Neural networks: evolution, topologies, learning algorithms and applications

Siddhartha Bhattacharyya
RCC Institute of Information Technology

Cross-Disciplinary Applications of Artificial Intelligence and Pattern Recognition:

Advancing Technologies

Vijay Kumar Mago
Simon Fraser University, Canada

Nitin Bhatia
DAV College, India

Information Science
REFERENCE

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Cross-disciplinary applications of artificial intelligence and pattern recognition : advancing technologies / Vijay Kumar Mago and Nitin Bhatia, editors.
p. cm.
Summary: “This book provides a common platform for researchers to present theoretical and applied research findings for enhancing and developing intelligent systems, discussing advances in and applications of pattern recognition technologies and artificial intelligence”-- Provided by publisher.
Includes bibliographical references and index.
ISBN 978-1-61350-429-1 (hardcover) -- ISBN 978-1-61350-430-7 (ebook) -- ISBN 978-1-61350-431-4 (print & perpetual access) 1. Pattern recognition systems. 2. Artificial intelligence. I. Mago, V. K. II. Bhatia, Nitin, 1978-
TK7882.P3C66 2012
006.3--dc23
2011046541

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 24
Neural Networks:
Evolution, Topologies, Learning Algorithms and Applications

Siddhartha Bhattacharyya
RCC Institute of Information Technology, India

ABSTRACT

Artificial neural networks form a class of soft computing tools, which are made up of interconnected computational primitives for the analysis of numeric data. These are inspired by the functional behavior of the human nervous system comprising millions of nerve cells or neurons. Different artificial neural network architectures have been evolved over the years based on the storage, transmission, and processing characteristics of the human nervous system.

These networks generally operate in two different modes, viz., supervised and unsupervised modes. The supervised mode of operation requires a supervisor to train the network with a training set of data. Networks operating in unsupervised mode apply topology preservation techniques so as to learn inputs. Representative examples of networks following either of these two modes are presented with reference to their topologies, configurations, types of input-output data and functional characteristics. Recent trends in this computing paradigm are also reported with due regards to the application perspectives.

INTRODUCTION

A neural network is a powerful data-modeling tool that is able to capture and represent complex input-output relationships (Haykin, 1999; Kumar, 2004). The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform intelligent tasks similar to those performed by the human brain. These tasks include understanding, cognition, perception, control of human body parts etc. Above all, the brain being the pivotal organ of the human body is also entrusted with the synchronization of these actions to avoid functional disorders with the help of the central nervous system, which by itself is an information

DOI: 10.4018/978-1-61350-429-1.ch024

processing entity. This multifaceted functionality of the human brain is attributed to the inherent massively parallel computing structure gifted by a highly dense structure of interconnected computing units referred to as *neurons* (approximately more than 10 billion in number), which evoke appropriate responses to external world signals (Rojas, 1996). These neurons or neuronal cells, which form the human nervous system, imbibe a series of biochemical reactions for the purpose of reception, storage, processing and transmission of information incident from the real world.

The neural cell body or *soma* houses the nucleus and is connected to treelike networks of nerve fibers called *dendrites* (Rojas, 1996). A single long fiber referred to as the *axon* extends from the soma. This axon eventually branches into strands and substrands to connect other neurons through junctions referred to as *synapses*. Complex biochemical reactions enable the transmission of signals from one neuron to another via the intermediate synapses when some transmitter compounds are released from the sending end of the junction. These compounds either raise or lower the electrical potential inside the body of the connected receiving soma. When the electric potential reaches above a threshold value, the signal pulse is sent from the sending soma to the receiving soma via the axon. At this point, the receiving cell is activated or fired. Thus, it is evident that the functioning of the nervous system is mainly carried out by the neurons (Rojas, 1996).

Artificial neural networks are targeted to model the information processing capabilities of the nervous system. The initial steps in this direction were envisaged by the introduction of the simplified neuron by McCulloch and Pitts in 1943 (McCulloch 1943). An artificial neural network comprises several computing primitives interconnected together in different topologies for processing and propagation of information. In analogy with the human nervous system, these primitives are referred to as neurons or *nodes*. The synapses or junctions between these neurons are represented by connection strengths (or *weights*), which weight or modulate the incident input signals. The nonlinear processing power of these neurons is guided by a characteristic *activation/transfer* function. To be precise, the impinging weighted signals from the anterior neurons sum up to generate a neuronal impulse by means of a transformation in the posterior neuron. In this way incident input information gets processed and propagated from the preceding neurons to the following ones. Eventually, relevant features of the information are learnt by the individual neurons by means of an application specific learning algorithm, which entails the adjusting of the interconnection weights until the desired precision is attained (Haykin 1999; Kumar, 2004). This learning of information content by the neurons of an artificial neural network is referred to as the training of the neural network. So much so forth, this trained information is manifested in the adjusted weights in a neural network.

A plethora of literature (Egmont-Petersen, 2002; Huang, 2009) is available on the possible incarnations of the artificial neural network models, which have evolved through the ages. These models differ in their structures, topological connectivities, operational modes, learning algorithms, activation characterizations etc. Since the functioning of the nervous system inspires these models, a proper understanding of the biological mechanisms involved inside the nervous system is a prerequisite for designing artificial neural network models appropriate for a specific task.

The proposed chapter is targeted at providing an understanding of the essence of the neural networking paradigm with reference to the biological processes involved in the nervous system. The following section discusses the mechanisms of transmission, storage and cognition of information in the nervous system. A detailed analysis of the artificial neural networking paradigm is presented in the subsequent section with reference to the basic neuronal model, its mathematical formalism, constituent components, structure and topology.

The different operational modes of artificial neural networks are also discussed in this section with due regards to the learning algorithms employed. The next section introduces the basic model of an artificial neural network due to McCulloch and Pitts (McCulloch, 1943). The architecture and operational dynamics of five different artificial neural network models are illustrated in the next section. These include the multilayer perceptron (MLP) (Haykin, 1999), Hopfield network (Hopfield, 1982), Kohonen's self-organizing feature map (SOFM) (Kohonen, 1982), the radial basis function network (RBFN) (Buhmann, 2003) and the adaptive resonance theory (ART) (Carpenter, 1987a). Highlights of the recent advances clinched in the neural networking paradigm as well as the paradigm shift towards the *neuro-fuzzy*, *neuro-genetic*, *neuro-fuzzy-genetic*, *rough set* theoretic and *quantum* domains are discussed in the subsequent section. The application perspectives of the neural networking paradigm with reference to scientific, engineering and financial applications are touched upon in the penultimate section. Finally, the chapter concludes with a summary of the topics elucidated in this chapter.

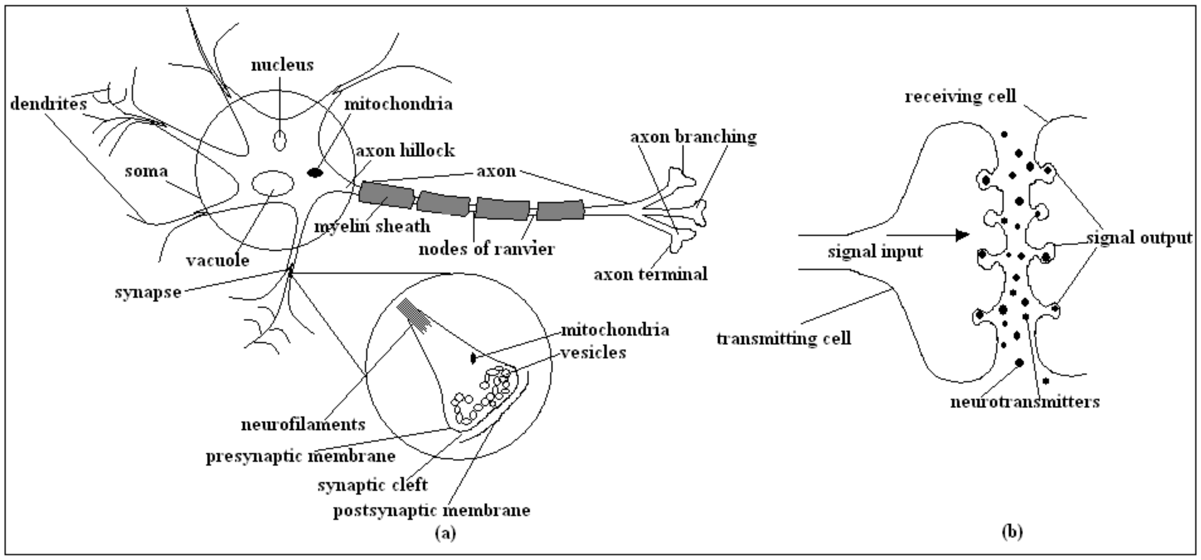
THE BIOLOGICAL NEURON: STRUCTURE AND OPERATION

As stated earlier, the neural networking paradigm is based upon the brain metaphor in that the operations of the neural networks mimic the behavior of human brain in as much as intelligent reasoning and decision-making are concerned. Hence there exists a strong analogy between the operations of the biological nervous system and the artificial neural networking paradigm. This section deals with an in-depth discussion regarding the cellular structure of biological neurons formed of selective permeable membranes, the guiding Nernst formulations (Matthews, 1991) for measurement of membrane potentials and the chemical action potentials at neuronal synapses which facilitate the storage, processing and subsequent transmission of signals.

Structure of Biological Neurons

The human nervous system is composed of similar neuronal cells or neurons, which otherwise vary in terms of functionalities and organizational complexities. Basically, there are two different kinds of nerves, viz., (i) motor nerves and (ii) sensory nerves. The motor nerves act as generators of signals/impulses to be propagated to the other nerves. The sensory nerves are the recipient nerves, which elicit responses to the signals received from the motor nerves. The neurons in the human cortex are organized in a hierarchical structure of six different layers differing in functionalities (Matthews, 1991; Brown, 1991). These layers carry out the tasks of processing and transmission of input signals. Figure 1a shows the structure of a biological neuron comprising the cell body or soma, dendrites, axon and synapses. Dendrites are the pathways for the incoming signals received from the neighboring cells via the synapses, which serve as the storehouse of information. There are three types of synapses in the nervous system, viz., (i) axon-to-axon synapses, (ii) axon-to-dendrite synapses and (iii) dendrite-to-dendrite synapses. The number of the axon-to-dendrite synapses is larger than the other two types of synapses with the axon-to-axon synapses being the smallest in number. Similar to the nerve cells, the axons are classified as afferent axons, which generate the outgoing signals and efferent axons, which receive the incoming signals. The *mitochondria* in the cell body supply energy to the cell elements and sustain the operation of the cell structure with the help of internal biochemical reactions. The myelin sheath membrane covers the axon protruding from the cell body. The uncovered parts of the axon are referred to as the nodes of Ranvier. Figure 1a also shows the *axon hillock* (the part of the axon

Figure 1. The biological neuron (a) structure and components (b) information transmission via synaptic cleft



nearest to the cell body) and the branching of the axon to transmit information to the following cells. On reception of a signal, the neurons produce an output signal or response, which is transmitted to another neighboring cell through the axon.

Transmission of Information in Biological Neurons

A cell encloses protoplasm and other cell components inside a selective permeable cell membrane composed of a double layer of molecules (Rojas, 1996). Negative and positive ions are formed in the intracellular and extra-cellular fluids in the human body out of dissociation of the dissolved salts and minerals therein. Salts like sodium chloride (NaCl) and potassium chloride (KCl) dissociate into positive ions Na^+, K^+ respectively and the negative ion Cl^- is produced in the process. The degree of permeability of the cell membrane depends among various factors on the number and size of pores in the membrane as well as on the types of these ions (Rojas, 1996). By virtue of the varying degrees of permeability, the cell membranes behave as ionic channels of varying diffusivity and act as diffusion barriers. Since the cell membranes/ionic channels are selectively permeable to the sodium, potassium or calcium ions, the specific permeability of a membrane leads to different ionic distributions in the interior and the exterior of the cells by means of diffusion of ions. The potential difference thus developed between the cell interior and exterior is referred to as the *diffusion potential*. This process of diffusion of ions occurs until a thermodynamic equilibrium is attained (Rojas, 1996). As an example, if the initial concentration of K^+ ions in the cell interior is greater than in its exterior, positive K^+ ions will diffuse out through the open potassium-selective channels in the cell. These channels would however, prevent any negative ion to diffuse out through the cell membrane, which would make the cell interior to become negatively charged with respect to the exterior. Thus, a potential difference is created between both the sides of the cell membrane. This balances the aforestated diffusion potential, thereby mitigating further diffusion of potassium ions through the cell membrane. Eventually, the net flow of the positive potassium ions from the interior to the exterior of the cell falls to zero. At this point, the biological system is said to attain a steady state (Rojas, 1996).

The potential difference E for a particular ion is given by the Nernst formula (Matthews, 1991; Rojas, 1996) as

$$E = \kappa\left(\ln \eta_o - \ln \eta_i\right) \tag{1}$$

where, η_i is the ionic concentration inside the cell, η_o is the ionic concentration in the extra-cellular fluid and κ is a proportionality constant (Matthews, 1991; Rojas, 1996). The equilibrium potential has been measured to be −80 mV for K^+ ions (Rojas, 1996).

However, the exact potential in the interior of a cell depends on the mixture of ionic concentrations. The cell's equilibrium potential lies closer to a value measured for potassium, due to the fact that the cell membrane's permeability is greater for potassium as compared to that for sodium (Rojas, 1996). This potential value leads to a net outflow of K^+ ions out of the cells and a net inflow of Na^+ ions into the cells. A self-sustained sodium ion pump inside the cell maintains the concentrations of the potassium and sodium ions on both sides of the cell membrane, thereby ensuring a polarization potential of –70 mV. This ion pump gets the requisite operational energy from a compound viz. adenosine triphosphate (ATP) produced by the mitochondria.

Neural signal information in the form of depolarization waves/spikes are transmitted into and out of a human cell by means of the diffusion of these dissociated ions across the cell membranes. These waves are also referred to as *action potentials*. These action potentials, as the name suggests, are generated by means of the formation of a potential out of a depolarization of the cell membrane (Rojas, 1996). The starting polarization potential rises from −70 mV up to +40 mV dipping to as low as −80 mV in the process (Rojas, 1996). After a switching time of 2.4 ms, the cell membrane returns back to the initial potential of –70 mV, which is referred to as the *resting potential* (Rojas, 1996).

These action potentials are responsible for producing small perturbations in the cell membrane thereby opening sodium-selective channels therein (Rojas, 1996). This phenomenon leads to the inflow of Na^+ ions into the cell. After a short interval of time, potassium-selective channels are opened enabling the outflow of K^+ ions out of the cell and preventing further inflow of Na^+ ions. This biological mechanism of opening of sodium and potassium-selective channels is propagated through the cell membrane in the form of impulses. These impulses, however, remain active as far as to perturb the next channel onwards. In this way, neural signals get transmitted across a cell modulated by these impulses (Rojas, 1996).

Storage and Processing of Signals in Biological Neurons

A synapse houses small *vacuoles* and generates the synaptic vesicles (shown in the magnified view of the synapse in Figure 1a), which contain some chemical transmitters. A small gap referred to as the synaptic cleft/gap, lies between a cell and its attached synapse. In the event of an arrival of an electric impulse these synaptic vesicles fuse with the cell membrane. Subsequently, the chemical transmitters flow into the synaptic gap and open the ionic channels as shown in Figure 1b. This allows ions to flow from the exterior to the interior of the cell thereby altering the cell's potential (Rojas, 1996). An action potential is generated if the potential in the interior of the cell raises and the synapse is then referred to as an excitatory synapse. An inhibitory synapse results if the cell's potential reduces, which prevents the generation of any action potential.

The synapses operate as the storehouse of information in neural networks. The storage mechanism can be appreciated by studying the role of NMDA (*N*-methyl *D*-aspartate) receptors (Brown, 1991; wikiNMDA; Li, 2009) in the cell membrane. The NMDA receptors (NMDAR) are ionotropic glutamate receptors, which control synaptic plasticity and memory functions of the human brain (wikiNMDA; Li, 2009). An interesting feature of these receptors is that the activations of these receptors are voltage dependent. On activation, these receptors allow specific ionic channels, which are nonselective to Na^+, $Ca^{2+,}$ or K^+ ions, to be opened (Li, 2009; Dingledine, 1999). However, when an Mg^{2+} ion is received by these receptors, these channels get blocked preventing further diffusion of Na^+ or Ca^{2+} ions. On the contrary, the NMDA receptors lose the Mg^{2+} ion and switch from the blocked state to an unblocked one when the cell is excited above a threshold level. Henceforth, ions with greater permeability flow into the cell effecting a change in the threshold level of the cell (Steward, 1989; Reichert, 1990).

Thus, it can be surmised that neurons do not transmit information only by electrical perturbations but also through chemical signaling (Rojas, 1996). Signal processing is effected in the nervous system by means of an integration of the electrical transmission of action potential induced perturbations inside the cell and the chemical transmission at the synaptic gaps between cells. Neural cells process signals by integrating several incoming signals, which are able to excite attached cells to open sufficient ionic channels. If enough channels are opened above a requisite threshold, an action potential is generated at the axon hillock. This is because the number of ionic channels is larger and the cell's threshold is lower at the axon-hillock (Thompson, 1990). Interested readers may refer to (Rojas, 1996; Matthews, 1991; Brown, 1991) for details regarding the storage, transmission and processing of information in the biological nervous system.

ARTIFICIAL NEURAL NETWORKS

Human intelligence is represented artificially by means of the artificial neural networks (ANNs) (Fausett, 1994; Haykin, 1999; Rojas, 1996), which are artificial counterparts of the biological neurons. This form of intelligence, also referred to as artificial intelligence, provides a means for representation of diverse kinds of knowledge in a knowledge base. It also forms a framework for reasoning of the underlying rules of the knowledge base and therefore induces a mechanism of learning of environmental (Hebb, 1949) data on which such an intelligent system operates. The basis of an artificial neural network would be dealt with in this section with due regards to the basic structure and components of an artificial neural network supported by a mathematical modeling and the different modes of operation.

Components, Structure and Topology

Any computational paradigm should entail basic primitives for implementing the fundamental set of functions characteristic of the paradigm and artificial neural networks are no exceptions in this regard (Rojas, 1996). Similar to the biological paradigm, the neurons/nodes act as elementary computational primitives in the neural networking paradigm. According to Kohonen (Kohonen, 1982):

Artificial neural networks are massively parallel adaptive networks of simple nonlinear computing elements called neurons, which are intended to abstract and model some of the functionality of the human nervous system in an attempt to partially capture some of its computational strengths.

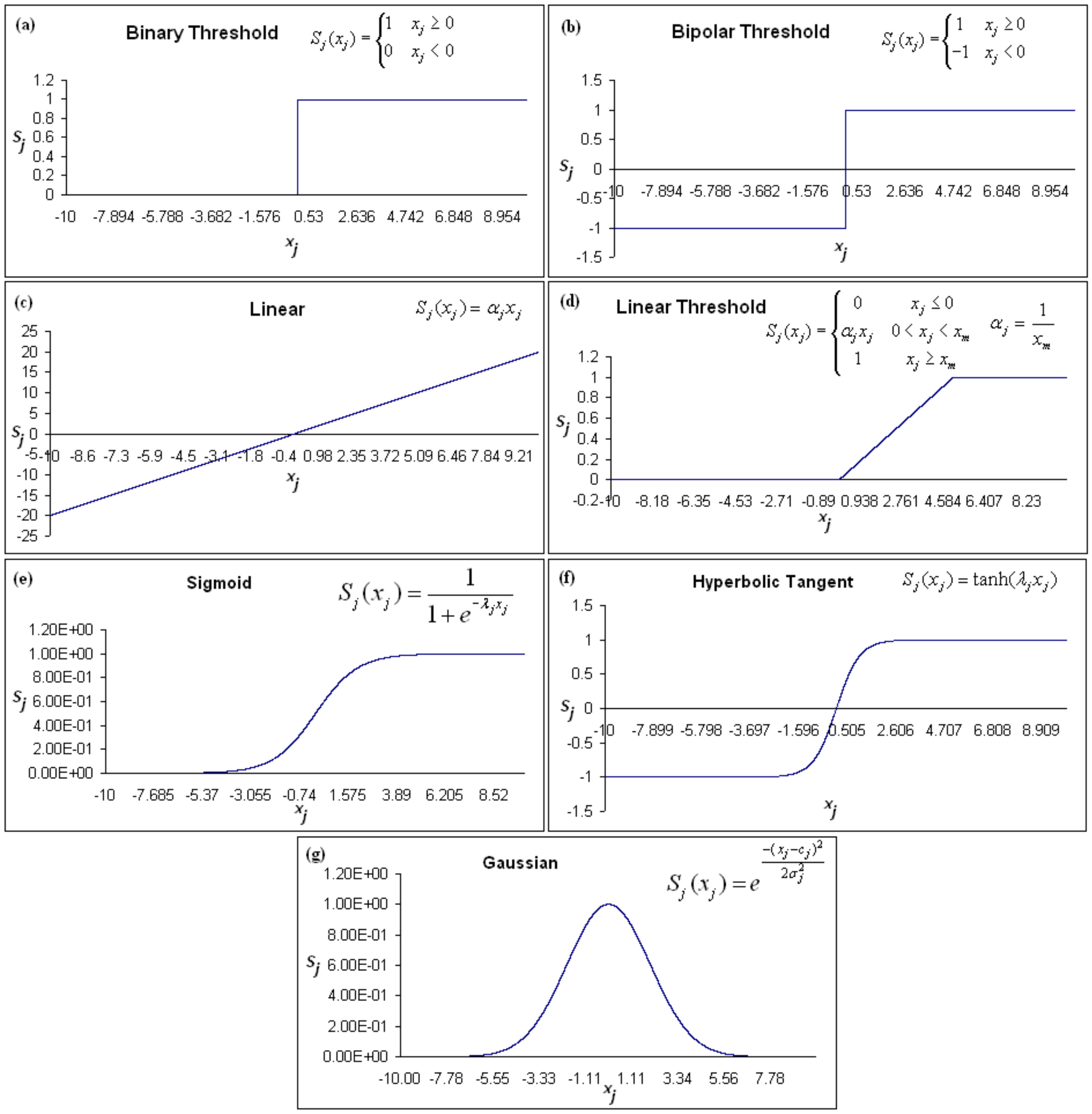
The corresponding flow of information in a neural network is achieved by the interconnection topology of the neurons. The distributed representation of the interconnections of the networks induces massive parallelism therein. Moreover, the generalization capabilities of the networks make them amenable to graceful degradation and failsafe. These network structures differ from one to another in the topology of the interconnection structure and the real life problems they are used for (Haykin, 1999; Kumar, 2004). Since the essence of neural network operation is based on the behavior of human brain, these networks require a form of training so as to acquire the learning ability of the human brain. Once these networks are trained with the different aspects of the problem at hand, viz. the input-output data relationships and distributions, they can be used to replicate the learnt relationships, given the immense generalization capabilities embedded therein. Thus the functionality of a neural network architecture depends on the nature of the characteristic functions performed by the primitive neurons, the interconnection topology and the methodology of transfer of information. A neural network generally comprises the following components (Haykin, 1999; Kumar, 2004)

- *Neurons:* As stated earlier, neurons are the basic computational elements of a neural network (Rojas, 1996; Kumar, 2004; Bhattacharyya, in press). Generally, there are three types of neurons in a neural network architecture, viz. input, hidden and output neurons. The input neurons accept real world data. The hidden neurons lie in the form of layers between the input and output neurons. These neurons operate on and process the data fed by the input neurons. The output neurons finally generate the network output responses based on the data processed by the hidden neurons.
- *Neuron activation state vector:* As stated earlier, the transmission of information through the different neurons of a human nervous system depends on the excitation/activation levels of the nerve cells or neurons. The same reasoning applies to the artificial neural networking paradigm as well. A neuron activation state vector determines the activation levels of the neurons. If there are n number of neurons, x_i, i=1, 2, …, n, the activation state vector $X \in R^n$ is represented as (Kumar, 2004; Bhattacharyya, in press).

$$X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} \tag{2}$$

- *Activation/transfer function:* This function represents the transfer characteristics of the individual neurons in a neural network in that it determines the activation behavior of the neurons to incident inputs (Kumar, 2004). The function responds to the range of incident input signals and transforms them by using a suitable learning algorithm. Though different constituent neurons may exhibit different transfer characteristics, yet most of the existing neural network architectures are field-homogeneous (Bhattacharyya, in press) with all the neurons in a layer characterized by the same activation. Commonly used activation functions include the binary threshold, sigmoid, linear threshold, or probabilistic activation functions. Figure 2 shows some of the widely used neural network activation functions and their mathematical forms.
- *Interconnection topology:* This refers to the mode of interconnections of the neurons in different layers of a neural network architecture similar to (but not so exactly complex) to the axon interconnections in biological nervous system. These interconnections act as the storage junctions/mem-

Figure 2. Commonly used neural network activation functions (a) Binary threshold (b) Bipolar threshold (c) Linear (d) Linear threshold (e) Sigmoid (f) Hyperbolic tangent (g) Gaussian



ory of the artificial neural network. Similar to the synapses of biological neurons, these interconnections are also characterized by a weight value, which ascertains their excitation/activation levels (Egmont-Petersen, 2002; Kumar, 2004; Bhattacharyya, in press).

- *Neuron firing behavior:* This component determines the processing capabilities of the neurons in a given layer given the characteristic activation functions employed therein (Kumar, 2004; Bhattacharyya, in press). The constituent neurons aggregate the incoming input signals by applying an inner product of the input vector and the neuron fan-in interconnection strength (weight) vector and impress the embedded activation function.

- *Learning algorithm:* The learning rule provides a mechanism for training a neural network with a training set of input-output pairs of data relationships (Kumar, 2004; Bhattacharyya, in press). The objective of the algorithm is to compensate the network system error by modifying the interconnection weights and to enable the network to arrive at the desired end result.

Common neural network architectures can be categorized in the following two main categories depending on the interconnection topologies.

- A *feedforward* manner where the network does not possess any loops. Representative examples include the single layer/multilayer perceptron (McCulloch, 1943; Haykin, 1999), support vector machines (SVM) (Vapnik, 1995; Vapnik, 1997; Burges, 1998), radial basis function networks (RBFN) (Buhmann, 2003) to name a few.
- A *feedback* (recurrent) manner where there exist loops due to feedback connections. Typical examples include the Hopfield network (Hopfield, 1982), brain-in-a-state-box (BSB) model (Anderson, 1977), Boltzmann machine (Ackley, 1985), bi-directional associative memories (BAM) (Kosko, 1988), adaptive resonance theory (ART) (Carpenter, 1987a), Kohonen's self-organizing feature maps (SOFM) (Kohonen, 1982) to name a few.

Mathematical Formalism

One important aspect of understanding the structure and operations of the artificial neural networks is to get an insight into the formalism and application perspectives of these networks. This section is devoted to providing a simple mathematical formalism of the neural networking paradigm so that both the structural and operational characteristics clearly stand out.

The constituent neurons in an artificial neural network are entrusted with the dual functions of accepting real world numeric inputs and processing them to yield a desired output depending on either an *a priori* knowledge base or some prototypic information regarding the system at hand. Put in simple terms, a neuron receives inputs from its preceding activated neuron/neurons, sums them up and applies a characteristic activation to generate an impulse. The summing mechanism is controlled by the interconnection weights between the receiving neuron and its preceding ones in as much they decide the activation state of the preceding neurons. The operational dynamics of any neural network architecture can thus be represented by the following mathematical formalism.

Let the n inputs $\{q_1, q_2, q_3, \ldots, q_n\}$ arriving at a receiving neuron from n preceding neurons, lie in a vector space Q^n. If the interconnection weight matrix from m such receiving neurons to their n preceding neurons is represented by $W_{m\times n}=\{w_{ij}, i=1, 2, 3, \ldots, n; j=1, 2, 3, \ldots, m\}$, then the summing mechanism in a neuron simplifies to a matrix-vector multiplication only, i.e.,

$$O = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdot & \cdot & w_{1n} \\ w_{21} & w_{22} & w_{23} & \cdot & \cdot & w_{2n} \\ w_{31} & w_{32} & w_{33} & \cdot & \cdot & w_{3n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ w_{m1} & w_{m2} & w_{m3} & \cdot & \cdot & w_{mn} \end{bmatrix} \times \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \cdot \\ \cdot \\ q_n \end{bmatrix} \quad (3)$$

which is simply a vector given by

$$O = \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ \cdot \\ \cdot \\ o_m \end{bmatrix} \tag{4}$$

Thus, each j^{th}, j=1, 2, 3, …, m, neuron is fed with an element of the vector $[o_1 o_2 o_3 \cdot \cdot o_m]^T$ resulting out of the summing mechanism. As a part of the processing task, each such j^{th}, j=1, 2, 3, …, m, neuron then applies the corresponding activation/transfer function (f) to the elements of this vector to yield the corresponding output response Y_j, j=1, 2, 3, …, m, as

$$Y_j = f(\begin{bmatrix} o_1 & o_2 & o_3 & \cdot & \cdot & o_m \end{bmatrix}^T) \tag{5}$$

Thus, given a set of inputs in Q^n, the interconnection weight matrix $W_{m\times n}$={w_{ij}, i=1, 2, 3, …, n; j=1, 2, 3, …, m} along with the characteristic activation function f decide the output responses of the neurons. Any variations in $W_{m\times n}$={w_{ij}, i=1, 2, 3, …, n; j=1, 2, 3, …, m} would yield different responses subject to the same characteristic neuronal activation f. Looking from the biological perspective, the interconnection weight matrix $W_{m\times n}$={w_{ij}, i=1, 2, 3, …, n; j=1, 2, 3, …, m} thus serves similar to the synaptic weights of a biological neuron and house the memory of an artificial neural network. Furthermore, for achieving a stable memory, which can be effective in recalling/recognizing test inputs, the interconnection weight matrix must be adapted in tune with the neuronal activation state vectors through a proper learning algorithm. Once the matrix achieves a stable configuration, a neural network is able to successfully recognize test inputs in the testing phase of the network learning procedure. The output responses $Y_j^{'}$, j=1, 2, 3, …, m, thus obtained corresponding to the test patterns, can also be represented in matrix-vector product formalism as

$$Y_j^{'} = f\left(\begin{bmatrix} w'_{11} & w'_{12} & w'_{13} & \cdot & \cdot & w'_{1n} \\ w'_{21} & w'_{22} & w'_{23} & \cdot & \cdot & w'_{2n} \\ w'_{31} & w'_{32} & w'_{33} & \cdot & \cdot & w'_{3n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ w'_{m1} & w'_{m2} & w'_{m3} & \cdot & \cdot & w'_{mn} \end{bmatrix} \times \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \cdot \\ \cdot \\ t_n \end{bmatrix}\right) \tag{6}$$

where, T={t_1, t_2, t_3, …, t_n}^{T} is the test input vector and $W'_{m\times n}$={w'_{ij}, i=1, 2, 3, …, n; j=1, 2, 3, …, m} is the stabilized interconnection weight matrix.

Operational Modes

The dynamics and operation of a neural network architecture depends among other factors on the nature of the learning mechanism employed (Haykin, 1999; Kumar, 2004; Bhattacharyya, in press). The operational modes of neural network architectures can be classified into two main categories viz., *supervised* mode and *unsupervised* mode.

- *Supervised mode of operation*: As the name suggests, the supervised mode of operation of a neural network architecture implies the use of some *a priori* knowledge base to guide the learning phase. This knowledge base puts forward to the neural network a training set of input-output patterns and an input-output relationship. The neural network is then supervised to embed an approximation function in its operational behavior, which replicates the relationship of the input-output training data vector presented to the network (Haykin, 1999). If an input vector $X_k \in R^n$ is related to an output vector $D_k \in R^p$ by means of a relationship $D_k = T\{X_k,\}$, the basic philosophy behind the supervised learning algorithm is to yield a mapping of the form f: $R^n \Rightarrow R^p$, in consonance with T such that the input-output data relationship is aptly replicated in the process (Kumar, 2004; Bhattacharyya, in press). To achieve this, the supervised algorithm aims to reduce the error ($E=D_k-S_k$), where S_k is the output response generated by X_k. This error reduction mechanism is effected by a technique referred to as the *gradient descent* method (Kumar, 2004; Bhattacharyya, in press; Snyman, 2005). Obviously, when the error E gets reduced to an acceptable limit, if the network is fed with an input similar to X_k, it would generate a response similar to D_k (Kumar, 2004; Bhattacharyya, in press). A network behaving in this fashion is said to be operating in a supervised mode.

- *Unsupervised mode of operation*: There are different schools of thoughts regarding this mode of operation of neural networks. According to one theory, this is an adaptive learning paradigm, which presents the neural network with an input X_k and allows it to self-organize the topological configuration depending on the distribution of input data by means of generation of exemplars/prototypes of input vectors presented. Thus, the algorithm stores the exemplars/prototypes of the input data set into different number of predefined categories/classes (Haykin, 1999; Kumar, 2004; Bhattacharyya, in press). When an unknown input data is presented to the network, it matches it to the stored prototypes and associates it to one of the stored categories. However, if the input data belongs to a completely new category/class, the storage states of the network are updated to incorporate the new class encountered. Another school of thought however, differs on the definition of the term "unsupervised" in the sense that suitable classical and soft computing approaches are employed to determine the number of classes to be assigned the network rather than using a predefined number of classes.

Apart from these two modes of operation, neural network architectures also employ the reinforcement learning mode (Kaelbling, 1996), semi-supervised mode (Chapelle, 2006) and transduction mode (Vapnik, 1998) of operation.

The Basic Neuron

As stated earlier, a neuron in a neural network architecture receives inputs from some other neurons, or the external world via coupling of synaptic weights. The neuron adds up its inputs and produces an output after comparing the sum to a threshold value (Bhattacharyya, in press). Based upon this basic functionality, Frank Rosenblatt (Bhattacharyya, in press, Rosenblatt, 1958) proposed the *single layer perceptron* model with the following features:

- The output of the neuron depends on its inputs and the synaptic weights which couple the inputs to the network neurons. This output would remain either on (activated or fired) or off (deactivated) depending on the activation level, which again depends on the strengths of the impinging inputs and weights (Bhattacharyya, in press, Rosenblatt, 1958).
- A certain number of inputs to a particular neuron must be high at any point of time in order to make that neuron fire. The coupling efficiency of the synapses is aptly represented by a multiplicative factor on each of the inputs to the neuron. This multiplicative factor often referred to as the interconnection weight w, can be modified so as to model synaptic learning (Bhattacharyya, in press, Rosenblatt,

1958). A more efficient synapse having a larger weight transmits more of the signal, while a weak synapse has a smaller weight. The output of a neuron, in turn, serves as an input to other units. This process of propagation of incident information to the network continues hierarchically through the different interconnected neurons of the network structure.

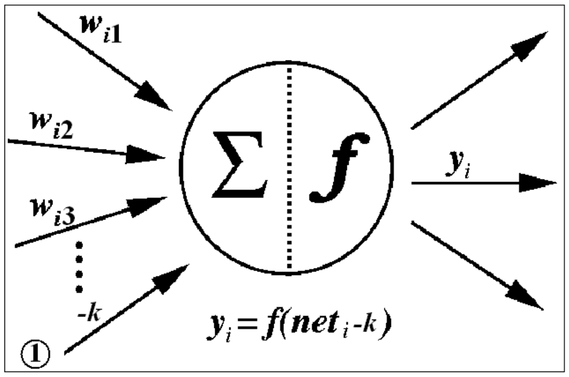
McCulloch and Pitts (McCulloch, 1943) devised the first model of the basic neuron as shown in Figure 3. It comprises an input layer and an output layer. Processed information propagates from the input layer to the output layer via the unidirectional interconnections between the input and output layer neurons. Each of these interconnections possesses a weight w, which indicates its strength. However, there is no interconnection from the output layer back to the input layer the other way round as is evident from the figure.

If m such input layer neurons are connected to n output layer neurons, then the input-output layer interconnections are represented by

$$W_{m\times n} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdot & \cdot & w_{1n} \\ w_{21} & w_{22} & w_{23} & \cdot & \cdot & w_{2n} \\ w_{31} & w_{32} & w_{33} & \cdot & \cdot & w_{3n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ w_{m1} & w_{m2} & w_{m3} & \cdot & \cdot & w_{mn} \end{bmatrix} \tag{7}$$

Apart from the m input layer neurons, an additional input of 1 (encircled) is also shown in Figure 3. This is referred to as the bias input to the neural network. The corresponding interconnection weight is $-k$, the bias/offset to the network. Hence, there are actually $m+1$ input lines, i.e., $j=1, 2, 3, \ldots, m+1$. Though this bias input is generally not shown in neural network diagrams, it has been shown in Figure 3 for the sake of self-sufficiency.

Figure 3. McCulloch and Pitts model of a basic neuron



The ith output neuron first computes the weighted sum of the m number of inputs (excluding the bias/offset) arriving from m interconnected input layer neurons as (Bhattacharyya, in press; McCulloch, 1943).

$$y_i = \sum_{j=1}^{n} w_{ji} y_j = net_i \tag{8}$$

where w_{ji} refers to the interconnection weight from jth input layer neuron to the ith output layer neuron. Taking the bias term $-k$ into consideration the output becomes (Bhattacharyya, in press; McCulloch, 1943).

$$y_i = \sum_{j=1}^{n} w_{ji} y_j - k = net_i - k \tag{9}$$

Subsequently, the neuron applies its activation function f, which performs a thresholding operation on the computed sum in equation 9. The output y_i of the ith neuron can then be written as (Bhattacharyya, in press; McCulloch, 1943)

$$y_i = f(\sum_{j=1}^{n} w_{ji} y_j - k) = f(net_i - k) \tag{10}$$

The thresholding process ensures that if the sum is greater than a threshold value defined by the function (*f*), then the neuron produces 1 else it produces 0. This output value signifies the status of the neuron, i.e. whether it is fired or not.

Learning Algorithm of the Network

The single layer perceptron operates in a supervised mode. Hence, it requires learning/training by means of some *a priori* knowledge base for its operation. This learning mechanism for the single layer perceptron comprises the following phases (Bhattacharyya, in press; McCulloch, 1943).

- *Input phase*: In this phase, the training set inputs are presented to the input layer of the network. The input-to-output layer interconnection weights are set to small random values. The activation threshold for the transfer function employed in the output layer neuron is also set randomly.
- *Input propagation phase*: The inputs to the input layer neurons are multiplied by the corresponding interconnection weights and the resultant products are summed up to obtain the input to the lone neuron in the output layer of the network.
- *Output generation phase*: The characteristic activation function (*f*) is applied by the output layer neuron to the input information to obtain its output response (*y*).
- *Error estimation phase*: In this phase, the output response (*y)* obtained in the previous phase is compared to the desired output (*d*) to estimate the network error (*E*).
- *Weight adjustment phase*: The computed error (*E*) is used to iteratively adjust the initially set random input-to-output layer interconnection weights so as to reduce the gradient of the error with time, i.e. to minimize the computed error. This process is referred to as *gradient descent* (Haykin, 1999; Snyman, 2005), whereby the network is forced to take correct decisions and abandon incorrect decisions.

Since E is obtained by the assimilation of all the individual error terms at the output layer neurons through the interconnection weights w_{ji}, j=1, 2, 3, …, m; i=1, 2, 3, …, n, the gradient ∇E of the error term is given by

$$\nabla E = \frac{\partial E}{\partial w_{ji}} \tag{11}$$

The weight adjustment term is then given by (Haykin, 1999; Snyman, 2005; Kumar, 2004; Rojas, 1996)

$$\Delta w_{ji} = -\mu \frac{\partial E}{\partial w_{ji}} \tag{12}$$

where, μ is the learning rate. μ defines the step length of weight adjustment in the gradient descent algorithm. This learning algorithm is repeated with the adjusted interconnection weights and training set inputs until the network converges, i.e. the network error falls down under a tolerable limit specified by the supervisor.

Computation of the Error Gradient

As stated earlier, the error minimization procedure during the training of a neural network involves the computation of the gradient of the error function with time (Haykin, 1999; Snyman, 2005). In fact, the gradient G_E of the network error E is computed with respect to each interconnection weight w_{ji} so as to decipher the contribution of the initially set random interconnection weights towards the overall E. The network error is represented as

$$E = \frac{1}{2}\sum_{j=1}^{N}(y_j - d_j)^2 \tag{13}$$

where, *N* is the number of neurons in the output layer of the network. The square of the error term is taken since it will always be positive. Moreover, it will be greater if the difference (*y* – *d*) is large and lesser if it is small.

For the *i*th output layer neuron, the error becomes (Haykin, 1999; Bhattacharyya, in press; Snyman, 2005).

$$E_i = \frac{1}{2}(y_i - d_i)^2 \tag{14}$$

Now, the gradient G_{E_i} is given by

$$G_{E_i} = \frac{\partial E_i}{\partial w_{ji}} \tag{15}$$

Using chain rule, one gets

$$\frac{\partial E_i}{\partial w_{ji}} = \frac{\partial E_i}{\partial y_i}\frac{\partial y_i}{\partial w_{ji}} \tag{16}$$

Since $E_i = \frac{1}{2}(y_i - d_i)^2$ (from equation 14), the first term of equation 16 is given by

$$\frac{\partial E_i}{\partial y_i} = (y_i - d_i) \tag{17}$$

Using $y_i = \sum_j w_{ji} y_j$, the second term of equation 16 becomes

$$\frac{\partial y_i}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\sum_j w_{ji} y_j = y_i \tag{18}$$

Hence, equation 16 becomes

$$\frac{\partial E_i}{\partial w_{ji}} = (y_i - d_i) y_i \tag{19}$$

Therefore, using equation 12, the interconnection weight w_{ji} between the *j*th input layer neuron and the *i*th output layer neuron is adjusted as (Haykin, 1999; Bhattacharyya, in press; Snyman, 2005).

$$\begin{aligned} w_{ji} &= w_{ji} - \Delta w_{ji} = w_{ji} - (-\mu \frac{\partial E_i}{\partial w_{ji}}) \\ &= w_{ji} + \mu(y_i - d_i) y_i \end{aligned} \tag{20}$$

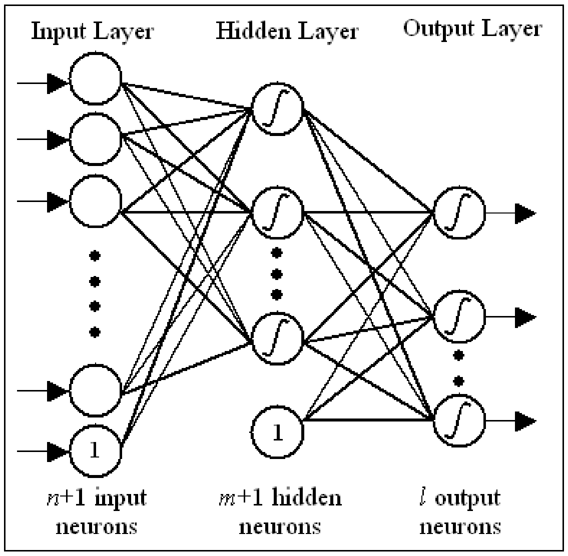
It may be noted that if μ is too small, the algorithm takes a longer time to converge, while for a larger value of μ, the algorithm diverges.

The single layer perceptron is efficient in classifying linearly separable data sets (Haykin, 1999; Kumar, 2004). However, it suffers from several limitations. It is not a good approximator. Moreover, it fails to classify in situations where the divisions between the classes are nonlinear and complex.

ARTIFICIAL NEURAL NETWORK ARCHITECTURES

Researchers and scientists have proposed several neural network architectures (Rojas, 1996; Egmont-Peterson, 2002; Kumar, 2004) differing in topology and operational characteristics so as to capture the essence of neural computation. These architectures have been evolved keeping in mind the operational behavior of the human brain. This section would introduce some of the important artificial neural network models, which are efficient in classification and recognition problems. The architectures discussed here fall into either of the two categories of network models, viz. supervised and unsupervised depending on the learning methodology adopted.

Figure 4. A three-layer neural network architecture



Multilayer Perceptron

As mentioned before, a single layer perceptron can only classify/discriminate linearly separable data (Haykin, 1999; Kumar, 2004). This limitation can be attributed to its solo output layer, which cannot incorporate nonlinearity prevalent in real life data. This limitation can be addressed by introducing additional layers (also referred to as hidden layers) between the input and output layers of the perceptron so as to incorporate some information regarding nonlinearly separable inputs. These multilayer neural network architectures therefore comprise several hidden layers between the input and output layers. The most commonly used multilayer perceptron architecture (Haykin, 1999) comprises three layers of nodes, viz., input, hidden and output layers. Such a three-layer architecture is shown in Figure 4 comprising n+1 (n real world data input layer neurons and one bias input neuron of 1 [encircled in the figure]) input neurons, m+1 (m hidden layer neurons for accepting inputs from n+1 input layer neurons and one bias input neuron of 1 [encircled in the figure]) hidden neurons and l output neurons. All the n+1 input layer neurons are connected to m hidden layer neurons (excluding the hidden neuron for providing the bias input) by means of input-to-hidden layer interconnections w_{ij}, i=1, 2, 3, …, n+1; j=1, 2, 3, …, m. Similarly, all the m+1 hidden layer neurons are connected to l output layer neurons by means of hidden-to-output layer interconnections w_{jk}, j=1, 2, 3, …, m+1; k=1, 2, 3, …, l. It may be noted that since the (n+1)th input layer neuron and the (m+1)th hidden layer neuron serve as biases/offsets for the respective layers, the respective interconnection weights w_{n+1j}, j=1, 2, 3, …, m and w_{m+1k}, k=1, 2, 3, …, l, are both equal to a threshold value of $-\theta$.

It is evident from the architecture of Figure 4 that all the hidden layer neurons excepting the (m+1)th neuron and all the output layer neurons exhibit nonlinear transfer characteristics shown by the nonlinear functions inside each neuron. Moreover, since the hidden layer neurons contribute to the inputs for the output layer neurons, it is not possible to know the target values for the hidden layers. Hence, the gradient descent algorithm (Snyman, 2005), which depends on the estimation of errors at the output layer neurons only, is not sufficient to train these types of networks with hidden layers. This is because of the fact that only a computation of the partial derivative of the error function at the output layer (as carried out in the gradient descent algorithm) with respect to each weight does not enable one to derive the errors at the hidden layers. Moreover, one cannot anticipate the direction of change of the network error in the gradient descent method as well. But, it is required to estimate the errors and their derivatives at each network layer. On the contrary, if the negatives of these derivatives are computed and added to the weights, then the overall system error would decrease until it reaches a local minimum. This method would also ascertain as to whether the errors are increasing when the weights are increasing, since then it would yield positive values for the derivatives. In case of a multilayer architecture with multiple hidden layers, this process of computing the partial derivatives and applying them to each of the weights can be initiated with the hidden-output layer weights and then proceeding to the preceding hidden layers to carry out the computation for the remaining hidden-hidden layer weights (for more than one hidden layers) and finally ending up with the input-hidden layer weights. This also implies that a change in a particular layer-to-layer set of weights requires that the partial derivatives in the preceding layer are also known at place. This algorithm of adjusting the weights of the different network layers (Rumelhart, 1986; Chauvin, 1995) starting from the output layer and proceeding downstream is referred to as the backpropagation algorithm (Rumelhart, 1986; Haykin, 1999; Rojas, 1996). It is readily used for overcoming the shortcomings of the gradient descent algorithm in dealing with the error compensation and weight adjustment procedure in higher layered neural network architectures.

Backpropagation Algorithm

In the backpropagation algorithm (Rumelhart, 1986; Haykin, 1999; Rojas, 1996), the outputs are computed in a multilayer neural network architecture after the inputs are presented to it. Subsequently, the network system error is calculated using supervised learning by comparing the obtained outputs with the expected ones. This training of the network with the expected outputs starts with random weights assigned to the network layer interconnections. The backpropagation algorithm aims to adjust the interconnection weights until the system error is minimized.

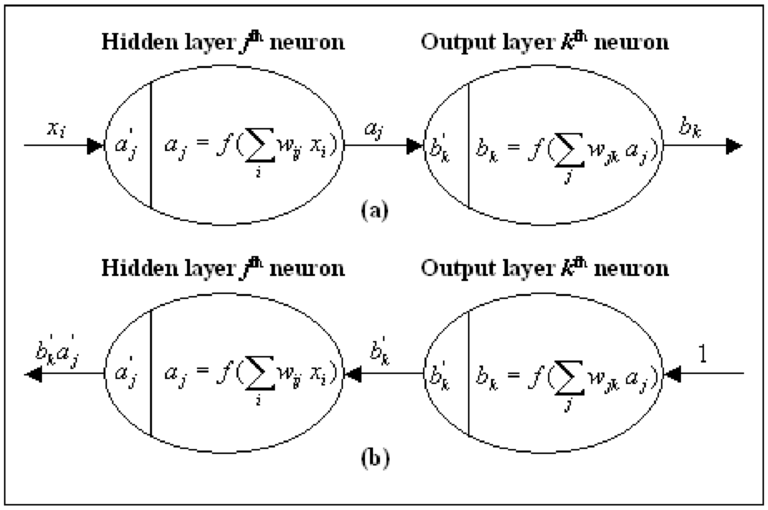
A good treatise on the backpropagation algorithm is provided in (Rojas, 1996). In the treatment, Rojas elucidated the computation of weight adjustments for a multilayer feedforward neural network architecture comprising any number of hidden layers with the neurons activated with sigmoidal activation functions by presenting an in-depth analysis of the functional primitives at each constituent neuron of the network. A summary of the concepts presented by Rojas in (Rojas, 1996) is presented below with reference to an example three-layer neural network architecture for a better understanding and analysis of the algorithm.

As stated earlier, each neuron, considered as an intelligent computational primitive (Rojas, 1996), essentially carries out the dual tasks of summation of incoming inputs and the application of the transfer function, thereby producing the output. Moreover, each neuron also computes two other quantities considered essential for the operation of the network, viz., (i) an error term due to the difference between the expected and obtained outputs and (ii) the derivative of the error term. The error term and its derivative are required for the computation of the overall network system error and the adjustment of the interconnection weights between the different network layers by the backpropagation algorithm.

Figure 5 shows the computational, storage and propagation characteristics of the individual neurons of the example three-layer neural network architecture, redrawn with the concepts presented in (Rojas, 1996).

Figure 5a illustrates the forward propagation of inputs from the input layer to the output layer

Figure 5. Computational, storage and propagation characteristics of a three-layer neural network architecture


Hidden layer j^{th} neuron
Output layer k^{th} neuron
x_i
a'_j
$a_j = f(\sum_i w_{ij} x_i)$
a_j
b'_k
$b_k = f(\sum_j w_{jk} a_j)$
b_k
(a)
Hidden layer j^{th} neuron
Output layer k^{th} neuron
$b'_k a'_j$
a'_j
$a_j = f(\sum_i w_{ij} x_i)$
b'_k
b'_k
$b_k = f(\sum_j w_{jk} a_j)$
1
(b)

of the network via its hidden layer neurons. Only a single hidden layer and a single output layer neuron are shown for the sake of simplicity. The j^{th} hidden layer neuron receives inputs x_i, i=1, 2, 3, …, n from the input layer neurons connected to it through the input-to-hidden layer interconnection weights w_{ij}, i=1, 2, 3, …, n; j=1, 2, 3, …, m. It also receives another constant input of 1 via an interconnection weight of $-\theta$ (the bias/offset) as shown in Figure 4. The j^{th} hidden layer neuron computes the sum of the weighted inputs and applies the characteristic activation function f to yield the output a_j. In addition, the j^{th} hidden layer neuron computes the derivative of the obtained output a_j, i.e. a'_j (Rojas, 1996). Finally, the neuron stores both a_j and a'_j (Rojas, 1996). a_j is propagated to the following output layer neurons by means of the hidden-to-output interconnection weights w_{jk}, j=1, 2, 3, …, m, k=1, 2, 3, …, l in the feedforward step along with an additional constant bias input of 1 with an interconnection weight of $-\theta$ (the bias/offset) as shown in Figure 4. Similarly, the k^{th} output layer neuron computes the weighted summation of a_j, j=1, 2, 3, …, m and the bias/offset and applies the same activation function f (considering the architecture to be field homogeneous) to yield the output b_k (Rojas, 1996). Subsequently, the k^{th} output layer neuron computes b'_k from b_k and stores both b_k and b'_k (Rojas, 1996). Each such b_k, k=1, 2, 3, …, l forms the outputs at the output layer neurons of the three-layer neural network architecture.

As stated earlier, the essence of the backpropagation algorithm lies in the adjustments of interconnection weights in the backward direction starting from the output layer backwards until the input layer is reached. This is similar to propagating information in the backward direction through the neural network architecture.

Rojas (Rojas, 1996) also illustrated the transfer characteristics of the computational primitives when information is fed into the neurons in the backward direction as well. Figure 5b shows the mechanism of transfer of information from the output layer backward to the input layer of the network. Taking 1 (the neuron status when it is fired or activated) as the backward input to the k^{th} output layer neuron, the backward direction output of this output layer neuron is simply the product of this backward input of 1 and the previously stored derivative of b_k, i.e. $1\times b_k^{'} = b_k^{'}$ (Rojas, 1996). Similarly, the backward direction output of the j^{th} hidden layer neuron is its backward input $b_k^{'}$ multiplied by the stored derivative term $a_j^{'}$, i.e. $b_k^{'}\times a_j^{'}$ (Rojas, 1996). With these computational and propagation characteristics of the individual neurons of the example three-layer architecture in the background, the network system error E can be evaluated and corresponding weight adjustments can be carried out using the backpropagation algorithm.

In general, the output b_k of a particular k^{th} output layer neuron due to the forward inputs a_j, j=1, 2, 3, …, m and the bias/offset from the m+1 interconnected neurons of the preceding hidden layer depends on a_j, w_{jk} and the impressed activation function f. With a sigmoidal activation function, the output produced by each such neuron is given by

$$b_k = \frac{1}{1+e^{-\lambda(\sum_{j=1}^{m} a_j w_{jk} - \theta)}} \tag{21}$$

where, λ controls the slope of the function. The obvious advantages offered by the use of a sigmoidal activation function are its continuity, asymptotic behavior and differentiability. Moreover, the sigmoidal activation function generates a response closer to 1 for larger positive numbers, a response of 0.5 at zero, and a response closer to zero for larger negative numbers. This feature sharply demarcates the lower and higher states of a neuron. Hence, the derivative of this output is given by

$$b_k^{'} = \frac{db_k}{da_j} = \frac{d\left[\frac{1}{1+e^{-\lambda(\sum_{i=1}^{m} a_j w_{jk} - \theta)}}\right]}{da_j} = b_k(1-b_k) \tag{22}$$

The error function E_k at the k^{th} output layer neuron with an expected output d_k for a particular pattern presented to the network is then given by equation 14 as

$$E_k = \frac{1}{2}[b_k - d_k]^2 \tag{23}$$

The error of the network will simply be the sum of the errors of all the neurons in the output layer taken over all the p patterns presented to the network. It is thus given by

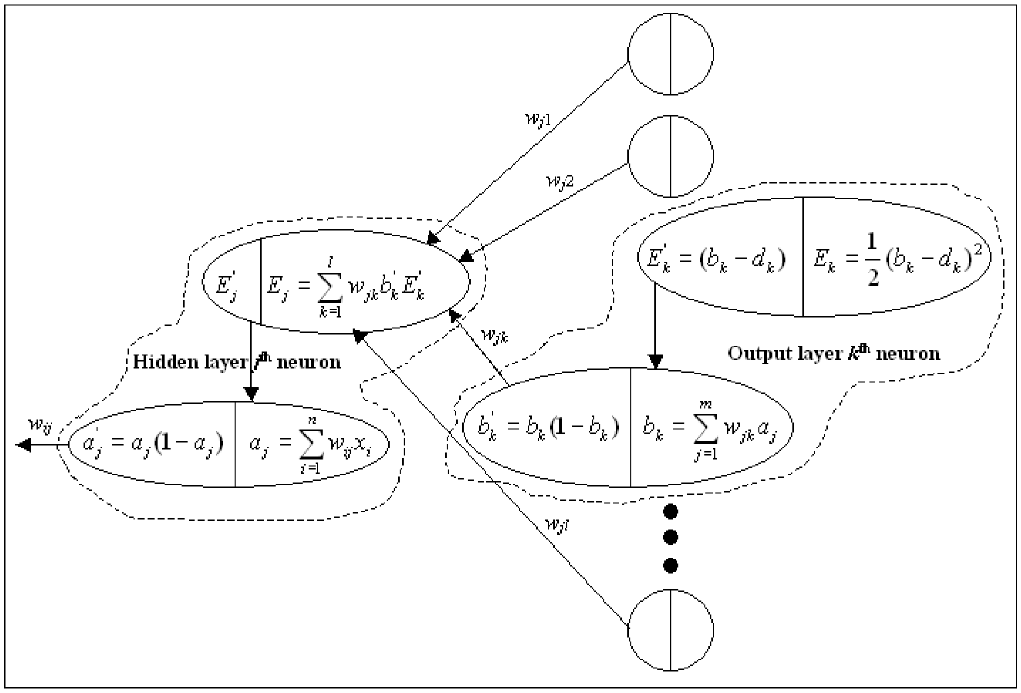
$$E = \frac{1}{2}\sum_p \sum_{k=1}^{l} E_k^p = \frac{1}{2}\sum_p \sum_{k=1}^{l} (b_k - d_k)^2 \tag{24}$$

Once the network system error is calculated, the adjustment of the interconnection weight w_{jk} between the k^{th} neuron of the output layer and the j^{th} connected neuron of the preceding hidden layer can be obtained through the gradient descent algorithm using equation 12. Subsequently the backpropagation algorithm can be used to determine the errors at the hidden layer neurons and adjust the intermediate layer interconnection weights.

The computational, storage and backpropagation characteristics of the individual neurons of the architecture of Figure 5 are redrawn in Figure 6 with regards to the network system error and all other information stored therein. The storage characteristics of the neurons have been generalized to incorporate the error and its derivative components as well. Moreover, Figure 6 also extends the simple architecture of Figure 5 by showing more than one output layer neuron along with their interconnection weights with the j^{th} hidden layer neuron.

From Figure 6 it is seen that the k^{th} output layer neuron (shown by dotted lines) can be regarded as a collection of two connected computational/storage parts, with one part housing the output b_k and its derivative $b_k^{'} = b_k(1-b_k)$ while

Figure 6. Computational, storage and backpropagation characteristics of a three-layer neural network architecture with error components (the arrowheads are for representative purposes only)


w_{j1}
w_{j2}
E_j'
$E_j = \sum_{k=1}^{l} w_{jk} b_k' E_k'$
$E_k' = (b_k - d_k)$
$E_k = \frac{1}{2}(b_k - d_k)^2$
w_{jk}
Hidden layer j^{th} neuron
Output layer k^{th} neuron
w_{ij}
$a_j' = a_j(1-a_j)$
$a_j = \sum_{i=1}^{n} w_{ij} x_i$
$b_k' = b_k(1-b_k)$
$b_k = \sum_{j=1}^{m} w_{jk} a_j$
w_{jl}

the other part houses the error term E_k, its derivative $E_k' = (b_k - d_k)$. Similar reasoning applies for all other output and hidden layer neurons. Following the propagation characteristics of the neurons as depicted in Figure 5, E_k' from one part of the kth output layer neuron is propagated to its other part containing b_k and b_k' (Rojas, 1996). Hence this kth output layer neuron computes the product of E_k' and b_k', i.e.,

$$\delta_k = E_k' \times b_k' = b_k(1-b_k)(b_k - d_k),$$

which represents the kth component of the error to be backpropagated to the connected hidden layer neurons. This backpropagated error term arises from the summed up product of two components (i) the forward direction propagated outputs a_j, j=1, 2, 3, …, m and the bias/offset of connected hidden layer neurons and (ii) the hidden-to-output layer interconnection weights w_{jk}. Hence, the back-propagated error (δ_k) is given by the gradient of E with respect to both a_j and w_{jk}, i.e. (Rojas, 1996)

$$\delta_k = \frac{\partial E}{\partial a_j w_{jk}} \tag{25}$$

Since, a_j is a processed training set input, it may be treated as a constant for all practical purposes. Hence,

$$\begin{aligned} \frac{\partial E}{\partial w_{jk}} &= a_j \frac{\partial E}{\partial a_j w_{jk}} = a_j \delta_k \\ &= a_j b_k (1-b_k)(b_k - d_k) \end{aligned} \tag{26}$$

What remains now is to compute the back-propagated error terms at the hidden layer neurons. From Figure 6, it is seen that the jth hidden layer neuron is connected to l output layer neurons by means of interconnection weights w_{jk}, k=1, 2, 3,

…, l. According to the propagation characteristics of the neurons, the backpropagation error δ_j at the jth hidden layer neuron is thus composed of a product of the collection of l number of error terms δ_k, k=1, 2, 3, …, l backpropagated from l output layer neurons multiplied by the corresponding hidden-to-output layer interconnections and the derivative of its forward direction output $a_j^{'} = a_j(1 - a_j)$ (from equation 22). It is given by (Rojas, 1996)

$$\begin{aligned}\delta_j &= a_j(1 - a_j)\sum_{k=1}^{l} w_{jk}\delta_k \\ &= a_j(1 - a_j)\sum_{k=1}^{l} w_{jk}b_k(1 - b_k)(b_k - d_k)\end{aligned} \tag{27}$$

Therefore the gradient of E with respect to the input-to-hidden layer interconnection weights w_{ij} is given by (Rojas, 1996)

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}} &= x_i\delta_j \\ &= x_i a_j(1 - a_j)\sum_{k=1}^{l} w_{jk}b_k(1 - b_k)(b_k - d_k)\end{aligned} \tag{28}$$

where, x_i is the input propagated from the ith input layer neuron to the jth hidden layer neuron.

These gradients of E are finally used to adjust the hidden-to-output layer interconnection weights w_{jk} and the input-to-hidden interconnection weights w_{ij} using equation 20. The respective weight adjustment expressions are given by (Rojas, 1996)

$$\begin{aligned}&w_{jk} = w_{jk} + \mu a_{jk}b_k(1 - b_k)(b_k - d_k); \\ &j = 1,2,3,\ldots,m+1; k = 1,2,3,\ldots,l\end{aligned} \tag{29}$$

and

$$\begin{aligned}&w_{ij} = w_{ij} + \mu x_i a_j(1 - a_j)\sum_{k=1}^{l} w_{jk}b_k(1 - b_k)(b_k - d_k), \\ &i = 1,2,3,\ldots,n+1; j = 1,2,3,\ldots,m\end{aligned} \tag{30}$$

These updated weights are used to compute the outputs of the network in the next epoch with fresh sets of input-output pairs and the network system error is again determined. The computed errors in the next epoch are similarly used to adjust the hidden-to-output and input-to-hidden layer interconnection weights using the backpropagation algorithm (Rumelhart, 1986; Haykin, 1999; Rojas, 1996). This procedure is continued until the network converges or the system error gets reduced to below some tolerable limits set by the supervisor. At this point, the network can be used for recognition/classification tasks on incomplete, corrupted or unknown inputs.

For another additional hidden layer, the same calculation would proceed to yield the error for the corresponding layer. However, it may be noted that the backpropagation algorithm is a time-complex procedure since the time for training a network grows exponentially for each additional layer.

Of late, several refinement strategies to the backpropagation algorithm have been proposed, to achieve a faster learning rate. Suitable choices of the initial weight selection mechanism, maximization of derivatives and the backpropagated errors at the neurons and avoidance of the computation of the nonlinear activation functions at the different network layers by use of look-up tables are few of the refinements, which can speed up the convergence of the backpropagation algorithm. Another significant approach in this direction is the data preconditioning technique by Silva and Almeida (Almeida, 1991). The proposed adaptive data decorrelation method (Almeida, 1991) brings about an acceleration of the convergence speed of the backpropagation algorithm. A variant backpropagation algorithm through time has been proposed for time lag recurrent networks (TLRN) (Mandic, 2001), which has lesser time complexity. Silva and Almeida (Silva, 1990) also proposed an adaptive step algorithm, which runs with different learning rates for each weight in a network thereby bringing about an exponential increase in the convergence rate. The delta-bar-delta algorithm (Jacobs, 1988) is another adaptive step algorithm to speed up the backpropagation algorithm. In (Gerke, 1997), Gerke and Hoyer applied fuzzy set (Zadeh, 1965) theoretic concepts to accelerate the speed of convergence of the backpropagation algorithm. Since the backpropagation algorithm often gets stuck to a local minimum, efforts (Wang, 2004; Bi, 2005) have been made to address this problem as well. Other contributions in this direction include the gradient descent with momentum and variable learning rate (GDMV) (Hagan, 1996), resilient backpropagation (RPROP) (Riedmiller, 1993) by Riedmiller and Braun, the conjugate gradient (CG) algorithm (Charalambous, 1992), Levenberg–Marquadrt (LM) method (Hagan, 1994), the adaptive momentum algorithm (Yu, 1993) and the dynamic adaption algorithm (Salomon, 1992), which relies on a global learning rate. Higher order algorithms have also been devised to reach a faster convergence. These include the QuickProp (Pfister, 1993; Pfister, 1995), QRPROP (a hybrid of RPROP and QuickProp), and second order backpropagation (Leonard, 1990) to name a few.

Associative Memory and Hopfield Network

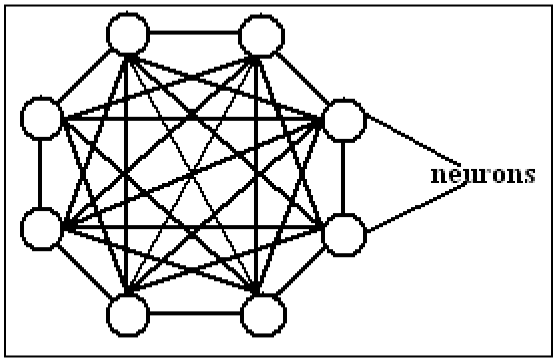
Associative memory (Kosko, 1988; Rojas, 1996) forms a class of systems intended to associate a known set of input vectors with a given set of output vectors. In addition, the concept of association also implies that the neighborhood of an input vector should also be mapped to the image to which it has been mapped. The selection of the mapping neighborhood depends on the distance of the neighbors from the input vector under consideration. Thus, even noisy input vectors can be associated with noise-free output vectors if a proper neighborhood selection mechanism is employed (Kosko, 1988). Associative memories find wide application in machine vision due to their capability of recalling a mapping already associated. Associative memories can be implemented with a feedback network, where the output is fed back repeatedly for every new input until the mapping process converges. The following three different kind of associative memories are in vogue (Kosko, 1988; Rojas, 1996).

- *Heteroassociative memories*, where n number of input vectors in an m dimensional vector space are mapped to n number of output vectors in an l dimensional space.
- *Autoassociative memories*, where a vector is associated to itself.
- *Pattern recognition memories*, where a vector is associated with a scalar.

The bidirectional associative memory (BAM) (Kosko, 1988) is a synchronous associative model with bidirectional interconnections/edges. It comprises two layers, viz., (i) an input layer and (ii) an output layer. Processed information is propagated recursively between these layers until the network operation converges. The input layer accepts inputs and sends the processed input information to the output layer via the bidirectional edges. The output layer operates on the incoming processed input information from the input layer and in turn sends the processed output information back to the input layer for further processing.

The Hopfield network (Hopfield, 1982) due to the American physicist John Hopfield is a recurrent asynchronous neural network architecture, which serves as a content-addressable memory (Rojas, 1996). A Hopfield network (shown in Figure 7) is a fully connected neural network architecture in that each neuron is connected to all other neurons except to itself. Moreover, the interconnections are symmetric (Rojas, 1996) such that the interconnection weight w_{ij} from the i^{th} neuron to the j^{th} neuron is equal to the interconnection

Figure 7. Schematic of Hopfield network comprising eight interconnected neurons



weight w_{ji} the other way round, i.e. from the jth neuron to the ith neuron. Due to this symmetric bidirectional interconnection topology, it forms a class of bidirectional associative memory (BAM) (Kosko, 1988).

In general, a Hopfield network comprises a set of N binary thresholded neurons connected following the topology as shown in Figure 7. There is no sharp delineation between the input and output neurons in this network. All the neurons act as both input and output neurons so that the inputs presented to the network retain their states until they are updated. The activation state of each neuron is updated asynchronously irrespective of the activation states of the other neurons.

Since the neurons in a Hopfield network are binary in nature, they exhibit binary/bipolar behavior. Hence, they can exist with activation values of either 1(+1) [or 0(-1)] depending on whether the activation values exceed [or do not exceed] the corresponding thresholdsθ_i. Thus, the activation value t_i of the ith neuron in a Hopfield network can be expressed as either [wikiHop]

$$t_i = \begin{cases} 1 & \text{if} \quad \sum_{j=1}^{N} w_{ij} s_j(t) > \theta_i \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

or [wikiHop]

$$t_i = \begin{cases} +1 & \text{if} \quad \sum_{j=1}^{N} w_{ij} s_j(t) > \theta_i \\ -1 & \text{otherwise} \end{cases} \tag{32}$$

where, w_{ij} represents the interconnection weight from the ith neuron to the jth neuron, $s_j(t)$ is the state of the jth neuron at time t and θ_i is the activation threshold of the ith neuron.

Activation Update Mechanism

The fully interconnected topology of the Hopfield network indicates that the network possesses a finite energy arising out of the activation states $s_i(t)$ and $s_j(t)$ of the neurons at any instant of time t. This energy is given by the Lyapunov function (Hahn, 1963) as

$$E(t) = -\frac{1}{2}\sum_{i,j} w_{ij} s_i(t) s_j(t) + \sum_{i} \theta_i s_i(t) \tag{33}$$

As the states $s_i(t)$ or $s_j(t)$ of the neurons are updated to newer states $s_i(t+1)$ or $s_j(t+1)$ of lower energy, the Hopfield network switches to an overall lower energy state $E(t+1)$. Subsequently, the network converges at a minimum of the energy state. However, until the current states of the neurons are updated, the neurons preserve their individual states.

One of the important applications of the Hopfield network is its ability to serve as an associative memory, where the interconnection weights are ultimately set in such a fashion so as to replicate the stable patterns already stored in the network. In other words, if a noisy or incomplete pattern is presented to the network, the network automatically converges to a stable state, which replicates a pattern similar to that presented. Details regarding the different incarnations of the Hopfield network are available in (Hopfield, 1982; Rojas, 1996).

Kohonen's Self-Organizing Feature Map (SOFM)

As mentioned in previous sections, neural networks can also be put to use in situations where there are neither target outputs nor class labels i.e. they can operate in an unsupervised manner as well in the absence of any supervisor. The main objective of such a learning paradigm is to either discover the underlying structure of data followed by the encoding, compression and transformation of data.

Kohonen's self-organizing feature map (SOFM) (Kohonen, 1982; Kohonen, 1988) is a good example of the unsupervised learning model. Kohonen's self-organizing feature map is centered on preserving the topology in the data presented to the network thereby discovering the underlying structure of data by means of reducing the dimensions of data through self-organization. The topological map of input data reflects its structure by means of a mapping that preserves the neighborhood relations therein. This is brought about by subjecting the output data units to certain neighborhood constraints so that more neurons are allocated to recognize those parts of the input space where there are larger number of input vectors and fewer neurons are allocated to the other parts where fewer input vectors occur. Hence, the topology preservation procedure ensures that the neighboring neurons respond to similar vectors and the network learns the categorization, topology, and distribution of input vectors in the process (Kohonen, 1982).

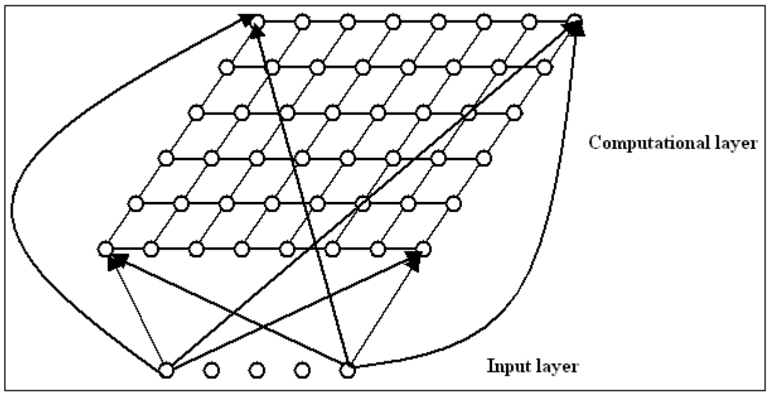
Kohonen's SOFM (Kohonen, 1982; Kohonen, 1988) differs from the perceptrons or other networks in terms of its architectural makeup. While in other network architectures, it is assumed that each neuron in a given layer is identical with respect to its connection to all the other nodes in the preceding and/or following layer, in SOFM, the physical arrangement of these neurons are also taken into consideration. Neurons tend to group together around a winning neuron, which bears the greatest similarity with an input pattern. Moreover, neurons that are closer together interact differently than those that are further apart from each other. A network that performs this kind of mapping is called a feature map (Kohonen, 1982; Kohonen, 1988). Figure 8 shows the topological structure of a SOFM comprising two layers, viz., (i) input layer and (ii) computational/output layer. Each computational layer neuron is connected to all the input layer neurons. Only fewer interconnections are shown in Figure 8 for the sake of clarity.

SOFM Self-Organizing Algorithm

The SOFM operates in a self-supervised manner. It organizes data into output responses based upon the correlations in the input data. The concept of grouping of similarly behaved neurons around a winning neuron enables the generation of localized output responses from these neurons. SOFM adaptively transforms an incoming signal pattern of arbitrary dimensions into a one or a two-dimensional discrete map in a topologically ordered fashion. The output layer of the SOFM therefore comprises neurons placed in either a one or a two-dimensional lattice. A competitive learning process between the neurons of the computational layer helps them to become selectively tuned to various input patterns. The winning neurons therefore, group together so as to replicate the features of the input patterns. The self-organizing process in a SOFM comprises the following phases (Kohonen, 1982; Kohonen, 1988).

1. *Initialization*: All the interconnection weights are initialized with small random values in this phase.
2. *Competitive learning*: In this phase, each computational layer neuron computes its value of a discriminant function for each input pattern presented, which provides the

Figure 8. Schematic of Kohonen's SOFM



basis for its selection as a winning neuron. The discriminant function generally employed is the squared Euclidean distance between the input vector **x**={x_1, x_2, x_3, …, x_n} (n being the input vector dimension) and the interconnection weight vector $\mathbf{w_j}$={w_{j1}, w_{j2}, w_{j3}, …, w_{jn}} for each jth neuron in the computational layer. It is given by (Kohonen, 1982; Kohonen, 1988)

$$D_j(\mathrm{x}) = \sum_{i=1}^{n}(x_i - w_{ji})^2 \qquad (34)$$

As a part of this competitive process, the neuron having the smallest value of the discriminant function is declared as the winner.

3. *Cooperative process*: This phase ensures the firing of the closest neighbors of the constituent neurons due to the lateral interaction between the excited neurons. It also ascertains that the topological features of the input patterns are preserved. Thus, the winning neuron determines the spatial location of a topological neighborhood of similarly excited neurons, thereby forming the so-called feature map. However, the degree of excitation decays with the distance from the firing neuron. The rate of decay is decided by a neighborhood function of the form (Kohonen, 1982; Kohonen, 1988)

$$U_{j,x_w} = \exp(-\frac{S^2_{j,x_w}}{2\sigma^2}) \qquad (35)$$

where, S_{j,x_w} is the lateral distance between the jth computational layer neuron and the winning neuron x_w and σ is the size of the neighborhood. As the cooperative process in SOFM proceeds, the size σ is decreased with time according to the following equation (Kohonen, 1982; Kohonen, 1988).

$$\sigma(t) = \sigma_0 \exp(-\frac{t}{\tau_\sigma}) \qquad (36)$$

Here, σ_0 is the initial neighborhood size and τ_σ is a time constant.

4. *Adaptation*: Once the topological feature map is constructed, it should be preserved. This phase ensures the stability of the topological feature map by allowing the excited neurons to decrease the values of their discriminant functions with time by means of suitable adjustments of the associated

interconnection weights w_{ji} by the following equation (Kohonen, 1982; Kohonen, 1988).

$$\begin{aligned} &w_{ji}(t+1) \\ &= w_{ji}(t) - \eta(t) U_{j,x_w}(t)[x_i - w_{ji}(t)] \end{aligned} \qquad (37)$$

where, $\eta(t) = \eta_0 \exp(-\frac{t}{\tau_\eta})$ is a time dependent learning rate. As a consequence, when an input pattern similar to the learnt pattern is presented to the network, the winning neurons elicit similar responses in consonance with the structure in the presented pattern.

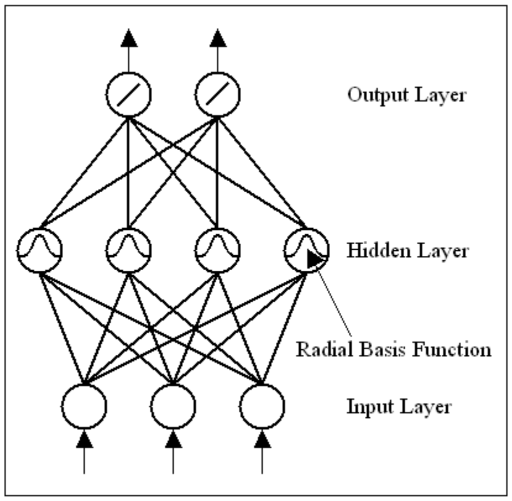
SOFM finds wide applications in feature extraction from multidimensional datasets due to its capabilities of dimensionality reduction and approximation. In addition, SOFM is widely used in classification/recognition problems in computer vision.

Radial Basis Function Networks (RBFN)

The evolution of the Radial Basis Function Network (RBFN) (Buhmann, 2003; Haykin, 1999; Paul, 2001) is motivated by the fields of statistical pattern classification and clustering, function approximation, spline interpolation techniques (Abraham, 2004; Bishop, 1995; Box, 1970; Poggio, 1990; Jones, 1990). The RBFN is a three-layer feed-forward network architecture comprising an input layer, a hidden layer and an output layer. The input layer simply propagates the input information to the hidden layer. Hence, there is no processing at the input layer. Nonlinear radial activation functions (normally Gaussian) are employed for the hidden layer neurons (Chen, 1991) while the output layer neurons use linear activations. The output layer neurons simply carry out a weighted sum of the outputs of the hidden layer neurons. Thus, the RBFN accepts nonlinear inputs and maps them to linear outputs. A typical RBFN is shown in Figure 9.

Due to this approximation capability, a RBFN is best suited for modeling complex mappings by resorting to only a single hidden layer (Buhmann, 2003; Haykin, 1999; Paul, 2001). The input-to-hidden layer interconnection weights are decided by taking into cognizance the clustered behavior of the input dataset. Given a training set of input-output pairs, the network parameters can be trained to establish a sought input-output relationship.

Figure 9. Schematic of a RBF network



Once a RBFN is trained, it can be used to map any data with statistical distributions and properties similar to those used to form the training set (Buhmann, 2003; Haykin, 1999; Paul, 2001). The dynamics of a RBFN varies from one application to another.

RBFN Activation Functions

Several types of nonlinear activation functions are used for the hidden layer (Buhmann, 2003; Lukaszyk, 2004) in a RBFN. These functions form a family of functions referred to as radial basis functions (RBFs) [wikiRadial1, wikiRadial2]. Some commonly used activation functions include (i) the Gaussian function $\psi(x) = e^{-\beta(||x-c||)^2}$; $\beta > 0$, where β is the RBF width parameter [wikiRadial1], (ii) the thin-plate spline function $\psi(x) = || x - c ||^2 \ln || x - c ||$ [wikiRadial1] (iii) the polyharmonic spline function $\psi(x) = || x - c ||^k$; $k = 1, 3, 5, \ldots, 2n-1$ or $\psi(x) = || x - c ||^k \ln || x - c ||^k$; $k = 2,4,6,\ldots,2n$ [wikiRadial1] and (iv) the multiquadric function $\psi(x) = \sqrt{(|| x - c ||) + \beta^2}$; $\beta > 0$ [wikiRadial1].

The general form of the Gaussian function for the ith hidden layer neuron in a RBFN is given by (Broomhead, 1988; Casdagli, 1989; Cha, 1996)

$$\psi_i(x) = e^{-\frac{(||x-c_i||)^T}{\sigma_i(||x-c_i||)}} \tag{38}$$

where, i=1, 2, 3, …, H (the number of hidden layer neurons). c_i and σ_i are the mean/center and covariance of the ith Gaussian function, respectively. c_i determines the position and σ_i controls the shape of the function.

As already stated, the output layer of a RBFN yields a linear combination of the weighted hidden layer neurons' outputs given by (Buhmann, 2003; Adrian, n.d.)

$$\varepsilon_k(x) = \sum_{i=1}^{H} w_{ik}\psi_i(x) \tag{39}$$

where, w_{ik} are the hidden-to-output layer weights and k=1, 2, 3, …, O (the number of output layer neurons). Since ε_k can have a wide range of values, ε_k is often squashed in the interval [0, 1] by the standard sigmoidal function in specific applications.

Training of RBF Networks

RBFN operates in both supervised and unsupervised modes of operation (Buhmann, 2003; Haykin, 1999; Paul, 2001). In the supervised mode of operation, a training set of input-output pairs is used to train the network so as to adapt the RBF width parameter β, the center c and the interconnection weight w such that an objective/cost function involving the expected outputs o and the input feature vector x is minimized. The most widely used cost function is given by (Adrian, n.d.)

$$F_t(w) = [o(t) - \varepsilon(x(t), w)]^2 \tag{40}$$

Since $F_t(w)$ depends on the interconnection weights, an optimal set of weights is needed to achieve the desired minimum of the cost function. Hence, the training procedure of a RBFN boils down to an optimization procedure. It may be noted this optimization procedure also enables the incorporation of multiple cost functions in specific applications. In those situations, the cost function may be carved out of an additive combination of the individual cost functions.

Similar to the multilayer perceptron, the gradient descent algorithm (Snyman, 2005) is used in RBFN to adaptively minimize the cost function. In addition, the standard backpropagation algorithm (Rumelhart, 1986; Haykin, 1999; Rojas, 1996) is also used for iteratively adapting the interconnection weights by computing the derivatives of the cost function with respect to the weights. Other notable algorithms employed for training of RBF networks include the Gram-Schmidt orthogonalization (Golub, 1996), learning vector quantization (Somervuo, 2004), the *k*-means clustering algorithm (MacQueen, 1967) etc.

RBF networks find wide applications in data interpolation (Davies, 1995), chaotic time-series modeling (Lukaszyk, 2004), speech recognition (Niranjan, 1990), image processing (Cha, 1996), motion estimation (Bors, 1998) to name a few.

Adaptive Resonance Theory (ART)

Among the early cognitive models, the exemplar model (Grossberg, n.d.; Medin, 1978; Murphy, 1985) and the prototype model (Smith, 1998; Minda, 1997) deserve special mention. The exemplar models rely on the storage of exemplars of each new event/pattern encountered during the learning phase for the purpose of eliciting cognitive responses. In the recognition phase, these models resort to comparing the test items to each of the exemplars of events/patterns stored in the learning phase. This comparison process finally assigns the test items presented, to a particular class of exemplars, which bears the greatest resemblance with the test items. Thus, it is evident that the cognitive capabilities of the exemplar models rest on the generation of suitable exemplars from events/patterns and subsequent searching for the nearest match with the test items.

On the contrary, the prototype models insist on storing only abstractions/prototypes of multiple exemplars generated from new events/patterns encountered (Grossberg, n.d.). During the recognition phase, the prototype models compare the test items to each stored prototype and assign the class of the test items to the one, which contains a prototype bearing a strong resemblance with the test items. Hence, both the storage and search complexity are lower in the prototype models as compared to the exemplar counterparts. However, both these models require a *bottom-up learning* mechanism in which exemplars/prototypes are generated and learnt followed by a *top-down learning* phase, where the similarity of the test items presented to the stored exemplars/prototypes is derived. Moreover, both these models suffer from the problems of generation of suitable abstractions and complexities arising out of the search procedures adopted (Grossberg, n.d.).

As a consequence, the rules-plus-exceptions cognitive model (Grossberg, n.d.; Nosofsky, 1998; Nosofsky, 2002) was introduced to overcome the shortcomings of the exemplar and prototype models. It is a hybrid model comprising a mixture of the exemplar and prototype models. The rules-plus-exceptions cognitive model learns both specific (exemplars) patterns, which form the guiding recognition rules and generic (prototypes) patterns, which incorporate out-of-rules exceptions for patterns, which deserve special attention, thereby keeping all avenues open for newer patterns without allowing the learnt ones to be destroyed. This is in sharp contrast to the competitive learning methodology adopted in feed-forward neural networks, where the trained synaptic weights tend to alter as newer events/patterns are learnt leading to a possible loss of older knowledge with time. Thus, to sustain the older knowledge, the synaptic weights have to be accommodative of the newer knowledge vis-à-vis the older ones. This paradox is referred to as the famous *stability-plasticity dilemma* (Grossberg, n.d.; Grossberg, 1976).

Carpenter and Grossberg introduced the Adaptive Resonance Theory (ART) (Grossberg, 1976; Carpenter, 1987a) during 1976-1986 as an effort to imbibe a dynamic balance in the so-called *stability-plasticity dilemma* by self-organizing newer input patterns in addition to those originally stored. The primary motivation behind the ART model lies in the postulate that humans possess cognitive capabilities due to an interaction between *top-down expectations*, which involve a learning of prototypes for selection of consistent bottom-up signals followed by suppression of inconsistent bottom-up signals (also referred to as attentional focusing) and *bottom-up sensory activation*, which involves automatic activation of target prototypes during the learning process. To be precise, the *top-down expectations* form the basis for a *bottom-up sensory activation* based comparison of newer patterns with the exemplars/ prototype generated in the learning phase leading to a class assignment. Moreover, a threshold value referred to as the *vigilance parameter* (χ) ensures an expected class assignment as long as the differences between the expectations and the sensation do not exceed the threshold value. As such, the ART model is suited as both a supervised and an unsupervised neural network model.

The ART model essentially comprises an *attentional subsystem* and an *orienting subsystem* (Grossberg, 1976; Carpenter, 1987a). The attentional subsystem carries out the stabilization of learning and activation procedures by means of matching of *bottom-up input activation* and *top-down expectations*. The orienting subsystem detects any mismatch, which may occur in the attentional subsystem during learning. Hence, it controls the attentional subsystem. An ART model possesses the following properties (Ekrafft, 2001; Carpenter, 1987a; Carpenter, 2003).

- *Self-scaling computational units:* The attentional subsystem functions using the principles of competitive learning, which ensures learning of patterns with features of importance. Moreover, it also ensures immunity to noisy patterns (Ekrafft, 2001).
- *Self-adjusting memory search:* This property enables an adaptive and parallel search of memory of stored patterns (Ekrafft, 2001).
- *Direct category access:* This feature enables the stored patterns to directly access their corresponding categories (Ekrafft, 2001).
- *Adaptive modulation of attentional vigilance:* As mentioned earlier, a vigilance parameter (χ) acts as a threshold in the categorization process. ART possesses the ability to adaptively tune the attentional vigilance parameter in accordance with the operating environment (Ekrafft, 2001).

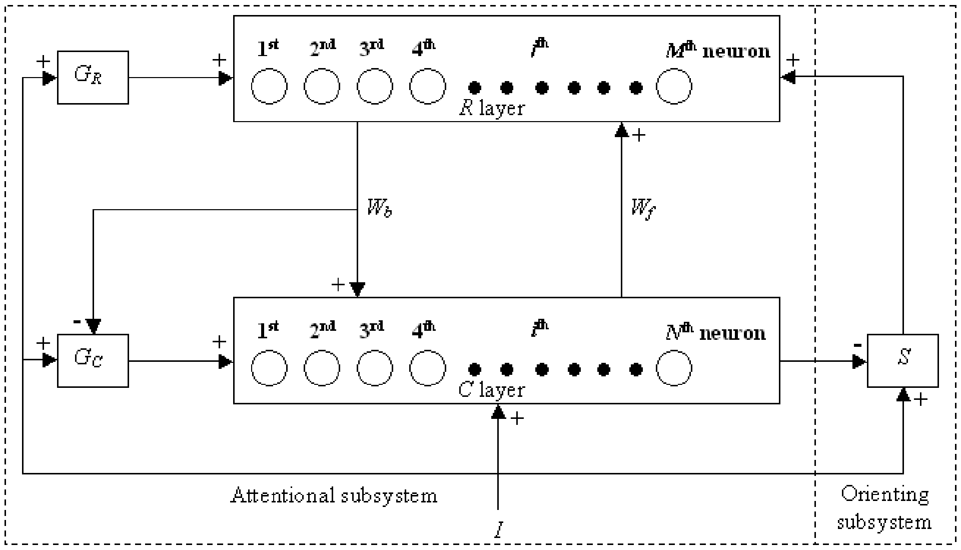
Several versions of the ART model are in vogue (Carpenter, 1987a; Carpenter, 2003). The basic ART model (ART1) (Carpenter, 1987a; Carpenter, 2003) is an unsupervised model for learning and recognition of binary input patterns. ART2 (Carpenter, 1987b; Carpenter, 1991a; Carpenter, 2003) is efficient in categorizing both analog and binary input patterns. ART3 (Carpenter, 1990; Carpenter, 2003), ARTMAP (Carpenter, 1991b; Carpenter, 2003), fuzzy ART (Carpenter, 1991c; Carpenter, 2003), fuzzy ARTMAP (Carpenter, 1992; Carpenter, 2003) are complex and distributed incarnations of the basic ART model, which are suitable for distributed recognition tasks. The following subsections discuss the architectures and operations of ART1 and ART2 models.

ART1

An extended version of the diagram (Carpenter, 1987a; Carpenter, 2003; art, n.d.) illustrating the basic components of the ART1 model (Carpenter, 1987a; Carpenter, 2003) is shown in Figure 10. The attentional subsystem (shown in the dotted right box) comprises two competitive network layers (i) the comparison layer C (consisting of N number of neurons) and (ii) the recognition layer R (consisting of M number of neurons) along with respective control gains G_C and G_R (Carpenter, 1987a; Carpenter, 2003; art, n.d.). The orienting subsystem (shown in the dotted right box) comprises a reset layer (S) to control the operational dynamics of the attentional subsystem.

Each neuron in C is connected to all the neurons in R by the continuous-valued forward long-term memory (LTM) gates/weights W_f. Only a single W_f weight is shown for the sake of clarity. On the other hand, each neuron in R is connected to all the neurons in C by the binary-valued back-

Figure 10. Schematic of ART1 model



ward LTM gates/weights W_b. Similarly, a single W_b weight is shown for the sake of clarity of the schematic. Both C and R generate short term memory (STM) activations/activity patterns when fed with inputs. Given this topology, each neuron in C receives three input signals, one each from (i) the external input vector I, (ii) the control gain G_C and (iii) internal network inputs corresponding to the STM activity pattern outputs of R (O_{R_j} j=1, 2, 3, …, M) multiplied by corresponding weights $W_{b_{ji}}$,j=1, 2, 3, …, M; i=1, 2, 3, …, N. The output (O_{C_i} , i=1, 2, 3, …, N) from each neuron in C is propagated forward to R and fed to the orienting subsystem as shown in Figure 10 (Carpenter, 1987a; Carpenter, 2003; art, n.d.). Similarly, each neuron in R receives three signals, one each from (i) the reset layer (S) of the orienting subsystem, (ii) the control gain G_R and (iii) internal network inputs corresponding to the outputs of C (O_{C_i} , i=1, 2, 3, …, N) multiplied by corresponding weights $W_{f_{ij}}$, i=1, 2, 3, …, N; j=1, 2, 3, …, M. The output (O_{R_j} ,j=1, 2, 3, …, M) of each of these neurons is fed back to C and propagated to G_C simultaneously. It may be noted that in the ART1 model, the neurons in both C and R follow the *two-thirds rule* to fire, i.e., they output a 1 if and only if at least two of their three inputs are high (Carpenter, 1987a; Carpenter, 2003; Alfredo, n.d.).

ART1 Operation

On reception of a binary input vector I, the comparison layer propagates it forward to the recognition layer for matching with a stored class template. The selected class is passed back to the comparison layer for verification with the input vector. If the verification process yields favorable results, a new input vector is read and the operations are repeated for the new vector. However, in case of a mismatch, the orienting subsystem comes into play. It inhibits the selected class and pushes the recognition layer to searches for a new class for the input vector.

The entire processing task carried out by ART1 can be summarized as follows (Carpenter, 1987a; Carpenter, 2003; Alfredo, n.d.).

- *Recognition phase/bottom-up activation*: In this phase, no input vector is applied to

C thereby disabling any recognition in R and making G_C=0; G_R=0. This sets all the neurons of R to zero, deeming them equally fit to win any subsequent recognition. However, when an input vector I arrives, one or more neurons are reset to 1 thereby setting G_C=1; G_R=1 (Carpenter, 1987a; Carpenter, 2003; Alfredo, n.d.).

Thus,

$$G_C = \begin{cases} 1 & \text{if} \quad I \neq 0 \quad \text{and} \quad O_{R_i} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

i.e., even if R does not produce any output, G_C is set to 1 if there is an input vector and

$$G_R = \begin{cases} 1 & \text{if} \quad I \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{42}$$

implying that R is activated by G_R when there is an input vector.

Since, both C and R layers represent the short-term memory (STM) of the ART1 model, each C layer neuron excites a positive nonzero (G_C=1) STM activity pattern which results in the generation of an output which is an exact replica of the input vector I i.e., $O_C = I$. The C layer neurons whose activity is greater than a threshold excite the R layer neurons by their outputs. Thus, each $j$th$R$ layer neuron receives an input I_{Rj} given by (Carpenter, 1987a; Carpenter, 2003; Alfredo, n.d.)

$$I_{R_j} = \sum_{i=1}^{N} O_{C_i} W_{f_{ij}} \tag{43}$$

Since the R layer acts as a winner-take-all layer, the neuron in R that receives the largest of all O_{C_i}, i=1, 2, 3, …, N in its input I_{R_j}, matches the input vector best and fires leaving all other neurons in R to be set to 0 by means of lateral inhibition.

Comparison phase/top-down expectation: In this phase, the outputs O_{R_j}, j=1, 2, 3, …, M generate a top-down template given by (Carpenter, 1987a; Carpenter, 2003; Alfredo, n.d.)

$$I_{C_i} = \sum_{j=1}^{M} O_{R_j} W_{b_{ji}} \tag{44}$$

This is fed back to the C layer for comparison with the input vector I. Meanwhile, G_C is inhibited in the process and set to 0. The reset layer (S) in the orienting subsystem is responsible for adjudging the measure of similarity between I and I_C. This measure of similarity is determined by the vigilance parameter χ,$0<\chi\leq 1$. After comparison, if a match is found, the system is ready to learn. On the other hand, mismatch occurs if I and I_C differ by a value greater thanχ. At this point, a reset signal generated by S inhibits R from further recognition. A larger value of χ allows the system to learn newer categories. On the other hand, a smaller value of χ ensures that more input patterns are classified into the same category (Carpenter, 1987a; Carpenter, 2003; Alfredo, n.d.).

Once R has been disabled, both O_R and I_C are discarded and the STM activity pattern of C is generated again. This cancels the reset signal and the recognition phase starts afresh. If no further mismatch occurs, the classification is said to have been completed (Carpenter, 1987a; Carpenter, 2003; Alfredo, n.d.).

ART2

ART2 (Carpenter, 1987b; Carpenter, 1991a; Carpenter, 2003; Alfredo, n.d.) operates on continuous valued inputs. In this model, the comparison layer C accepts n number of six types of units, viz., *W*, *X*, *U*, *V*, *P* and *Q*, where *n* is the input vector dimension. The recognition layer *R* is a competitive layer. The connection patterns in *C* follow: $P \rightarrow Q \rightarrow V \rightarrow U \rightarrow P$ and $W \rightarrow X \rightarrow V \rightarrow U \rightarrow W$ (Alfredo, n.d.). The top-down and bottom-up connections between the different units of *C* and *R* preserve the prototypes already learnt. Initially, the activations of all the six units are set to zero. Computation starts at *U*, which holds the normalized value of the activation at *V*. This computation result is forwarded to both *W* and *P*. *W* receives the input vector from the external world and sums it up with the result received from *U*. Similarly, *P* sums up the result from *U* and the top-down activations from *R* when *R* is active. *X* and *Q* hold the normalized values of the activations at *W* and *P*, respectively. *V* sums up the values received concurrently from *X* and *Q*. When the *C* layer activations have stabilized, *P* sends its activation for recognition to the *R* layer. The reset unit *S* sums up the reset signals it receives from both *P* and *U* and sends the result to the vigilance parameter χ. χ determines as to whether *R* is to be deactivated or not. Control gains G_C and G_R are responsible for the normalization of the activity patterns in the different units (Alfredo, n.d.).

The ART networks are efficient in fast category learning and category recognition/classification of binary/analog patterns. The networks also find wide use in incremental supervised learning of multidimensional data and feature spaces.

ADVANCES IN NEURAL NETWORKING PARADIGM

Over the years, several novel neural network models suitable for different applications have been evolved. The task of listing each and every one of them would mean a book and is beyond the scope of this chapter. Moreover, it may be noted that the plethora of models evolved from time to time has been motivated by different fields of engineering and science. Notable among these fields are the machine vision, image processing, pattern recognition, data clustering and mining. This section throws some light on some of the important neural network architectures, which have been developed for different applications with due regards to the fields of image processing and pattern recognition.

One of the important breakthroughs in this direction is the evolution of modular and recurrent neural networks. Modular networks (Gallinari, 1995) are formed by the admixture of different neural network models discussed in the previous sections, thereby incorporating the features of several networks together. The advantages offered by such networks include (i) reduced time complexity, (ii) heterogeneous knowledge base, (iii) flexibility in learning strategies and (iv) scalability. Typical examples include the counterpropagation networks due to Hecht-Nielsen (Hecht, 1987) and the spline networks (Walter, 1995). Among the recurrent models proposed, significant are the Elman and Jordan recurrent networks (Elman, 1990). The Hidden Markov Models (HMM) (Huang, 1990) is another recurrent network model, which is capable of existing in either of different possible states at any instant of time. The change of state in a HMM is governed by some probabilistic dynamics. Chua and Yang (Chua, 1988a; Chua, 1988b; Nossek, 1993) proposed the cellular neural network (CNN) architecture. A CNN comprises neurons locally connected in a cellular structure. Though the CNN architecture is faster in operation, it requires user specified parameters to control its operation, which limits its use in real time applications. Convolutional networks (CN) (Cun, 1995) have been proposed for feature detection and character recognition. The central idea behind the operation of a CN lies in the concept of local receptive fields, which decide the interconnections between the different layers of the network. The network also incorporates the concept of shared weights and spatial subsampling, which result in shift and deformation invariance with reduced number of parameters. Phung and

Bouzerdoum (Phung, 2007) proposed a new neural architecture motivated by the concepts of image pyramids and local receptive fields. The proposed architecture, referred to as the pyramidal neural network (PyraNet), is efficient for classification of visual patterns. The architecture has a hierarchical structure of neural layers. A subset of the constituent layers of the network is organized in form of a pyramid and the remaining layers are simply one-dimensional layers. The PyraNet operates in a supervised mode and is capable of classifying two-dimensional patterns. An exhaustive survey of the advances in the neural networking paradigm can be found in (Egmont-Petersen, 2002; Huang, 2009).

So much so forth, the proposed neural network models are able to classify/recognize patterns after they are suitably trained with real world inputs. However, real world data exhibit varying degrees of uncertainty and vagueness. Traditional neural network models are unable to handle the uncertainty in knowledge base. Moreover, the complexity of the models increases with the number of model parameters as well as with the volume of the knowledge base. Hence, researchers have incorporated other soft computing tools viz., fuzzy sets, genetic algorithms, wavelets and rough sets to evolve more robust neural network models, which can effectively handle the vagueness in data as well as operate with an optimum set of parameters. These models, which have resulted out from the synergism of the neural networking paradigm with the other tenets of the soft computing paradigm are referred to as the neuro-fuzzy-genetic models.

Models (Kasabov, 2000; Russo, 1999; Russo, 2000; Pal, 1996; Estevez, 2005; Villmann, 2005; Muhammed, 2002; Martinetz, 1993; Hammer, 2005) based on the neuro-fuzzy paradigm have been successfully used to deal with imprecise knowledge bases. Fuzzy neural networks are used in (Horikawa, 1992) to automatically identify the fuzzified model of a nonlinear system. Simpson (Simpson, 1993) proposed a fuzzy min-max neural network, which is efficient for image segmentation and clustering. In (Lin, 1996), Lin *et al*. incorporated fuzzy clustering into a Hopfield network so as to avoid the problems of finding weighting factors in the energy function, thereby resulting in more efficient neural network architecture. The proposed fuzzy Hopfield neural network converges faster than its conventional counterpart. Tzeng and Chen (Tzeng, 1998) proposed a fuzzy neural network and applied it to SAR image classification. Jang (Jang, 1993) introduced the ANFIS (Adaptive-Network-based Fuzzy Inference System) framework useful in modeling real world systems. Ghosh *et al*. (Ghosh, 1993) proposed a multilayer self-organizing neural network (MLSONN) architecture based on a fuzzy measure driven estimation of the system errors. The architecture comprises an input layer, one or more hidden layers and an output layer of neurons. The novelty of the architecture lies in the use of indices of fuzziness at the output layer of the network to determine the system errors. The network is efficient in extracting binary objects from a noisy/blurred perspective. However, the use of the standard backpropagation algorithm for the adjustment of interconnection weights limits its use in real time applications. In addition, the use of the bilevel sigmoidal activation function also restricts its usage to the binary domain. The limitations of the MLSONN (Ghosh, 1993) architecture as regards to its inability to deal with multilevel/color inputs as well its high time complexity have led to the evolution of more efficient network architectures and activation functions. Bhattacharyya *et al*. (Bhattacharyya, 2008a; Bhattacharyya, 2010) introduced a multilevel version of the sigmoidal activation function, which is capable of handling multilevel inputs. Hence, the MLSONN (Ghosh, 1993) architecture guided by the proposed multilevel sigmoidal (MUSIG) activation function (Bhattacharyya, 2008a; Bhattacharyya, 2010) is able to deal with the extraction and segmentation tasks of multilevel input data. Bhattacharyya *et al*. (Bhattacharyya, 2007a; Bhattacharyya, 2007b) also devised a parallel version of the MLSONN architecture to process multiple featured inputs. The proposed parallel self-organizing neural network (PSONN) architecture is capable of extracting/segmenting pure and true color images. Bhattacharyya *et al*. (Bhattacharyya, 2006a; Bhattacharyya, 2007c) alleviated the usage of the backpropagation algorithm, which is used for the adjustment of interconnection weights in most of the traditional neural network architectures, by proposing a bi-directional self-organizing neural network (BDSONN) architecture. The proposed architecture comprises three interconnected fuzzy layers of neurons, viz., an input layer, an intermediate layer and an output layer. The input layer as usual accepts fuzzy inputs from the real world. The remaining two layers viz. the intermediate and the output layers are counterpropagating in nature. The neurons in all the layers are connected in a cellular fashion. In addition, the neurons in the different network layers are connected to each other using a neighborhood-based topology. The architecture uses fuzzy membership based weight assignment and subsequent updating procedure, thereby obviating the use of the time-complex backpropagation algorithm based weight adjustment mechanism. The network architecture uses a dynamic context sensitive thresholding mechanism and self organizes the input information by counter-propagation of the fuzzy network states between the intermediate and the output layers of the network.

Genetic algorithms (Goldberg, 1989) are heuristic search techniques, which use a feature space to arrive at an optimal solution space by means of application of several genetic operators. Significant contributions in the neuro-genetic domain are available in the literature (Zamparelli, 1997; Sklansky, 1996; Estevez, 2003). De *et al*. (De, 2009; De, 2010) applied a genetic algorithm based approach to determine the optimum transition levels of the MUSIG activation function (Bhattacharyya, 2008a; Bhattacharyya, 2010). The resultant neuro-fuzzy-genetic architecture is efficient in image processing applications.

A *wavelet* transform (Goswami, 1999; Daubechies, 1988; Mallat, 1989; Meyer, 1993) is an efficient tool for data approximation, compression, and noise removal. The decomposition and approximation powers of wavelets have been used to develop wavelet-based neural network architectures (Cristea, 2000; Shashidhara, 2000; Ho, 2001), which possess more noise immunity and generalization capabilities than the conventional neural network architectures. Applications of wavelet-based models also appear in the literature (Deschˆenes, 1998; Iftekharuddin, 1995).

Rough sets (Pawlak, 1982) are extensions of the classical set theory for describing imprecision and vagueness in real world data. Rough sets do not require any *a priori* information regarding the system at hand. Moreover, imprecision is defined in rough sets in terms of the boundary region of a set. The approximating capabilities of rough set theory have been put to use to evolve robust neural network architectures as well (Pan, 2003; Peters, 2000; Peters, 2001).

Support vector machines (SVMs) (Vapnik, 1995; Vapnik, 1997; Burges, 1998; Cristianini, 2000) fall into the recent advances in the neural networking paradigm as far as classification is concerned. SVMs are based on the principles of structural risk minimization. These are a set of supervised generalized linear classifiers more efficient than the traditional neural network based classifiers. In fact, a SVM model using a sigmoid kernel function is equivalent to a two-layer perceptron. A SVM performs classification using support vector methods. Input vectors are first mapped onto a higher-dimensional space in order to widen the separation between the data samples. Subsequently it constructs a hyperplane, which forms an optimal separator/discriminator of these vectors. Thus, it does not suffer from being stuck to a local minimum.

Most of the conventional neural network architectures discussed so far, inclusive of those resorting to a neighborhood-based topology, employ a large number of interconnected neurons. Larger the number of neurons, larger is the number of interconnections between them. A larger network implies larger network complexity and more processing time. Moreover, redundancy in the network interconnections also leads to the problem of false classification and recognition. Researchers have tried out several methods for arriving at an optimized neural network topology, which would sever the redundant interconnections in the architecture, thereby evolving space and time-efficient topologies (de Kruif, 2003; White, 1993; Whitley, 1990; Castellano, 1997; Zhou, 1999; Weymaere, 1994; Reed, 1993; Zeng, 2005). Several attempts (Ghosh, 2002) have been made to optimize the Hopfield network architecture, thereby reducing the network complexity. Methods are also reported (Harp, 1989) where the network structure is represented as a bit string and optimized using genetic algorithms. In (Miller, 1990), Miller *et al*. used a connectivity constraint matrix so as to optimize a layered feed-forward network of *N* units. Genetic algorithm based optimization techniques have been combined with backpropagation for speeding up the convergence of neural networks (Kitano, 1990). The optimization algorithm is used to determine the starting weights of the network, which are subsequently refined using backpropagation. Some methods (Miller, 1989; Oliker, 1992), which aim at encoding each network connections, are found to perform well for smaller sized networks. Larger sized networks have been optimized effectively (Harp, 1989; Dodd, 1990) by encoding only a subset of the network features. Opitz and Shavlik (Opitz, 1994) introduced a refining algorithm (REGENT) for arriving at an optimized neural network topology generated from a knowledge base using the genetic algorithm. Bhattacharyya *et al*. (Bhattacharyya, 2006b; Bhattacharyya, 2008b) proposed a fuzzy-set theoretic method to design pruned neighborhood neural network architectures. The resultant pruned architectures have been efficient compared to their fully connected counterparts both in terms of network complexity and extraction efficiency. Other notable contributions in this direction are the intuitive pruning methods based on weight and unit activation values (Hagiwara, 1993) or the magnitude-based pruning (MBP) method (Sietsma, 1991), which assumes that small weights are irrelevant.

Statistical analysis of the network parameters has often been used for pruning neural network architectures. Steppe *et al*. (Steppe, 1996) used the likelihood-ratio test statistics of the interconnection weights to refine the network topology. They pruned those hidden units having weights statistically different from zero. Belue and Bauer (Belue, 1995) injected a noisy input parameter into a neural network model and decided upon the relative significance of the original network parameters with the injected noisy parameter. Parameters with lower significance than the noisy parameter are pruned. Prechelt (Prechelt, 1997) developed methods to determine the number of weights to be pruned.

Sensitivity analysis pruning techniques (Ruck, 1990; Karnin, 1990; Mozer, 1989), which model the network parameters as small perturbations to the network sensitivity, have been widely used for evolving pruned network topologies. Zurada et al. (Zurada, 1994; Zurada, 1997) removed redundant network input units by means of the sensitivity analysis of the network output function with respect to perturbations of input units. This approach was further extended in (Engelbrecht, 2001). Ponnapalli *et al*. (Ponnapalli, 1999) devised a formal selection and pruning technique based on the concept of sensitivity index proposed by Karnin (Karnin, 1990). The approach is efficient in reducing redundancy in feedforward neural network architectures without requiring any *a priori* knowledge about the system. In the optimal brain damage (OBD) (Cun, 90), optimal brain surgeon (OBS) (Hassibi, 1993) and optimal cell damage (OCD) (Cibas, 1996), sensitivity analysis is performed with regard to the training error.

The field of *quantum computing* (Steane, 2000; Shor, 1997; Grover, 1998; Williams, 1998; Berman, 1998) entails the applications of the concepts of quantum mechanics/physics in the development of a computing paradigm much faster as compared to the conventional computing paradigm. The increase in the computing speed is obtained by means of utilizing the inherent parallelism observed in the *qubits*, the building blocks of a quantum computer. Researchers are actively involved in the design and implementation of neural network architectures inspired by the features offered by the quantum computing paradigm. Oliveira (de Oliveira, 2009) proposed a quantum RAM based neural network, which enables a direct realization of the architecture in quantum circuits. Liu *et al*. (Liu, 2007) developed an image segmentation method based on the discrete Tchebichef moments and quantum neural networks using multilevel transfer functions. A score of quantum neural network models exist in the literature (Faber, 1989; Menneer, 1995; Ventura, 1998; Goertzel, n.d.; deOliveira, 2008; Gopathy, 1997).

APPLICATION PERSPECTIVES

Artificial neural networks (ANNs) find wide applications to a broad spectrum of different data-intensive application areas in science, engineering and finance. These can be classified into following.

- Cognitive science and machine learning (Egmont-Petersen, 2002), where ANNs have been widely used to mimic the behavior of human brain.
- Process modeling and control (Agarwal, 1997; Balakrishnan, 1996; Hunt, 1992; Kerr, 1998; Sanner, 1992), where ANNs have been used for modeling higher and lower levels of reasoning in the form of (i) language, (ii) problem solving, (iii) vision, (iv) speech recognition for generating the best process control settings.
- Optimization (Chen, 1994; Scales, 1985; Shanno, 1990), where ANNs have been used to find out an optimal solution of a problem for a particular set of constraints and a cost function.
- Classification (Egmont-Petersen, 2002), which is one of the most important application areas where ANNs have been put to use. This involves grouping of data based patterns into classes for proper understanding and analysis of the same.
- Signal processing applications (Zaknich, 2003) for suppressing line noise and blind source separation.
- Pattern recognition (Phung, 2007; Egmont-Petersen, 2002), for faithful analysis of corresponding datasets.
- Associative memory (Kosko, 88), where ANNs are used to recall memory based on a partial match of previously trained patterns.
- Regression analysis (Sengupta, 1995), where ANNs are used for mapping of functions and time series predictions.

Apart from the aforestated application areas, other applications involving artificial neural networks include portfolio management (Zimmermann, 2001; Freitas, 2009), robotics (Kwan, 2000; Sanger, 1994; Kim, 1999; Barambones, 2002), target recognition (Egmont-Petersen, 2002), medical diagnosis (Moein, 2008), quality control (Bahlmann, 1999; Benvenuto, 2000) to name a few.

DISCUSSIONS AND CONCLUSION

An overview of artificial neural networks has been presented with due regards to the evolution, topologies, learning algorithms and applications. The structure and functions of the human nervous system, which remains the motivation behind the evolution of the neural networking paradigm, is elucidated. A detailed analysis of the different components that make up this computing paradigm is also presented with due regards to the possible variations thereby. The topology, structure, operational characteristics and applications of some of the important neural network models are illustrated with emphasis on the learning methodologies adopted by these networks. The latest trends in researches in this direction are also discussed with reference to the significant breakthroughs achieved over the years. Finally, the application areas that these neural networks are put to are mentioned for a proper understanding of the relevance of the computing paradigm.

ACKNOWLEDGMENT

The author would like to take this opportunity to render his reverence to Prof. (Dr.) Asit Kumar Datta, Retired Professor of the Department of Applied Optics and Photonics, University College of Science and Technology, University of Calcutta, Kolkata and presently Head of the Department of Electronics & Communication Engineering, Aliah University, Kolkata for being a constant source of inspiration in his endeavors. The author feels highly privileged to have a professor and a mentor in Prof. (Dr.) A. K. Datta in the early days of his enlightenment of the topic under consideration.

REFERENCES

Abraham, A. (2004). Meta-learning evolutionary artificial neural networks. *Neurocomputing*, *56c*, 1–38. doi:10.1016/S0925-2312(03)00369-2

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, *9*, 147–169. doi:10.1207/s15516709cog0901_7

Adrian. (n.d.). *Paper*. Retrieved from www-users.cs.york.ac.uk/~adrian/Papers/Others/OSEE01.pdf

Agarwal, A. (1997). A systematic classification of neural-network-based control. *IEEE Control Systems Magazine*, *17*(2), 75–93. doi:10.1109/37.581297

Alfredo. (n.d.). *Website*. Retrieved from www.cannes.itam.mx/Alfredo/English/Nslbook/MitPress/Publications/157_170.CH08.pdf

Almeida, L. B., & Silva, F. (1991). Speeding-up backpropagation by data orthonormalization. *Artificial Neural Networks*, *2*, 56–149.

Anderson, J. A., Silverstein, J. W., Ritz, S. A., & Jones, R. S. (1977). Distinctive features, categorical perception, probability learning: Some applications of a neural model. *Psychological Review*, *84*, 413–451. doi:10.1037/0033-295X.84.5.413

ART. (n.d.). *Website*. Retrieved from http://www.learnartificialneuralnetworks.com/art.html

Bahlmann, C., Heidemann, G., & Ritter, H. (1999). Artificial neural networks for automated quality control of textile seams. *Pattern Recognition*, *32*(6), 1049–1060. doi:10.1016/S0031-3203(98)00128-9

Balakrishnan, S. N., & Weil, R. D. (1996). Neurocontrol: A literature survey. *Mathematical Modeling and Computing*, *23*, 101–117. doi:10.1016/0895-7177(95)00221-9

Barambones, O., & Etxebarria, V. (2002). Robust neural control for robotic manipulators. *Automatica*, *38*, 235–242. doi:10.1016/S0005-1098(01)00191-1

Belue, L. M., & Bauer, K. W. (1995). Determining input features for multilayer perceptrons. *Neurocomputing*, *7*, 111–121. doi:10.1016/0925-2312(94)E0053-T

Benvenuto, F., & Marani, A. (2000). Neural networks for environmental problems: Data quality control and air pollution nowcasting. *International Journal of Global Nest.*, *2*(3), 281–292.

Berman, G. P., Doolen, G. D., Mainieri, R., & Tsifrinovich, V. I. (1998). *Introduction to quantum computers*. London, U. K.: World Scientific. doi:10.1142/9789812384775

Bhattacharyya, S., & Dutta, P. (2006b). Designing pruned neighborhood neural networks for object extraction from noisy background. *International Journal of Foundations of Computing and Decision Sciences*, *31*(2), 105–134.

Bhattacharyya, S., Dutta, P., & DuttaMajumder, D. (2007b). A parallel self-organizing neural network (SONN) based color object extractor using MUSIG activation function with CONSENT. In D. Chakraborty, S. Nanda, & D. DuttaMajumder (Eds.), *Fuzzy logic and its application in technology and management* (pp. 206-213). New Delhi, India: Narosa Publishing House.

Bhattacharyya, S., Dutta, P., & Maulik, U. (2007c). Binary object extraction using bi-directional self-organizing neural network (BDSONN) architecture with fuzzy context sensitive thresholding. *International Journal of Pattern Analysis and Applications,* 345-360.

Bhattacharyya, S., Dutta, P., & Maulik, U. (2008a). Self organizing neural network (SONN) based gray scale object extractor with a Multilevel Sigmoidal (MUSIG) activation function. *International Journal of Foundations of Computing and Decision Sciences*, *33*(2), 46–50.

Bhattacharyya, S., Dutta, P., Maulik, U., & Nandi, P. K. (2006a). A self supervised bi-directional neural network (BDSONN) architecture for object extraction guided by beta activation function and adaptive fuzzy context sensitive thresholding. *International Journal of Intelligent Systems and Technologies*, *1*(4), 345–365.

Bhattacharyya, S., Dutta, P., Maulik, U., & Nandi, P. K. (2007a). Multilevel activation functions for true color image segmentation using a self supervised parallel self organizing neural network (PSONN) architecture: A comparative study. *International Journal of Computer Science, 2*(1), 09-21.

Bhattacharyya, S., Maulik, U., & Bandyopadhyay, S. (in press). Soft computing and its applications. In Dai, Y., Chakraborty, B., & Shi, M. (Eds.), *Kansei engineering and soft computing: Theory and practice*. Hershey, PA: IGI Global.

Bhattacharyya, S., Maulik, U., & Dutta, P. (2008b). A pruning algorithm for efficient image segmentation with neighborhood neural networks. *IAENG International Journal of Computer Science*, *35*(2), 191–200.

Bhattacharyya, S., Maulik, U., & Dutta, P. (2010). Multilevel image segmentation with adaptive image context based thresholding. *International Journal of Applied Soft Computing*, *11*, 946–962. doi:10.1016/j.asoc.2010.01.015

Bi, W., Wang, X. G., Tang, Z., & Tamura, H. (2005). Avoiding the local minima problem in backpropagation algorithm with modified error function. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E (Norwalk, Conn.)*, *88-A*, 3645–3653.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press.

Bors, A. G., & Pitas, I. (1998). Optical flow estimation and moving object segmentation based on median radial basis function network. *IEEE Transactions on Image Processing*, *7*(5), 693–702. doi:10.1109/83.668026

Box, G. E. P., & Jenkins, G. M. (1970). *Time series analysis, forecasting and control*. CA: Holden Day.

Broomhead, D. S., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, *2*, 321–355.

Brown, A. (1991). *Nerve cells and nervous systems*. Berlin, Germany: Springer-Verlag.

Buhmann, M. D. (2003). *Radial basis functions: Theory and implementations*. Cambridge University Press. doi:10.1017/CBO9780511543241

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167. doi:10.1023/A:1009715923555

Carpenter, G. A., & Grossberg, S. (1987a). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision Graphics and Image Processing*, *37*, 54–115. doi:10.1016/S0734-189X(87)80014-2

Carpenter, G. A., & Grossberg, S. (1987b). ART2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, *26*, 4919–4930. doi:10.1364/AO.26.004919

Carpenter, G. A., & Grossberg, S. (1990). ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, *3*, 129–152. doi:10.1016/0893-6080(90)90085-Y

Carpenter, G. A., & Grossberg, S. (2003). Adaptive resonance theory (ART). In Arbib, M. A. (Ed.), *The handbook of brain theory and neural networks* (pp. 79–82). Cambridge, MA: MIT Press.

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., & Rosen, D. B. (1992). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, *3*, 698–713. doi:10.1109/72.159059

Carpenter, G. A., Grossberg, S., & Reynolds, J. H. (1991b). ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, *4*, 565–588. doi:10.1016/0893-6080(91)90012-T

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991a). ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, *4*, 493–504. doi:10.1016/0893-6080(91)90045-7

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991c). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, *4*, 759–771. doi:10.1016/0893-6080(91)90056-B

Casdagli, M. (1989). Nonlinear prediction of chaotic time series. *Physica D. Nonlinear Phenomena*, *35*, 335–356. doi:10.1016/0167-2789(89)90074-2

Castellano, G. (1997). An iterative pruning algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks*, *8*(3), 519–537. doi:10.1109/72.572092

Cha, I., & Kassam, S. A. (1996). RBFN restoration of nonlinearly degraded images. *IEEE Transactions on Image Processing*, *5*(6), 964–975. doi:10.1109/83.503912

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Cambridge, MA: MIT Press.

Charalambous, C. (1992). Conjugate gradient algorithm for efficient training of artificial neural networks. *Proceedings of the IEEE*, *139*(3), 301–310.

Chauvin, Y., & Rumelhart, D. E. (1995). *Backpropagation: Theory, architectures, and applications*. Hillsdale, NJ: L. Erlbaum Associates Inc.

Chen, Q., & Weigand, W. A. (1994). Dynamic optimization of nonlinear processes by combining neural net model with UDMC. *AIChE Journal. American Institute of Chemical Engineers*, *40*, 1488–1497. doi:10.1002/aic.690400908

Chen, S., Cowan, C. F. N., & Grant, P. M. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, *2*(2), 302–309. doi:10.1109/72.80341

Chua, W., & Yang, L. (1988a). Cellular networks: Theory. *IEEE Transactions on Circuits and Systems*, *35*(10), 1257–1272. doi:10.1109/31.7600

Chua, W., & Yang, L. (1988b). Cellular networks: Applications. *IEEE Transactions on Circuits and Systems*, *35*(10), 1273–1290. doi:10.1109/31.7601

Cibas, T., Souli, F. F., Gallinari, P., & Raudys, S. (1996). Variable selection with neural networks. *Neurocomputing*, *12*, 223–248. doi:10.1016/0925-2312(95)00121-2

Correction Neural Networks*Modeling and forecasting financial data, techniques of nonlinear dynamics* (Soofi, A., & Cao, L., Eds.). Kluwer Academic Press.

Cristea, P., Tuduce, R., & Cristea, A. (2000). Time series prediction with wavelet neural networks. In *Proceedings of IEEE Neural Network Applications in Electrical Engineering*, *25-27*, 5–10.

Cristianini, N., & Taylor, J. S. (2000). *An introduction to support vector machines and other kernel-based learning methods*. New York, NY: Cambridge University Press.

Cun, Y. L., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. In Arbib, M. A. (Ed.), *The handbook of brain theory and neural networks* (pp. 255–258). Cambridge, MA: MIT Press.

Cun, Y. L., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In Touretzky, D. S. (Ed.), *Advances in neural information processing systems, 2* (pp. 598–605). San Mateo, CA: Morgan Kaufmann.

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, *41*, 909–996. doi:10.1002/cpa.3160410705

Davies, J. R., Coggeshall, S. V., Jones, R. D., & Schutzer, D. (1995). Intelligent security systems. In Freedman, S. R., Flein, A. R., & Lederma, J. (Eds.), *Artificial intelligence in the capital markets*. Chicago, IL: Irwin.

De, S., Bhattacharyya, S., & Dutta, P. (2009). Multilevel image segmentation using OptiMUSIG activation function with fixed and variable thresholding: A comparative study. In Mehnen, J., Koppen, M., Saad, A., & Tiwari, A. (Eds.), *Applications of soft computing: From theory to praxis, advances in intelligent and soft computing* (pp. 53–62). Berlin: Springer-Verlag. doi:10.1007/978-3-540-89619-7_6

De, S., Bhattacharyya, S., & Dutta, P. (2010). Efficient gray level image segmentation using an optimized MUSIG (OptiMUSIG) activation function. *International Journal of Parallel, Emergent and Distributed Systems,* 1-39.

Deschenes, S., Sheng, Y., & Chevrette, P. C. (1998). Three-dimensional object recognition from two-dimensional images using wavelet transforms and neural networks. *Optical Engineering (Redondo Beach, Calif.)*, *37*(3), 763–770.

de Kruif, B. J., & de Vries, T. J. (2003). Pruning error minimization in least squares support vector machines. *IEEE Transactions on Neural Networks*, *14*(3), 696–702. doi:10.1109/TNN.2003.810597

de Oliveira, W. R. (2009). Quantum RAM based neural networks. In *Proceedings of European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning* (pp. 331-336).

de Oliveira, W. R., et al. (2008). Quantum logical neural networks. In *Proceedings of 10th Brazilian Symposium on Neural Networks SBRN '08* (pp. 147-152).

Dingledine, R., Borges, K., Bowie, D., & Traynelis, S. F. (1999). The glutamate receptor ion channels. *Pharmacological Reviews*, *51*(1), 7–61.

Dodd, N. (1990). Optimization of network structure using genetic techniques. In *Proceedings of IEEE International Joint Conference on Neural Networks, 3*, 965-970.

Egmont-Petersen, M., de Ridder, D., & Handels, H. (2002). Image processing with neural networks—A review. *Pattern Recognition*, *35*, 2279–2301. doi:10.1016/S0031-3203(01)00178-9

Ekrafft. (2001). *Wayfinding*. http://www.ifi.uzh.ch/Ekrafft/papers/2001/wayfinding/html/node97.html

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211. doi:10.1207/s15516709cog1402_1

Engelbrecht, A. P. (2001). A new pruning heuristic based on variance analysis of sensitivity information. *IEEE Transactions on Neural Networks*, *12*(6), 1386–1399. doi:10.1109/72.963775

Estevez, P. A., Flores, R. J., & Perez, C. A. (2005). Color image segmentation using fuzzy min-max neural networks. In *Proceedings of IEEE International Joint Conference on Neural Networks,* (vol. 5, pp. 3052-3057).

Estevez, P. A., Perez, C. A., & Goles, E. (2003). Genetic input selection to a neural classifier for defect classification of radiata pine boards. *Forest Prod. Journal*, *53*(7/8), 87–94.

Faber, J., & Giraldi, G. (1989). *Quantum models of artificial neural networks*. Retrieved from http://arquivosweb.lncc.br/pdfs/QNN-Review.pdf

Fausett, L. (1994). *Fundamentals of neural networks*. USA: Prentice Hall.

Freitas, F. D., De Souza, A. F., & de Almeida, A. R. (2009). Prediction-based portfolio optimization model using neural networks. *Neurocomputing*, *72*(10-12), 2155–2170. doi:10.1016/j.neucom.2008.08.019

Gallinari, P. (1995). Training of modular neural net systems. In M. A, Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 582–585), Cambridge, MA: MIT Press.

Gerke, M., & Hoyer, H. (1997). Fuzzy backpropagation training of neural networks. In Reusch, B. (Ed.), *Computational intelligence theory and applications* (pp. 416–427). Berlin, Germany: Springer.

Ghosh, A., Pal, N. R., & Pal, S. K. (1993). Self-organization for object extraction using multilayer neural network and fuzziness measures. *IEEE Transactions on Fuzzy Systems*, *1*(1), 54–68. doi:10.1109/TFUZZ.1993.390285

Ghosh, S., & Ghosh, A. (2002). A GA-fuzzy approach to evolve Hopfield type optimum networks for object extraction. In. *Proceedings of AFSS, 2002*, 444–449.

Goertzel, B. (1989). *Quantum neural networks*. Retrieved from http://goertzel/org/ben/quantnet.html

Goldberg, D. E. (1989). *Genetic algorithms: Search, optimization and machine learning*. New York, NY: Addison-Wesley.

Golub, G. H., Van, L., & Charles, F. (1996). *Matrix computations*. Johns Hopkins.

Gopathy, P., Nicolaos, B., & Karayiannis, N. B. (1997). Quantum neural networks are inherently fuzzy feedforward neural networks. *IEEE Transactions on Neural Networks*, *8*(3), 679–693. doi:10.1109/72.572106

Goswami, J. C., & Chan, A. K. (1999). *Fundamentals of wavelets-Theory, algorithm, and applications*. Wiley Inter-Science.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, *23*, 121–134. doi:10.1007/BF00344744

Grossberg, S. (n.d.). *Anticipatory brain dynamics in perception, cognition, and action.* Retrieved from http://www.cns.bu.edu/Profiles/Grossberg

Grover, L. K. (1998). Quantum computers can search rapidly by using almost any transformation. *Physical Review Letters*, *80*(19), 4329–4332. doi:10.1103/PhysRevLett.80.4329

Hagan, M. T., Demuth, H. B., & Beale, M. H. (1996). *Neural network design*. Boston, MA: PWS-Kent.

Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, *5*(6), 989–993. doi:10.1109/72.329697

Hagiwara, M. (1993). Removal of hidden units and weights for backpropagation networks. In *Proceedings of International Joint Conference on Neural Networks, 1*, 351-354.

Hahn, W. (1963). *Theory and application of Liapunov's direct method*. Englewood Cliffs, NJ: Prentice-Hall.

Hammer, B., Strickert, M., & Villmann, T. (2005). Supervised neural gas with general similarity measure. *Neural Processing Letters*, *21*(1), 21–44. doi:10.1007/s11063-004-3255-2

Harp, S. A., & Samad, T., & Guha. (1989). A. Designing application-specific neural networks using the genetic algorithm. *Advances in Neural Information Processing Systems*, *2*, 447–454.

Hassibi, B., & Stork, D. G. (1993). Second-order derivatives for network pruning: Optimal Brain Surgeon Advances. In Giles, C. L., Hanson, S. J., & Cowan, J. D. (Eds.), *Neural information processing systems, 5* (pp. 164–171). San Mateo, CA: Morgan-Kaufmann.

Haykin, S. (1999). *Neural networks: A comprehensive foundation*. Upper Saddle River, NJ: Prentice Hall.

Hebb, D. O. (1949). *The organization of behavior*. New York: John Wiley.

Hecht-Nielsen, R. (1987). Counterpropagation networks. *Applied Optics*, *26*, 4979–4984. doi:10.1364/AO.26.004979

Ho, D. W. C., Zhang, P. A., & Xu, J. (2001). Fuzzy wavelet networks for function learning. *IEEE Transactions on Fuzzy Systems*, *9*, 200–211. doi:10.1109/91.917126

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*(8), 2554–2558. doi:10.1073/pnas.79.8.2554

Horikawa, S., Furuhashi, T., & Uchikawa, Y. (1992). On fuzzy modeling using fuzzy neural networks with back propagation algorithm. *IEEE Transactions on Neural Networks*, *3*, 801–806. doi:10.1109/72.159069

Huang, X., Jack, M., & Ariki, Y. (1990). *Hidden Markov models for speech recognition*. Edinburgh University Press.

Huang, Y. (2009). Advances in artificial neural networks – Methodological development and application. *Algorithms*, *2*, 973–1007. doi:10.3390/algor2030973

Hunt, K. J., Sbarbaro, D., Zbikowski, R., & Gawthrop, P. J. (1992). Neural networks for control system - A survey. *Automatica*, *28*, 1083–1112. doi:10.1016/0005-1098(92)90053-I

Iftekharuddin, K. M., Schechinger, T. D., & Jemili, K. (1995). Feature-based neural wavelet optical character recognition system. *Optical Engineering (Redondo Beach, Calif.)*, *34*(11), 3193–3199. doi:10.1117/12.213654

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, *1*, 295–307. doi:10.1016/0893-6080(88)90003-2

Jang, J. S. R. (1993). (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, *23*, 665–685. doi:10.1109/21.256541

Jones, R. D., Lee, Y. C., Barnes, C. W., Flake, G. W., Lee, K., Lewis, P. S., & Qian, S. (1990). Function approximation and time series prediction with neural networks. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 17-21).

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Karnin, E. D. (1990). A simple procedure for pruning backpropagation trained neural networks. *IEEE Transactions on Neural Networks*, *1*, 239–242. doi:10.1109/72.80236

Kasabov, N. K. Israel, & S. I., Woodford, B. J. (2000). Adaptive, evolving, hybrid connectionist systems for image pattern recognition. In S. K. Pal, A. Ghosh, & M. K. Kundu (Eds.), *Soft computing for image processing.* Heidelberg, Germany: Physica-Verlag.

Kerr, T. H. (1998). Critique of some neural network architectures and claims for control and estimation. *IEEE Transactions on Aerospace and Electronic Systems*, *34*(2), 406–419. doi:10.1109/7.670323

Kim, Y. H., & Lewis, F. L. (1999). Neural network output feedback control of robot manipulators. *IEEE Transactions on Robotics and Automation*, *15*(2), 301–309. doi:10.1109/70.760351

Kitano, H. (1990). Empirical studies on the speed of convergence of neural network training using genetic algorithms. In *Proceedings of the Eleventh International Conference on Artificial Intelligence* (pp. 789-795).

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*, 59–69. doi:10.1007/BF00337288

Kohonen, T. (1988). *Self-organization and associative memory*. New York, NY: Springer-Verlag.

Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics*, *18*(1), 49–60. doi:10.1109/21.87054

Kumar, S. (2004). *Neural networks: A classroom approach*. New Delhi, India: Tata McGraw-Hill.

Kwan, C., & Lewis, F. L. (2000). Robust backstepping control of nonlinear systems using neural networks. *IEEE Transactions on Systems, Man, and Cybernetics. Part A, Systems and Humans*, *30*(6), 753–766. doi:10.1109/3468.895898

Leonard, J., & Kramer, M. A. (1990). Improvement of the backpropagation algorithm for training neural networks. *Computers & Chemical Engineering*, *14*(3), 337–341. doi:10.1016/0098-1354(90)87070-6

Li, F., & Tsien, J. Z. (2009). Clinical implications of basic research: Memory and the NMDA receptors. *The New England Journal of Medicine*, *361*(302).

Lin, J.-S., Cheng, K.-S., & Mao, C.-W. (1996). A fuzzy Hopfield neural network for medical image segmentation. *IEEE Transactions on Nuclear Science*, *43*(4), 2389–2398. doi:10.1109/23.531787

Liu, Z., Bai, Z., Shi, J., & Chen, H. (2007). Image segmentation by using discrete Tchebichef moments and quantum neural network. In *Proceedings of Third International Conference on Natural Computation (ICNC 2007)*.

Lukaszyk, S. (2004). A new concept of probability metric and its applications in approximation of scattered data sets. *Computational Mechanics*, *33*, 299–3004. doi:10.1007/s00466-003-0532-2

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281-297).

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *11*(7), 674–693. doi:10.1109/34.192463

Mandic, D., & Chambers, J. (2001). *Recurrent neural networks for prediction: Learning algorithms, architectures and stability*. New York, NY: John Wiley & Sons. doi:10.1002/047084535X

Martinetz, T., Berkovich, S., & Schulten, K. (1993). Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, *4*(4), 558–569. doi:10.1109/72.238311

Matthews, G. G. (1991). *Cellular physiology of nerve and muscle*. Boston, MA: Blackwell Scientific Publications.

McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*, 115–133. doi:10.1007/BF02478259

Medin, D. L., & Schaffer, M. M. (1978). Context theory of classification learning. *Psychological Review*, *85*, 207–238. doi:10.1037/0033-295X.85.3.207

Menneer, T., & Narayanan, A. (1995). *Quantum-inspired neural networks*. Technical report R329. Department of Computer Science, University of Exeter, UK.

Meyer, Y. (1993). *Wavelets: Algorithms and applications*. Philadelphia, PA: SIAM.

Miller, G. F., & Todd, P. M. (1990). Exploring adaptive agency in theory and methods for simulating the evolution of learning. In *Connectionist models*. Morgan Kaufmann.

Miller, G. F., Todd, P. M., & Hegde, S. (1989). Designing neural networks using genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 379-384), Arlington, VA: Morgan Kaufmann.

Moein, S., Monadjemi, S. A., & Moallem, P. (2008). A novel fuzzy-neural based medical diagnosis system. *World Academy of Science, Engineering and Technology, 37*.

Mozer, M. C., & Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from a network via relevance assessment. In D. S. Touretzky (Ed.), *Advances in neural information processing systems, 1*, 107-115, San Mateo, CA: Morgan Kaufmann.

Muhammed, H. H. (2002). A new unsupervised fuzzy clustering algorithm (FC-WINN) using the new weighted incremental neural network. In *Proceedings of WINN 2002*.

Murphy, G. L., & Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review*, *92*, 289–316. doi:10.1037/0033-295X.92.3.289

Niranjan, M., & Fallside, F. (1990). Neural networks and radial basis functions in classifying static speech patterns. *Computer Speech & Language*, *4*, 275–289. doi:10.1016/0885-2308(90)90009-U

Nosofsky, R. M., & Palmeri, T. J. (1998). A rule-plus-exception model for classifying objects in continuous-dimension spaces. *Psychonomic Bulletin & Review*, *5*(3), 345–369. doi:10.3758/BF03208813

Nosofsky, R. M., & Zaki, S. R. (2002). Exemplar and prototype models revisited: Response strategies, selective attention, and stimulus generalization. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, *28*(5), 924–940. doi:10.1037/0278-7393.28.5.924

Nossek, J. A., & Roska, T. (1993). Special issue on cellular neural networks. *IEEE Transactions on Circuits and Systems*, *40*(3).

Oliker, S., Furst, M., & Maimon, O. (1992). A distributed genetic algorithm for neural network design and training. *Complex Systems*, *6*, 459–477.

Opitz, D. W., & Shavlik, J. W. (1994). Using genetic search to refine knowledge-based neural networks. In W. Cohen, & H. Hirsh (Eds.), *Machine learning*: *Proceedings of the Eleventh International Conference*. San Fransisco, CA: Morgan Kaufmann.

Pal, S. K., & Ghosh, A. (1996). Neuro-fuzzy computing for image processing and pattern recognition. *International Journal of Systems Science*, *27*(12), 1179–1193. doi:10.1080/00207729608929325

Pan, L., Zheng, H., & Nahavandi, S. (2003). The application of rough set and Kohonen network to feature selection for object extraction. *In Proceedings of the Second International Conference on Machine Learning and Cybernetics* (pp. 1185-1189).

Paul, V. Y., & Haykin, S. (2001). *Regularized radial basis function networks: Theory and applications*. John Wiley.

Pawlak, Z. (1982). Rough sets. *International Journal of Computer and Information Sciences*, *11*, 341–356. doi:10.1007/BF01001956

Peters, J. F., Han, L., & Ramanna, S. (2001). Rough neural computing in signal analysis. *Computational Intelligence*, *17*(3), 493–513. doi:10.1111/0824-7935.00160

Peters, J. F., Skowron, A., Han, L., & Ramanna, S. (2000). Towards rough neural computing based on rough membership functions: Theory and application. In *RSCTC* (*Vol. 2005*, pp. 611–618). LNAI. doi:10.1007/3-540-45554-X_77

Pfister, M. (1995). *Hybrid learning algorithms for neural networks*. PhD Thesis, Free University, Berlin.

Pfister, M., & Rojas, R. (1993). Speeding-up backpropagation – A comparison of orthogonal techniques. *In Proceedings of International Joint Conference on Neural Networks* (pp. 517–523), Japan.

Phung, S. L., & Bouzerdoum, A. (2007). A pyramidal neural network for visual pattern recognition. *IEEE Transactions on Neural Networks*, *18*(2), 329–343. doi:10.1109/TNN.2006.884677

Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, *78*(9), 1484–1487. doi:10.1109/5.58326

Ponnapalli, P. V. S., Ho, K. C., & Thomson, M. (1999). A formal selection and pruning algorithm for feedforward artificial neural network optimization. *IEEE Transactions on Neural Networks*, *10*(4), 964–968. doi:10.1109/72.774273

Prechelt, L. (1997). Connection pruning with static and adaptive pruning schedules. *Neurocomputing*, *16*(1), 49–61. doi:10.1016/S0925-2312(96)00054-9

Reed, R. (1993). Pruning algorithms - A survey. *IEEE Transactions on Neural Networks*, *4*(5), 740–747. doi:10.1109/72.248452

Reichert, H. (1990). *Neurobiologie*. Stuttgart, Germany: Georg Thieme.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proceedings of IEEE International Conference on Neural Networks* (pp. 586-591), San Francisco.

Rojas, R. (1996). *Neural networks: A systematic introduction*. Berlin, Germany: Springer-Verlag.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–408. doi:10.1037/h0042519

Ruck, D. W., Rogers, S. K., & Kabrisky, M. (1990). Feature selection using a multilayer perceptron. *Neural Network Computing*, *2*(2), 40–48.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., & McClelland, J. L. (Eds.), *Parallel distributed processing: Explorations in the microstructures of cognition* (*Vol. I*, pp. 318–362). Cambridge, MA: MIT Press.

Russo, F. (1999). Hybrid neuro-fuzzy filter for impulse noise removal. *Pattern Recognition*, *32*(11), 1843–1855. doi:10.1016/S0031-3203(99)00009-6

Russo, F. (2000). Image filtering using evolutionary neural fuzzy systems. In *Soft computing for image processing* (pp. 23-43). S. K. Pal, A. Ghosh, M. K. Kundu (Eds.), *Proceedings of IEEE International Conference on Industrial Technology Vol. II* (pp. 335-340). Physica-Verlag, Heidelberg.

Salomon, R. (1992). Verbesserung konnektionistischer Lernverfahren. *die nach der Gradientenmethode arbeiten*, PhD Thesis, Technical University of Berlin.

Sanger, T. D. (1994). Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE Transactions on Robotics and Automation*, *10*, 323–333. doi:10.1109/70.294207

Sanner, R. M., & Slotine, J. J. E. (1992). Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, *3*, 837–863. doi:10.1109/72.165588

Scales, L. E. (1985). *Introduction to Non-linear optimization*. New York, NY: Springer-Verlag.

Sengupta, S. (1995). A comparative study of neural network and regression analysis as modeling tools. In *Proceedings of the Twenty-Seventh Southeastern Symposium on System Theory* (pp. 202-205).

Shanno, D. F. (1990). Recent advances in numerical techniques for large-scale optimization. In S. Miller, & Werbos (Eds.), *Neural networks for control.* Cambridge, MA: MIT Press.

Shashidhara, H. L., Lohani, S., & Gadre, V. M. (2000). Function learning wavelet neural networks. In *Proceedings of IEEE International Conference on Industrial Technology Vol. II* (pp. 335-340).

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, *26*(5), 1484–1509. doi:10.1137/S0097539795293172

Sietsma, J., & Dow, R. J. F. (1991). Creating artificial neural networks that generalize. *Neural Networks*, *4*, 67–79. doi:10.1016/0893-6080(91)90033-2

Silva, F., & Almeida, L. (1990). Speeding-up backpropagation. In Eckmiller, R. (Ed.), *Advanced neural computers* (pp. 151–156). Amsterdam, The Netherlands: North-Holland.

Simpson, P. K. (1993). Fuzzy min-max neural networks: Part 2-Clustering. *IEEE Transactions on Fuzzy Sets*, *1*, 32–45. doi:10.1109/TFUZZ.1993.390282

Sklansky, J., & Vriesenga, M. (1996). Genetic selection and neural modeling of piecewise-linear classifiers. *International Journal of Pattern Recognition and Artificial Intelligence*, *10*(5), 587–612. doi:10.1142/S0218001496000360

Smith, J. D., & Minda, J. P. (1998). Prototypes, sage, exemplars and thyme. In *Proceedings of 90th Annual Meeting of the Southern Society for Philosophy and Psychology*, New Orleans, LA. Minda, J. P., & Smith, J. D. (1997). Prototypes in category learning. In *Proceedings of the 38th Annual Meeting of the Psychonomic Society*, Philadelphia, PA.

Snyman, J. A. (2005). *Practical mathematical optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms*. New York, NY: Springer Publishing.

Somervuo, P., & Kohonen, T. (2004). *Self-organizing maps and learning vector quantization for feature sequences*. Netherlands: Kluwer Academic Publishers.

Steane, A. M., & Rieffel, E. G. (2000). Beyond bits: The future of quantum information processing. *IEEE Computers*, *33*(1), 38–45. doi:10.1109/2.816267

Steppe, J. M., Bauer, K. W., & Rogers, S. K. (1996). Integrated feature and architecture selection. *IEEE Transactions on Neural Networks*, *7*, 1007–1014. doi:10.1109/72.508942

Steward, O. (1989). *Principles of cellular, molecular, and developmental neuroscience*. New York, NY: Springer-Verlag. doi:10.1007/978-1-4612-3540-8

Thompson, R. (1990). *Das Gehirn: Von der Nervenzelle zur Verhaltenssteuerung*. Heidelberg, Germany: Spektrum der Wissenschaft.

Tzeng, Y. C., & Chen, K. S. (1998). A fuzzy neural network to SAR image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *36*(1), 301–307. doi:10.1109/36.655339

Vapnik, V., Golowich, S., & Smola, A. (1997). Support vector method for function approximation, regression estimation, and signal processing. In Mozer, M., Jordan, M., & Petsche, T. (Eds.), *Advances in neural information processing systems 9* (pp. 281–287). Cambridge, MA: MIT Press.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. London, UK: Springer-Verlag.

Vapnik, V. N. (1998). *Statistical learning theory*. New York, NY: Wiley.

Ventura, D., & Martinez, T. (1998). Quantum associative memory with exponential capacity. In *Proceedings of the International Joint Conference on Neural Networks* (pp.509-513).

Villmann, T., Hammer, B., Schleif, F.-M., & Geweniger, T. (2005). Fuzzy labeled neural GAS for fuzzy classification. In *Proceedings of the Workshop on Self-Organizing Maps WSOM* (pp. 283-290).

Walter, J., & Ritter, H. (1995). Local PSOMs and Chebyshev PSOMs improving the parameterized self-organizing maps. In F. Fogelman-Soulie (Ed.), *Proceedings of International Conference on Artificial Neural Networks*.

Wang, X. G., Tang, Z., Tamura, H., Ishii, M., & Sun, W. D. (2004). An improved backpropagation algorithm to avoid the local minima problem. *Neurocomputing*, *56*, 455–460. doi:10.1016/j.neucom.2003.08.006

Weymaere, N., & Martens, J. P. (1994). On the initialization and optimization of multilayer perceptrons. *IEEE Transactions on Neural Networks*, *5*, 738–751. doi:10.1109/72.317726

White, D., & Ligomenides, P. (1993). GANNet: A genetic algorithm for optimizing topology and weights in neural network design. In J. Mira, J. Cabestany, & A. Prieto (Eds.), *New Trends in neural computation* In *Proceedings of International Workshop on Artificial Neural Networks* (pp. 332-327). Berlin, Germany: Springer-Verlag.

Whitley, D., & Bogart, C. (1990). The evolution of connectivity: Pruning neural networks using genetic algorithms. In *Proceedings of International Joint Conference on Neural Networks, 1*, 134-137.

Wikipedia. (n.d.). *Hopfield net.* Retrieved from www.wikipedia.org/wiki/Hopfield_net

Wikipedia. (n.d.). *NMDA receptor*. Retrieved from www.wikipedia.org/wiki/NMDA_receptor

Wikipedia. (n.d.). *Radial base function*. Retrieved from www.wikipedia.org/wiki/Radial_basis_function

Wikipedia. (n.d.). *Radial basis function network*. Retrieved from www.wikipedia.org/wiki/Radial_basis_function_network

Williams, C. P., & Clearwater, S. H. (1998). *Explorations in quantum computing*. New York, NY: Springer-Verlag.

Yu, X., Loh, N. K., & Miller, W. C. (1993). A new acceleration technique for the backpropagation algorithm. In *Proceedings of IEEE International Conference on Neural Networks, Vol. III* (pp. 1157-1161), San Diego, CA.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*, 338–353. doi:10.1016/S0019-9958(65)90241-X

Zaknich, A. (2003). *Neural networks for intelligent signal processing*. World Scientific.

Zamparelli, M. (1997). Genetically trained cellular neural networks. *Neural Networks*, *10*(6), 1143–1151. doi:10.1016/S0893-6080(96)00128-1

Zeng, X., & Chen, X.-W. (2005). SMO-based pruning methods for sparse least squares support vector machines. *IEEE Transactions on Neural Networks*, *16*(6), 1541–1546. doi:10.1109/TNN.2005.852239

Zhou, G., & Si, J. (1999). Subset-based training and pruning of sigmoid neural networks. *Neural Networks*, *12*(1), 79–89. doi:10.1016/S0893-6080(98)00105-1

Zimmermann, H. G., Neuneier, R., & Grothmann, R. (2001). Modeling of dynamical systems by error correction neural networks. In Soofi, A., & Cao, L. (Eds.), *Modeling and forecasting financial data: Techniques of nonlinear dynamics*. Kluwer Academic Pub.doi:10.1007/978-1-4615-0931-8_12

Zurada, J. M., Malinowski, A., & Cloete, I. (1994). Sensitivity analysis for minimization of input data dimension for feedforward neural network. In *Proceedings of IEEE International Symposium on Circuits and Systems*.

Zurada, J. M., Malinowski, A., & Usui, S. (1997). Perturbation method for deleting redundant inputs of perceptron networks. *Neurocomputing*, *14*(2), 177–193. doi:10.1016/S0925-2312(96)00031-8

ADDITIONAL READING

Chong, E. K. P., & Zak, S. H. (1996). *An Introduction to Optimization*. New York: Wiley.

Egmont-Petersen, M., de Ridder, D., & Handels, H. (2002). Image processing with neural networks—a review. *Pattern Recognition*, *35*, 2279–2301. doi:10.1016/S0031-3203(01)00178-9

Faber, J., & Giraldi, G. Quantum models of artificial neural networks. Retrieved from http://arquivosweb.lncc.br/pdfs/QNN-Review.pdf

Fausett, L. (1994). *Fundamentals of Neural Networks*. USA: Prentice Hall.

Goertzel, B. Quantum Neural Networks. http://goertzel/org/ben/quantnet.html Buhmann, M. D. (2003). *Radial Basis Functions: Theory and Implementations*. Cambridge University Press.

Goldberg, D. E. (1989). *Genetic Algorithms: Search, Optimization and Machine Learning*. New York: Addison-Wesley.

Goswami, J. C., & Chan, A. K. (1999). *Fundamentals of Wavelets-theory, Algorithm, and Applications*. Wiley Inter-Science.

Hagan, M. T., Demuth, H. B., & Beale, M. H. (1996). *Neural Network Design*. Boston, MA: PWS-Kent.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. (National Academy of Sciences: Washington, DC, USA.). *Proceedings of the National Academy of Sciences of the United States of America*, *79*(8), 2554–2558. doi:10.1073/pnas.79.8.2554

Huang, Y. (2009). Advances in Artificial Neural Networks – Methodological Development and Application. *Algorithms*, *2*, 973–1007. doi:10.3390/algor2030973

Kohonen, T. (1988). *Self-Organization and Associative Memory*. New York: Springer-Verlag.

Meyer, Y. (1993). *Wavelets: Algorithms and Applications*. Philadelphia: SIAM.

Pal, S. K., & Ghosh, A. (1996). Neuro-fuzzy computing for image processing and pattern recognition. *International Journal of Systems Science*, *27*(12), 1179–1193. doi:10.1080/00207729608929325

Pawlak, Z. (1982). Rough sets. *International Journal of Computer and Information Sciences*, *11*, 341–356. doi:10.1007/BF01001956

Rojas, R. (1996). *Neural Networks: A Systematic Introduction*. Berlin: Springer-Verlag.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. London: Springer-Verlag.

www.wikipedia.org/wiki/Hopfield_net

www.wikipedia.org/wiki/Radial_basis_function

www.wikipedia.org/wiki/Radial_basis_function_network

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*, 338–353. doi:10.1016/S0019-9958(65)90241-X

KEY TERMS AND DEFINITIONS

Artificial Neural Network: A simulated model of the biological neural network to mimic the functions of the human nervous system.

Backpropagation: An iterative algorithm to adjust the interlayer interconnection weights of an artificial neural network architecture based on the minimization of the network errors with time.

Biological Neuron: The basic unit of a human nerve cell comprising a cell body, dendrites, axon and synapses.

Fuzzy Set: A soft computing tool to explain the ambiguity, vagueness and uncertainty in real world knowledge bases.

Genetic Algorithm: A heuristic search methodology for achieving an optimum solution to combinatorial problems.

Multilayer Perceptron: A multiple layered artificial neural network model capable of classifying nonlinear datasets.

Perceptron: The basic model of an artificial neural network comprising of a single input layer and a single output layer.

Quantum Neural Networks: Quantum computing inspired efficient and faster neural network architectures.

Rough Set: A soft computing tool to describe imprecise data based on approximations.

Self-Organizing Feature Map: A topology preserving artificial neural network architecture capable of learning through self-supervision of incident input features.

Supervised Learning: An artificial neural network learning paradigm using a training set of input-output pairs and their relationships in presence of an external supervisor.

Unsupervised Learning: An artificial neural network learning paradigm where the network adapts to the incident inputs and its features.