NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

Intelligent Pattern Recognition System Using BPNN. (Signatures Recognition)

COM - 400

Student: Abd Almuti Albaba (20033334)

Supervisor: Asst. Prof. Dr. Elbrus Imanov

Lefkoşa – 2007

ACKNKNOWLEDGMENTS

First of all, I want to thanks my almighty God for reaching me my goals, and gave me bravery and helped me carry on.

More over I want to pay special regards to my lovely parents who are enduring these all expenses and supporting me in all events. I am nothing without their prayers. They also encouraged me in crises . I shall never forget their sacrifices for my education so that I can enjoy my successful life as they are expecting. They may get peaceful life in heaven. At the end I am again thankful to those persons who helped me or even encouraged me to complete me, my project. My all efforts to complete this project might be fruitful.

The special thank is to my best teachers who are really supports me and they were close to mind with all of their efforts to create from us a qualified persons, those are my project advisor Assist. Prof. Dr Elbrus Imanov , special thanks goes to my advisor during the Mr. Okan Donangel.

I will never forget the help that I got it from this university for continuing my education .

Lastly I would also like to thank all my friends for their help and for their patience specialy Osama al Kurd, Khaled A'amar, Mohammad al Fa'ory .

ABSTRACT

As the pattern recognition excersises its influence during a several fields of life ,where it is necessary to be implemented proparely to get the desired result .
Artificial neural network using backpropagation method is used for a signature recognition as a convenient mean to detect and validate the documents and reliable techniques for signature verification which are consequently requested , thus , three different types of forgeries are taken into consideration , random forgeries , simple forgeries and skilled forgeries that from enough reasons to achieve a mission bu taking signature samples for training the neural network then after testing the given signature accordingly .
neural network structures represent models for "thinking" that are still being evolved in the laboratories. Yet, all of these networks are simply tools and as such the only real demand they make is that they require the network architect to learn how to use them.
Artificial Neural Networks are being touted as the wave of the future in computing. They are indeed self learning mechanisms which don't require the traditional skills of a programmer. But unfortunately, misconceptions have arisen. Writers have hyped that these neuron-inspired processors can do almost anything. These exaggerations have created disappointments for some potential users who have tried, and failed, to solve their problems with neural networks. These application builders have often come to the conclusion that neural nets are complicated and confusing.
Unfortunately, that confusion has come from the industry itself. Avalanches of articles have appeared touting a large assortment of different neural networks, all with unique claims and specific examples. Currently, only a few of these neuron-based structures, paradigms actually, are being used commercially. One particular structure, the feed forward, back-propagation network, is by far and away the most popular

INTRODUCTION

Pattern recognition covers a wide range of activities from many walks of life. It is something which we humans are particularly good at; we receive data from our senses and are often able, immediately and without conscious effort, to identify the source of the data. For example, many of us can recognize faces we have not seen for many years, even in disguise, recognize voices over a poor telephone line, as babies recognize our mothers by smell, distinguish the grapes used to make a wine, and sometimes even recognize the vineyard and year, identify thousands of species of flowers and spot an approaching storm. Science, technology and business has brought to us many similar tasks, including diagnosing diseases, detecting abnormal cells in cervical smears, recognizing dangerous driving conditions, identifying types of car, aero plane identifying suspected criminals by fingerprints and DNA profiles, reading Zip codes (US postal codes) on envelopes, reading hand-written symbols (on a pen pad computer), reading maps and circuit diagrams, classifying galaxies by shape, picking an optimal move or strategy in a game such as chess, identifying incoming missiles from radar or sonar signals, detecting shoals of fish by sonar, checking packets of frozen peas for 'foreign bodies', spotting fake 'antique' furniture, deciding which customers will be good credit risks and spotting good opportunities on the financial markets. [1]

Artificial Neural Networks are being touted as the wave of the future in computing. They are indeed self learning mechanisms which don't require the traditional skills of a programmer. But unfortunately, misconceptions have arisen. Writers have hyped that these neuron-inspired processors can do almost anything. These exaggerations have created disappointments for some potential users who have tried, and failed, to solve their problems with neural networks. These application builders have often come to the conclusion that neural nets are complicated and confusing. Unfortunately, that confusion has come from the industry itself. Avalanches of articles have appeared touting a large assortment of different neural networks, all with unique claims and specific examples. Currently, only a few of these neuron-based structures, paradigms actually, are being used commercially. One particular structure, the feed forward, back-propagation network, is by far and away the most popular. Most of the other

neural network structures represent models for "thinking" that are still being evolved in the laboratories. Yet, all of these networks are simply tools and as such the only real demand they make is that they require the network architect to learn how to use them.

This assignment will consist from four sections.

chapter one is an artificial neural network.

chapter two is about intelligent pattern recognition.

chapter three is Back propagation algorithm.

chapter four will introduce the signature recognition.

The fully detailed explanation will be included about the algorithm block diagram, the neural network parameters adjustment, software program and the results of the signature recognition with addition to the future work

TABLE OF CONTENTS

1. ARTIFICIAL NEURAL NETWORKS

1.1 Overview

This chapter presents an overview of neural networks, its history, simple structure, biological analogy and the Backpropagation algorithm.

In both the Perceptron Algorithm and the Backpropagation Producer, the correct output for the current input is required for learning. This type of learning is called **supervised learning**. Two other types of learning are essential in the evolution of biological intelligence: **unsupervised learning** and reinforcement learning. In unsupervised learning a system is only presented with a set of exemplars as inputs. The system is not given any external indication as to what the correct responses should be nor whether the generated responses are right or wrong. Statistical clustering methods, without knowledge of the number clusters, are examples of *unsupervised learning*. **Reinforcement learning** is somewhere between *supervised learning*, in which the system is provided with the desired output, and *unsupervised learning*, in which the system gets no feedback at all on how it is doing. In reinforcement learning the system receivers a feedback that tells the system whether its output response is right or wrong, but no information on what the right output should be is provided.[27]

1.2 Neural Network Definition

First of all, when we are talking about a neural network, we should more properly say "artificial neural network" (ANN) because that is what we mean most of the time. Biological neural networks are much more complicated than the mathematical models we use for ANNs, but it is customary to be lazy and drop the "A" or the "artificial".

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems

involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

- **Definition:**
  A machine that is designed to model the way in which the brain preference a particular taste or function. The neural network is usually implemented using electronic components or simulated as software.

- **Simulated:**
  A neural network is a massive, parallel-distributed processor made up of simple processing units, which has neural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:
  1. The network from its environment through a learning process acquires knowledge.
  2. Interneuron connection strength, known as synaptic weights, are used to store the acquired knowledge.

- **Simulated:**
  A neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.

- **Simulated:**
  A neural network is a massive, parallel-distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:
  1. Knowledge is acquired by the network through a learning process.
  2. Interneuron connection strengths, known as synaptic weights are used to store the knowledge.

- **Simulated:**

  A neural network is a computational model that shares some of the properties of the brain. It consists of many simple units working in parallel with no central control; the connections between units have numeric weights that can be modified by the learning element.

- **Simulated:**

  A new form of computing inspired by biological models, a mathematical model composed of a large number of processing elements organized into layers.
  "A computing system made up of a number of simple ,highly interconnected elements, which processes information by its dynamic state response to external inputs"

Neural networks go by many aliases. Although by no means synonyms the names listed in figure 1.1 below.

- Parallel distributed processing models
- Connectivist /connectionism models
- Adaptive systems
- Self-organizing systems
- Neurocomputing
- Neuromorphic systems

Figure 1.1 Neural Network Aliases

All refer to this new form of information processing; some of these terms again when we talk about implementations and models. In general though we will continue to use the words "neural networks" to mean the broad class of artificial neural systems. This appears to be the one most commonly used

1.3 History of Neural Networks

1.3.1 Conception (1890-1949)

Alan Turing was the first to use the brine as a computing paradigm, a way of looking at the world of computing. That was in 1936. In 1943, a Warren McCulloch, a neurophysiologist, and Walter Pitts, an eighteen–year old mathematician, wrote a paper about how neurons might work. They modeled a simple neural network with electrical circuits. John von Neumann used it in teaching the theory of computing machines. Researchers began to look to anatomy and physiology for clues about creating intelligent machines.

Another important book was Donald Hebb's the Organization of Behavior (1949) [2], which highlights the connection between psychology and physiology, pointing out that a neural pathway is reinforced each time it is used. Hebb's "Learning Rule", as it is sometime known, is still used and quoted today.

1.3.2 Gestation (1950s)

Improvements in hardware and software in the 1950s ushered in the age of computer simulation. It became possible to test theories about nervous system functions. Research expanded and neural network terminology came into its own.

1.3.3 Birth (1956)

The Dartmouth Summer Research Project on Artificial Intelligence (AI) in the summer of 1956 provided momentum for both the field of AI and neural computing. Putting together some of the best minds of the time unleashed a whole raft of new work. Some efforts took the "high-level" (AI) approach in trying to create computer programs that could be described as "intelligent" machine behavior; other directions used mechanisms modeled after "low-level" (neural network) processes of the brain to achieve "intelligence".[7]

1.3.4 Early Infancy (Late 1950s-1960s)

The year following the Dartmouth Project, John von Neumann wrote material for his book The Computer and the Brain (Yale University Press, 1958). Here he makes such suggestions as imitating simple neuron function by using telegraph relays or vacuum. The Perceptron, a neural network model about which we will hear more later, built in hardware, is the oldest neural network and still has use today in various form for applications such as character recognition.

In 1959, Bernard Widrow and Marcian Hoff (Stanford) developed models for ADALINE, then MADALINE (Multiple ADAptive LINer Elements). This was the first neural network applied to real-world problem-adaptive filers to eliminate echoes on phone lines. As we mentioned before, this application has been in commercial use for several decades.

One of the major players in the neural network reach from to the 1960s to current time is Stephen Grossberg (Boston University). He has done considerable writing (much of it tedious) on his extensive physiological research to develop neural network models. His 1967 network, Avalanche, uses a class of networks to perform activities such as continuous-speech recognition and teaching motor commands to robotic arms.[10]

1.3.5 Excessive Hype

Some people exaggerated the potential of neural networks. Biological comparisons were blown out of proportion in the October 1987 issue of the "Neural Network Review", newsletter editor Craig Will quoted Frank Rosenblatt from a 1958 issue of the "New Yorker".

1.3.6 Stunted Growth (1969-1981)

In 1969 in the midst of such outrageous claims, respected voices of critique were raised that brought a halt too much of the funding for neural network research. Many researchers turned their attention to AI, which looked more promising at the time.

- Amari (1972) independently introduced the additive model of a neural and used it to study the dynamic behavior of randomly connected neuron like elements.
- Wilson and Cowan (1972) derived coupled nonlinear differential equations for the dynamic of spatially localized populations containing both excitatory and inhibitory model neurons.

- Little and Shaw (1975) described a probabilistic of a neuron, either firing or not firing an action potential and used the model to develop a theory of short term memory.
- Anderson Silverstein Ritz and Jones (1977) proposed the brain state in a box (BSB) model consisting of simple associative network coupled to nonlinear dynamics. [14]

1.3.7 Late Infancy (1982 –Present)

Important development in 1982 was the publication of Kohonen's paper on self-organizing maps "Kohonen 1982", which used a one or two dimensional lattice structure.

In 1983,Kirkpatrick, Gelatt, and Vecchi described a new procedure called simulated annealing, for solving combinatorial optimization problems. Simulated annealing is rooted in statistical mechanics.

Jordan (1996) by used a mean-field theory a technique also in statistical mechanics.

A paper by Bator, Sutton and Anderson on reinforcement learning was published in 1983. Although, they were not the first to use reinforcement learning (Minsky considered it in his 1954 Ph.D. thesis for example).

In 1984 Braitenberg's book, Vehicles: Experiments in Synthetic Psychology, was published.

In 1986 the development of the back-propagation algorithm was reported by Rumelhart Hinton and Williams (1986).

In 1988 Linsker described a new principle for self-organization in a perceptual network (Linsker, 1988a) Also in 1988, Broomhead and Lowe described a procedure for the design of layered feed-forward networks using radial basis functions (RBF) which provide an alter native to multiplayer perceptrons.

In 1989 Mead's book, Analog VLSI and Neural Systems, was published. This book provides an unusual mix of concepts drawn from neurobiology and VLSI technology.

In the early 1990s, Vapnik and coworkers invented a computationally powerful class of supervised leaning networks called Support Vector Machines, for solving pattern recognition regression, and the density estimation problem. "Boser, Guyon and Vapnik, 1992, Cortes and Vapnik, 1995; Vapnik, 1995,1998."

In 1982 the time was rip for renewed interest in neural networks. Several events converged to make this a pivotal year.

John Hopfield (Caltech) presented his neural network paper to the National Academy of Sciences. Abstract ideas became the focuse as he pulled together previous work on neural networks.

But there were other threads pulling at the neural network picture as well. Also in 1982 the U.S. – Japan Joint Conference on Cooperative Competitive Neural Network, was held in Kyoto Japan.

In 1985 the American Institute of Physics began what has become an annual Neural Networks for computing meeting. This was the first of many more conference to come in 1987 the institute of Electrical and Electronic Engineers (IEEE). The first international conference on neural networks drew more than 1800 attendees and 19 vendors (although there were few products yet to show). Later the same year, the International Neural Network Society (INNS), was formed under the leadership of Grossberg in the U.S., Kohonen in Finland, and Amari in Japan.

AI though there were two competing conferences in 1988, the spirit of cooperation in this new technology has resulted in joint spontional Joint Conference on Neural Networks (IJCNN) held in Japan in 1989 which produce 430 papers, 63 of which focused on application development. January 1990 IJCNN in Washington, D.C. clouded an hour's concert of music generated by neural networks. The Neural Networks for Defense meeting, held in conjunction with the June 1989 IJCNN above, gathered more than 160 represntives of government defense and defense contractors giving presentations on neural network efforts. When the U.S. Department of Defense announced its 1990 Small Business Innovation Program 16 topics specifically targeted neural networks. An additional 13 topics mentioned the possibility of using neural network approaches. The year of 1989 was of unfolding application possibilities. On

September 27, 1989, the IEEE and the Learning Neural Networks Capabilities created applications for today and the Future.

The ICNN in 1987 included attendees from computer science electrical engineering, physiology cognitive psychology, medicine and even a philosopher of two. In May of 1988 the North Texas Commission Regional Technology Program convened a study group for the purpose of reviewing the opportunities for developing the field of computational neuroscience. Their report of October 1988 concluder that the present is a critical time to establish such a center. [1]

Believing that a better scientific understanding of the brain and the subsequent application to computing technology could have significant impact. They assess their regional strength in electronics and biomedical science and their goals are both academic and economic. You can sense excitement and commitment in their plans.

Hecht-Nielsen (1991) attributes a conspiratorial motive to Minsky and Papert. Namely, that the MIT Al Laboratory had just been set up and was focussing on LISP based Al, and needed to spike other consumers of grants. A good story, whatever the truth, and given extra spice by the coincidence that Minsky and Rosenblatt attended the same class in high-school. Moreover, any bitterness is probably justified because neural network researchers spent the best part of 20 years in the wilderness.

Work did not stop however, and the current upsurge of interest began in 1986 with the famous PDP books which announced the invention of a viable training algorithm (backpropogation) for multilayer networks (Rumelhart and McClelland, 1986). [23]

Table 1.1. Summarize the history of the development of N.N.

Table 1.1 Development of N.N.

Present	Late 80s to now	Interest explodes with conferences, articles, simulation, new companies, and government funded research.
Late Infancy	1982	Hopfiled at National Academy of Sciences
Stunted Growth	1969	Minsky & Papert's critique Perceptrons
Early Infancy	Late 50s, 60s	Excessive Hype Research efforts expand
Birth	1956	AI & Neural computing Fields launched Dartmouth Summer Research Project
Gestation	1950s	Age of computer simulation
	1949	Hebb, the Organization of Behavior
	1943	McCulloch & Pitts paper on neurons
	1936	Turing uses brain as computing paradigm
Conception	1890	James, Psychology (Briefer Curse)

1.4 Analogy to the Brain

The human nervous system may be viewed as a three stage system, as depicted in the block diagram of the block diagram representation of the nervous system.

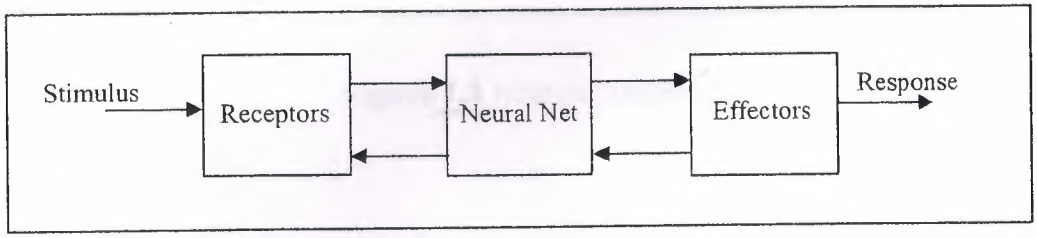


Figure 1.2 Block Diagram of the Nervous System.

(Arbib,1987) Central to the system is the brain, represented by the neural (nerve) network which continually receives information, perceives if, and makes appropriate decisions. Two sets of arrows are shown in the block diagram. Those pointing from left

to right indicate the forward transmission of information-bearing signals through the system. The receptors convert stimuli from the human body or the external environment into electrical impulses which convey information to the neural network (brain). The effectors convert electrical impulses by the neural network into discernible responses as system outputs.

1.4.1 Natural Neuron

A neuron is a nerve cell with all of its processes. Neurons are one of the main distinctions of animals (plants do not have nerve cells). Between seven and one hundred different classes of neurons have been identified in humans. The wide variation is related to how restrictively a class is defined. We tend to think of them as being microscopic, but some neurons in your legs are as long three meters. The type of neuron found in the retina is shown in figure 1.3.

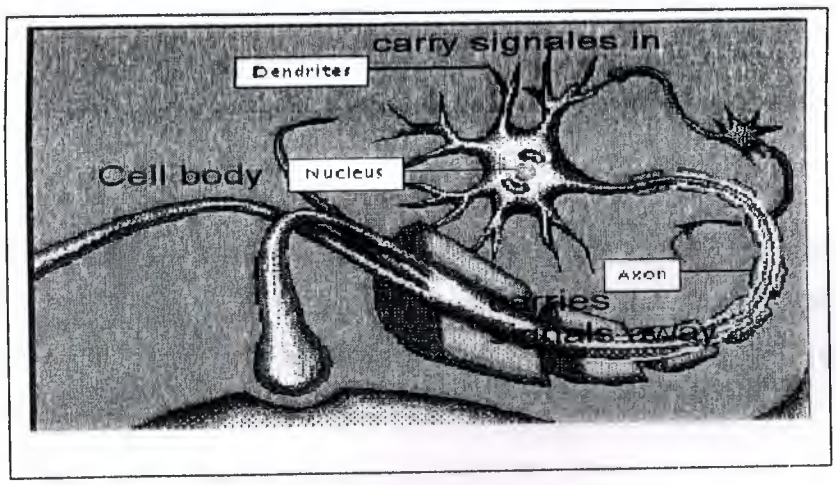


Figure 1.3 Neuron Natural. [23]

An example is a bipolar neuron. Its name implies that has two processes. The cell body contains the nucleus, and leading into the nucleus are one or more dendrites. These branching, tapering processes of the nerve cell, as a rule, conduct impulses toward the cell body. The axon is the nerve cell process that conducts the impulse type of neurons. This one gives us the functionality and vocabulary we need to make analogies.

1.4.2 Artificial Neuron

Our paper and pencil model starts by copying the simplest element the neuron call our artificial neuron a processing element or PE for short. The word node is also used for this simple building block, which is represented by circle in the figure 1.4 "a single mode or processing element PE or Artificial Neuron"

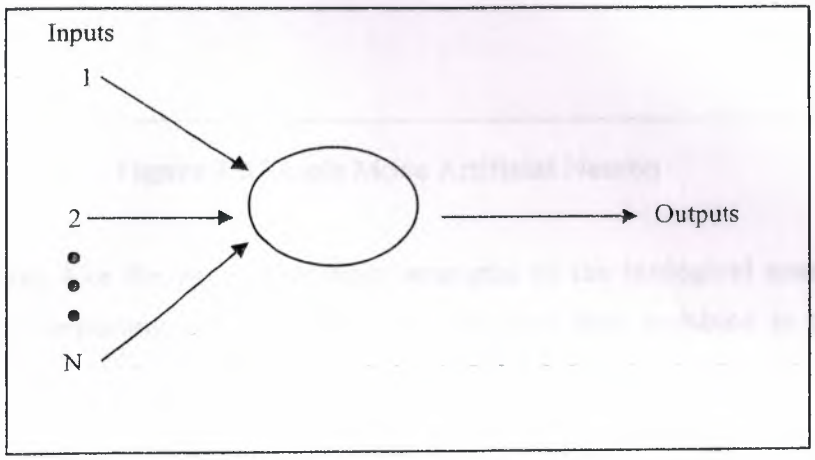


Figure 1.4 Artificial Neuron

The PE handles several basic functions: (1) Evaluates the input signals and determines the strength of each one, Calculates the total for the combined input signals and compare that total to some threshold level, and (3) Determines what the output should be.

Input and Output: Just as there are many inputs (stimulation levels) to a neuron there should be many input signals to our PE. All of them should come into our PE simultaneously. In response a neuron either "fires" or "doesn't fire" depending on some threshold level. The PE will be allowed a single output signal just as is present in a biological neuron. There are many inputs and only one output.

Weighting Factors: Each input will be given a relative weighting which will affect the impact of that input. In figure 1.5, "a single mode or processing element PE or Artificial Neuron" with weighted inputs.

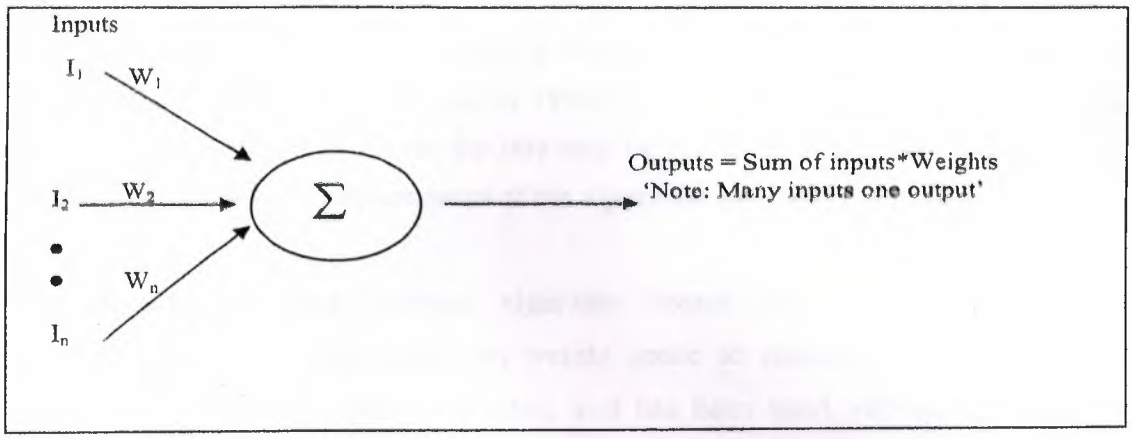


Figure 1.5 Single Mode Artificial Neuron

This is something like the varying synaptic strengths of the biological neurons. Some inputs are more important than others in the way that they combine to produce an impulse.

1.5 Model of a Neuron

The neuron is the basic processor in neural networks. Each neuron has one output, which generally related to the state of the neuron –its activation, which may fan out to several other neurons. Each neuron receives several inputs over these connections, called synapses. The inputs are the activations of the neuron. This is computed by applying a threshold function to this product. An abstract model of the neuron is shown in figure 1.6.

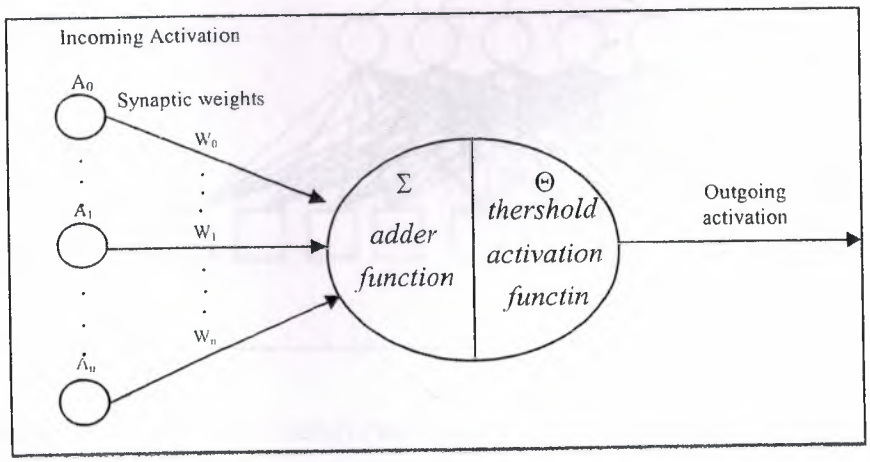


Figure 1.6 Diagram of Abstract Neuron Model. [23]

1.6 Back-Propagation

The most popular method for learning in the multiplayer network is called "back-propagation." It was first invented in 1996 by Bryson, but was more or less ignored until the mid-1980s. The reason for this may be sociological, but may also have to do with the computational requirements of the algorithm on nontrivial problems.

The **back-propagation** learning algorithm works on multiplayer feed-forward networks, using gradient descent in weight space to minimize the output error. It converges to a locally optimal solution, and has been used with some success in a variety of applications. As with all hill-climbing techniques, however, there is no guarantee that it will find a global solution. Furthermore, its converge is often very slow.

1.6.1 Back-Propagation Learning

Suppose we want to construct a network for the restaurant problem. So we will try a two-layer network. We have ten attributes describing each example, so we will need ten input units. In figure 1.7, we show a network with four hidden nits. This turns out to be about right for this problem.

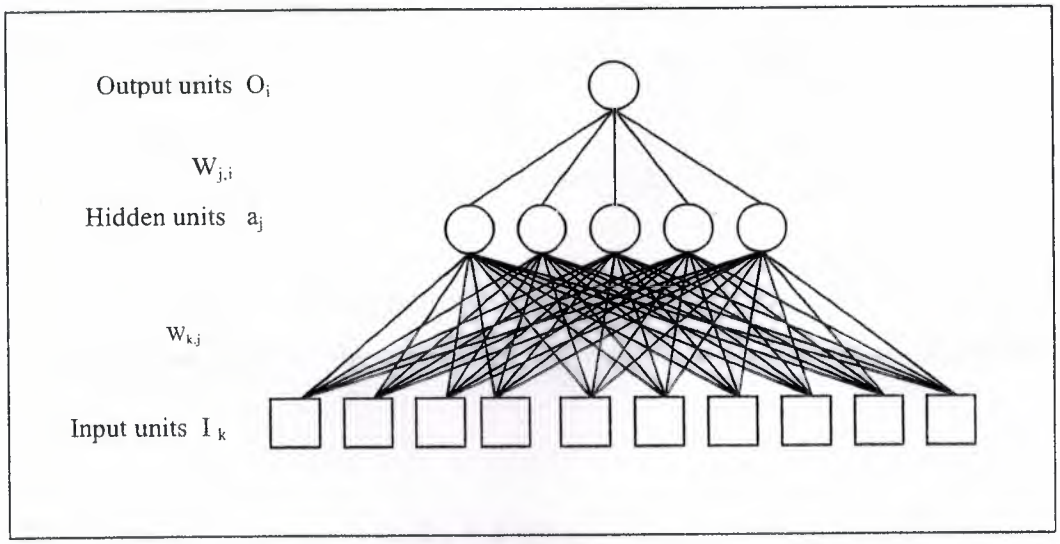


Figure 1.7 A two layer feed forward network for the restaurant problem. [23]

Example inputs are presented to the network, and if the network computes an output vector that matches the target, nothing is done. If there is an error (a difference between the output and target), then weights are adjusted to reduce this error. The trick is to assess the blame for an error and divide it among the contributing weights. In Perceptrons, this is easy, because there is only one weight connecting each input and output. But in multiplayer networks, there are many weights connecting each input to an output and each of these weights contributes to more than one output.

The back-propagation algorithm is a sensible approach to dividing the contribution of each weight. As in the Perceptron Learning Algorithm, we try to minimize the error between each target output and the output actually computed by the network. At the output layer the weight update rule is very similar to the rule for the perceptron. However, there are two differences. The activation of the hidden unit a_j is used instead of the input value; and the rule contains a term for the gradient of the activation function. If Err_i is the error (T_i-O_i) at the output node, then the weight update rule for the link from unit j to unit i is

$$W_{ji} \leftarrow W_{ji} + \alpha \times Err_i \times g'(in_i) \tag{1.1}$$

Where g' is the derivative of the activation g will find it convenient to define a new error term Δ_i which for output node is defined as $\Delta_i = Err_i g'(in_i)$. The update rule then becomes:

$$W_{ji} \leftarrow W_{ji} + \alpha \times a_j \times \Delta_i \tag{1.2}$$

For updating the connections between the input and the hidden units, we need to define a quantity analogous to the error term for output node. The propagation rule so the following:

$$\Delta_j = g'(in_j) \sum_i W_{ji} \Delta_i \tag{1.3}$$

Now the weight update rule for the weights between the inputs and the hidden layer is almost identical to the update rule for the output layer.

$$W_{kj} \leftarrow W_{kj} + \alpha \times I_k \times \Delta_j \tag{1.4}$$

Function Back-Prop-UPDATE (network, examples, α) returns a network with modified weights.
Inputs: network, a multiplayer network
Examples, asset of input/output pairs α, the learning rate.
Repeat
For each e **in** example **do**
$O \leftarrow TUN - NETWORK(network, I^e)$
$Err^e \leftarrow T^e - O$
$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times Err_i^e \times g'(in_i)$
for each subsequent layer **in** network **do**
$\Delta_j \leftarrow g'(in_j)\sum_i W_{j,i}\Delta_j$
$W_{k,j} \leftarrow W_{k,j} + \alpha \times I_k \times \Delta_j$
end
end
until network has converged
return network

Figure 1.8 Back propagation algorithm for updating weights in a multiplayer network

Back-propagation provides a way of dividing the calculation of the gradient among the unit so the change in each weight can be calculated by the unit to which the weight is attached using only local information.

We use the sum of squared errors over the output values:

$$E = \frac{1}{2}\sum_i (T_i - O_i)^2 \tag{1.5}$$

The key insight again is that the output values O_i are a function of the weights for general two-layer network, we can write:

$$E(W) = \frac{1}{2}\sum_i (T_i - g(\sum_j W_{j,i} a_j))^2 \tag{1.6}$$

$$E(W) = \frac{1}{2}\sum_{i}(T_i - g(\sum_{j} W_{j,i} g(\sum W_{k,j} I_k)))^2 \quad (1.7)$$

1.7 Learning Processes

Learning is a process by which the free parameters of a neural network are a adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by a manner in which the parameter change takes place.

This definition of the learning process implies the following sequence of events:

- The neural network is stimulated by an environment.
- The neural network undergoes changes in its parameters as a result of this stimulation.
- The neural network responds in a new way to the environment because of the changes that have occurred in its internal structure.

A prescribed set of well-defined rules for the solution of a learning problem is called a "learning algorithm."

Basically learning algorithms differ from each other in the way in which the adjustment to a synaptic weight of neurons is formulated. Another factor to be considered is the manner in which a neural network (learning machine) is made up of a set of interconnected neurons. Learning paradigm refers to a model of the environment in whish the neural network operates.

1.7.1 Memory-Based Learning

In memory-based learning, all (or most) of the past experiences are explicitly stored in a large memory of correctly classified input-output examples.

$$\{(x_i, d_i)\}_i^N = 1 \quad (1.8)$$

Where xi denotes an input vector and di denotes the corresponding desired response.

1.7.2 Hebbian Learning

When an axon of cell A is near enough to excite a cell B, it repeatedly or persistently takes part in firing it. Some growth processes or metabolic changes take place in one or both cell such that A is efficiency as one of the cells firing B is increased.

1. If two neurons on either side of a synapse are selectively (connection) activated simultaneously (i.e. then the strength of that synapse is selectively increased).
2. If two neurons on either side of a synapse are active asynchronously, then that synapse is selectively weakended or eliminated.

The following are four key mechanisms that characterize a Hebbian Synapse:

1. Time-dependent mechanism. This mechanism refers to the fact that the modification in the Hebbian synapse depend on the exact time of occurrence of the presynaptic and postsynaptic signals.
2. Local mechanism. By its nature a synapse is the transmission site where information-bearing signals (representing ongoing activity in the presynaptic and postsynaptic units) are in spatiotemporal congtiguity.
3. Interactive mechanism. The occurrence of a change in the Hebbian synapse depends on signals on both sides of the synapse.
4. Conjunctional or correlational mechanism. One interpretation of Hebb's postulate of learning is that the condition for a change in synaptic efficiency is the conjunction of presynaptic and posynaptic signals.

1.7.2.1 Synaptic Enhancement and Depression

The conception of a Hebbian modification by is recognizing that positively correlated activity produces synaptic weakening; synaptic for depression may also be of a noninteractive type. The classification of modifications such as Hebbian, anti-Hebbian, and non-Hebbian, according to this scheme, increases its strength when these signals are either uncorrelated or negatively correlated.

1.7.2.2 Mathematical Models of Hebbian Modifications

To formulate Hebbian learning in mathematical terms, consider a synaptic weight W_{kj} of neuron k with presynaptic and postsynaptic signals denoted by x_j and y_k respectively. The adjustment applied to the synaptic weight W_{kj}, at time step n, is expressed in the general form:

$$\Delta w_{kj}(n) = f(y_k(n), x_j(n)) \tag{1.9}$$

Where F(.,.) is a function of both postsynaptic and presynaptic signals the signals $x_j(n)$ and $y_k(n)$ are often treated as dimensionless.

1.7.2.3 Hebbian Hypothesis

The simplest form of Hebbian learning is described by:

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n) \tag{1.10}$$

Where η is a positive constant that determine the rate of learning, it clearly emphasizes the correlational nature of a Hebbian synapse. It is sometimes referred to as the activity product rule. (The top curve of figure 1.9).

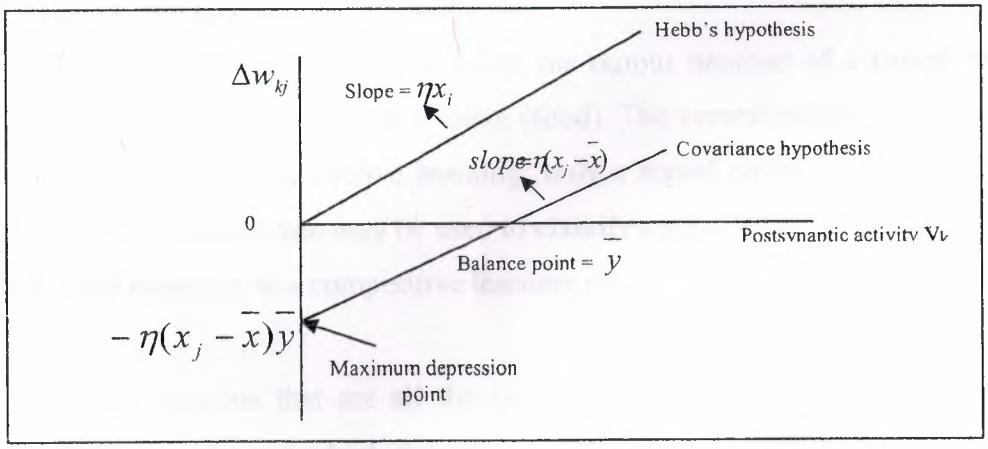


Figure 1.9 Illustration of Hebb's Hypothesis and the Covariance Hypothesis. [23]

With the change Δw_{kj} plotted, versus the output signal (postsynaptic activity) y_k, therefore exponential growth finally drives the synaptic connection into staturation. At that point no information will be stored in the synapse and selectivity is lost.

Covariance hypothesis: One way of overcoming the limitation of Hebb's hypothesis is to use covariance hypothesis introduced by Sejnowski. In this hypothesis, the presynaptic and postsynaptic signals in are replaced by the departure of presynaptic and postsynaptic signals from their respective values over a certain time interval. Let $\bar{x}$ and $\bar{y}$ denote the time average values of the presynaptic signal x_j, and postsynaptic signal y_k respectively according to the covariance hypothesis. The adjustment applied to the synaptic weight w_{kj} is defined by:

$$\Delta w_{kj} = \eta(x_j - \bar{x})(y_k - \bar{y}) \tag{1.11}$$

Where η is the learning rate parameter, the average values x and y constitute presynaptic and postsynaptic thresholds. This determines the sign of synaptic modification.

1.7.3 Competitive Learning

In competitive learning as the name implies the output neurons of a neural network compete among themselves to become active (fired). The several output neurons may be active simultaneously in completive learning; only a signal output neuron is active at any time. It is this features that may be used to classify a set of input patterns.
The three basic elements to a competitive learning rule.

- A set of neurons that are all the same except for some randomly distributed synaptic weight and which therefore respond differently to a given set of input patterns
- A limit imposed on the strength of each neuron.
- A mechanism that permits the neurons to compete for the right to respond to a given subset of input; such that only one output neurons.

In the simplest form of competitive learning the neuronal network has a single layer of output neurons. Each of which is fully connected to the input nodes. The network may include feedback connection among the neurons as indicated in figure 1.10.

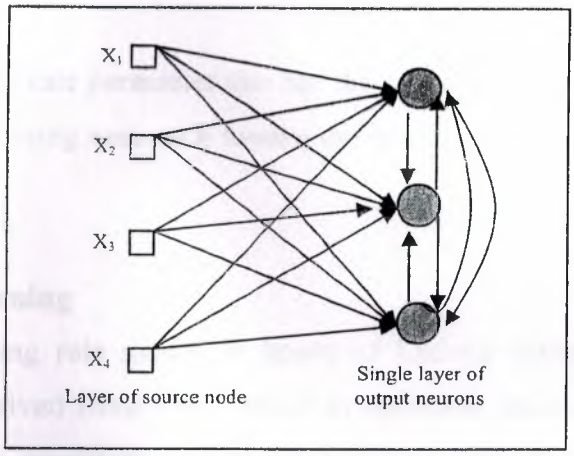


Figure 1.10 Feedback Connections Among the Neurons. [23]

For a neuron k, to be the winning neuron, its induced local field v_k for a specified input pattern. X must be the largest among all the neurons in the network. The output signal y_k, of winning neurons k is set equal to one. The output signals of all the neurons that lose the competition are set equal to zero. We thus write:

$$y_k = \begin{cases} 1 \; if v_k > v_j \; for all j, j \neq k \\ o \qquad otherwise \end{cases} \tag{1.12}$$

The induced local field v_k represents the combined action of all the forward and feedback inputs to neuron k.

Let w_{kj} denote the synaptic weight connecting input node j to neuron k. Suppose that each neurons is allotted a fixed amount of synaptic weight, which is distributed among its input node that is:

$$\sum_j w_{kj} = 1 \quad \text{for all k} \tag{1.13}$$

The change Δw_{kj} applied to synaptic weight w_{kj} is defined by:

$$w_{kj} = \begin{cases} \eta(x_j - w_{kj}) if\ neuron\ \ k\ \ wins\ \ the compention \\ 0 \quad\ \ if\ neuron\ \ k\ \ loses\ \ the compention \end{cases} \tag{1.14}$$

Where η is the learning rate parameter this has the overall effect of moving the synaptic weight vector w_k of winning neurons k toward the input pattern x.

1.7.4 Boltzmann Learning

The Boltzmann learning rule named in honor of Ludwig Boltzmann is a stochastic learning algorithm derived from ideas rooted in statistical mechanics. In a Boltzmann machine, the neurons constitute a recurrent structure and they operate in a binary manner. Since, for example, they are either in an on state denoted by +1 or in an off state denoted of which is determined by the particular states occupied by the individual neurons of the machine as shown by:

$$E = -\frac{1}{2}\sum_j \sum_k w_{kj} x_k x_j \tag{1.15}$$

Where x_j is the state of neuron j and w_{kj} is the synaptic weight connecting neuron j to neuron k, the fact that $j \neq k$ means simply that none of the neurons in the machine has self feedback. The machine operates by choosing a neuron at random, for example neuron k at some step of the learning process then flipping the state of neuron k from state x_k at some temperature T with probability.

$$p(x_k \rightarrow -x_k) = \frac{1}{1 + \exp(-\Delta E_k / T)} \tag{1.16}$$

Where ΔE_k is the energy change resulting from such a flip notice that T is not physical temperature but rather a pseudotemperature.

The neurons of a Boltzmann machine partition into two functional groups: visible and hidden. The visible neurons provide an interface between the network and the

environment in which it operates, whereas the hidden neurons always operate freely. There are two modes of operation to be considered.

- Clamped condition in which the visible neurons are all clamped onto specific states determined by the environment.
- Free running condition in which all the neurons visible and hidden are allowed to operate freely.

According to the Boltzmann learning rule, the change Δw_{kj} applied to the synaptic weight w_{kj} from neuron j to neuron k by:

$$\Delta w_{kj} = \eta(p_{kj}^{+} - p_{kj}^{-}), \quad j \neq k \tag{1.17}$$

Where η is a learning rate parameter, note that both p_{kj}^{+} and p_{kj}^{-} range in value from –1 to +1.

1.8 Learning Tasks

In this context we will identify six learning tasks that apply to the use of neural network in one form or another.

a. Pattern Association

An associative memory is a brain-like, distributed memory that learns by association. Association has been known to be a prominent feature of human memory since Aristotle and all models of cognition use in one form or another as the basic operation. There are two phases involved in the operation of an associative memory:

- Storage phase, which refers to the training of the network in accordance with $x_k \rightarrow y_k, \quad k = 1,2,3.....q$
- Recall phase, which involves the retrieval of a memorized pattern in response to the presentation of a noisy or distorted version of a key pattern to the network.

b. Pattern Recognition

Humans are good at pattern recognition. We receive data from the world around us via our senses and are able to recognize the source of the data.

Pattern recognition is formally defined as the process whereby a received pattern/signal is assigned to one of a prescribed number of classes (categories).

c. Function Approximation

The third learning task of interest is that of function approximation.

d. Control

The control of a plant is another learning task that can be done by a neural network; by a plant we mean a process or critical part of a system that is to be maintained in a controlled condition.

e. Filtering

The term filter often refers to a device of algorithm used to extract information about a prescribed quantity of interest from a set of noisy data.

f. Beamforming

Beamforming is a spatial form of filtering and is used to distinguish between the spatial properties of a target signal and background noise. The device used to do the beamforming is called a "beamformer."

1.9 Activation Functions

This threshold function is generally some form of nonlinear function. One simple nonlinear function that is appropriate for discrete neural nets is the step function. One variant of the step function is:

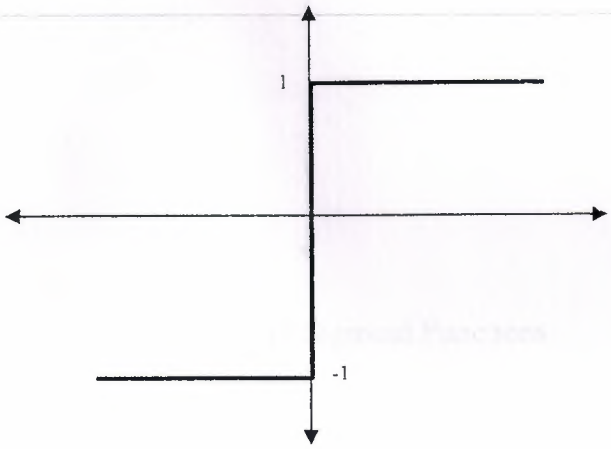


Figure 1.11 Hard Activation Functions

$$f(x) = \begin{cases} 1 & x > 0 \\ f'(x) & x = 0 \\ -1 & x < 0 \end{cases} \qquad (1.18)$$

Where f' (x) refers to the previous value of f(x) (that is the activation of the neuron will not change)

Where x is the summation (over all the incoming neurons) of the product of the incoming neuron's activation, and the connection:

$$X = \sum_{i=0}^{n} A_i w_i \qquad (1.19)$$

The number of incoming neurons, is A the vector of incoming neurons and w is the vector of synaptic weights connecting the incoming neurons to the neurons we are examining. One more appropriate to analog is the sigmoid, or squashing, function; an example is the logistic functions illustrated in figure 1.12.

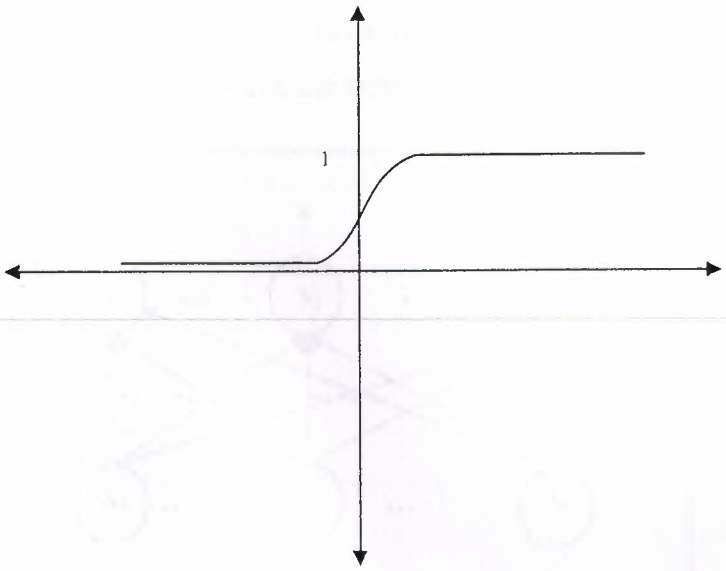


Figure 1.12 Sigmoid Functions

$$f(x) = \frac{1}{1+e^{-x}} \tag{1.20}$$

Another popular alternative is:

$$f(x) = \tanh(x) \tag{1.21}$$

The most important characteristic of our activation function is that it is nonlinear. If we wish to use activation function as a multiplayer network, the activation function must be nonlinear, or the computational will be equivalent to a single-layer network.

1.9.1 A.N.N.

All of the knowledge that a neural network possesses is stored in the synapses. The weights of the connections between the neurons of diagram of the synapse layer model.

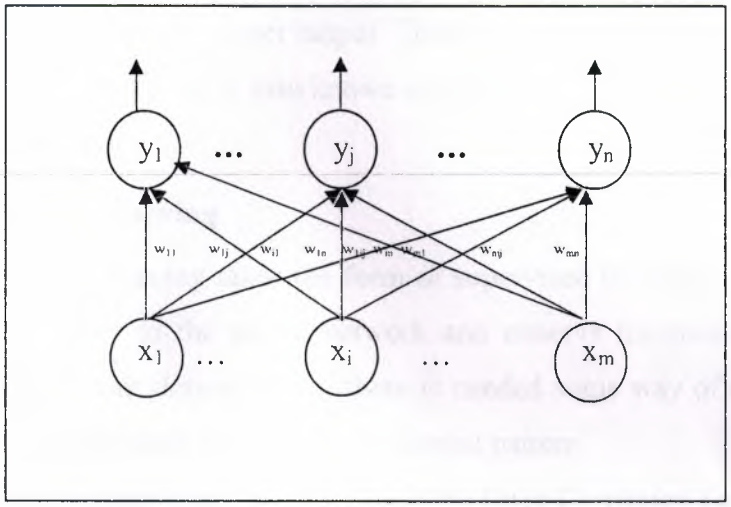


Figure 1.13 Diagram of Synapse Layer Model

However the network acquires that knowledge, this happens during training$\alpha\alpha$ g pattern associations are presented to the network in sequence, and the weights are adjusted to capture this knowledge. The weight adjustment scheme is known as the "learning law". One of the first learning methods formulated was Hebbian Learning.

Donald Hebb, in his organization of behavior formulated the concept of "correlation learning". This is the idea that the weight of a connection is adjusted based on the values of the neurons its connects:

$$\Delta w_{ij} = \alpha a_i a_j \tag{1.22}$$

Where α is the learning rate a_i is the activation of the ith neuron in one neuron layer, a_j is the activation of the jth neuron in another layer, and w_{ij} is the connection strength between the two neurons. A variant of this learning rule is the signal Hebbian Law:

$$\Delta w_{ij} = -w_{ij} + s(a_i)s(a_j) \tag{1.23}$$

S is a sigmoid

1.9.2 Unsupervised learning

One method of learning is the unsupervised learning method. In general, an unsupervised learning method is one in which weight adjustments are not made based on comparison with some target output. There is no teaching signal feed into the weight adjustments. This property is also known as self – organization.

1.9.3 Supervised learning

In many models, learning takes the form of supervised training. I present input pattern one after the other to the neural network and observe the recalled output pattern in comparison with our desired result, there is needed some way of adjusting the weights which takes into account any error in the output pattern.

An example of a supervised learning law is the Error Correction Law:

$$\Delta w_{ij} = \alpha a_i \left[c_j - b_j \right] \tag{1.24}$$

A before α is again the learning rate, ai the activation of the ith neuron, bj is the activation of the jth neuron in the recalled pattern, and cj is the deired activation of the jth neuron.

1.9.4 Reinforcement learning

Another learning method, known as reinforcemnet learing fits into the general category of supervised learning. However, its formula differs from the error correction formula just presented. This type of learning is similar to supervised learning except that each ouput neuron gets an error value. Only one error value is computed for each ouput neuron. The weight adjustment formula is then:

$$\Delta w_{ij} = \alpha \left[v - \Theta j \right] e_{ij} \tag{1.25}$$

Again α is the learning rate, v is the single value indicting the total error of the output pattern, and Θ is the threshold value for the jth output neuron. We need to spread out this generalized error for the jth output neuron to each of the incoming i neurons, is a value representing the eligibility of the weight for updating. This may be computed as:

$$e_{ij} = \frac{d \ln g_i}{dw_{ij}} \tag{1.26}$$

Where g_i is the probability of the output being correct given the input from the ith incoming neuron. (This is vague description; the probability function is of necessity a heuristic estimate and manifests itself differently from specific model to specific model).

1.10 Backpropagation Model

Backpropagation of errors is a relatively generic concept. The Backpropagation model is applicable to a wide class of problems. It is certainly the predominant supervised training algorithm. Supervised learning implies that we must have a set of good pattern associations to train with. The backpropagation model presented in figure 1.14.

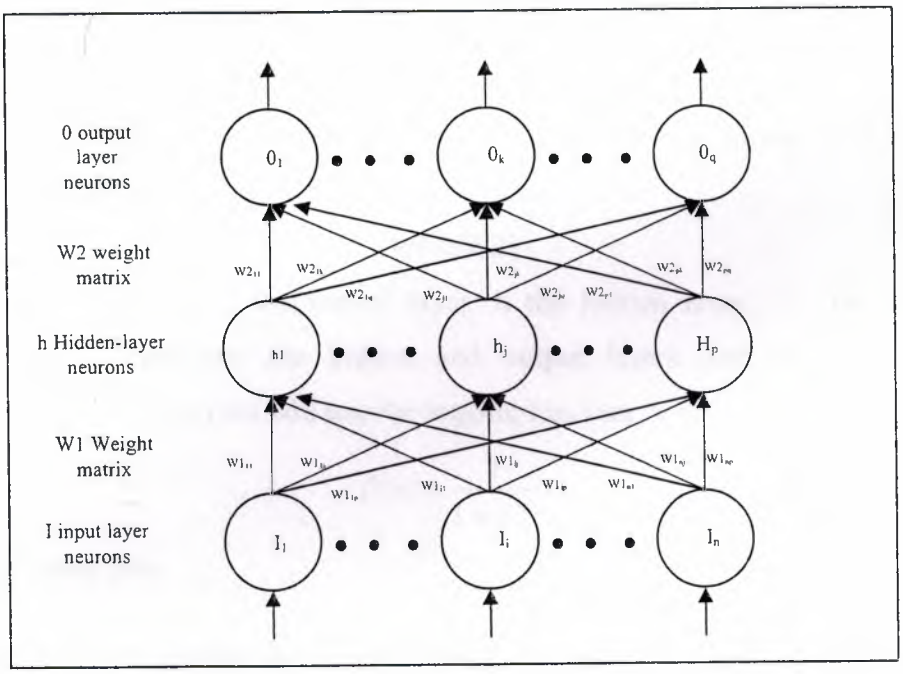


Figure 1.14 Diagram of Backpropagation Topology. [23]

It has three layers of neurons: an input layer, a hidden layer, and an output layer. There are two layers of synaptic weights. There is a learning rate term, α in the subsequent formulas indicating how much of the weight changed to effect on each pass this is typically a number between 0 and 1. There is a momentum term Θ indicating how much a previous weight change should influence the current weight change. There is also a term indicating within what tolernce we can accept an output as good.

1.10.1 Back Propagation Algorithm

Assign random values between –1 and +1 to the weghts between the input and hidden layers, the weights between the hidden and output layers, and the thershold for the hidden layer and output layer neurnos train the network by preforming the following procedure for all pattern pairs:

Forward Pass.

1. Computer the hidden layer neuron activations:

   $$h=F(iW1) \tag{1.27}$$

   Where h is the vector of hidden layer neurons i is the vector of input layer neurons, and W1 the weight matrix between the input and hidden layers.

2. Compute the output layer neuron activation:

   $$O=F(hW2) \tag{1.28}$$

   Where o represents the output layer, h the hidden layer, W2 the matrix of synapses connecting the hidden and output layers, and FO is a sigmoid activation function we will use the logistic function:

   $$f(x) = \frac{1}{1+e^{-x}} \tag{1.29}$$

   Backward Pass.

3. Compute the output layer error (the difference between the target and the observed output):

   $$d = O(1-O)(O-t) \tag{1.30}$$

   Where d is the vector of errors for each output neuron, o is the output layer, and t is the target correct activation of the output layer.

4. Compute the hidden layer error:

$$e = h(1-h)W2d \tag{1.31}$$

Where is e is the vector of errors for each hidden layer neuron.

5. Adjust the weights for the second layer of synapses:

$$W2 = W2 + \Delta W2 \tag{1.32}$$

Where $\Delta W2$ is a matrix representing the change in matrix $W2$. It is computed as follows:

$$\Delta W2_t = \alpha hd + \Theta \Delta W2_{t-1} \tag{1.33}$$

Where α is the learning rate, and Θ is the momentum factor used to allow the previous weight change to influence the weight change in this time period. This does not mean that time is somehow incorporated into the mode. It means only that a weight adjustment has been made. This could also be called a cycle.

6. Adjust the weights for the first layer of synapses:

$$W1 = W1 + W1_t \tag{1.34}$$

Where

$$W1_t = \alpha ie + \Theta \Delta W1_{t-1} \tag{1.35}$$

Repeat step 1 to 6 on all pattern pairs until the output layer error (vector d) is within the specified tolerance for each pattern and for each neuron.

Recall:

Present this input to the input layer of neurons of our backpropagation net:

- Compute the hidden layer activation:

$$h = F(W1i) \tag{1.36}$$

- Computer the output layer:

$$O = F(W2h) \tag{1.37}$$

The vector o is our recalled pattern.

1.10.2 Strengths and Weaknesses

The Back Propagation Network has the ability to learn any arbitrarily complex nonlinear mapping this is due to the introduction of the hidden layer. It also has a capacity much greater than the dimensionality of its input and output layers as we will see later. This is not true of all neural net models.

However Backpropagation can involve extremely long and potentially infinite training time. If you have a strong relationship between input and outputs and you are willing to accept results within a relatively broad time, your training time may be reasonable.

1.11 Summary

In this chapter the followings were discussed Perceptron Algorithm, supervised and unsupervised algorithms, Neural network definition, some history of the Neural network, Natural Neuron, Artificial Neuron, the Backpropagation algorithm and their models, Learning processes and their tasks, and the Activation function.

2. Intelligent Pattern Recognition

2.1 Overview

In this section the intelligent pattern recognition will be added as fully detailed explanation over whole related topics respectively.

2.2 What is pattern recognition?

It is generally a easy for a person to differentiate the sound of a human voice, from that of a violin; a handwritten numeral "3," from an "8"; and the aroma of a rose, from that of an onion. However, it is difficult for a programmable computer to solve these kinds of perceptual problems. These problems are difficult because each pattern usually contains a large amount of information, and the recognition problems typically have an inconspicuous, high-dimensional, structure.

Pattern recognition is the science of making inferences from perceptual data, using tools from statistics, probability, computational geometry, machine learning, signal processing, and algorithm design. Thus, it is of central importance to artificial intelligence and computer vision, and has far-reaching applications in engineering, science, medicine, and business. In particular, advances made during the last half century, now allow computers to interact more effectively with humans and the natural world (e.g., speech recognition software). However, the most important problems in pattern recognition are yet to be solved.[17]

Pattern recognition is the scientific discipline whose goal is the classification of objects into a number of categories or classes. Depending on the application, these objects can be images or signal waveforms or any type of measurements that need to be classified. We will refer to these objects using the generic term patterns. Pattern recognition has a long history, but before the 1960s it was mostly the output of theoretical research in the area of statistics. As with everything else, the advent of computers increased the demand for practical applications of pattern recognition, which in turn set new demands for further theoretical developments.[17]

As our society evolves from the industrial to its postindustrial phase, automation in industrial production and the need for information handling and retrieval are becoming increasingly important. This trend has pushed pattern recognition to the high edge of today's engineering applications and research. Pattern recognition is an integral part in most machine intelligence systems built for decision making.

Pattern is a description of an object. Recognition: classifying an object to a pattern class. PR is the science that concerns the description or classification (recognition) of measurements. PR techniques are an important component of intelligent systems and are used for

- Decision making
- Object and pattern classification
- Data preprocessing

2.3 Machine perception

The task is Seek to design and build machines that can recognize patterns. Applications is Biomedical (Neuroscience, ECG monitoring, drug development, Dna sequences) speech recognition fingerprint identification, optical character recognition Industrial inspection.[17]

The unrelenting pace of innovation in computer and communication technology continues to affirm the central role played by information in modern society. Indeed, the typical amount of digital storage capacity is doubling every year, and bandwidth in both wired and wireless networks increases at an even faster rate, supporting instant and almost ubiquitous access to an abundance of resources. As a consequence, there is a pressing need for tools that can assist us in exploring the huge quantities of data with which we are confronted when managing large multimedia databases or monitoring complex sensor streams.

These data can have intricate spatial, spectral or dynamic structures (eg text that refers to images, audio punctuating video) and are potentially an extremely valuable source of information. However, unless we can extract the knowledge buried

in the bits and bytes, all this data serves little purpose. In this respect, it is becoming increasingly clear that in order to be efficient, data processing needs to be content-based. As the enormous size of these collections precludes comprehensive human supervision, the only viable alternative is the development of reliable machine perception and understanding, and in particular, the automatic creation of semantically rich metadata that can be used as input for ensuing high-level processing or decision support.

Addressing these challenges is quite a daunting task. Fortunately, we are witnessing prodigious activity in key scientific and technological areas that promise to have a profound impact on the way we tackle this deluge of information. First, progress in signal processing for the different modalities (image, audio, speech, etc) has given rise to the creation of sophisticated tools capable of performing reliably for specialised sub-problems (eg face- or motion-detection and text-to-speech translation). In addition, researchers are increasingly turning their attention to cross-modal integration, combining different modalities to maximise information extraction and robustness. Concurrently, progress in statistical and machine learning has boosted the wider acceptance of automated learning methodologies, and it has transpired that these techniques can contribute significantly to the automatic exploration and structuring of large datasets.

To underscore the urgency of the data-mining problem and its potential impact on society and industry, the European Commission has made semantic-based knowledge systems a priority theme in its call for Information Society Technologies for the upcoming Sixth Framework. ERCIM has positioned itself to play an important role by submitting a Network of Excellence (NoE) on Multimedia Understanding through Semantics, Computation and Learning (MUSCLE, currently in the negotiation phase). This NoE will run for four years and should stimulate closer collaboration between European groups working on research projects that aim to integrate machine perception and learning for multimedia data-mining.

The consortium members have agreed to focus on different aspects of single- and cross-modal processing (in video, audio and speech) as well as various flavours of statistical learning.

To encourage close coordination of effort and durable scientific integration, MUSCLE will set itself two 'Grand Challenges'. These are ambitious research projects that involve the whole spectrum of expertise that is represented within the consortium and as such, will act as focal points. The first challenge focuses on natural high-level interaction with multimedia databases. In this vision it should become possible to query a multimedia database at a high semantic level.

Think Ask Jeeves for multimedia content: one can address a search engine using natural language and it will take appropriate action, or at least ask intelligent, clarifying questions. This is an extremely complicated problem and will involve a wide range of techniques, including natural language processing, interfacing technology, learning and inferencing, merging of different modalities, federation of complex meta-data, appropriate representation and interfaces, etc.

The second Grand Challenge is related more closely to machine perception and addresses the problem of detecting and recognising humans and their behaviour in videos. At first glance, this might seem rather a narrow scope, but it has become clear that robust performance will heavily rely on the integration of various complementary modalities such as vision, audio and speech.

Applications are legion: surveillance and intrusion detection, face recognition and registration of emotion or affect, and automatic analysis of sports videos and movies, to name just a few. For more information on this Network, we invite the reader to visit the MUSCLE Web page (see below).

This special issue highlights the breadth and depth of the research related to machine perception and understanding that is currently being conducted by various ERCIM groups.

There continues to be a strong interest in biologically inspired approaches, which is hardly surprising, since nature still easily outperforms the most intricate technical solutions. Another trend that re-affirms itself is the reliance on computationally intensive methodology to extract statistical information and simulate computational models. Equally varied are the applications on display, ranging from mobile robots to support for virtual studios.

Researchers are also enthusiastically accommodating the advances in sensor and (wireless) communication technology that support the creation of networks of interacting and context-aware components, thus moving closer to the vision of genuine ambient intelligence. It is our hope that this Special Issue will offer the reader a taste of the exciting developments on which the various ERCIM laboratories are working.

2.3.1 Examples

- Differentiate between salmon and sea-bass
- Animal footprints
- Hand-written numeral recognition
- SPAM identification

2.4 Feature extraction

The traditional goal of the feature extractor is to characterize an object by making numerical measurements: In animal footprints-example features were square ness and solidness of the footprint shape. Good features are those whose values are similar for objects belonging to the same category and distinct for objects in different categories. Feature extraction is very problem dependent: Good features for sorting fish are of little use for recognizing fingerprints. Usually one feature is not enough to differentiate between objects from different categories. Multiple features representing the same object are organized into feature vectors. The set of all possible feature vectors is called the feature space. Invariant features are such that remain the same if something (irrelevant) is done to the sensed input.[17]

2.5 Classification

The task of the classifier component is to use the feature vector provided by the feature extractor to assign the object to a category. Classification is the main topic of this course. The abstraction provided by the feature vector representation of the input data enables the development of a largely domain-independent theory of classification.

Essentially the classifier divides the feature space into regions corresponding to different categories. The degree of difficulty of the classification problem depends on the variability in the feature values for objects in the same category relative to the feature value variation between the categories. Variability is natural or is due to noise. Variability can be described through statistics leading to statistical pattern recognition. Questions: How to design a classifier that can cope with the variability in feature values? What is the best possible performance.[17]

Artificial neural networks are relatively crude electronic networks of "neurons" based on the neural structure of the brain. They process records one at a time, and "learn" by comparing their classification of the record (which, at the outset, is largely arbitrary) with the known actual classification of the record. The errors from the initial classification of the first record is fed back into the network, and used to modify the networks algorithm the second time around, and so on for many iterations. Roughly speaking, a neuron in an artificial neural network is

1. A set of input values (xi) and associated weights (wi)
2. A function (g) that sums the weights and maps the results to an output (y) .

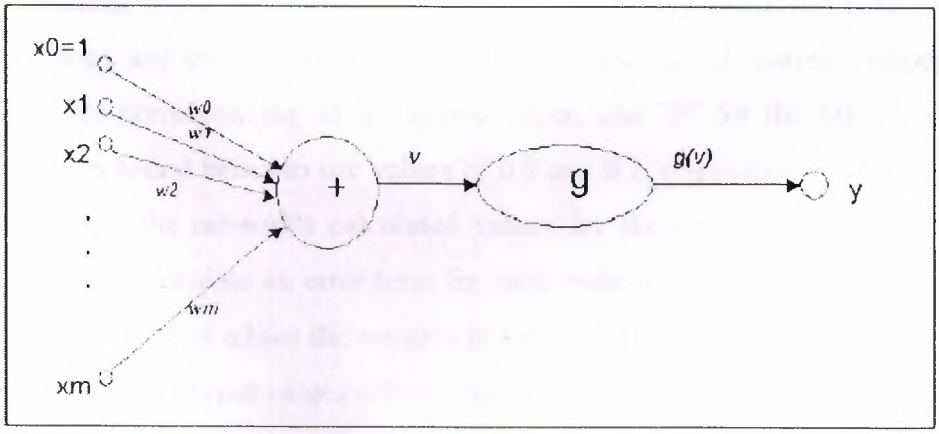


Figure 1.1 The Neural Network algorithm.

The input layer is composed not of full neurons, but rather consists simply of the values in a data record, that constitute inputs to the next layer of neurons. The next layer is called a hidden layer; there may be several hidden layers. The final layer is the output layer, where there is one node for each class.

A single sweep forward through the network results in the assignment of a value to each output node, and the record is assigned to whichever class's node had the highest value.

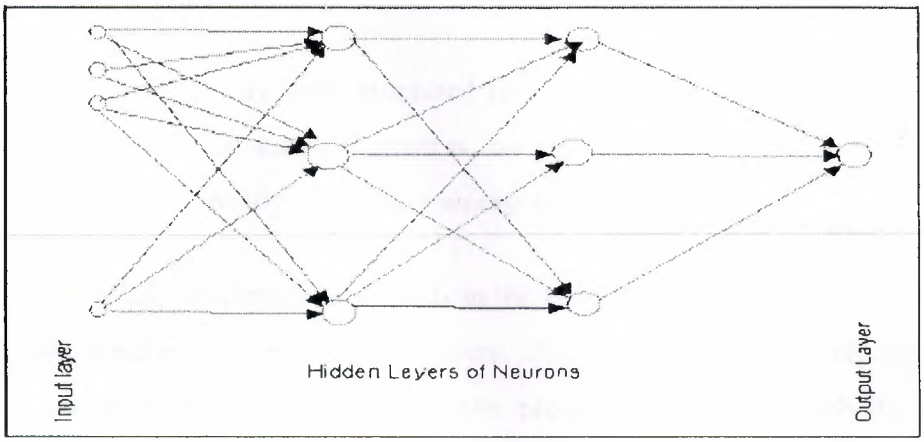


Figure 1.2 Neurons are organized into layers.

2.6 Training an Artificial Neural Network

In the training phase, the correct class for each record is known (this is termed supervised training), and the output nodes can therefore be assigned "correct" values -- "1" for the node corresponding to the correct class, and "0" for the others. (In practice it has been found better to use values of 0.9 and 0.1, respectively.) It is thus possible to compare the network's calculated values for the output nodes to these "correct" values, and calculate an error term for each node (the "Delta" rule). These error terms are then used to adjust the weights in the hidden layers so that, hopefully, the next time around the output values will be closer to the "correct" values.

2.7 The Iterative Learning Process

A key feature of neural networks is an iterative learning process in which data cases (rows) are presented to the network one at a time, and the weights associated with the input values are adjusted each time. After all cases are presented, the process often starts over again. During this learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of input samples.

Neural network learning is also referred to as "connectionist learning," due to connections between the units. Advantages of neural networks include their high tolerance to noisy data, as well as their ability to classify patterns on which they have not been trained. The most popular neural network algorithm is back-propagation algorithm proposed in the 1980's.

Once a network has been structured for a particular application, that network is ready to be trained. To start this process, the initial weights (described in the next section) are chosen randomly. Then the training, or learning, begins.

The network processes the records in the training data one at a time, using the weights and functions in the hidden layers, then compares the resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights for application to the next record to be processed. This process occurs over and over as the weights are continually tweaked. During the training of a network the same set of data is processed many times as the connection weights are continually refined.

Note that some networks never learn. This could be because the input data do not contain the specific information from which the desired output is derived. Networks also don't converge if there is not enough data to enable complete learning. Ideally, there should be enough data so that part of the data can be held back as a validation set.

In the broadest sense, any method that incorporates information from training samples in the design of a classifier employs learning. Due to complexity of classification problems, we cannot guess the best classification decision ahead of time, we need to learn it. Creating classifiers then involves positing some general form of model, or form of the classifier, and using examples to learn the complete classifier.[2]

2.8 Feedforward, Back-Propagation

The feedforward, back-propagation architecture was developed in the early 1970's by several independent sources (Werbor; Parker; Rumelhart, Hinton and Williams). This independent co-development was the result of a proliferation of articles and talks at various conferences which stimulated the entire industry.

Currently, this synergistically developed back-propagation architecture is the most popular, effective, and easy-to-learn model for complex, multi-layered networks. Its greatest strength is in non-linear solutions to ill-defined problems. The typical back-propagation network has an input layer, an output layer, and at least one hidden layer.

There is no theoretical limit on the number of hidden layers but typically there are just one or two. Some work has been done which indicates that a maximum of five layers (one input layer, three hidden layers and an output layer) are required to solve problems of any complexity. Each layer is fully connected to the succeeding layer.

As noted above, the training process normally uses some variant of the Delta Rule, which starts with the calculated difference between the actual outputs and the desired outputs. Using this error, connection weights are increased in proportion to the error times a scaling factor for global accuracy.

Doing this for an individual node means that the inputs, the output, and the desired output all have to be present at the same processing element. The complex part of this learning mechanism is for the system to determine which input contributed the most to an incorrect output and how does that element get changed to correct the error. An inactive node would not contribute to the error and would have no need to change its weights. To solve this problem, training inputs are applied to the input layer of the network, and desired outputs are compared at the output layer. During the learning process, a forward sweep is made through the network, and the output of each element is computed layer by layer.

The difference between the output of the final layer and the desired output is back-propagated to the previous layer(s), usually modified by the derivative of the transfer function, and the connection weights are normally adjusted using the Delta Rule. This process proceeds for the previous layer(s) until the input layer is reached.

2.9 Structuring the Network

The number of layers and the number of processing elements per layer are important decisions. These parameters to a feedforward, back-propagation topology are also the most ethereal - they are the "art" of the network designer. There is no quantifiable, best answer to the layout of the network for any particular application. There are only general rules picked up over time and followed by most researchers and engineers applying this architecture to their problems.

- **Rule One:** As the complexity in the relationship between the input data and the desired output increases, the number of the processing elements in the hidden layer should also increase.

- **Rule Two:** If the process being modeled is separable into multiple stages, then additional hidden layer(s) may be required. If the process is not separable into stages, then additional layers may simply enable memorization of the training set, and not a true general solution effective with other data.

- **Rule Three:** The amount of training data available sets an upper bound for the number of processing elements in the hidden layer(s). To calculate this upper bound, use the number of cases in the training data set and divide that number by the sum of the number of nodes in the input and output layers in the network. Then divide that result again by a scaling factor between five and ten. Larger scaling factors are used for relatively less noisy data. If you use too many artificial neurons the training set will be memorized. If that happens, generalization of the data will not occur, making the network useless on new data sets.

2.10 Summary

Pattern recognition systems aim to decide for an action based on the data provided. Classification is an important step in pattern recognition systems. A classifier uses feature values to assign an object into a category. Feature values contain variability which needs to be modeled) statistics. It is challenging to construct good classifiers. Nevertheless many problems can be solved.

3. BACK PROPAGATION ALGORITHM

3.1 Overview

In this section the back propagation algorithm will be added as fully detailed explanation over whole related topics respectively.

3.2 What is a back propagation algorithm?

Back propagation is a form of supervised learning for multi-layer nets, also known as the generalized delta rule. Error data at the output layer is "backpropageted" to earlier ones, allowing incoming weights to these layers to be updated. It is most often used as training algorithm in current neural network applications. The back propagation algorithm was developed by Paul Werbos in 1974 and rediscovered independently by Rumelhart and Parker. Since its rediscovery, the back propagation algorithm has been widely used as a learning algorithm in feed forward multilayer neural networks.[3]

3.3 Leaning with the back propagation algorithm

The back propagation algorithm is an involved mathematical tool; however, execution of the training equations is based on iterative processes, and thus is easily implementable on a computer. During the training session of the network, a pair of patterns is presented (X_k, T_k), where X_k in the input pattern and T_k is the target or desired pattern. The X_k pattern causes output responses at teach neurons in each layer and, hence, an output O_k at the output layer. At the output layer, the difference between the actual and target outputs yields an error signal. This error signal depends on the values of the weights of the neurons I each layer. This error is minimized, and during this process new values for the weights are obtained. The speed and accuracy of the learning process-that is, the process of updating the weights-also depends on a factor, known as the learning rate.[4]

Before starting the back propagation learning process, we need the following:

- The set of training patterns, input, and target
- A value for the learning rate
- A criterion that terminates the algorithm
- A methodology for updating weights
- The nonlinearity function (usually the sigmoid)
- Initial weight values (typically small random values)

3.4 Network Design Parameters

Employing a backpropagational neural network requires an understanding of a number of network design options, however a brief discussion of some key network parameters is given below. Be advised that there are no definate rules for choosing the settings of these parameters a priori. Since the solution space associated with each problem is not known, an number of different network runs must be undertaken before the user can determine with relative confidence a suitable combination.[4]

3.4.1 Number of Input Nodes:

These are the independent variables which must be adjusted to fall into a range of 0 to 1. The number of nodes is fixed by the number of inputs. Inputs must not be nominal scale, but can be binary, ordinal or better. Such inputs can be accommodated by providing a separate input node for each category which is associated with a binary (0 or 1) input.

3.4.2 Number of Output Nodes:

For the purposes of this research there was always a single output - also adjusted to fall within the range of 0-1.

3.4.3 Number of middle or hidden layers:

The hidden layers allow a number of potentially different combinations of inputs that might results in high (or low) outputs. Each successive hidden layer represents the possibility of recognizing the importance of combinations of combinations.

3.4.4 Number of Hidden Layers:

The more nodes there are the greater the number of different input combinations that the network is able to recognize.

3.4.5 Number of Nodes Per Hidden Layer:

Generally all nodes of any one layer are connected to all nodes of the previous and the following layers. This can be modified at the discretion of the user however.

3.4.6 Initial Connection Weights:

The weights on the input links are initialized to some random potential solution. Because the training of the network depends on the initial starting solution, it can be important to train the network several times using different starting points. Some users may have reason to start the training with some particular set of link weights. It is possible, for example to find a particularly promising starting point using a genetic algorithm approach to weight initialization.

3.4.7 Initial Node Biases:

Node bias values impart a significance of the input combinations feeding into that node. In general node biases are allowed to be modified during training, but can be set to particular values at network initialization time. Modification of the node biases can be also allowed or disallowed.

3.4.8 Learning Rate:

At each training step the network computes the direction in which each bias and link value can be changed to calculate a more correct output. The rate of improvement at that solution state is also known. A learning rate is user-designated in order to determine how much the link weights and node biases can be modified based on the change direction and change rate. The higher the learning rate (max. of 1.0) the faster the network is trained. However, the network has a better chance of being trained to a local minimum solution. A local minimum is a point at which the network stabilizes on a solution which is not the most optimal global solution.

3.4.9 Momentum Rate:

To help avoid settling into a local minimum, a momentum rate allows the network to potentially skip through local minima. A history of change rate and direction are maintained and used, in part, to push the solution past local minima. A momentum rate set at the maximum of 1.0 may result in training which is highly unstable and thus may not achieve even a local minima, or the network may take an inordinate amount of training time. If set at a low of 0.0, momentum is not considered and the network is more likely to settle into a local minimum. A process of "simulated annealing" is performed if the momentum rate starts high and is slowly shifted to 0 over a training session. Like other statistical and mathematical solutions, back propagation networks can be over- parameterized. This leads to the ability of the statistics to find parameters which can accurately compute the desired output at the expense of the systems ability to interpolate and compute appropriate output for different inputs. To ensure that a back propagation neural network is not over parameterized, the training data must be split into a training and a testing set. It is the performance of the trained network on the data reserved for testing that is the most important measure of training success.

3.5 Mathematical Approach

A sequence of steppes should be followed during the mathematic approach, thus they are listed as below, and Figure 3.1 shows the multilayer BP network.[5]

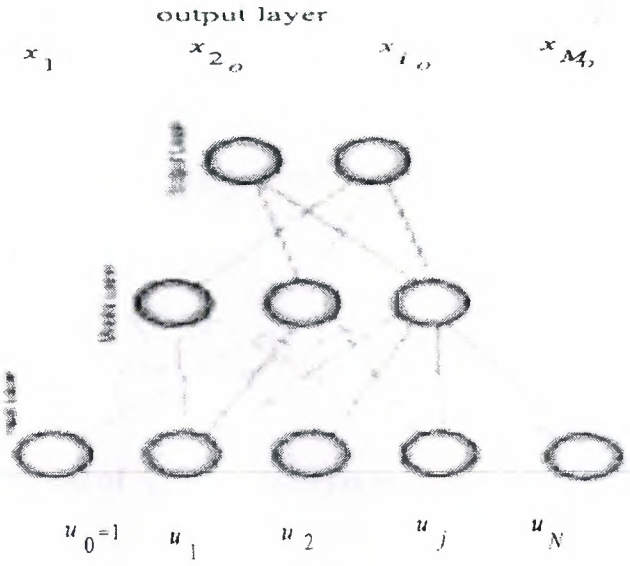


Figure 3.1 Multilayer BP network

Step 0: Initialize weights: to small random values;

Step 1: Apply a sample: apply to the input a sample vector u_k having desired output vector y_k ;

Step 2: Forward Phase:

Starting from the first hidden layer and propagating towards the output layer:

2.1. Calculate the activation values for the units at layer L as:

- If L-1 is the input layer

$$a_{h_L}^k = \sum_{j=0}^{N} W_{jh_L} u_j^k$$

- If L-1 is a hidden laye

$$a_{h_L}^k = \sum_{j_{L-1}=0}^{N} W_{j_{(L-1)}h_L} x_{j_{(L-1)}h_L}^k$$

2.2. Calculate the output values for the units at layer L as:

$$x_{h_L}^k = f(a_{h_L}^k)$$

in which use i_o instead of h_L if it is an output layer

Step 4: Output errors: Calculate the error terms at the output layer as:

$$\delta_{i_o}^k = (y_{i_o}^k - x_{i_o}^k) f_o'(a_{i_o}^k)$$

Step 5: Backward Phase Propagate error backward to the input layer through each layer L using the error term

$$\delta_{h_L}^k = f_L'(a_{h_L}^k) \sum_{i_{L+1}=1}^{N_{L+1}} \delta_{i_{(L+1)}}^k w_{h_L i_{(L+1)}}^k$$

in which, use i_o instead of $i_{(L+1)}$ if $(L+1)$ is an output layer;

Step 6: Weight update: Update weights according to the formula

$$w_{j_{(L-1)}h_L}(t+1) = w_{j_{(L-1)}h_L}(t) + \eta \delta_{h_L}^k x_{j_{(L-1)}}^k$$

Step7: Repeat steps 1-6 until the stop criterion are satisfied, which may be chosen as the mean of the total error is sufficiently small.

$$< e^k > = < \tfrac{1}{2} \sum_{i_o=1}^{M} (y_{i_o}^k - x_{i_o}^k)^2 >$$

3.6 Summary

In this section the back propagation algorithm was added as fully detailed explanation over whole related topics respectively.

4. SIGNATURE VERIFICATION

4.1 Overview

Document certification is very important in many cases. Suppose a cheque with a million dollars or a university certificate signed by forgeries signature. Signature verification is requested in any document validation. This section will represent the problem of signature verification, previous approaches for signature recognition, and multi expert systems for signature verification.

4.2 Analysis of the problem of signature verification

Several types of document need to be validated by signature, and reliable techniques for signature verification are consequently requested. Even if the most recent research is focused on the digital signature of electronic documents (i.e. an encrypted key code associated with a document in its electronic version, especially designed for avoiding manipulation of the file by unauthorized people), a very large number of signed paper documents are still produced daily.[21]

Until now, the problem of signature verification on this class of documents has been faced by taking into account three different types of forgeries: random forgeries, produced without knowing either the name of the signer nor the shape of its signature; simple forgeries, produced knowing the name of the signer but without having an example of his signature; and skilled forgeries, produced by people who looking at an original instance of the signature, attempt to imitate it as closely as possible.

It is obvious that the problem of signature verification becomes more and more difficult when passing from random to simple and skilled forgeries, the latter being so difficult a task that even human beings make errors in several cases.

In fact, exercises in imitating a signature often allow people to produce forgeries so similar to the originals that discrimination is practically impossible; in many cases, the distinction is complicated even more by the large variability introduced by some signers when writing their own signatures. In relation to this, studies on signature shape found that North American signatures are typically more stylistic in contrast to the highly personalized and 'variable in shape' European ones [6].

A number of biometric techniques have been proposed for personal identification in the past. Among the vision-based ones, we can mention face recognition, fingerprint recognition, iris scanning and retina scanning.

Voice recognition or signature verification are the most widely known among the non-vision based ones. Signature verification requires the use of electronic tablets or digitizers for on-line capturing and optical scanners for off-line conversion. These interfaces have the drawback that they are bulky and complicated to use, increasing the complexity of the whole identification system.

Cameras, on the other hand, are much smaller and simple to handle, and are becoming ubiquitous in the current computer environment. The goal of this project is to develop a signature verification technique suitable for signatures captured by our camera-based acquisition system. The following figure shows the signature verification system. Given the subject's name and a new acquired signature, the system decides whether the signature corresponds to the subject's name. The decision is based on a measurement of the similarity between the acquired signature and a prototype of the subject's signature. The prototype has been previously extracted from a training set of signatures.

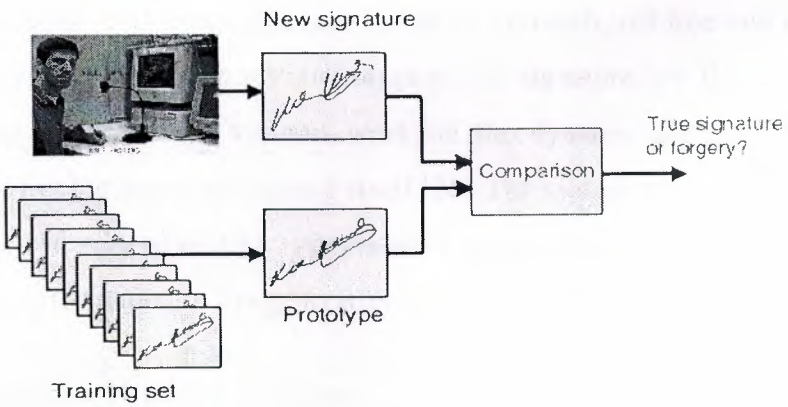


Figure 3.2 The Signature Verification Scheme.

The signature verification system is based upon the measurement of similarity between signatures. Therefore, this computation is the key element of the system. There are several methods proposed in the literature in order to compare signatures. We choose to use Dynamic Time Warping (DTW) in order to perform the comparison between signatures. The present implementation of DTW for signature verification attempts to perform the best alignment of the 2D shape of the signatures, i.e., we find the warping function that has the minimum cost of aligning the planar curves that represent signatures. We note that the pen up strokes drawn by each subject were as consistent as the pen down strokes.

This observation agrees with the belief that signatures are produced as a ballistic or reflex action, without any visual feedback involved. Therefore, we used the full signing trajectory in our experiments.We do not perform any type of normalization on the signatures since we consider that users are very consistent on their style of signing, they write their signatures with a similar slant, in a similar amount of time, with similar dimensions and with a similar motion.

Handwriting recognition is still an open problem, even though it has been extensively studied for many years. Signature verification is a reduced problem that still poses a real challenge for researchers. The literature on signature verification is quite extensive and shows two main areas of research, off-line and on-line systems. Off-line systems deal with a static image of the signature, i.e. the result of the action of signing while on-line systems work on the dynamic process of generating the signature, i.e. the action of signing itself.[26] The system proposed in this paper falls within the category of on-line systems since the visual tracker of handwriting captures the timing information in the generation of the signature.

4.2.1 Signature Parameterization.

In most of the literature on signature verification, a time-based parameterization of the functions to be aligned was used. There is no clear reason for using this parameterization of the signatures rather than another one, e.g. arc-length parameterization.

The arc-length pameterization of the signature is loosely dependent on time and on the dynamics of signing, even though it keeps the causality of the signature's generation.[24]

This weak dependence on the dynamics of signing seems contrary to the traditional idea that pen dynamics is a key element in detecting forgeries. However, the use of the arc-length parameterization is a first step towards achieving invariance with respect to Euclidean transformations of the signatures. Going one step further, we could use a parameterization that provides a certain degree of invariance with respect to affine transformations of the signatures. This parameterization has been described in the literature and has been called affine arc-length.[25]

4.3 Previous Approaches of Signature Verification

Until now, a lot of different recognition systems have been proposed. Among them, it is found that solutions in the case of random forgeries are mainly based on the use of global shape descriptors as the shadow code, initially proposed by Burr [8] and successively enhanced by Sabourin et al [18], who report the results with reference to a database of 20 writers.

Another approach, which performs well in the case of random forgeries, is based on the gradient operator applied to the signature image. The gradient image obtained, integrated over the signature in a predefined range of directions, provides a directional probability density functions, which is used as a signature descriptor; the results of the method on a database of 800 random forgeries are reported by Drouhard et al [8].

Other approaches using global shape descriptors such as shape envelope projections on the coordinate axes, geometric moments, or even more general global features such as area, height and width, have been widely investigated [9].

Sometimes these approaches, although tailored for detecting random forgeries, produce interesting results with simple forgeries.

As soon as the problem of simple forgery detection is considered, the use of global features often does not allow discriminating simple forgeries from the original, so limiting the overall performance of the system. In this case, most of the systems known in the literature perform the validation by using more local features obtained by starting from the skeleton of the image or its outline. In particular, some authors consider peculiarities of the writing process, such as the evaluation of the ink distribution at different resolution levels and the high-pressure regions [19], or other features characterizing the shape of the signature, such as the edges evaluated along different directions.[13]

Another way of extracting local features is proposed by Murshed et al . The input signature image is divided into areas of equal size, and a set of features is extracted by evaluating in each area the occurrence of some graphical segments. It is worth pointing out that most of the systems proposed up to now, while performing reasonably well on a single category of forgeries (random, simple or skilled), decrease in performance when working with all the categories of forgeries simultaneously, and generally this decrement is bigger than one would expect. The main reason for this behavior lies in the difficulty of defining a feature set that is adequate to work with all the classes of forgery simultaneously.[11]

In fact, the use of global features allows discriminators to isolate, as required, the random forgeries and most of the simple forgeries, but often does not allow them to discriminate between a genuine signature and a skilled forgery that is similar in shape to the corresponding original. [22]

On the other hand, as more specific features are considered, the systems increase the ability to discriminate among forgeries and originals, allowing the recognition of skilled forgeries, but those genuine signatures drawn with higher shape variations risk being classified as forgeries.

4.4 Multi-Expert Systems for Signature Verification

In the light of considerations above, it seems justifiable that a single-stage system, devised for recognizing forgeries of all the categories, should simultaneously use both general and specialized features; this could result in unavoidable errors related to the adjustment of the discriminatory power of the feature set depending on the variability of the genuine signatures[22].

A good solution to this kind of problem could be obtained by considering Multi-Expert Systems (MES), i.e. systems made of a variety of simple experts, each able to solve a particular problem.

The rationale of the multi-expert approach lies in the assumption that, by combining the results of a set of experts according to a combining criterion, it is possible to compensate for the weakness of each single expert while preserving its own strength. Experimental analysis on different areas has demonstrated that the performance of the MES can be better than that of any single expert, especially when using complementary experts, as far as possible, and adopting a combining rule for determining the most likely class a sample should be attributed to, given the class to which it is attributed by each single expert [12]. The idea of using a MES has recently been investigated in the literature of signature verification; in this area, most of the existing systems refine the decision process by adopting a multi-resolution scheme. Consequently, the set of features is always the same, but is applied to the input image at different resolution levels [19]. Other approaches are based on a parallel combination of a set of experts, each using a relatively small number of features.

4.5 Signature Recognition Block Diagram

The block diagram of the signature recognition consists from several processing blocks as it is shown in figure 4.1.

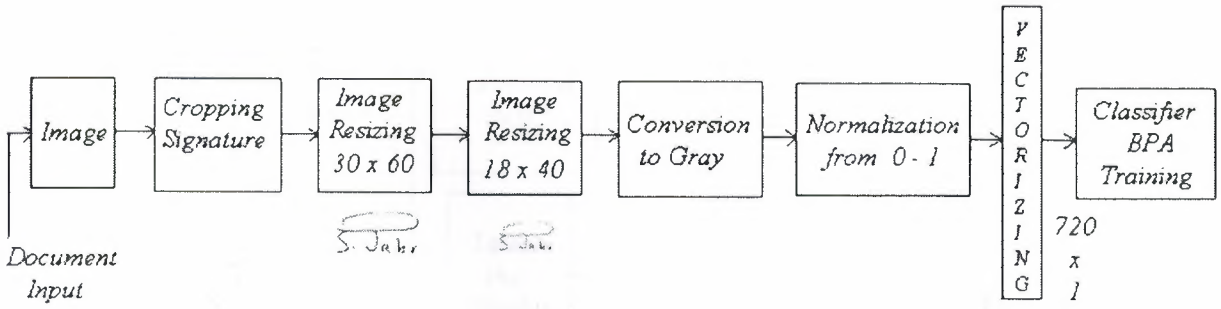


Figure 4.1: Signature Recognition Block Diagram

- Image: The document is converted to digitized image for example by a scanner.
- Cropping Signature: By one of the image tools like Photoshop, or Microsoft Picture Manager.
- Image Resizing 30 x 60 Pixel: By one of the image tools like Photoshop, or Microsoft Picture Manager.
- Image Resizing 20 x 40 Pixel: By the software program using MATLAB.
- Conversion to Gray: By the software program using MATLAB.
- Normalization from 0 – 1: By dividing every image matrix entry by 255 in the software program using MATLAB.
- Vectorizing: Reshaping the normalized image matrix from 18x40 to a victor 720x1 in the software program using MATLAB.
- Classifier BPA Training: Entering the 450 elements as inputs to the backpropagation neural network.

4.6 Backpropagation Algorithm Block Diagram

The block diagram of the Backpropagation Algorithm consists from several processing blocks as it is shown in figure 4.2.

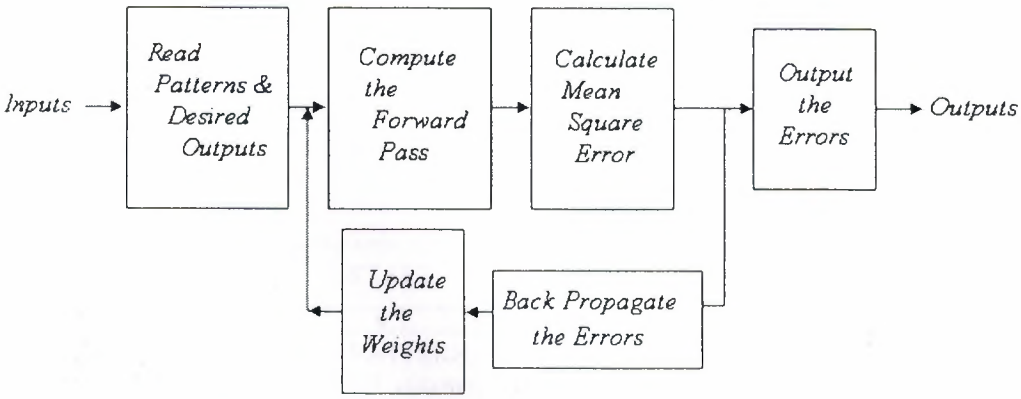


Figure 4.2: Backpropagation Algorithm Block Diagram

4.7 Signature Recognition Network Structure

The neural network of the logo classification consists from three layers, input layer with 720 neurons, forward fully connected with 40 neurons in the hidden layer, and forward fully connected with 4 neurons in the output layer as seen in Figure 4.3.

4.8 Flowchart

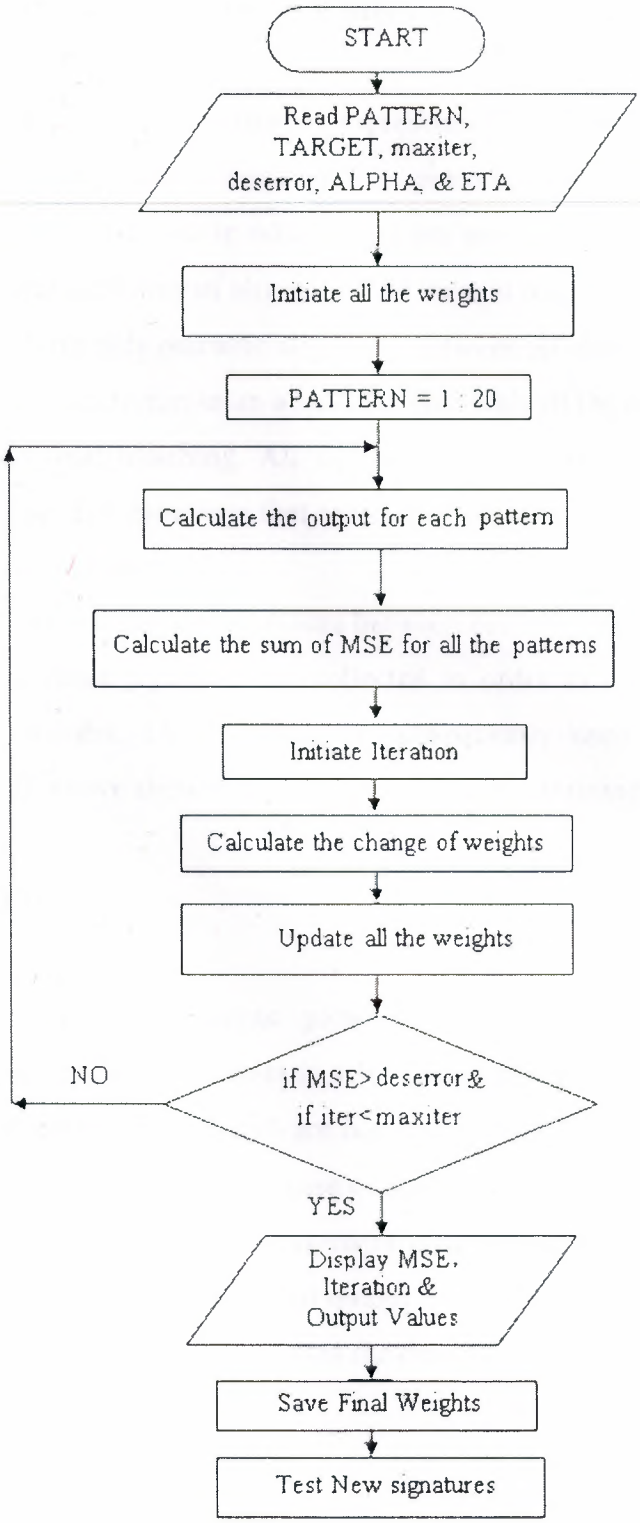


Figure 4.4 Signature Recognition Flowchart of Neural Network Training

4.9 LEARNING A PROTOTYPE SIGNATURE (TRAINING)

The system needs to extract a representation of the training set that will yield minimum generalization error. DTW provides the optimal alignment of two signatures, so in the case in which there are more than two examples in the training set, there is no clear way of aligning all of them at the same time.

We perform only pairwise alignment between all elements in the training set. The signature that yields minimum alignment cost with all the remaining ones is chosen to perform the final matching. All signatures are placed in correspondence with this particular one. The prototype that represents the training set is computed as the mean of the aligned signatures.

The individual residual distances between each of the signatures in the training set and this reference signature are collected in order to estimate the statistics of the alignment process. This statistics is subsequently used for classification. In the followgin figure we show a few signatures from the training set and the prototype.

4.10 Algorithm

STEP 1: Reading the input patterns and the desired outputs. In case logo classification, each input pattern is a 30 x 30 matrix given as a column-major ordered sequence of pixel values, which are between 0 and 1, and each desired output is a 5 x 1 vector. On the other hand, in case of signature recognition, each input pattern is an 18 x 40 matrix given as a column-major ordered sequence of pixel values, which are between 0 and 1, and each desired output is a 4 x 1 vector., while desired outputs are listed in table 4.1. Table 4.3 contains the desired outputs of logo classification.

Table 4.1 Desired Outputs for Signature Recognition

T1	1	0	0	0
T2	0	1	0	0
T3	0	0	1	0
T4	0	0	0	1

STEP 2: Initiating the hidden and output layers weights matrixes. The weights matrices pixels values have been randomly selected between -0.35 and +0.35 for signature recognition and between -0.25 and +0.25 for logo classification.

STEP 3: Initiating the change of weights matrixes equals to zero.

STEP 4: Calculating the output and the error for each input pattern. Compute the sum of error to all patterns.

STEP 5: Initiating the iteration to training.

STEP 6: Updating the hidden and output layers weights.

STEP 7: Calculating the outputs and error after updating the weights.

STEP 8: Displaying the error, the iteration number, and the processing time when the program gets the goal error or exceeds the maximum number of iterations.

4.11 Signature Recognition Results

We collected signatures from 56 subjects, 18 of them women and 4 left handed. Each of them provided 25 signatures, 10 of them to be used as the training set and the other 15 to be used as the test set. We also collected 10 forgeries for each of the subjects in the database. The test set allows us to compute the FRR. We computed the FAR in two different ways. First, we used all the signatures from the other subjects as random forgeries, and second, we used the acquired forgeries.

One common problem of many on-line systems for signature verification is the lack of examples needed to build a reliable model for a signature and to asses the performance of the algorithm.

This problem is inherent to the application since it is not feasible to ask a subject for all the examples of his/her signature required to perform these two tasks reliably.

Thus, we have to build a model of the signature that will perform well in practice and we have to infer the generalization error of the algorithm, all with very few examples. We could increase the number of examples in both the training and test set by using Duplicate Examples, if we know that the model that we are building should be invariant with respect to some transformation of the examples. The essential parameters of signature recognition program are listed in table 4.2 below.

Table 4.2 Signature Recognition Essential Parameters

Number of Input Neurons	450
Number of Hidden Neurons	17
Number of Output Neurons	4
Learning Rate (ETA)	0.001
Momentum Factor (ALP)	0.3
Max Iterations	6000
Desired Error	0.001

Figure 4.5 illustrate the mean square error versus iteration number. It seems that the MSE is decreasing smoothly as desired.

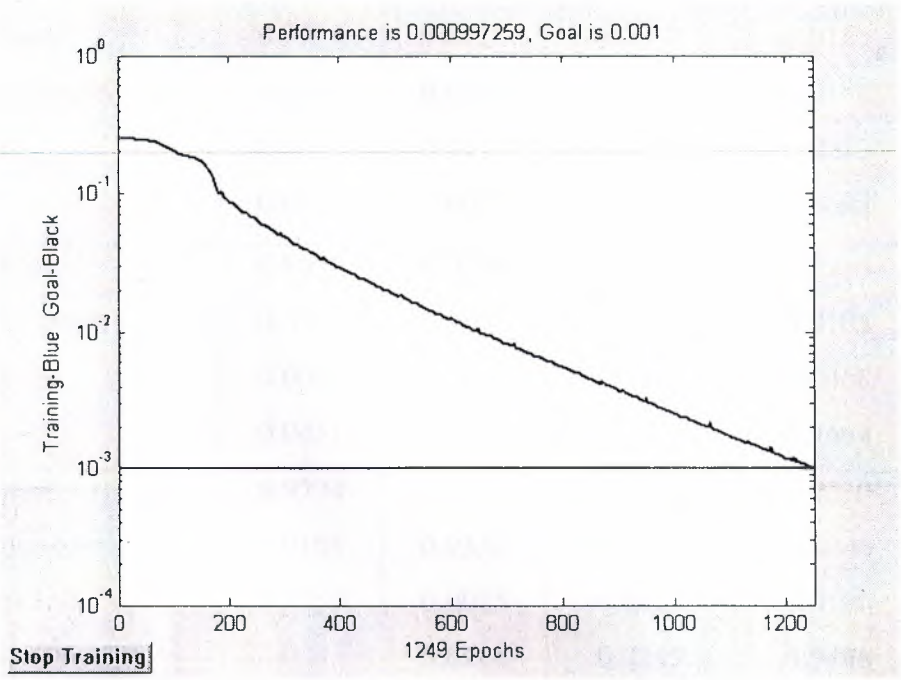


Figure 4.5 Mean Square Error versus Iterations for Signature Recognition

Table 4.3 contains the training results for the twenty patterns, while table 4.4 contains the test results for the eight test signature, as two for every person.

Number of Iterations is 1249

Table 4.3 Train Results for Signature Recognition

T. Set	Sign no	Sign 1	Sign 2	Sign 3	Sign 4
Train Signature Set 1		**0.9718**	0.0123	0.0452	0.0145
		0.0547	**0.9539**	0.0098	0.0198
		0.0030	0.0410	**0.9496**	0.0215
		0.0050	0.0086	0.0278	**0.9608**
Train Signature Set 2		**0.9655**	0.0196	0.0055	0.0114
		0.0273	**0.9842**	0.0407	0.0105
		0.0006	0.0195	**0.9477**	0.0468
		0.0451	0.0098	0.0348	**0.9664**
Train Signature Set 3		**0.9774**	0.0312	0.0087	0.0210
		0.0101	**0.9336**	0.0315	0.0246
		0.0459	0.0085	**0.9623**	0.0100
		0.0043	0.0216	0.0249	**0.9486**
Train Signature Set 4		**0.9591**	0.0137	0.0272	0.0113
		0.0294	**0.9709**	0.0131	0.0484
		0.0023	0.0123	**0.9522**	0.0040
		0.0275	0.0477	0.0338	**0.9571**
Train Signature Set 5		**0.9466**	0.0175	0.0276	0.0031
		0.0224	**0.9729**	0.0196	0.0445
		0.0422	0.0313	**0.9611**	0.0494
		0.0021	0.0141	0.0181	**0.9414**
Train Signature Set 6		**0.9580**	0.0562	0.0087	0.0124
		0.0315	**0.9259**	0.0315	0.0121
		0.0155	0.0049	**0.9623**	0.0102
		0.0018	0.0290	0.0249	**0.9823**

Table 4.4 Test Results for Signature Recognition

T. Set	Sign no	Sign 1	Sign 2	Sign 3	Sign 4
Test		**0.9844**	0.0407	0.0001	0.0353
Signature		0.0121	**0.8311**	0.3206	0.0271
Set 1		0.1177	0.0339	**0.9570**	0.0014
		0.0003	0.0114	0.5031	**0.9530**
Test		**0.7887**	0.0103	0.0172	0.0116
Signature		0.4260	**0.9051**	0.0093	0.1256
Set 2		0.0040	0.0160	**0.9321**	0.2059
		0.0069	0.2850	0.1311	**0.1694**

Program Test Display:

The Signature is for Mr. Musa'b Recognition Percentage = 98.44
The Signature is for Mr. Samer Recognition Percentage = 83.11
The Signature is for Mr. Waled Recognition Percentage = 95.70
The Signature is for Mr. Adham Recognition Percentage = 95.30
The Signature is for Mr. Musa'b Recognition Percentage = 78.87
The Signature is for Mr. Samer Recognition Percentage = 90.51
The Signature is for Mr. Waled Recognition Percentage = 93.21
The Signature is for Mr. Adham Recognition Percentage = 01.69

Table 4.5 Recognition Rate & Recognition Accuracy for Signature Recognition

Signature set	Recognition Rate	Recognition Accuracy
Training Set	**(24/24) = 100%**	**95.88%**
Testing Set	**(7/8) = 87.5%**	**81.50%**

4.12 Analysis of Results

After training and testing of both logo classification and signature recognition it seems that the neural network is good tool for document certification.

After looking to train and test results, it seems that the neural network can learn with six training sets and recognizes other signature sets with high percentage (91.16%). During experiment it is noticed that the network could not learn with four training sets. As a result, the training should be more than six signatures for every person.

4.13 PERFORMANCE OF THE SYSTEM (TESTING)

There are two different errors that characterize the performance of the algorithm. The Type I error (or False Rejection Rate (FRR)), measures the number of true signatures classified as forgeries as a function of the classification threshold. The Type II error (or False Acceptance Rate (FAR)), evaluates the number of false signatures classified as real ones as a function of the classification threshold. The performance of the system is provided by these two errors as a function of the classification threshold. Clearly, we can trade-off one type of error for the other type of error. As an extreme example, if we accept every signature, we will have a 0% of FRR and a 100% of FAR, and if we reject every signature,we will have a 100% of FRR and a 0% of FAR. The curve of FAR as a function of FRR, using the classification threshold as a parameter, is called the error trade-off curve. It provides the behavior of the algorithm for any operation regime and it is the best descriptor of the performance of the algorithm. In practice, this curve is often characterized by the equal error rate, i.e., the error rate at which the percentage of false accepts equal the percentage of false rejects. This equal error rate provides an estimate of the statistical performance of the algorithm, i.e., it provides an estimate of its generalization error.

4.13.1 Future Work

We have developed a vision-based system for personal identification based on signature tracking. Our system achieves and equal error rate of 0.2% for random forgeries and 3.3% for intentional forgeries. We have shown the importance of choosing the proper parameterization of the signatures in order to achieve good performance. We have also described a method that allows to overcome the lack of examples in order to estimate the generalization error of the algorithm.

As a future work,test is done on signatures written with black ink, and if it is written with blue or red or green ink it is possible that the network will not recognize it. The future work can continue to solve this problem.

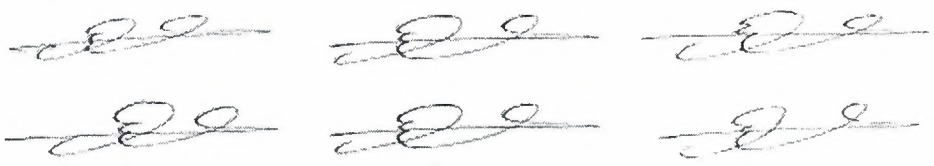
4.14 Summary

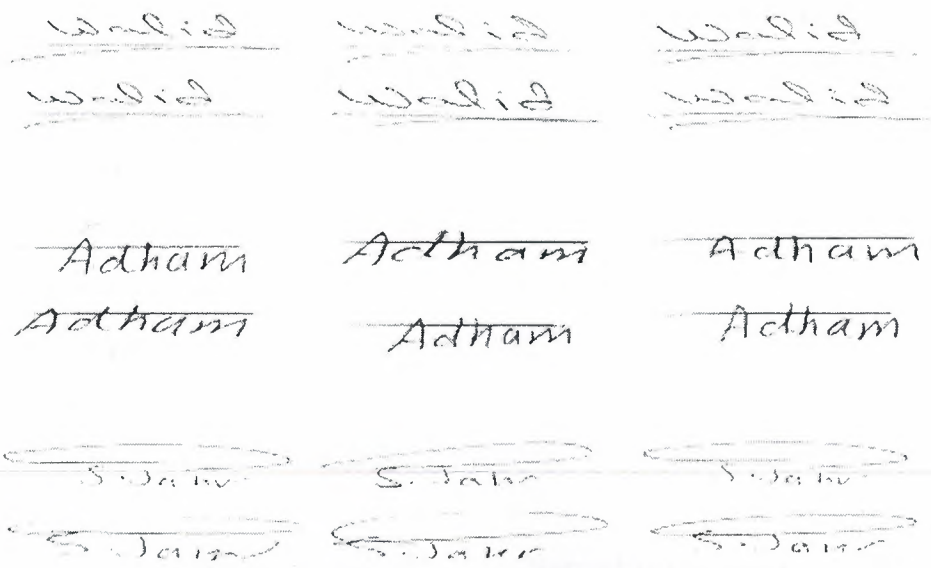
In this chapter the signature verification has been added as fully detailed explanation over whole related topics respectively.

4.15 Appendix

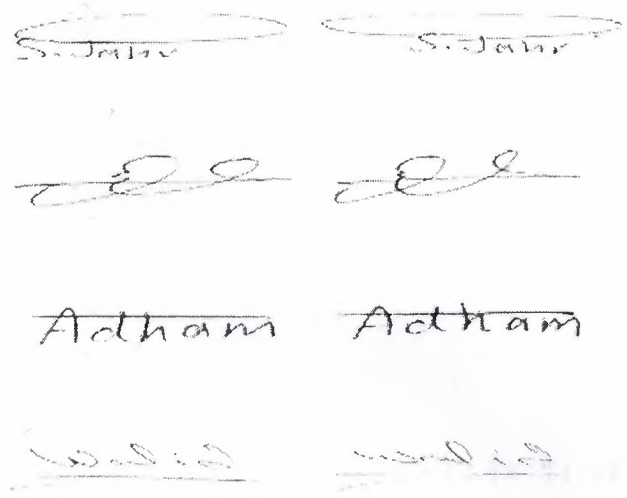
1. DATA BASE

1.1 Signature Training Set





1.2 Signature Testing Set



2. Signature Recognition Software Program

```
clear all
close all
clc
sig_number = 4;
sig_expression = 6;
PATTERNS = [];
```

```
cd( 'D:\Documents and Settings\Khaled Waleed\Desktop\sig khaled\')
% put the directory to get the images in
for l = 1:sig_expression                % Start for loop to read signature image
   for k = 1:sig_number
   image = strcat(['signtrain' int2str(k),'_' int2str(l),'.jpg']);
   sig_image = imread(image);
   sig_Image = imresize(sig_image,[15 30],'bicubic');        % Resize the image to
18x40 pixel
   gray_sig = rgb2gray(sig_Image);          % Convert the colour image to grayscale
image
   gray_sig = double(gray_sig)/255;      % Normalize the image to 0 - 1
   [row col] = size(gray_sig);
   vector_sig = reshape(gray_sig,[],1);  % Convert the image matrix to vector
   PATTERNS = [PATTERNS vector_sig];
   end
end
%%%%  Desired Output %%%%
T1 = [1;0;0;0];
T2 = [0;1;0;0];
T3 = [0;0;1;0];
T4 = [0;0;0;1];

D_output = [T1 T2 T3 T4  T1 T2 T3 T4  T1 T2 T3 T4  T1 T2 T3 T4  T1 T2 T3 T4
T1 T2 T3 T4 ];
[g,h]=size(PATTERNS);
[m,h]=size(D_output);
% CREATING AND INITIATING THE NETWORK
net = newff(minmax(PATTERNS),[17 4],{'logsig','logsig'},'traingdx'); %( for change
**********
net = init(net);
net.LW{2,1} = net.LW{2,1}*0.01;
net.b{2} = net.b{2}*0.01;

% TRAINING THE NETWORK
```

```
net.trainParam.goal = 0.001; % Sum-squared error goal.
net.trainParam.lr = 0.001;% Learning Rate ( for change *************
net.trainParam.show = 10; % Frequency of progress displays (in epochs).
net.trainParam.epochs = 6000; % Maximum number of epochs to train.
net.trainParam.mc = 0.3; % Momentum Factor.( for change *************

[net,tr] = train(net,PATTERNS,D_output);
% TEST
for k=1:h
   PATTERNS(:,k);
sigtrain = sim(net,PATTERNS(:,k))
end
disp(sprintf('If You Want To Continue To Test Press Enter'));
pause
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sig_number = 4;
sig_expression = 2;
TEST_PATTERNS = [];
Thresh = 0.8;
cd( 'D:\Documents and Settings\Khaled Waleed\Desktop\sig khaled\')




for l = 1:sig_expression
   for k = 1:sig_number
   image = strcat(['signtest' int2str(k),'_' int2str(l),'.jpg']);
   sig_image = imread(image);
   sig_Image = imresize(sig_image,[15 30],'bicubic');
   gray_sig = rgb2gray(sig_Image);
   gray_sig = double(gray_sig)/255;
   [row col] = size(gray_sig);
```

```
   vector_sig = reshape(gray_sig,[],1);
   TEST_PATTERNS=vector_sig;

   [m,n]=size(TEST_PATTERNS);
s=0;
for k=1:n
   TEST_PATTERNS(:,k);
   s=s+1;
sigtest = sim(net,TEST_PATTERNS(:,k))
end
end
end
```

CONCLUSION

The hand written signatures are to be classified. In fact, exercises in imitating a signature often allow people to produce forgeries so similar to the originals that discrimination is practically impossible; in many cases, the distinction is complicated even more by the large variability introduced by some signers when writing their own signatures. The old procedure for signature verification was done by experts of hand writes. It takes a long time, when the time is money or war or human life.

This assignment proposes an approach to classify the hand written signature using backpropagation neural network. It proposes to train the net for four hand written signatures for four persons. When any related document is to be certified, its logos and signatures are input to the net to classify them.

This assignment will consist from four sections.

Chapter one is an artificial neural network .

Chapter two is about intelligent pattern recognition.

Chapter three is Back propagation algorithm.

Chapter four will introduce the signature recognition.

The fully detailed explanation was included about the algorithm block diagram, the neural network parameters adjustment, software program and the results of the signature recognition with addition to the future work.

REFERENCES

[1] Partt. William k. Digital Image Processing. New York: John Wiley & Sons. Inc. 1991. p. 634.

[2] Horn. Berthold P . K. Robot Vision New York: McGraw-Hill, 1986. pp. 73-77.

[3] http://www.cs.brandeis.edu/~cs113/classprojects/~uday/cs113/bp1.htm

[4] http://www.cs.brandeis.edu/~cs113/classprojects/~uday/cs113/bp1.htm

[5] http://www.dictionary.com/cgi-bin/dict.pl?term=EEE

[6] Cardot H, Revenu M, Victorri B, Revillet MJ. A static signature verification system based on a cooperative neural network architecture. International Journal of Pattern Recognition and Artificial Intelligence 1994; 8(3):679–692

[7] Jain, Anil k. Fundamentals of Digital Image Processing. Englewood Cliffs, NJ: Prentice Hall ,1989 .pp. 150-153.

[8] Drouhard JP, Sabourin R, Godbout M. A neural network approach to off-line signature verification using directional PDF. Pattern Recognition 1996;29(3):415–424

[9] Bajaj R, Chaudhury S. Signature verification using multiple neural classifiers. Pattern Recognition 1997; 30(1):1–75. Dimauro G, Impedovo S, Pirlo G, Salzo A. A multi-expert

[10] Haralick, Robert M, and Linda G, Shapirio. Computer and Robert Vision, Volume I . Addison-Wesley, 1992. p.158.

[11] Murshed NA, Sabourin R, Bortolozzi F. A cognitive approach to off-line signature

verification. International Journal of Pattern Recognition and Artificial Intelligence 1997; 11(5):801–825

[12] Rahman AFR, Fairhurst MC. An Evaluation of Multi-Expert Configurations for the Recognition of Handwritten Numerals. Pattern Recognition 1998; 31(9):1255–1273.

[13] Canny, John "A Computational Approach to Edge Detection " IEEE Transaction on Pattern Analysis And Machine Intelligence, 1986. Vol. PAMI-8, No. 6,pp. 679-684.

[14] Lim Jae S. Two-Dimensional Signal and Image Processing. Englewood Cliffs, NJ: Prentice Hall ,1990 .pp. 478-488.

[15] http://www2.psy.uq.edu.au/~brainwav/Manual/BackProp.html

[16] http://www.stats.ox.ac.uk/~northrop/teaching/PAN/PR2005.pdf

[17] http://www.cs.tut.fi/sgn/m2obsi/courses/IPR/Lectures/IPR_Lecture_1.pdf

[18] Plamondon R, Lorette G. Automatic signature verification and writer

[19] Huang K, Yan H. Off-line signature verification based on geometric feature extraction an neural network classification. Pattern Recognition 1997; 30(1):9–17

[20] Qi Y, Hunt BR. A multiresolution approach to computer verification of handwritten signatures. IEEE Transactions on Image Processing 1995; 4(6):870–874.

[21] Van Den Boomgaard and Van Balen " Image Transforms Using Bitmapped Binary Images," Computer Vision Graphics, and Image Processing: Graphical Models and Image Processing, vol. 54. no 3, May 1992. pp. 254-256

[22] Kak, Avinsah C, and Malcolm Slaney, Principles of Computerized Tomographic Imaging. New York: IEEE Press.

[23] J. P. Lewis "Fast Normalized Cross-Ccorrelation" Industrial Light & Magic.

[24] Robert M. Haralick and Linda, Shapiro, Computer and Robot Vision, Volume II, Addison-esley, 1992.pp.316-317

[25] Bracewell, Ronald N. Two-Dimensional Imaging. Englewood Cliffs, NJ: Prentice Hall, 1995.pp. 505-507,

[26] Lim. Jae S. Two-Dimensional Signal and Image Processing. Englewood Cliffs, NJ: Prentice Hall, 1990.pp.42-44

[27] Rien van den Boomgard and Richard van Balen, "Methods for Fast Morphological Image Transforms Using Bitmapped Images " Computer vision, Graphics, and Image processing : graphical Models and Image processing. Vol 54.no. 3.May 1992.pp.252-254.