing criterion c_2 let $\bar{B}_1$ be the value of a_1 versus a_2, $\bar{B}_2$ for a_1 to a_3, and $\bar{B}_3$ the outcome of comparing a_2 to a_3. For the criteria we compare c_1 to c_2 and obtain $\bar{C}$. We need only one fuzzy set here because there are only two criteria.

In fuzzy hierarchical analysis these fuzzy sets $\bar{A}_1, \bar{A}_2, \bar{A}_3, \bar{B}_1, \bar{B}_2, \bar{B}_3$, and $\bar{C}$ are combined to

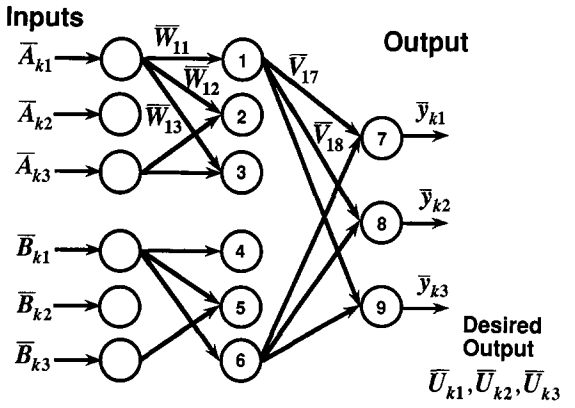

Fig. 8. Fuzzy hierarchical analysis as a fuzzy neural network.

produce the final weights $\bar{U}_i$ for the a_i, $1 \leqslant i \leqslant 3$. Then, employing some method of ranking the fuzzy sets $\bar{U}_i$ we rank the a_i from most important (best) to least important (worst).

A regular fuzzy neural net for fuzzy hierarchical analysis is shown in Fig. 8. This net is a regular FNN_3 except for the fact that all input neurons are not connected to all the neurons in the hidden layer [4, 31, 34, 35]. We now enlist the help of N experts (judges) to rank the alternatives. The judges produce the training data for the net with fuzzy ratios $\bar{A}_{k1}, \bar{A}_{k2}, \bar{A}_{k3}, \bar{B}_{k1}, \bar{B}_{k2}, \bar{B}_{k3}$, and $\bar{C}_k$ from expert k and weights, from fuzzy hierarchical analysis, $\bar{U}_{k1}, \bar{U}_{k2}, \bar{U}_{k3}$, $1 \leqslant k \leqslant N$. The training set is $\{\bar{A}_{k1}, \ldots, \bar{B}_{k3}\}$ for input and target $\{\bar{U}_{k1}, \bar{U}_{k2}, \bar{U}_{k3}\}$, $1 \leqslant k \leqslant N$. The input nodes have their output the same as their input. Nodes 1–3 are for criterion c_1, nodes 4–6 are for c_2, and nodes 7 to 9 combine the results across all the criteria to produce outputs $\bar{Y}_{k1}, \bar{Y}_{k2}, \bar{Y}_{k3}$.

The fuzzy neural net is trained from the combined results of N experts. Once trained, its generalization property can be used to perform fuzzy hierarchical analysis for other "judges" who give data on only the comparison of the alternatives for each criterion.

4.5. Fuzzy matrix equations

We wish to solve

$$\bar{A}\bar{x} = \bar{b} \tag{26}$$

for $\bar{x}$ given $m \times n$ fuzzy matrix $\bar{A} = [\bar{a}_{ij}]$ for triangular fuzzy numbers $\bar{a}_{ij}$ and given $\bar{b}^t = (\bar{b}_1, \ldots, \bar{b}_m)$ a $m \times 1$ vector made up of triangular shaped fuzzy numbers, and $\bar{x}^t = (\bar{x}_1, \ldots, \bar{x}_n)$ is the unknown $n \times 1$ vector of fuzzy numbers. See [15] for a discussion of solving fuzzy matrix equations and [3] for a general discussion on solving fuzzy equations.

A FNN_3 solution to Eq. (26) is given in Fig. 9 [4, 13]. The input to the net is the ith row of $\bar{A}$. The input neurons make no change in their inputs, so the input to the output neuron is

$$\bar{a}_{i1}\bar{x}_1 + \cdots + \bar{a}_{in}\bar{x}_n, \tag{27}$$

which is the ith component in the product $\bar{A}\bar{x}$. The output, in the output neuron, equals its input, so

$$y_i = \bar{a}_{i1}\bar{x}_1 + \cdots + \bar{a}_{in}\bar{x}_n \quad \text{for } 1 \leqslant i \leqslant m. \tag{28}$$

How is the FNN_3 going to solve the fuzzy matrix equation? The training data is $(\bar{a}_{i1}, \ldots, \bar{a}_{in})$ for input and target output is $\bar{b}_i$, $1 \leqslant i \leqslant m$. A learning algorithm, possibly one of those discussed in Section 3, is then employed to find the best weights which will produce the unknown fuzzy vector $\bar{x}$.

The fuzzy matrix equation may have no solution for fuzzy numbers $\bar{x}_i$, $1 \leqslant i \leqslant m$. In this case there is no hope in making the error measure close to zero. One could then try more general fuzzy sets for the $\bar{x}_i$ and possibly try a fuzzy genetic algorithm to train the FNN_3 (see Section 3.5.4).

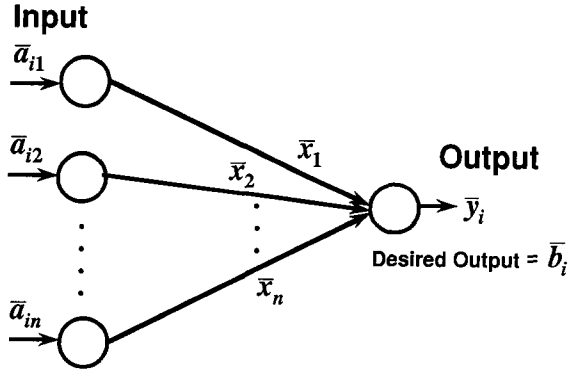

Fig. 9. Fuzzy neural net to solve fuzzy matrix equation.

Alternatively, the FNN_3 in Fig. 9 could be used to evaluate fuzzy matrix equations. If we know $\bar{A}$ and $\bar{x}$, then the FNN_3 computes $\bar{b}$.

4.6. Universal approximators

Numerous papers have appeared showing that a regular neural net (Fig. 1) is a universal approximator. What this means is that given continuous $g: \mathbb{R}^2 \to \mathbb{R}$, a compact subset J of $\mathbb{R}^2$ and $\varepsilon > 0$, there is a NN (weights w_{ik}, v_k, and K = number of hidden neurons) so that

$$|\mathrm{NN}(x_1, x_2) - g(x_1, x_2)| < \varepsilon \quad \text{for all } (x_1, x_2) \text{ in } J. \tag{29}$$

We have used the notation $\mathrm{NN}(x_1, x_2)$ to denote the output y from the neural net given inputs x_1, x_2. Can this result be generalized to fuzzy neural nets?

Let continuous $G: \mathscr{F} \times \mathscr{F} \to \mathscr{F}$, $\bar{J}$ be a compact subset of $\mathscr{F} \times \mathscr{F}$ and $\varepsilon > 0$. Can we build a regular FNN_3 so that

$$\|\mathrm{FNN}_3(\bar{X}_1, \bar{X}_2) - G(\bar{X}_1, \bar{X}_2)\| < \varepsilon \quad \text{for all } (\bar{X}_1, \bar{X}_2) \text{ in } \bar{J}? \tag{30}$$

See [9] for the details on the topology for $\mathscr{F}$ and the metric $\|\cdot\|$ in Eq. (30). In general, the answer to the above question is no because FNN_3 is a monotonic mapping from $\mathscr{F} \times \mathscr{F}$ into $\mathscr{F}$. If G is also monotonic then we might hope to build a FNN_3 that can uniformly approximate G on compact subsets of $\mathscr{F} \times \mathscr{F}$.

However, it was also shown in [9] that certain $HFNN_3$'s are universal approximators. But these $HFNN_3$'s were not continuous. Future research will be concerned with finding continuous $HFNN_3$'s which are universal approximators.

It is important to note that the fuzzy neural nets discussed in this subsection are to be built for fast parallel computation and learning algorithms are not relevant.

4.7. Other applications

A regular FNN_3 has been applied to a fuzzy classification problem [4,13,36]. In [37,39] the authors employ a regular FNN_2 to learn/interpolate fuzzy if–then rules. The FNN_1 in [48,49] was used to learn the if–then rules in a fuzzy controller. Finally, Yamakawa's fuzzy neuron was applied to pattern recognition [47,50,56–58] and system identification [59].

5. Summary and conclusions

In this paper we reviewed learning algorithms and applications of fuzzy neural networks. Our definition of a fuzzy neural network is a layered, feedforward, network that processes fuzzy set signals and/or has fuzzy set weights.

The main topic for future research is to develop learning algorithms for fuzzy neural nets that have more general fuzzy sets. Initially researchers assumed the signals/weights were (symmetric) triangular fuzzy numbers. This is too restrictive for applications. We need now to assume the signals/weights are fuzzy numbers (any type) or general fuzzy sets. It appears that (fuzzy) genetic algorithms might be the tool needed to handle the more general fuzzy sets.

Once we have a general learning algorithm for fuzzy neural nets the applications will follow naturally.

References

[1] A. Bardossy, I. Bogardi and L. Duckstein, Fuzzy nonlinear regression analysis of dose-response relationships, *European J. Oper. Res.* **66** (1993) 46–51.

[2] J.J. Buckley, Fuzzy hierarchical analysis, *Fuzzy Sets and Systems* **17** (1985) 233–247.

[3] J.J. Buckley, Solving fuzzy equations, *Fuzzy Sets and Systems* **50** (1992) 1–14.

[4] J.J. Buckley and Y. Hayashi, Fuzzy neural nets and applications, *Fuzzy Systems and AI*, **3** (1992) 11–41.

[5] J.J. Buckley and Y. Hayashi, Fuzzy simulation based on fuzzy chaos, *Proc. 2nd IEEE Internat. Conf. on Fuzzy Systems*, San Francisco (1993) Vol. II, 1039–1043.

[6] J.J. Buckley and Y. Hayashi, Fuzzy genetic algorithms for optimization, *Proc. Internat. Joint Conf. on Neural Networks*, Nagoya, Japan (1993) Vol. I, 725–728.

[7] J.J. Buckley and Y. Hayashi, Numerical relationships between neural networks, continuous functions, and fuzzy systems, *Fuzzy Sets and Systems* **60** (1993) 1–8.

[8] J.J. Buckley and Y. Hayashi, Hybrid neural nets can be fuzzy controllers and fuzzy expert systems, *Fuzzy Sets and Systems* **60** (1993) 135–142.

[9] J.J. Buckley and Y. Hayashi, Can fuzzy neural nets approximate continuous fuzzy functions? *Fuzzy Sets and Systems* **61** (1993) 43–52.

[10] J.J. Buckley and Y. Hayashi, Genetic algorithms for fuzzy neural nets, unpublished manuscript.

[11] J.J. Buckley and Y. Hayashi, Fuzzy backpropagation for fuzzy neural nets, unpublished manuscript.

[12] J.J. Buckley and Y. Hayashi, Fuzzy genetic algorithm and applications, to appear in *Fuzzy Sets and Systems*.

[13] J.J. Buckley and Y. Hayashi, Fuzzy neural networks, in: R.R. Yager and L.A. Zadeh, Eds., *Fuzzy Sets, Neural Networks and Soft Computing* (to appear).

[14] J.J. Buckley and Y. Hayashi, Applications of fuzzy chaos to fuzzy simulation, to appear in *Fuzzy Sets and Systems*.

[15] J.J. Buckley and Y. Qu, Solving systems of fuzzy linear equations, *Fuzzy Sets and Systems* **43** (1991) 33–43.

[16] L. Davis, *Handbook of Genetic Algorithms* (Van Nostrand Reinhold, New York, 1991).

[17] P. Diamond, Fuzzy least squares, *Inform. Sci.* **46** (1988) 141–157.

[18] P. Diamond, Chaos and fuzzy representations of dynamical systems, *Proc. 2nd Internat. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan (1992) 51–58.

[19] P. Diamond, Chaos and information loss in fuzzy dynamical systems, unpublished manuscript.

[20] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, MA 1989).

[21] M.M. Gupta, Fuzzy logic and neural networks, *Proc. 2nd Internat. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan (1992) 157–160.

[22] M.M. Gupta and G.K. Knopf, Fuzzy neural network approach to control systems, *Proc. 1st Internat. Symp. on Uncertainty Modeling and Analysis*, Maryland, MD (1990) 483–488.

[23] M.M. Gupta and J. Qi, Fusion of fuzzy logic and neural networks with applications to decision and control problem, *Proc. NAFIPS-'91*, Columbia, MO (1991) 327–328.

[24] M.M. Gupta and J. Qi, On fuzzy neuron models, *Proc. Internat. Joint Conf. on Neural Networks*, Seattle (1991) Vol. II, 431–436.

[25] M.M. Gupta and J. Qi, On fuzzy neuron models, in: L. Zadeh and J. Kacprzyk, Eds., *Fuzzy Logic for the Management of Uncertainty* (Wiley, New York, 1992) 479–491.

[26] Y. Hayashi and J.J. Buckley, Fuzzy controllers and fuzzy expert systems as hybrid neural nets, *Proc. 5th IFSA*, Seoul, Korea (1993) Vol. I, 70–72.

[27] Y. Hayashi and J.J. Buckley, Are regular fuzzy neural nets universal approximators? *Proc. Internat. J. Conf. Neural Networks*, Nagoya, Japan (1993) Vol. I, 721–724.

[28] Y. Hayashi and J.J. Buckley, Direct fuzzification of neural networks, *Proc. 1st Asian Fuzzy Systems Symp.*, Singapore (1993) 560–567.

[29] Y. Hayashi and J.J. Buckley, Fuzzy max-min neural controller, *Fuzzy Sets and Systems*, under revision.

[30] Y. Hayashi, J.J. Buckley and E. Czogala, Systems engineering applications of fuzzy neural networks, *J. Systems Engrg.* **2** (1992) 232–236.

[31] Y. Hayashi, J.J. Buckley and E. Czogala, Fuzzy neural network with fuzzy signals and weights, *Proc. Internat. J. Conf. Neural Networks*, Baltimore (1992) Vol. II, 696–701.

[32] Y. Hayashi, J.J. Buckley and E. Czogala, Fuzzy neural controller, *Proc. IEEE Internat. Conf. Fuzzy Systems*, San Diego (1992) 197–202.

[33] Y. Hayashi, J.J. Buckley and E. Czogala, Systems engineering applications of fuzzy neural networks, *Proc. Internat. Joint Conf. on Neural Networks*, Baltimore (1992) Vol. II, 412–418.

[34] Y. Hayashi, J.J. Buckley and E. Czogala, Direct fuzzification of neural network and fuzzified delta rule, *Proc. 2nd Internat. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan (1992) 73–76.

[35] Y. Hayashi, J.J. Buckley and E. Czogala, Fuzzy neural network with fuzzy signals and weights, *Internat. J. Intelligent Systems* **8** (1993) 527–537.

[36] H. Ishibuchi, R. Fujioka and H. Tanaka, An architecture of neural networks for input vectors of fuzzy numbers, *Proc. IEEE Internat. Conf. on Fuzzy Systems*, San Diego (1992) 1293–1300.

[37] H. Ishibuchi, R. Fujioka and H. Tanaka, Neural networks that learn from fuzzy if–then rules, *IEEE Trans. Fuzzy Systems* **1** (1993) 85–97.

[38] H. Ishibuchi, K. Kwon and H. Tanaka, Learning of fuzzy neural networks from fuzzy inputs and fuzzy targets, *Proc. 5th IFSA World Congr.*, Seoul, Korea (1993) Vol. I, 147–150.

[39] H. Ishibuchi, H. Okada and H. Tanaka, Interpolation of fuzzy if–then rules by neural networks, *Proc. 2nd Internat. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan (1992) 337–340.

[40] H. Ishibuchi, H. Okada and H. Tanaka, Learning of neural networks from fuzzy inputs and fuzzy targets, *Proc. Int. J. Conf. on Neural Networks*, Beijing, China (1992) Vol. III, 447–452.

[41] H. Ishibuchi, H. Okada and H. Tanaka, Fuzzy neural networks with fuzzy weights and fuzzy biases, *Proc. IEEE Internat. Conf. Neural Networks*, San Francisco (1993) Vol. III, 1650–1655.

[42] H. Ishibuchi, H. Okada and H. Tanaka, An architecture of neural networks with interval weights and its application to fuzzy regression analysis, to appear in *Fuzzy Sets and Systems*.

[43] H. Ishibuchi and H. Tanaka, Fuzzy regression analysis using neural networks, *Fuzzy Sets and Systems* **50** (1992) 257–265.

[44] J.M. Keller and D. Hunt, Incorporating fuzzy membership functions into the perceptron algorithm, *IEEE Trans. Pattern Anal. Mach. Intell.* **7** (1985) 693–699.

[45] S.C. Lee and E.T. Lee, Fuzzy sets and neural networks, *J. Cybernetics* **4** (1974) 83–103.

[46] S.C. Lee and E.T. Lee, Fuzzy neural networks, *Math. Biosci.* **23** (1975) 151–177.

[47] K. Nakamura, T. Fujimaki, R. Horikawa and Y. Ageishi, Fuzzy network production system, *Proc. 2nd Internat. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan (1992) 127–130.

[48] D. Nauck and R. Kruse, A neural fuzzy controller learning by fuzzy error propagation, *Proc. NAFIPS 1992*, Puerto Vallarta, Mexico (1992) Vol. II, 388–397.

[49] D. Nauck and R. Kruse, A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation, *Proc. IEEE Internat. Conf. on Neural Networks*, San Francisco (1993) Vol. II, 1022–1027.

[50] M. O'Hagan, A fuzzy neuron based upon maximum entropy ordered weighted averaging, in: B. Bouchon-Meunier, R.R. Yager and L.A., Zadeh, Eds., *Uncertainty in Knowledge Bases*, Lecture Notes in Computer Science, Vol. 521 (Springer, Berlin, 1991) 598–609.

[51] I. Requena and M. Delgado, R-FN: A model of fuzzy neuron, *Proc. 2nd Internat. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan (1992) 793–796.

[52] R. Serra and G. Zanarini, *Complex Systems and Cognitive Processes* (Springer, Berlin, 1990).

[53] H. Takagi, Fusion technology of fuzzy theory and neural networks – survey and future directions, *Proc. Internat. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan (1990) 13–26.

[54] H. Tanaka, S. Uejima and K. Asai, Linear regression analysis with fuzzy model, *IEEE Trans. Systems Man Cybernet.* **12** (1982) 903–907.

[55] M. Tokunaga, K. Kohno, Y. Hashizume, K. Hamatani, M. Watanabe, K. Nakamura and Y. Ageishi, Learning Mechanism and an application of FFS-Network reasoning system, *Proc. 2nd Internat. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan (1992) 123–126.

[56] T. Yamakawa, Pattern recognition hardware system employing a fuzzy neuron, *Proc. Internat. Conf. on Fuzzy Logic*, Iizuka, Japan (1990) 943–948.

[57] T. Yamakawa and M. Furukawa, A design of membership functions for a fuzzy neuron using example based learning, *Proc. IEEE Internat. Conf. Fuzzy Systems*, San Diego, CA (1992) 75–82.

[58] T. Yamakawa and S. Tomoda, A fuzzy neuron and its application to pattern recognition, *Proc. 3rd IFSA Cong.*, Seattle (1989) 30–38.

[59] T. Yamakawa, E. Uchino, T. Miki and H. Kusanagi, A neo fuzzy neuron and its application to system identification and prediction of the system behavior, *Proc. 2nd Internat. Conf. on Fuzzy Logic and Neural Networks*, Iizuka, Japan (1992) 477–483.