

Invited Review

On the principles of fuzzy neural networks

M.M. Gupta and D.H. Rao

Intelligent Systems Research Laboratory, College of Engineering, University of Saskatchewan, Saskatoon, Canada, S7N 0W0

Received August 1993

Revised September 1993

Abstract: Over the last decade or so, significant advances have been made in two distinct technological areas: fuzzy logic and computational neural networks. The theory of fuzzy logic provides a mathematical framework to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning. Also, it provides a mathematical morphology to emulate certain perceptual and linguistic attributes associated with human cognition. On the other hand, the computational neural network paradigms have evolved in the process of understanding the incredible learning and adaptive features of neuronal mechanisms inherent in certain biological species. Computational neural networks replicate, on a small scale, some of the computational operations observed in biological learning and adaptation. The integration of these two fields, fuzzy logic and neural networks; has given birth to an emerging technological field – the fuzzy neural networks. The fuzzy neural networks have the potential to capture the benefits of the two fascinating fields, fuzzy logic and neural networks, into a single capsule. The intent of this tutorial paper is to describe the basic notions of biological and computational neuronal morphologies, and to describe the principles and architectures of fuzzy neural networks. Towards this goal, we develop a fuzzy neural architecture based upon the notion of T-norm and T-conorm connectives. An error-based learning scheme is described for this neural structure.

Keywords: Fuzzy logic; neural networks; fuzzy neural networks; confluence operation; synaptic and somatic operations.

Correspondence to: Dr. M.M. Gupta, Intelligent Systems Research Laboratory, College of Engineering, University of Saskatchewan, Saskatoon, Canada, S7N 0W0. E-mail: guptam@sask.usask.ca

1. Introduction

1.1. Motivation

The incredible flexibility and adaptability of biological neuronal control mechanisms may be used as a plausible source of motivation and framework for the design of intelligent and autonomous robots. Unlike most of the conventional control techniques, biological control mechanisms are non-model based; and such non-model based mechanisms are quite successful in dealing with uncertainty, complexity, imprecision and approximate data. For example, we can reach a destination with vague and approximate information: “to get to my office take a left turn after about 100 feet and then go straight approximately 75 feet”. With this fuzzy information, our carbon-based computer, the brain, will generate motor commands and smoothly coordinate many degrees of freedom during the execution of manipulative tasks in an unstructured environment. Biological control mechanisms are usually very complex and do not depend upon exact mathematical formulation of their operations. They carry out complex tasks without having to develop their mathematical models or that of the environment, and without solving, in an explicit form, any integral, differential or any complex mathematical equations.

On the other hand, to make a mobile robot perform the same task, reach a destination with vague and imprecise information, is an extremely complex task for it involves the fusion of most of the existing control methodologies such as adaptive control, knowledge-base engineering, fuzzy logic and computational neural networks. The computations required to coordinate different robot joints to produce a desired

trajectory may be obtained by solving complex trigonometric relationships between different structural members of the robot. The control methodology developed in this traditional way may completely fail should the desired task or the environment change.

It is our hypothesis that if the fundamental principles of neural computations used by biological control systems are understood, it seems most likely that an entirely a new generation of control methodologies can be developed which are more robust and intelligent, far beyond the capabilities of the present techniques based upon explicit mathematical modeling.

In this process of understanding biological computational power, and the desire of system scientists to capture the power, the two most powerful fields in modern technology, namely fuzzy logic and neural networks along with genetic algorithms, have emerged [1]. Over the last decade or so, however, these two fields have grown independently to form distinct branches of science and technology. During the recent years, an integration of these two fields has presented the system designers with another powerful computational tool called the fuzzy neural networks.

1.2. Integration of fuzzy logic and neural networks

Fuzzy logic provides an inference morphology that enables approximate human reasoning

capabilities to be applied to knowledge-based systems [2, 3]. The theory of fuzzy logic provides a mathematical strength to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning. Also, it provides a mathematical morphology to emulate certain perceptual and linguistic attributes associated with human cognition.

While fuzzy theory provides an inference mechanism under cognitive uncertainty, computational neural networks offer exciting advantages such as learning, adaptation, fault-tolerance, parallelism and generalization. The computational neural networks, comprising of processing elements called neurons, are capable of coping with computational complexity, non-linearity and uncertainty. In view of this versatility of neural networks, it is believed that they hold great potential as building blocks for a variety of behaviors associated with human cognition.

A brief comparative study between fuzzy systems and neural networks in their operations in the context of knowledge acquisition, uncertainty, reasoning and adaptation is presented in Table 1.

To enable a system to deal with cognitive uncertainties in a manner more like humans, one may incorporate the concept of fuzzy logic into the neural network. Although fuzzy logic is a natural mechanism for modeling cognitive uncertainty, it may involve an increase in the amount of computation required (compared with a system using classical binary logic). This can be

Table 1. A comparative study between fuzzy systems and neural networks

Skills		Fuzzy systems	Neural networks
<i>Knowledge acquisition</i>	Inputs	Human experts	Sample sets
	Tools	Interaction	Algorithms
<i>Uncertainty</i>	Information	Quantitative and qualitative	Quantitative
	Cognition	Decision making	Perception
<i>Reasoning</i>	Mechanism	Heuristic search	Parallel computations
	Speed	Low	High
<i>Adaptation</i>	Fault-tolerance	Low	Very high
	Learning	Induction	Adjusting synaptic weights
<i>Natural language</i>	Implementation	Explicit	Implicit
	Flexibility	High	Low

readily offset by using fuzzy neural network approaches having the potential for parallel computations with high flexibility.

A fuzzy neuron is designed to function in much the same way as a non-fuzzy neuron, except that it reflects the fuzzy nature of a neuron and has the ability to cope with fuzzy information. Inputs to the fuzzy neuron are fuzzy sets (x_1, x_2, \dots, x_N) in the universe of discourse (X_1, X_2, \dots, X_N) respectively. These fuzzy sets may be labeled by such linguistic terms as high, large, warm, medium, etc. The fuzzy inputs are then 'weighted' in synapses in a much different way from that used in a non-fuzzy case. The weighted fuzzy inputs are then aggregated not by the summation but by the fuzzy aggregation operations (fuzzy union, weighted mean, or intersection).

An important difference between the computational aspects of a non-fuzzy neuron and that of a fuzzy neuron is the definition of mathematical operations. The mathematical operation in a non-fuzzy neuron may be defined in terms of confluence operation (usually, an inner product) between adjustable synaptic weights and neural inputs. Any learning and adaptation occurring within the neuron involves modifying these synaptic weights. In a fuzzy neuron, the synaptic connections are represented by a two-dimensional fuzzy relation between the synaptic weights and neural inputs. In this situation, learning involves changing the two-dimensional relation surface at each synapse.

The term 'fuzzy neural network (FNN)' has existed for more than a decade now, however, the recent resurgence of interest in this area is motivated by the increasing recognition of the potential of fuzzy logic and neural networks as two of the most promising approaches for exploring the functioning of human brains. Many researchers are currently investigating ways and means of building fuzzy neural networks by incorporating the notion of fuzziness into a neural network framework [1, 4–13].

Yamakawa et al. [6] described a fuzzy neural network model and applied it successfully to a pattern recognition problem. Kuncicky and Kandel [7] proposed a fuzzy neuron model in which the output of one neuron is represented by a fuzzy level of confidence and the firing process in a neuron is regarded as an attempt to

find a typical value among inputs. Kiszka and Gupta [8] studied a fuzzy neuron model described by the logic equations. However, no specific learning algorithms are developed in these three cases. Gupta and Knopf [5] proposed a fuzzy neuron model which is similar to the first two cases except that a specific modification scheme was proposed for weights adaptation during learning. Nakanishi et al. [9] and Hayashi et al. [10] used the non-fuzzy neural approach for the design of fuzzy logic controllers with adaptive and learning features. Similarly, Cohen and Hudson [4] used non-fuzzy neural network learning techniques to determine the weights of antecedents for use in fuzzy expert systems. However, no fuzzy neural structures were used.

One way to incorporate fuzziness into the neural network is by arranging the integrator/transfer functions at each node to perform some sort of fuzzy aggregation on the numerical information arriving at each node [11]. Another way to introduce fuzziness into the neural network is through the input data itself, which may be 'fuzzified' in one of several ways. Gupta and Qi [12] proposed three different fuzzy neural models. Recently, Carpenter et al. [13] proposed Fuzzy ARTMAP as an extension of their well known Adaptive Resonance Theory (ART) based neural network. The Fuzzy ARTMAP is a synthesis of fuzzy logic and ART network. The neural network structure realizes a new min-max learning rule that minimizes predictive error and improves generalization. Furthermore, it learns each input as it is received on-line, rather than performing an off-line optimization of a certain function. The applications of Fuzzy ARTMAP have concentrated on pattern classification and image recognition.

In order to provide a motivation and some basic notions of 'neurons', we present a basic descriptive of biological neuronal morphology in Section 2. Following this description, we present some basic mathematical operations in terms of synaptic and somatic operations, or, equivalently, in terms of confluence and activation operations of a single computational neuron in Section 3. This mathematical neuronal morphology of a single neuron is extended to fuzzy neuron in Section 4. A basic introduction to fuzzy logic, and T-norm and T-conorm connectives are also described in this section. The

learning scheme for a single fuzzy neuron is presented in Section 5, followed by conclusions in the last section. Hopefully, this tutorial presentation will help the readers to extend and apply these concepts to develop advanced notions of fuzzy neural systems for engineering and other decision making problems.

2. Biological neuronal morphology

In this section, we briefly describe the biological neuronal morphology (structure) which forms the basis for the study of computational neural networks.

The basic building block of the central nervous system (CNS) is the neuron, the cell that processes and communicates information to and from various parts of the body. From an information processing point of view an individual neuron consists of the following three parts each associated with a particular mathematical function (Figures 1 and 2).

- (i) the synapses are a storage area of the past experience (knowledge base) and receive information from other neurons;
- (ii) the cell body, called soma, receives synaptic information and performs further processing of information; and
- (iii) the neuron transmits information to other neurons through a single fiber called axon.

The junction point of an axon with a dendrite is called synapse. Synapses provide long term memory (LTM) to the past accumulated experience and this is a storage for the knowledge base. A single biological neuron may

be having, on the average, 10 000 synaptic connections.

A schematic diagram of the biological neuron is shown in Figure 1. From a systems theoretic point of view, the neuron can be considered as a multiple-input–single-output (MISO) system.

Neurons are filled with and surrounded by fluids containing dissolved chemical ions. The main chemical ions are sodium (Na^+), calcium (Ca^{++}), potassium (K^+) and chloride (Cl^-). Na^+ and K^+ ions are largely responsible for generating the active neural response called an action potential, also called the nerve impulse. K^+ ions are mainly concentrated inside the cell of the neuron, whereas the Na^+ ions are concentrated outside the cell membrane.

The process of generating action potential either in the neuron (where the processing of information takes place) or in the axons (through which the transmission of information takes place) is due to exchange of ions (K^+ and Na^+) caused by a change in permeability of the cell membrane. When the nerve impulse, in the form of action potentials, reaches the synaptic junction at the end of the axon, the transmitter substance in the synaptic vesicles is released onto the dendrite of the neuron causing an electrical response. This electrical response can be either excitatory or inhibitory, depending upon the type of transmitter released and the nature of the dendrite membrane. The dendritic inputs originating from the inhibitory synapses tend to decrease this firing rate. The magnitude of the dendritic signal is proportional to the average frequency at which the pulses arrive at the synaptic junction. A schematic diagram of the

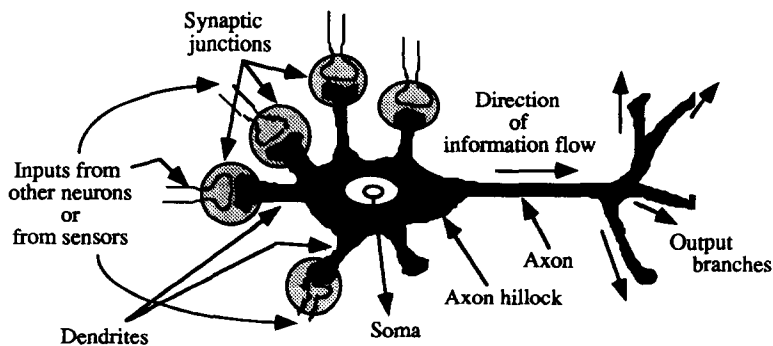


Fig. 1. A schematic view of the biological neuron. The soma of each neuron receives parallel inputs from its synapses, and generates a output at the soma.

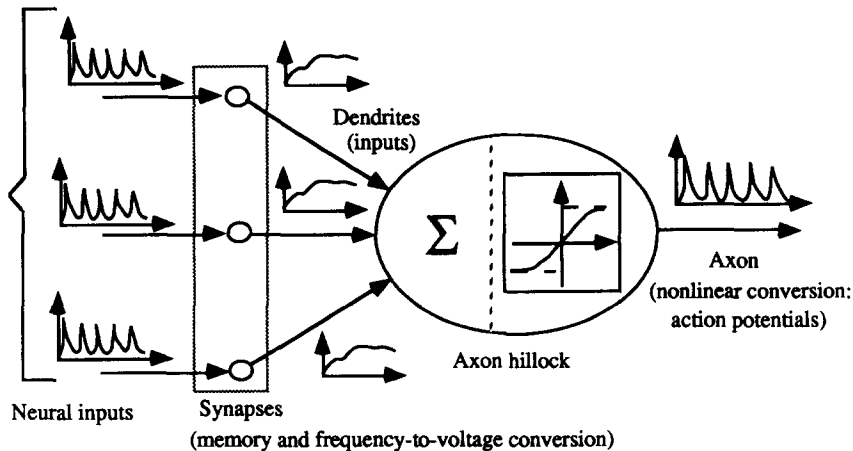


Fig. 2. A simplified model of the signal processing characteristics of a biological neuron.

signal coding characteristics of a biological neuron is shown in Figure 2. In terms of information processing, the synapse also performs a crude pulse frequency-to-voltage conversion.

From the mathematical point of view, it may be concluded that the processing of information within a neuron involves two distinct mathematical operations namely

(i) *the synaptic operation*: The strength (weight) of the synapse is a representation of the stored knowledge. The synaptic operation provides a weight to the neural inputs. Thus, the synaptic operation assigns a relative weight (significance) to each incoming signal according to the past experience (knowledge or memory) stored in the synapse.

(ii) *the somatic operation*: This provides aggregation, thresholding and nonlinear activation to the synaptic inputs. If the weighted aggregation of the neural inputs exceeds a certain threshold, soma will produce an output signal.

More details of mathematical synaptic and somatic operations for a computational neuron are given in the following section.

3. Computational neuronal morphology

3.1. Mathematical model of a neuron: Synaptic and somatic operations

As shown in Figures 1 and 2, a biological neuron consists of synapses (junction points) and

a soma – the main body of the neuron. The numerous synapses which adjoin a neuron receive neural inputs from other neurons and transmit modified (weighted) versions of these signals to the soma via the dendrites. Each soma receives, on the average, 10^4 dendritic inputs. The role of the soma is to perform a spatio-temporal weighted aggregation (often a summation) of all these inputs. If this weighted aggregation is greater than an intrinsic threshold, then the weighted aggregation is converted into an action potential yielding a neural output. These action potentials are transmitted along the axon to the other neurons for further processing, Figure 2.

From a signal processing point-of-view, the biological neuron has two key elements, synapse and soma, which are responsible for performing computational tasks such as learning, acquiring knowledge (storage or LTM of the past experience) and recognizing patterns. Each synapse is a storage element that contains some attribute of the past experience. The synapse learns by continuously adapting its strength (weight) to the new neuronal inputs. The soma combines the weighted inputs such that if it exceeds a certain threshold, then the neuron will fire. This axonal (output) signal undergoes a nonlinear transformation prior to leaving the axonic hillock in the soma. Mathematically, the synapses and early stage of the soma provide a confluence operation between the fresh neuronal inputs and stored knowledge (past experience). The latter part of the soma provides a nonlinear

bounded activation operation to the aggregated signals.

In simple terms, a neuron can be depicted as an information processing element (PE) which receives an n -dimensional neural input vector,

$$\mathbf{X}(t) = [x_1(t), x_2(t), \dots, x_i(t), \dots, x_n(t)]^T \in \mathbb{R}^n, \quad (1)$$

and yields a scalar neural output $y(t) \in \mathbb{R}^1$. The input vector, $\mathbf{X}(t) \in \mathbb{R}^n$, represents the signals being transmitted from the n -neighboring neurons (including self-feedback signal) and/or the outputs (measurements) from the sensory neurons. Mathematically, the information processing ability of a neuron can be represented as a nonlinear mapping operation, Ne, from the input vector $\mathbf{X}(t) \in \mathbb{R}^n$ to the scalar output $y(t) \in \mathbb{R}^1$; that is,

$$\text{Ne}: \mathbf{X}(t) \in \mathbb{R}^n \rightarrow y(t) \in \mathbb{R}^1 \quad (2)$$

as depicted in Figure 3.

Alternatively, we can write (2) as

$$y(t) = \text{Ne} [\mathbf{X}(t) \in \mathbb{R}^n] \in \mathbb{R}^1. \quad (3)$$

Mathematically, the neuronal nonlinear mapping function Ne can be divided into two parts: (i) confluence and (ii) nonlinear activation operations. The confluence operation provides the weighting, aggregating and thresholding operations to the neural inputs. In order to account for the thresholding operation, we will define the augmented vectors of neural inputs and synaptic weights as follows:

$$\begin{aligned} \mathbf{X}_a(t) &= [x_0(t), x_1(t), \dots, x_i(t), \dots, x_n(t)]^T \in \mathbb{R}^{n+1}, \\ x_0(t) &= 1, \end{aligned} \quad (4)$$

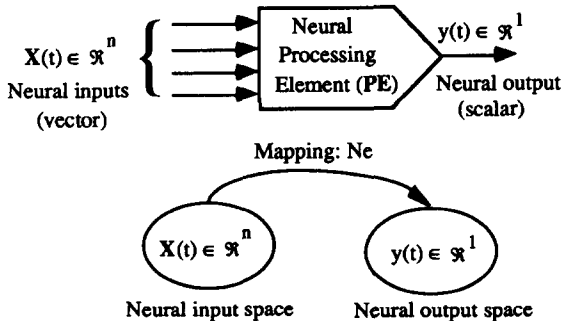


Fig. 3. The information processing ability of a neuron as represented by the nonlinear mapping function $\text{Ne}: \mathbf{X}(t) \in \mathbb{R}^n \rightarrow y(t) \in \mathbb{R}^1$.

and

$$\begin{aligned} \mathbf{W}_a(t) &= [w_0(t), w_1(t), \dots, \\ &w_i(t), \dots, w_n(t)]^T \in \mathbb{R}^{n+1} \end{aligned} \quad (5)$$

where $x_0(t) = 1$ and $w_0(t)$ introduces a thresholding (bias) term in the confluence operation. The confluence operation, \odot , essentially provides a measure of similarity between the augmented neural input vector $\mathbf{X}_a(t)$ (new information) and the augmented synaptic weight vector $\mathbf{W}_a(t)$ (accumulated knowledge-base). The nonlinear activation operation then performs a nonlinear mapping on the similarity measure. As shown in Figure 4, the first operation provides a linear mapping from $\mathbf{X}_a(t) \in \mathbb{R}^{n+1}$ to $u(t) \in \mathbb{R}^1$ through the weighting vector $\mathbf{W}_a(t) \in \mathbb{R}^{n+1}$. The second operation provides a nonlinear mapping from $u(t) \in \mathbb{R}^1$ to $y(t) \in \mathbb{R}^1$ through a nonlinear activation function $\psi[\cdot]$. These two basic mathematical operations of a computational neuron will now be described in greater detail.

3.1.1. Confluence operation: Measure of similarity

From a biological perspective the confluence operation represents the weighting of the input signals, $\mathbf{X}_a(t) \in \mathbb{R}^{n+1}$, with the accumulated knowledge stored at the synapses, $\mathbf{W}_a(t)$, and the spatio-temporal aggregation of these weighted inputs, as performed by the soma. The synaptic weighting assigns a relative weight to each incoming signal component $x_i(t)$ according to an attribute of the past experience (knowledge or memory) stored in synaptic weight $w_i(t)$.

One can mathematically view this confluence operation as a linear weighted mapping from the $(n+1)$ -dimensional neural input space $\mathbf{X}_a(t) \in \mathbb{R}^{n+1}$ to the one-dimensional space $u(t) \in \mathbb{R}^1$. The synaptic (weighting) and somatic (aggregation and thresholding) linear mapping can be modeled as

$$u(t) = \mathbf{W}_a(t) \odot \mathbf{X}_a(t) \quad (6)$$

where \odot is a confluence operation*. Equation

* The confluence operation defined in (6) is a combination of the synaptic weighting, somatic aggregating and somatic thresholding operations. This linear weighted mapping yields a scalar output $u(t)$ which is a measure of the similarity (mutual relationship) between the augmented neural input vector $\mathbf{X}_a(t)$ and the knowledge stored in the augmented synaptic weight vector $\mathbf{W}_a(t)$.

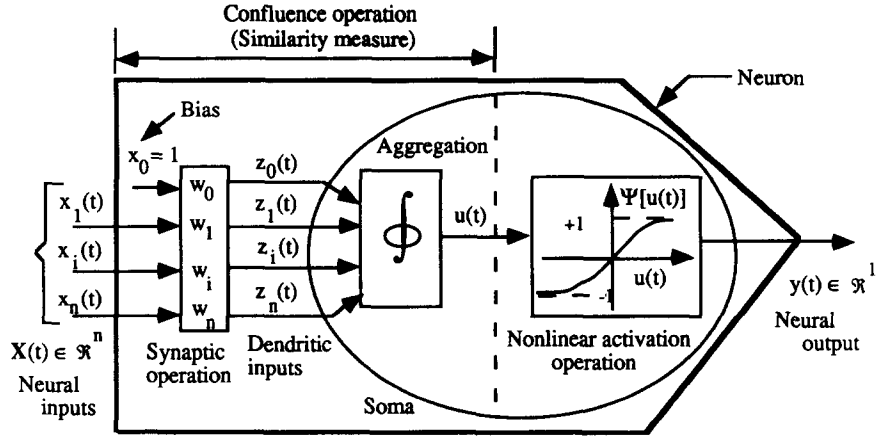


Fig. 4. Mathematical representation of a generalized neuron. The confluence operation, ©, compares new neural information $X_a(t)$ with the past experience stored in the synaptic weights $W_a(t)$, and the nonlinear activation operation, $\psi[\cdot]$, provides a bounded neural output $y(t)$.

(6) represents a measure of similarity between $X_a(t)$ (input vector) and $W_a(t)$ (synaptic weight vector). We will present two types of similarity measures:

- (i) the scalar (inner) product of the vectors $X_a(t)$ and $W_a(t)$, and
- (ii) the Euclidean distance between vectors $X_a(t)$ and $W_a(t)$.

The computational neurons of most neural networks described in the literature assume a confluence operation given by the scalar product. A popular exception to this is the radial basis function (RBF) network which employs the distance measure between $W_a(t)$ and $X_a(t)$ for describing the confluence between the inputs and weights. These two models of the confluence operation will now be described.

(i) *Inner Product of $X_a(t)$ and $W_a(t)$.* The inner product of $X_a(t)$ and $W_a(t)$ is defined geometrically as the projection of the neural inputs $X_a(t)$ (new information) onto the synaptic weights $W_a(t)$ (the accumulated knowledge) in the vector space as graphically illustrated in Figure 5; that is,

$$u(t)W_a(t)^T X_a(t) = \sum_{i=0}^n w_i x_i \tag{7}$$

where $X_a(t)$ and $W_a(t)$ are defined in (4) and (5).

(ii) *Euclidean distance measure between $X_a(t)$ and $W_a(t)$.* An alternative approach for measuring the similarity between the vectors $X_a(t)$ and $W_a(t)$ is to use the distance measure as shown in Figure 6. The Euclidean distance between the

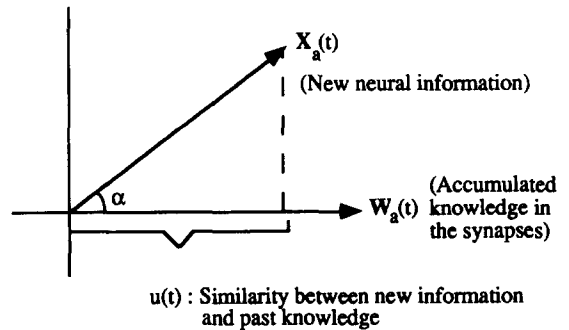


Fig. 5. A measure of similarity based on the projection (inner product) of the augmented neural vector $X_a(t)$ onto the augmented synaptic weight vector $W_a(t)$. Note that if angle $\alpha=0^\circ$ in the vector space, then $u(t)$ becomes a maximum value (most similar). Conversely, if $\alpha=90^\circ$, then the two vectors are orthogonal and the similarity measure is $u(t)=0$. Thus, if $X_a(t)$ lies in the first or the fourth quadrant with respect to $W_a(t)$, $u(t)$ is an excitatory signal (positive), whereas if $X_a(t)$ lies in the second or the third quadrant, $u(t)$ is an inhibitory signal (negative).

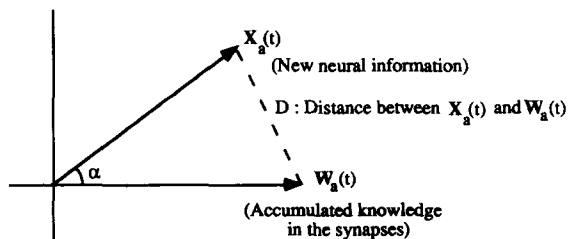


Fig. 6. Euclidean distance measure of similarity between the new neural information and the previously accumulated synaptic knowledge. Note that if $D=0$ then $X_a(t)$ has a lot in common with $W_a(t)$, and $u(t)=1.0$. Conversely, if $D=1$ then $X_a(t)$ and $W_a(t)$ have zero similarity yielding $u(t)=0$.

new neural information, $X_a(t)$, and the accumulated knowledge, $W_a(t)$, is given by

$$D = \beta \sqrt{[W_a(t) - X_a(t)]^T [W_a(t) - X_a(t)]} \in \mathbb{R}^1 \quad (8)$$

where β is a normalization constant such that $0 \leq D \leq 1$. The measure of similarity between $X_a(t)$ and $W_a(t)$ may then be defined as

$$u(t) = [1 - D]. \quad (9)$$

3.1.2. Somatic nonlinear activation function

The somatic nonlinear activation function, $\psi[\cdot]$ maps the confluence value $u(t) \in [-\infty, \infty]$ to a bounded neural output. In general, the neural output is in the range of $[0, 1]$ for unipolar signals, and $[-1, 1]$ for bipolar signals. The nonlinear activation operator transforms the signal $u(t)$ into a bounded neural output $y(t)$; that is,

$$y(t) = \psi[u(t)] \quad (10a)$$

$$= \psi[W_a(t) \odot X_a(t)] \in \mathbb{R}^1. \quad (10b)$$

Many different forms of mathematical functions can be used to model the nonlinear activation function, such as linear, hard limiter, unipolar and bipolar sigmoidal, multimodal sigmoidal and radial basis operators. Some typical activation functions are given in Table 2. The most widely

used is the sigmoidal function given by

$$\psi[x] = \frac{[e^{gx(t)} - e^{-gx(t)}]}{[e^{gx(t)} + e^{-gx(t)}]} = \tanh[x(t)] \quad (11)$$

where g is the parameter which controls the slope of the sigmoidal function.

3.2. Multi-layered neural networks

In the preceding subsection the mathematical details of a single neuron were described. Although a single neuron can perform certain simple pattern detection functions, the power of neural computation comes from the neurons connected in a network structure. Larger networks generally offer greater computational capabilities. Arranging neurons in layers or stages is supposed to mimic the layered structure of a certain portion of the brain. These multi-layered networks have been proven to have capabilities beyond those of a single layer. The most commonly used neural network architecture in applications, such as pattern recognition, system identification and control, is the multi-layered neural network (MNN) with an error back propagation (BP) algorithm. A typical MNN comprises of an input layer, output layer, and one hidden layer of neurons is shown

Table 2. Examples of typical nonlinear activation operators $\psi[\cdot]$

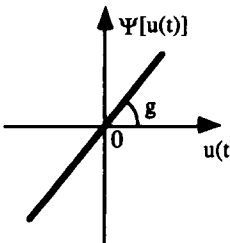
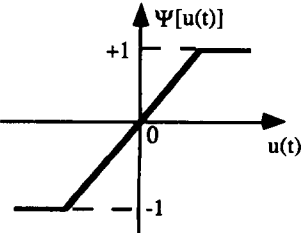
Type	Equation	Functional form
(i) Linear	$\psi[u(t)] = g u,$ $g > 0, \text{ activation gain}$	
(ii) Piecewise linear	$\psi[u(t)] = \begin{cases} +1 & \text{if } g u > 1, \\ g u & \text{if } g u < 1, \\ -1 & \text{if } g u < -1, \end{cases}$ $g > 0, \text{ activation gain}$	

Table 2. (continued)

Type	Equation	Functional form
(iii) Hard limiter	$\psi[u(t)] = \text{sgn}[u]$	
(iv) Unipolar sigmoidal	$\psi[u(t)] = \frac{1}{1 + \exp(-g u)}$ $g > 0, \text{ activation gain}$	
(v) Bipolar sigmoidal	$\psi[u(t)] = \tanh[g u(t)],$ $g > 0, \text{ activation gain}$	
(vi) Unipolar multimode sigmoidal	$\psi[u(t)] = \frac{1}{2} \left[1 + \frac{1}{M} \sum_{i=1}^M \tanh(g^i (u - w_0^i)) \right]$ $g^i > 0, \text{ activation gain}$	
(vii) Radial basis function (RBF)	$\psi[u(t)] = \exp(u(t))$ $u(t) = \left[\frac{-\sum_{i=0}^N (w_i(t) - x_i(t))^2}{2\sigma^2} \right]$	
(viii) Maximum	$\psi[u(t)] = \begin{cases} 1 & \text{if } x_i(t) = \text{MAX}\{x_n(t)\}, \\ 0 & \text{otherwise} \end{cases}$ $n \in \mathbb{N}; n = \text{set of all possible winners}$	

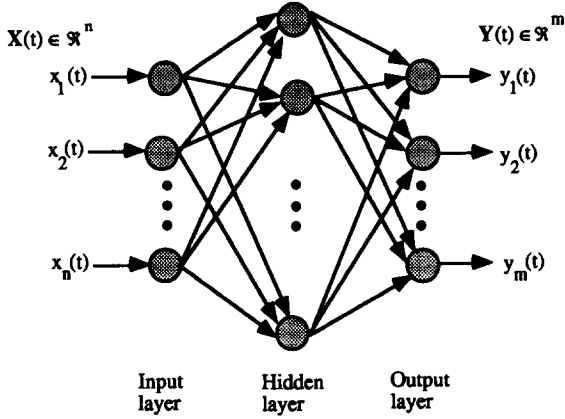


Fig. 7a. A densely interconnected three layered static neural network. Each shaded circle, or node, represents the neuron shown in Figure 4. This neural network consists an input layer (stage) with input vector, $X(t) = [x_1(t), \dots, x_i(t), \dots, x_n(t)]^T \in \mathbb{R}^n$, and the output layer with output vector $Y(t) = [y_1(t), \dots, y_i(t), \dots, y_m(t)]^T \in \mathbb{R}^m$. Layers between the input and output layers are normally referred to as hidden (intermediate) layers.

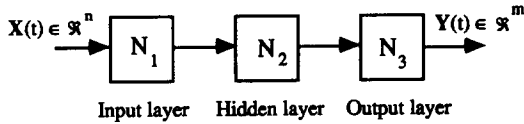


Fig. 7b. A block diagram representation of a three layered static neural network (MNN) with the input vector $X(t) \in \mathbb{R}^n$ and output vector $Y(t) \in \mathbb{R}^m$.

in Figure 7a. A simplified block diagram representation of the MNN is given in Figure 7b.

The input–output mapping of the MNN shown in Figure 7 can be mathematically represented by

$$Y(t) = N_3[N_2[N_1[X(t) \in \mathbb{R}^n]]] \in \mathbb{R}^m. \quad (12)$$

In terms of the confluence and nonlinear activation operators, (12) can be rewritten as

$$Y(t) = \psi^3[W_a^3(t) \odot \psi^2[W_a^2(t) \odot \psi^1[W_a^1(t) \odot X_a(t)]]] \quad (13)$$

where $\psi^i[\cdot]$ is the nonlinear activation operator, \odot is the confluence operator (scalar product or distance measure), and $W_a^1(t)$, $W_a^2(t)$ and $W_a^3(t)$ are the augmented synaptic weight vectors for the input, hidden and output layers, respectively.

4. Fuzzy neural network architectures

4.1. Fuzzy logic: Basic introduction

The concept of graded membership in fuzzy sets was introduced by Zadeh [14] in 1965. This notion of graded membership was introduced in order to provide a mathematical precision to information arising from our cognitive process. The theory of fuzzy sets provides a mechanism for representing linguistic constructs such as ‘many’, ‘low’, ‘medium’, ‘often’, ‘few’. In general, the fuzzy logic provides an inference structure that enables approximate human reasoning capabilities [14–19]. On the contrary, the traditional binary set theory describes crisp events, events that either do or do not occur. It uses probability theory to explain if an event will occur, measuring the chance with which a given event is expected to occur. The theory of fuzzy logic is based upon the notion of relative graded membership and so are the functions of mentation and cognitive processes. Thus, the utility of fuzzy sets lies in their ability to model uncertain or ambiguous data so often encountered in real life.

Fuzzy set definition. Let X be a space of points (or objects) with a generic element of X denoted by x . X is often referred to as the universe of discourse. A fuzzy set (class) A in X is characterized by a membership (characteristic) function $\mu_A(t)$ which associates with each point in X a real number in the interval $[0, 1]$, with the value of $\mu_A(x)$ representing the ‘grade of membership’ of x in A . Thus, the nearer the value of $\mu_A(x)$ to unity, the higher the grade of membership of x in A .

A fuzzy set A is a subset of the universe of discourse X that admits partial membership. The fuzzy set A is defined as an ordered pair

$$A = \{(x, \mu_A(x))\} \quad (14)$$

where $x \in X$ and $0 \leq \mu_A(x) \leq 1$. The membership function $\mu_A(x)$ describes the degree to which the object x belongs to the set A . $\mu_A(x)$ is also referred to as the characteristic function or graded membership of x in A . If $\mu_A(x) = 0$ then it is certain that x is not in A , and $\mu_A(x) = 1$ then it is certain that x is in A . For x over $0 < \mu_A(x) < 1$, there is an uncertainty associated

with x , that is, x belongs to A with the possibility $\mu_A(x)$.

4.2. Fuzzy neural architectures

Neural network structures can deal with imprecise data and ill-defined activities. However, the subjective phenomena such as reasoning and perceptions are often regarded beyond the domain of conventional neural network theory. It is interesting to note that fuzzy logic is another powerful tool for modeling uncertainties associated with human cognition, thinking and perception. In fact, the neural network approach fuses well with fuzzy logic [1, 4–6] and some research endeavors have given birth to the field of ‘fuzzy neural networks’ or ‘fuzzy neural systems’. Paradigms based upon this integration are believed to have considerable potential in the areas of expert systems, medical diagnosis, control systems, pattern recognition and system modeling. Two possible models of fuzzy neural systems are schematically shown in Figures 8(a) and (8b).

The computational process envisioned for fuzzy-neural systems is as follows. It starts with the development of a ‘fuzzy neuron’ based on

the understanding of biological neuronal morphologies, followed by learning mechanisms. This leads to the following three steps in a fuzzy-neural computational process:

- (i) development of fuzzy neural models motivated by biological neurons,
- (ii) models of synaptic connections which incorporates ‘fuzziness’ into neural network, and
- (iii) development of learning algorithms (that is, the method of adjusting the synaptic weights).

Based upon the computational process involved in a fuzzy-neural system, one may broadly classify the fuzzy neural structures as feedforward (static) and feedback (dynamic), Figure 9.

In a feedforward (static) architecture, the neuron responds instantaneously to the fuzzy inputs because of the absence of dynamic elements in the structure. The neural mathematical operations in a feedforward network can be performed either by fuzzy arithmetic or fuzzy logic operations. As was mentioned in the preceding section, the function of a non-fuzzy neuron can be modeled as

$$y(t) = \psi \left[\sum_{i=0}^n w_i x_i \right] \tag{15}$$

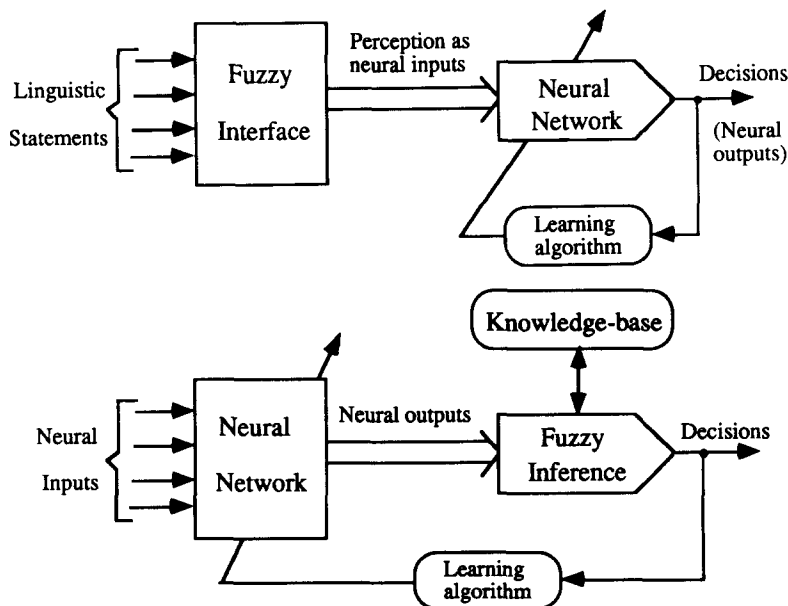


Fig. 8. Two models of fuzzy neural systems. (a) In response to linguistic statements, the ‘fuzzy interface’ block provides an input vector, to a multi-layered neural network. The neural network can be adapted (trained) to yield desired command outputs or decisions. (b) In this scheme, a multi-layered neural network drives the fuzzy inference mechanism.

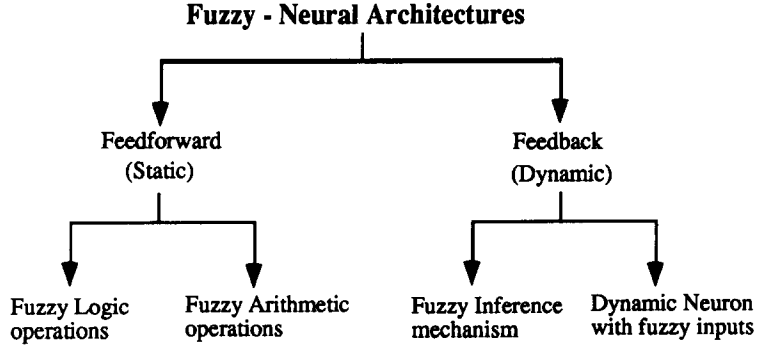


Fig. 9. Classification of fuzzy-neural systems.

where $[x_1, \dots, x_n]$ represent neural inputs $[w_1, \dots, w_n]$ the synaptic weights, $y(t)$ the neural output and $\psi[\cdot]$ is some nonlinear activation function. From (15) it may be observed that the mathematical operations involved in a computational neuron are:

- (i) the scalar product between the neural inputs and the synaptic weights, and
- (ii) the summation of these products.

The scalar product in (15) can be replaced by fuzzy multiplication and the summation operation by fuzzy addition. Detailed descriptions of fuzzy arithmetic operations may be found in [3]. These modifications lead to a fuzzy neural architecture based on fuzzy arithmetic operations. The function of such a fuzzy neuron can be modeled by the following equation:

$$y(t) = \psi \left[\sum_{i=0}^n (+) w_i(\cdot) x_i \right] \quad (16)$$

where $(+)$ and (\cdot) are fuzzy addition and fuzzy multiplication operators respectively.

Alternatively, fuzzy logic operations, such as **OR**, **AND**, **NOT**, or their generalized versions can be introduced in (15). In this paper, we confine our discussion to a neural architecture based upon fuzzy logic operations.

The other classification, as shown in Figure 9, is the feedback (dynamic) architecture. The dynamic networks not only provide some robust computing characteristics but also bring about greater insights into biological neural structures. The dynamics in fuzzy neural computing does provide some functional basis of the cerebellum and its associated circuitry, and can offer great computational advantages over purely feedfor-

ward architectures. Based on fuzzy inference mechanism, Gupta and Knopf [20] proposed a dynamic fuzzy-neural architecture and developed a Fuzzy Expert Navigator (FEN) for an autonomous vehicle.

4.3. Fuzzy-neural architecture based on fuzzy logic operations [1]

If we express the neural input signals in terms of their membership functions each over the interval $[0, 1]$, rather than in their absolute amplitudes, then we can write the augmented vector of neural inputs as

$$\mathbf{X}_a(t) = [x_0(t), x_1(t), \dots, x_i(t), \dots, x_n(t)]^T \in [0, 1]^{n+1} \quad (17)$$

where these neural signal (including the bias term, x_0) are bounded by the $(n+1)$ dimensional hypercube $[0, 1]^{n+1}$. Similarly, the augmented synaptic weighting vector $\mathbf{W}_a(t)$ can be expressed over the unit hypercube $[0, 1]^{n+1}$.

We perform mathematical operations on these signals using logical operations (connectives) [21, 22] such as **OR**, **AND** (or their generalized form based upon triangular norm of T-operators) and **negation**.

Let us express the inputs x_1 and x_2 over $[0, 1]$. Then we define the generalized **AND** (T-operation) as a **T** mapping function:

$$\mathbf{T}: [0, 1] \times [0, 1] \rightarrow [0, 1]$$

given by

$$y_1 = [x_1 \text{ AND } x_2] \triangleq [x_1 \mathbf{T} x_2] = \mathbf{T}[x_1, x_2]. \quad (18)$$

Similarly, we define the generalized **OR**, (**T**-conorm) as a **S** mapping function

$$\mathbf{S}: [0, 1] \times [0, 1] \rightarrow [0, 1]$$

given by

$$y_2 = [x_1 \text{ OR } x_2] \triangleq [x_1 \mathbf{S} x_2] = \mathbf{S}[x_1, x_2]. \quad (19)$$

Negation **N** on $x_1 \in [0, 1]$ is defined as a mapping

$$\mathbf{N}: [0, 1] \rightarrow [0, 1]$$

with the following properties

$$y_3 = \mathbf{N}[x_1] = 1 - x_1. \quad (20)$$

Thus, $\mathbf{N}(0) = 1$, $\mathbf{N}(1) = 0$, and $\mathbf{N}(\mathbf{N}(x)) = x$.

Now, we describe some important properties of the **T** and **S** operators:

$$\begin{aligned} \mathbf{T}(0, 0) &= 0, & \mathbf{T}(1, 1) &= 1, \\ \mathbf{T}(1, x) &= x, & \mathbf{T}(x, y) &= \mathbf{T}(y, x), \end{aligned} \quad (21a)$$

$$\begin{aligned} \mathbf{S}(0, 0) &= 0, & \mathbf{S}(1, 1) &= 1, \\ \mathbf{S}(0, x) &= x, & \mathbf{S}(x, y) &= \mathbf{S}(y, x). \end{aligned} \quad (21b)$$

Also, De Morgan's Theorems are stated as follows:

$$\mathbf{T}(x_1, x_2) = 1 - \mathbf{S}(1 - x_1, 1 - x_2),$$

and

$$\mathbf{S}(x_1, x_2) = 1 - \mathbf{T}(1 - x_1, 1 - x_2). \quad (22)$$

In the development of fuzzy logic based neural morphology, we will use the following combined synaptic and somatic operations: Let the augmented vector of neural inputs and synaptic weights be represented by

$$\mathbf{X}_a(t) \in [0, 1]^{n+1}, \text{ and } \mathbf{W}_a(t) \in [0, 1]^{n+1}$$

respectively. Then, in (6), by replacing the \odot -operation by the **T**-operation, and the Σ -operation by the **S**-operation, we get

$$u(t) = \mathbf{S}_{i=0}^n [w_i(t) \mathbf{T} x_i(t)] \in [0, 1], \quad (23a)$$

and

$$y(t) = \psi[u(t)] \in [0, 1] \quad (23b)$$

where $\psi[\cdot]$ is a nonlinear mapping function.

4.3.1. Unipolar to bipolar transformation [1]

The logical operations defined in the preceding section are unipolar signals over the positive unit interval $[0, 1]$. Such logical operations

provide only the neural state corresponding to the excitatory (positive) interactions. In order to account for both the excitatory (positive) and the inhibitory (negative) interactions, of the neural input vector, we must consider both $\mathbf{X}_a(t)$ and its negated values $\mathbf{N}[\mathbf{X}_a(t)]$, thus making the neural inputs of dimensions $(2n + 2)$.

Alternatively, we may express the neural inputs and synaptic weights as bipolar signals and weights over the interval $[-1, 1]$ and redefine the logical operations over this interval. We will provide a brief description of this transformation for unipolar $[0, 1]$, to bipolar $[-1, 1]$, and of the definition of logical operations over the interval $[-1, 1]$.

Let $x(t) \in [0, 1]$ be a unipolar signal. The corresponding bipolar signal $z(t)$ is defined as

$$z(t) = 2x(t) - 1. \quad (24)$$

The negation is defined as

$$\mathbf{N}[x] = 1 - x(t), \text{ for unipolar signals,}$$

and

$$\mathbf{N}[z] = -z(t), \text{ for bipolar signals.}$$

The **T** and **S** operations defined in the interval $[0, 1]$ can be transformed to the interval $[-1, 1]$ using (24).

In Table 3, we give a summary of logical **T** and **S** operations for both unipolar and bipolar signals. We define in Table 4 some important logical functions and operations such as Godel's implication, degree of equality using Godel's implication, degree of equality using Łukasiewicz conjunction, and degree of error (inequality) for two bipolar signals z_1 and $z_2 \in [-1, 1]$.

5. Learning scheme: Adapting the knowledge base

The weighting and spatio-temporal aggregation operations performed by the synapses and soma, respectively, provide a similarity measure between the input vector $\mathbf{X}_a(t)$ (new neural information) and the synaptic weight vector $\mathbf{W}_a(t)$ (accumulated knowledge base). When a new input pattern that is significantly different from the previously learned patterns is presented to the neural network, the similarity between

Table 3. Summary of logical operations on unipolar and bipolar signals

Unipolar signals $x(t) \in [0, 1]$	Bipolar signals $z(t) \in [-1, 1]$
(i) Bipolar to unipolar transformation $x(t) = \frac{1}{2}(z(t) + 1)$	(i) Unipolar to bipolar transformation $z(t) = 2x(t) - 1$
(ii) Negation: $\mathbf{N}[x(t)] = x(t) = 1 - x(t)$	(ii) Negation: $\mathbf{N}[z(t)] = z(t) = -z(t)$
(iii) Boundary conditions	(iii) Boundary Conditions
(a) T-Operator (generalized AND) $\mathbf{T}(0, 0) = 0; \quad \mathbf{T}(1, 1) = 1$ $\mathbf{T}(1, x) = x; \quad \mathbf{T}(x_1, x_2) = \mathbf{T}(x_2, x_1)$	(a) T-operator (generalized AND) $\mathbf{T}(-1, -1) = -1; \quad \mathbf{T}(1, 1) = 1$ $\mathbf{T}(1, z) = z; \quad \mathbf{T}(z_1, z_2) = \mathbf{T}(z_2, z_1)$
(b) S-operator (generalized OR) $\mathbf{S}(0, 0) = 0; \quad \mathbf{S}(1, 1) = 1$ $\mathbf{S}(0, x) = x; \quad \mathbf{S}(x_1, x_2) = \mathbf{S}(x_2, x_1)$	(b) S-operator (generalized OR) $\mathbf{S}(-1, 1) = -1; \quad \mathbf{S}(1, 1) = 1$ $\mathbf{S}(-1, z) = z; \quad \mathbf{S}(z_1, z_2) = \mathbf{S}(z_2, z_1)$
(iv) Generalized De Morgan's Theorem $\mathbf{T}(x_1, x_2) = 1 - \mathbf{S}(1 - x_1, 1 - x_2)$ $\mathbf{S}(x_1, x_2) = 1 - \mathbf{T}(1 - x_1, 1 - x_2)$	(iv) Generalized De Morgan's Theorem $\mathbf{T}(z_1, z_2) = -\mathbf{S}(-z_1, -z_2)$ $\mathbf{S}(z_1, z_2) = -\mathbf{T}(-z_1, -z_2)$

Table 4. Summary of some important logical functions and operations (these logical functions and operations can be defined on both the unipolar, $[0, 1]$ and bipolar $[-1, 1]$, signals, but here, we will consider the bipolar signals)

(a) *Godel's implication* \mathbf{G} : Godel's implication $[z_1 \mathbf{G} z_2]$ (read as, z_1 implies z_2) is defined as

$$[z_1 \mathbf{G} z_2] = [z_1 \rightarrow z_2] = \begin{cases} 1 & z_1 \leq z_2, \\ z_2, & z_1 > z_2, \end{cases}$$

(b) *Degree of equality* (using Godel's implication) $\eta(z_1, z_2)$. Given z_1 and z_2 over $[-1, 1]$, to what degree they are equal is defined as

$$\eta(z_1, z_2) = \frac{1}{2} \{ [z_1 \mathbf{G} z_2] \mathbf{T}\{z_2 \mathbf{G} z_1\} + \{ \bar{z}_1 \mathbf{G} \bar{z}_2 \mathbf{T}\{ \bar{z}_2 \mathbf{G} \bar{z}_1 \} \} \in [-1, 1]$$

where $\bar{z} = -z$.

(c) *Degree of equality* (using Łukasiewicz's conjunction): $\eta(z_1, z_2)$. Again, given z_1 and z_2 over $[-1, 1]$, to what degree they are equal is defined as

$$\eta(z_1, z_2) = [1 - |z_1 z_2|] \in [-1, 1].$$

(d) *Degree of inequality* (degree of error): $E(z_1, z_2) \in [-1, 1]$. Given z_1 and z_2 over $[-1, 1]$, in order to find the degree of difference or degree of inequality, we define $E[z_1, z_2]$ as the negation on the degree of equality; that is,

$$E[z_1, z_2] = \mathbf{N}[\eta(z_1, z_2)] = -\eta(z_1, z_2).$$

Thus, the degree of error, using Łukasiewicz conjunction can be defined as

$$E(z_1, z_2) = |z_1 - z_2| - 1.$$

this input and the existing knowledge base is small. As the neural network learns this new pattern, by changing the strength of the synaptic weights, the distance between the new information and accumulated knowledge decreases. In other words, the purpose of learning is to make $W_a(t)$ very similar to a given pattern $X_a(t)$.

Most of the neural network structures undergo

a 'learning' procedure during which the synaptic weights (connection strengths) are adapted. Algorithms for varying these connection strengths such that learning ensues are called 'learning rules'. The objective of learning rules depends upon the applications. For example, the objective in pattern classification from sample data is to classify and predict successfully on new

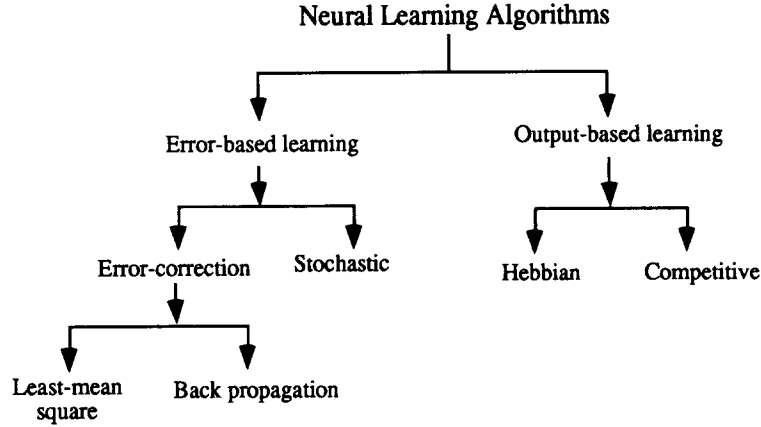


Fig. 10. A flow diagram of learning algorithms employed in different neural structures to adapt the synaptic weights.

data, while the objective in control applications is to approximate nonlinear functions, and/or to make unknown systems follow the desired response. In classification and functional approximation problems, each cycle of presentation of all cases is usually referred to as a ‘learning epoch’. However, there has been no generalization as to how a neural network can be adapted. A flow diagram illustrating the different learning algorithms normally employed for the adaptation of synaptic weights is shown in Figure 10. As shown in this figure, learning algorithms may be broadly categorized as ‘error-based (supervised)’ and ‘output-based (unsupervised)’.

Error-based (also known as supervised) learning algorithms employ an external reference signal (teacher) and generate an error signal by comparing the reference with the obtained response. Based on error signal, neural network modifies its synaptic connections to improve the system performance. In this learning scheme, it is assumed that the desired answer is known apriori. The error-based learning procedure is schematically shown in Figure 11.

A general equation for the error-based learning algorithm is

$$w_i(t + 1) = w_i(t) + \Delta w_i(t) \tag{25a}$$

where

$$\Delta w_i(t) = \mu_i x_i(t) [y_d(t) - y(t)] \tag{25b}$$

and $w_i(t)$ is the synaptic weight corresponding to the input $x_i(t)$. The parameter $\Delta w_i(t)$ is the change in synaptic connection $w_i(t)$ over an instant in time, μ_i is the learning rate, $y_d(t)$ is the

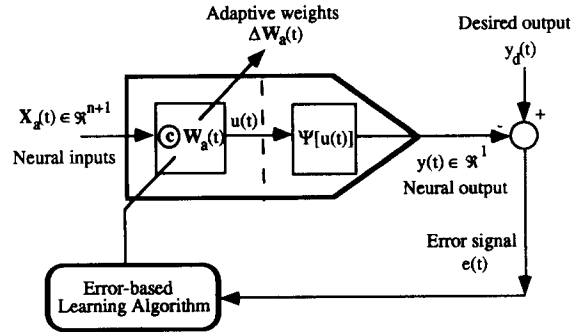


Fig. 11. An error-based (supervised) learning scheme where the learning process is guided by the error signal $e(t)$.

desired neural output, and $y(t)$ is the actual neural response. The proper selection of μ is of critical importance in these learning rules. A very small value of μ_i will result in extremely slow learning. On the other hand, a large value of μ_i will make learning faster, but it may also result in oscillations or make the system unstable.

In contrast, output-based learning algorithms do not incorporate a reference signal, and generally involve self-organization principles that rely only upon local information and internal control mechanisms in order to discover emergent collective properties. The two most important forms of output-based learning are Hebbian learning and competitive learning. Hebbian learning [23, 24], Figure 12, involves the adjustment of a synaptic weight according to the correlation of the response of the two neurons that adjoin it. A simple Hebbian learning rule used to describe the correlation of

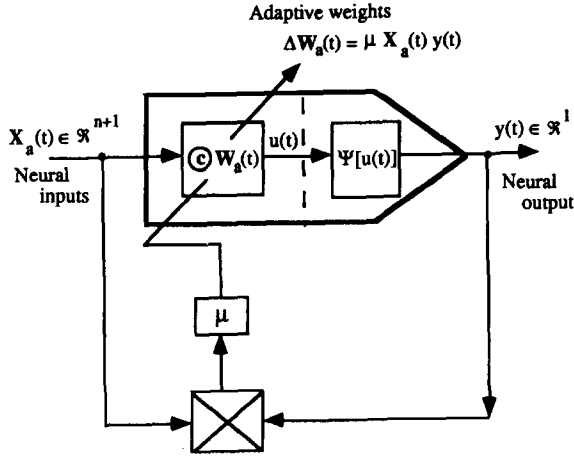


Fig. 12. An output-based (unsupervised) learning scheme, often called Hebbian learning, is guided by the neural output rather than by the output error as in error-based (supervised) learning scheme.

the input $x_i(t)$ with the neuron output $y(t)$ is

$$\Delta w_i(t) = \mu_i x_i(t) y(t) \tag{26}$$

where $\Delta w_i(t)$ represents the temporal change of the synaptic weight $w_i(t)$ and μ_i is the learning rate.

We can extend these basic learning rules to a fuzzy neuron. Figure 13 shows an error-based learning scheme for a fuzzy neuron with bipolar signals and synaptic weights. The augmented neural input signals $X_a(t)$ are defined over the

unit hypercube $[0, 1]^{n+1}$. Using the transformation given in (24), we transform the unipolar neural inputs $X_a(t)$ into bipolar signals $Z_a(t) \in [-1, 1]^{n+1}$.

The logical operation of this neuron is summarized as follows:

$$u(t) = \sum_{i=0}^n [w_i(t) T z_i(t)] \tag{27a}$$

which is equivalent to

$$u(t) = W_a^T(t) \text{ AND } Z_a(t) \in [-1, 1] \tag{27b}$$

(a logical scalar product operation), where $w_0(t)$ and $z_0(t)$ correspond to the bias terms and $z_0 = 1$. The neural output is defined as

$$y(t) = \psi[u(t)] \in [-1, 1] \tag{28a}$$

where $\psi[\cdot]$ is defined as

$$\psi[u(t)] = |u(t)|^g \cdot \text{sgn}[u(t)], \quad g > 0 \tag{28b}$$

where the parameter g is the somatic gain which controls the slope of the activation (sigmoidal) function. Let us define an error signal with respect to the desired neural output, $y_d(t) \in [-1, 1]$, as $e(t) = y_d(t) - y(t) \in [-1, 1]$. The objective of learning and adaptation in neural networks is to adapt the parameters of the neural structures, in this case $W_a(t)$ and g in (21a) and (21b), in order to minimize an error function. The learning rules to modify $W_a(t)$ and

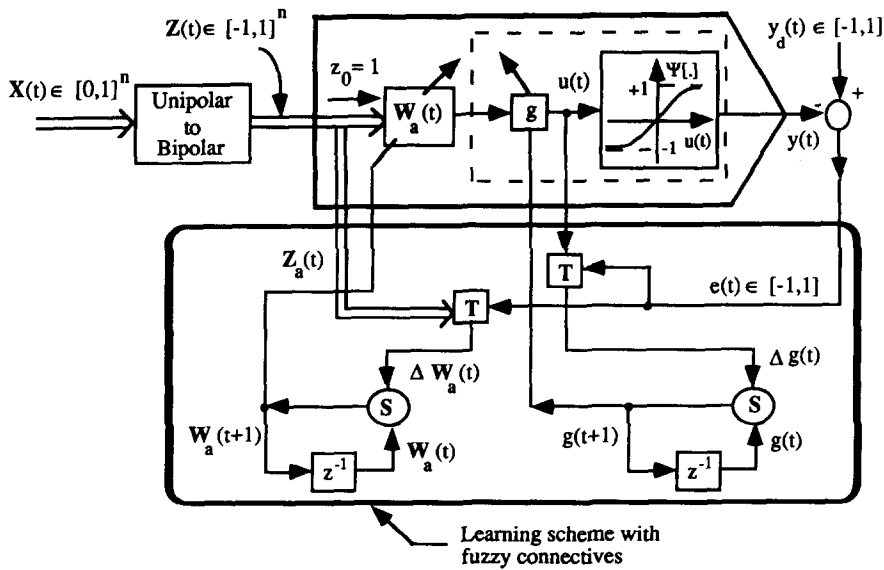


Fig. 13. Implementation of the learning scheme, Equations (29) and (30), to modify the synaptic weights, W_a , and the somatic gain, g , of a fuzzy neuron.

$g(t)$ may be developed as follows:

$$W_a(t+1) = W_a(t) \text{ OR } \Delta W_a(t) = S[W_a(t), \Delta W_a(t)], \quad (29a)$$

and

$$g(t+1) = g(t) \text{ OR } \Delta g(t) = S[g(t), \Delta g(t)] \quad (29b)$$

where

$$\Delta W_a(t) = Z_a(t) \text{ AND } e(t) = T[Z_a(t), e(t)], \quad (30a)$$

and

$$\Delta g(t) = u(t) \text{ AND } e(t) = T[u(t), e(t)]. \quad (30b)$$

The above description provides a learning scheme to update the neural weights of a fuzzy neuron. It represents one particular model of a fuzzy neural architecture. It is postulated that the fuzzy neural networks can learn by experience if their synaptic connections are interpreted as fuzzy relations between the external inputs and the dendritic inputs. Complex decisions may be derived from a shallow hierarchy of fuzzy neurons and their related network architectures with electronic circuitry.

6. Conclusions

This paper is a tutorial presentation on the principles of biological and conventional neuronal morphologies. Biology does provide a motivation and framework for the development of computational neural structures. Biological neuronal principles can be extended to generate several neural topologies and algorithms for both non-fuzzy and fuzzy situations. In this paper, we have emphasized the basic principles rather than giving some advanced structures of neural networks which are available in the literature.

Our emphasis in this paper, both from mathematical structure and information processing point of view, has been on the operations such as confluence and nonlinear activation. The confluence operation provides a measure of similarity between the neural inputs and accumulated stored experience in synaptic weights, and the activation function provides a graded output to similarity measure. Based upon

these neuronal operations, we developed the learning algorithms as discussed in Section 5 for both error-based and output-based learning. With the development of fuzzy neural networks, it is envisaged that learning schemes for autonomous vehicles will have the following features:

(i) easy to implement fuzzy natural languages so that the structure of knowledge is very clear and efficient,

(ii) any changes in the task and environment can be easily taken care of by adapting the neural weights, and

(iii) since a fuzzy system is one kind of interpolation [25], drastic reduction of data and software/hardware overheads can be achieved.

However, it should be noted that more research endeavors are necessary to develop general topology of fuzzy neural models, learning algorithms, and approximation theory so that these models are made applicable in system modeling and control of complex systems. The area of fuzzy neural networks is still in its infancy, and is a very fertile area of theoretical and applied research.

References

- [1] M.M. Gupta, Fuzzy logic and neural networks, *Tenth Int. Conf. on Multicriterion Decision Making*, Taipei, July 19–24, 1992, 281–294.
- [2] L.A. Zadeh, Outline of a new approach to the analysis of complex systems and decision process, *IEEE Trans. Systems, Man and Cybernetics* 3(1) (1973) 28–44.
- [3] A. Kaufmann and M.M. Gupta, *Introduction to Fuzzy Arithmetic: Theory and Applications*, 2nd edition (Van Nostrand Reinhold, New York, 1991).
- [4] M.E. Cohen and D.L. Hudson, An expert system on neural network techniques, in: I.B. Turksen, Ed., *The Proceedings of NAFIP*, Toronto, June 1990, 117–12.
- [5] M.M. Gupta and G.K. Knopf, Fuzzy neural network approach to control systems, *Proc. of First Int. Symp. on Uncertainty Modeling and Analysis*, Maryland, Dec. 3–5, 1990, 483–488.
- [6] T. Yamakawa and S. Tomoda, A fuzzy neuron and its application to pattern recognition, *Proc. of the Third IFSA Congress*, Seattle, Aug. 1989, 30–38.
- [7] D.C. Kuncicky and A. Kandel, A fuzzy interpretation of neural networks, in: J.C. Bezdek, Ed., *The Proceedings of the Third IFSA Congress*, Seattle, 1989, 113–116.
- [8] J.B. Kiszka and M.M. Gupta, Fuzzy logic neural network, *BUSEFAL* 4 (1990) 104–109.
- [9] S. Nakanishi, T. Takagi, K. Uehara and Y. Gotoh,

- Self-organizing fuzzy controllers by neural networks, *Int. Conf. on Fuzzy Logic and Neural Networks, IIZUKA '90, Japan 1990*, 187–192.
- [10] I. Hayashi, H. Nomura and N. Wakami, Artificial neural network driven fuzzy control and its application to learning of inverted pendulum system, in: J.C. Bezdek, Ed., *Proc. of the Third IFSA Congress, Seattle, 1989*, 610–613.
- [11] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms* (Plenum Press, NY, 1991).
- [12] M.M. Gupta and J. Qi, On fuzzy neuron models, *Int. Joint Conf. on Neural networks (IJCNN)*, Seattle, July 1991, 431–456.
- [13] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds and D.B. Rosen, Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps, *IEEE Trans. on Neural Networks*, 3(5) (Sept. 1992) 698–713.
- [14] L.A. Zadeh, Fuzzy Sets, *Information and Control* 8 (1965) 338–353.
- [15] P.K. Simpson, Fuzzy min–max neural networks – Part I: Classification, *IEEE Trans. on Neural Networks* 3(5) (Sept. 1992) 776–786.
- [16] M.M. Gupta, Uncertainty and information: The emerging paradigms, *Int. J. of Neuro and Mass-Parallel Computing and Information Systems* 2 (1991) 65–70.
- [17] S.K. Pal and S. Mitra, Multilayer perceptron, fuzzy sets, and classification, *IEEE Trans. on Neural Networks* 3(5) (Sept. 1992) 683–697.
- [18] M.M. Gupta, Fuzzy neural computing systems, *2nd Int. Conf. on Fuzzy Logic and Neural Networks*, July 17–22, Japan, 1992.
- [19] H.-J. Zimmermann, *Fuzzy Set Theory and Its Applications* (Dordrecht, Kluwer Academic Press, 1991).
- [20] M.M. Gupta and G.K. Knopf, Dynamic neural network for fuzzy inference, *SPIE's Conf. on Applications of Fuzzy Logic Technology*, Boston, Sept 7–10, 1993.
- [21] M.M. Gupta and J. Qi, Connections (AND, OR, NOT) and T-operators in fuzzy reasoning, in: I.R. Goodman, M.M. Gupta, H.T. Nguyen and G.S. Rodgers (Eds.), *Conditional Logic in Expert Systems*, (North-Holland, Amsterdam, 1991) 211–233.
- [22] M.M. Gupta and J. Qi, Design of fuzzy logic controllers based on generalized T-operators, *Fuzzy Sets and Systems* 40(3) (1991) 473–489.
- [23] D.O. Hebb, *The Organization of Behavior* (John Wiley and Sons, New York, 1949).
- [24] D. Hammerstrom, Working with neural networks, *IEEE Spectrum* July 1993, 46–53.
- [25] T. Yamakawa, A fuzzy inference engine in nonlinear analog mode and its application to a fuzzy logic control, *IEEE Trans. on Neural Networks* 4(3) (May 1993) 496–522.