Proc. Nat. Sem on Fuzzy Math. & its
appl. Nov. 25-26, 2006; 31-46

Artificial Neural Network as a Soft Computing Tool – A case study

Mrinal Kanti Bhowmik
mkb_cse@yahoo.co.in
Department of Computer Science and Engineering, Engineering, Faculty
Tripura University, Suryamaninagar-799130

Abstract : This paper describes Artificial Neural Networks (ANN) as a tool used in soft computing domain. At the beginning, development of ANN from human brain is reported. Next, various types of neural networks are explained and demonstrated. The basic idea behind the ANN is learning from exemplars and handling real life ambiguities by recognizing similar situations which are not exactly same as some exemplar. The connection between the artificial and the real thing is also investigated and explained. Finally, applications of neural networks and some case studies like ANNs in medicine, ANNs in business are described.

1. Introduction to neural networks

1.1 What is a Neural Network?

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous system, 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 (neurones) 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 connectiions that exist between the neurones. The is true of ANNs as well.

1.2 Why use neural networks?

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.

2. Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.

3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

1.3 Neural networks versus conventional computers

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurones) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

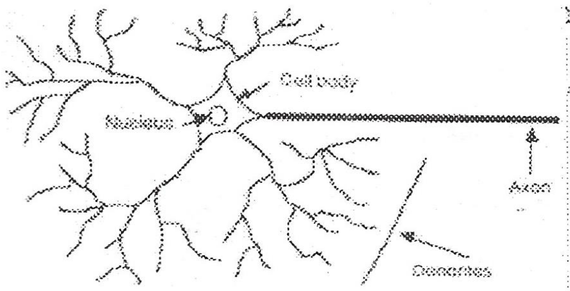
Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

Neural networks do not perform miracles. But if used sensibly they can produce some amazing results.

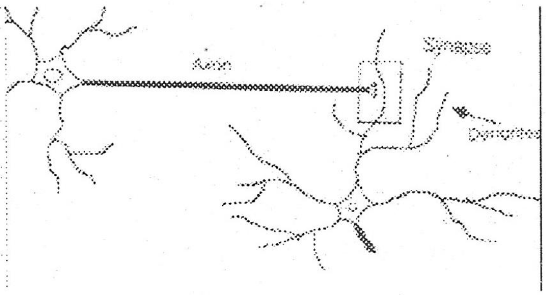
2. Human and Artificial Neurones - investigating the similarities.

2.1 How the Human Brain Learns?

Much is still unknown about how the brain trains itself to process information, so theories abound. In the human brain, a typical neuron collects signals from others through a host of fine structures called dendrites. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurones. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes.



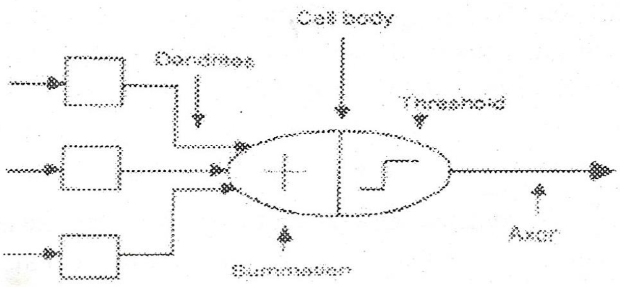
Components of a neuron



The synapse

2.2 From Human Neurones to Artificial Neurones

We conduct these neural networks by first trying to deduce the essential features of neurones and their interconnections. We then typically program a computer to simulate these features. However because our knowledge of neurones is incomplete and our computing power is limited, our models are necessarily gross idealisations of real networks of neurons.

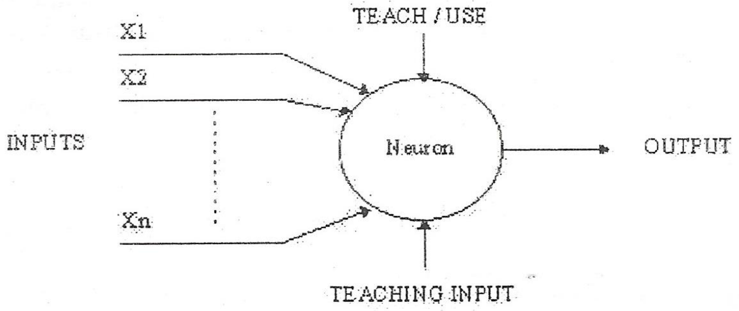


The neuron model.

3. An engineering approach

3.1 A simple neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.



A simple neuron

3.2 Firing rules

The firing rule is an important concept in neural networks and accounts for their high flexibility. A firing rule determines how one calculates whether a neuron should fire for any input pattern. It relates to all the input patterns, not only the ones on which the node was trained.

A simple firing rule can be implemented by using Hamming distance technique. The rule goes as follows:

Take a collection of training patterns for a node, some of which cause it to fire (the 1-taught set of patterns) and others which prevent it from doing so (the 0-taught set). Then the patterns not in the collection cause the node to fire if, on comparison, they have more input elements in common with the 'nearest' pattern in the 1-taught set than with the 'nearest' pattern in the 0-taught set. If there is a tie, then the pattern remains in the undefined state.

For example, a 3-input neuron is taught to output 1 when the input (X1,X2 and X3) is 111 or 101 and to output 0 when the input is 000 or 001. Then, before applying the firing rule, the truth table is;

X1:	0	0	0	0	1	1	1	1
X2:	0	0	1	1	0	0	1	1
X3:	0	1	0	1	0	1	0	1
OUT:	0	0	0/1	0/1	0/1	1	0/1	1

As an example of the way the firing rule is applied, take the pattern 010. It differs from 000 in 1 element, from 001 in 2 elements, from 101 in 3 elements and from 111 in 2 elements. Therefore, the 'nearest' pattern is 000 which belongs in the 0-taught set. Thus the firing rule requires that the neuron should not fire when the input is 001. On the other hand, 011 is equally distant from two taught patterns that have different outputs and thus the output stays undefined (0/1).

By applying the firing in every column the following truth table is obtained:

X1:	0	0	0	0	1	1	1	1
X2:	0	0	1	1	0	0	1	1
X3:	0	1	0	1	0	1	0	1
OUT:	0	0	0	0/1	0/1	1	1	1

The difference between the two truth tables is called the generalisation of the neuron. Therefore the firing rule gives the neuron a sense of similarity and enables it to respond 'sensibly' to patterns not seen during training.

3.3 Pattern Recognition - an example

An important application of neural networks is pattern recognition. Pattern recognition can be implemented by using a feed-forward (figure 1) neural network that has been trained accordingly. During training, the network is trained to associate outputs with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern.

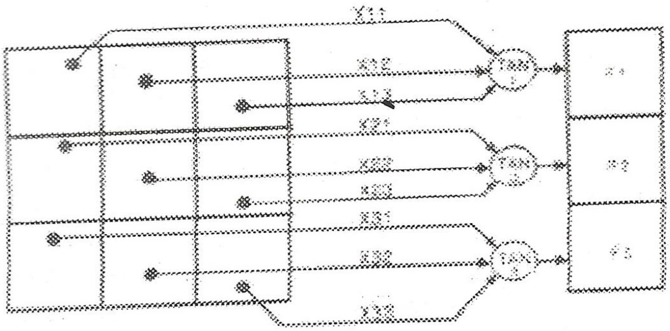
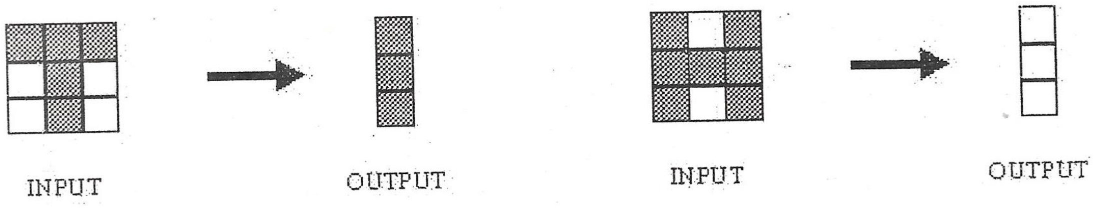


Figure 1.

For example:

The network of figure 1 is trained to recognise the patterns T and H. The associated patterns are all black and all white respectively as shown below.



If we represent black squares with 0 and white squares with 1 then the truth tables for the 3 neurones after generalisation are;

X11:	0	0	0	0	1	1	1	1
X12:	0	0	1	1	0	0	1	1
X13:	0	1	0	1	0	1	0	1
OUT:	0	0	1	1	0	0	1	1

Top neuron

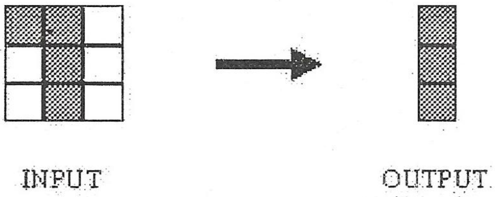
X21:	0	0	0	0	1	1	1	1
X22:	0	0	1	1	0	0	1	1
X23:	0	1	0	1	0	1	0	1
OUT:	1	0/1	1	0/1	0/1	0	0/1	0

Middle neuron

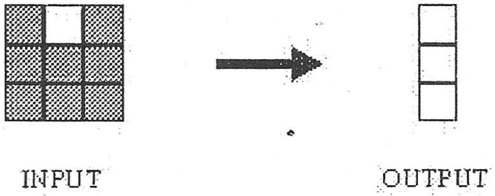
X21:	0	0	0	0	1	1	1	1
X22:	0	0	1	1	0	0	1	1
X23:	0	1	0	1	0	1	0	1
OUT:	1	0	1	1	0	0	1	0

Bottom neuron

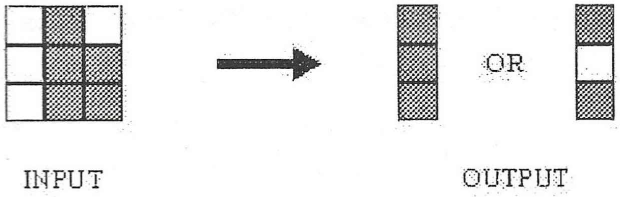
From the tables it can be seen the following associasions can be extracted:



In this case, it is obvious that the output should be all blacks since the input pattern is almost the same as the 'T' pattern.



Here also, it is obvious that the output should be all whites since the input pattern is almost the same as the 'H' pattern.



Here, the top row is 2 errors away from the a T and 3 from an H. So the top output is black. The middle row is 1 error away from both T and H so the output is random. The bottom row is 1 error away from T and 2 away from H. Therefore the output is black. The total output of the network is still in favour of the T shape.

3.4 A more complicated neuron

The previous neuron doesn't do anything that conventional computers don't do already. A more sophisticated neuron (figure 2) is the McCulloch and Pitts model (MCP). The difference from the previous model is that the inputs are 'weighted', the effect that each input has at decision making is dependent on the weight of the particular input. The weight of an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire.

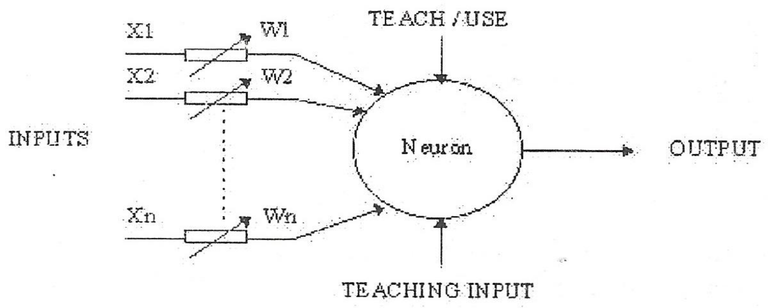


Figure 2. An MCP neuron

In mathematical terms, the neuron fires if and only if;

$$X_1W_1 + X_2W_2 + X_3W_3 + \ldots > T$$

The addition of input weights and of the threshold makes this neuron a very flexible and powerful one. The MCP neuron has the ability to adapt to a particular situation by changing its weights and/or threshold. Various algorithms exist that cause the neuron to 'adapt'; the most used ones are the Delta rule and the back error propagation. The former is used in feed-forward networks and the latter in feedback networks.

4 Architecture of neural networks

4.1 Feed-forward networks

Feed-forward ANNs (figure 1) allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organisation is also referred to as bottom-up or top-down.

4.2 Feedback networks

Feedback networks (figure 1) can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organisations.

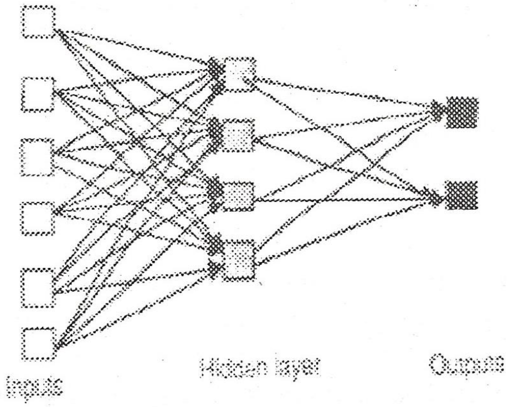


Figure 4.1 An example of a simple feedforward network

4.3 Network layers

The commonest type of artificial neural network consists of three groups, or layers, of

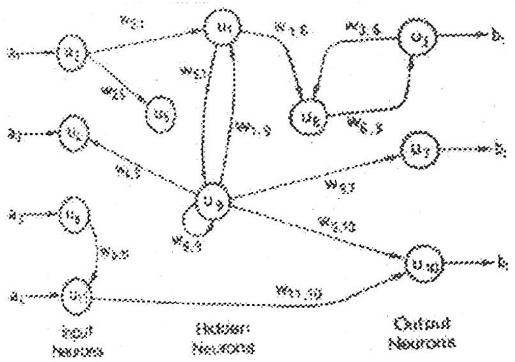


Figure 4.2 An example of a complicated network

units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units. (See Figure 4.1)

1. The activity of the input units represents the raw information that is fed into the network.

2. The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.

3 The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

We also distinguish single-layer and multi-layer architectures. The single-layer organisation, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organisations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

4.4 Perceptrons:

The most influential work on neural nets in the 60's went under the heading of

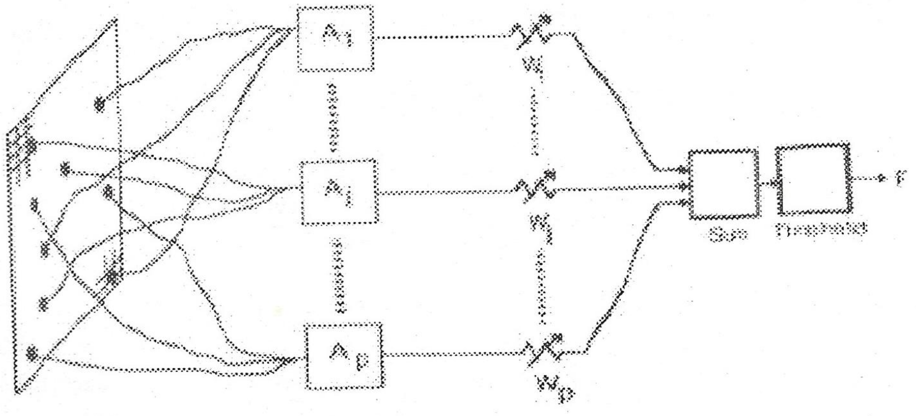


Figure 4.4

'perceptrons' a term coined by Frank Rosenblatt. The perceptron (figure 4.4) turns out to be an MCP model (neuron with weighted inputs) with some additional, fixed, pre—processing. Units labelled A1, A2, Aj, Ap are called association units and their task is to extract specific, localised featured from the input images. Perceptrons mimic the basic idea behind the mammalian visual system. They were mainly used in pattern recognition even though their capabilities extended a lot more.

In 1969 Minsky and Papert wrote a book in which they described the limitations of single layer Perceptrons. The impact that the book had was tremendous and caused a lot of neural network researchers to loose their interest. The book was very well written and showed mathematically that single layer perceptrons could not do some basic pattern recognition operations like determining the parity of a shape or determining whether a shape is connected or not. What they did not realised, until the 80's, is that given the appropriate training, multilevel perceptrons can do these operations.

5. The Learning Process

The memorisation of patterns and the subsequent response of the network can be categorised into two general paradigms:

1. Associative mapping in which the network learns to produce a particular pattern on the set of input units whenever another particular pattern is applied on the set of input units. The associtive mapping can generally be broken down into two mechanisms:

2. Auto-association: an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern completition, ie to produce a pattern whenever a portion of it or a distorted pattern is presented. In the second case, the network actually stores pairs of patterns building an association between two sets of patterns.

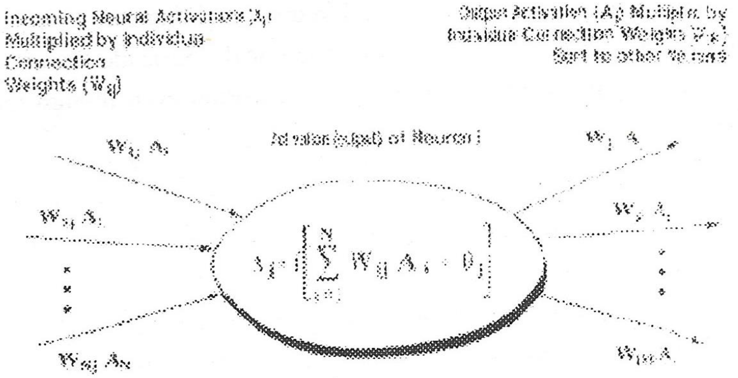
3. hetero-association: is related to two recall mechanisms:

4. nearest-neighbour recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented, and

5. Interpolative recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. Yet another paradigm, which is a variant associative mapping, is classification, ie when there is a fixed set of categories into which the input patterns are to be classified.

6. Regularity detection in which units learn to respond to particular properties of the input patterns. Whereas in asssociative mapping the network stores the relationships among

patterns, in regularity detection the response of each unit has a particular 'meaning'. This type of learning mechanism is essential for feature discovery and knowledge representation.


Incoming Neural Activations (A_i) Multiplied by Individual Connection Weights (W_{ij})
Output Activation (A_j) Multiplied by Individual Connection Weights (W_{jk}) Sent to other Neurons
Activation (output) of Neuron j
$W_{1j}A_1$
$W_{2j}A_2$
$W_{nj}A_N$
$W_{j1}A_j$
$W_{j2}A_j$
$W_{jm}A_j$
$A_j = f\left[\sum_{i=1}^{N} W_{ij}A_i - \theta_j\right]$

Every neural network posseses knowledge which is contained in the values of the connections weights. Modifying the knowledge stored in the network as a function of experience implies a learning rule for changing the values of the weights.

Information is stored in the weight matrix W of a neural network. Learning is the determination of the weights. Following the way learning is performed, we can distinguish two major categories of neural networks:

1. **Fixed networks** in which the weights cannot be changed, ie dW/dt=0. In such networks, the weights are fixed a priori according to the problem to solve.

2. **Adaptive networks** which are able to change their weights, ie dW/dt not= 0.

All learning methods used for adaptive neural networks can be classified into two major categories:

1. **Supervised learning** which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning.

An important issue conserning supervised learning is the problem of error convergence, ie the minimisation of error between the desired and computed unit values. The aim is to determine a set of weights which minimises the error. One well-known method, which is common to many learning paradigms is the least mean square (LMS) convergence.

2. **Unsupervised learning** uses no external teacher and is based upon only local information. It is also referred to as self-organisation, in the sense that it self-organises data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hebbian lerning and competitive learning.

Ano2.2 From Human Neurones to Artificial Neuronesther aspect of learning concerns the distinction or not of a seperate phase, during which the network is trained, and a subsequent operation phase are distinct. A neural network learns on-line if it learns and operates at the same time. Usually, supervised learning is performed off-line, whereas usupervised learning is performed on-line.

5.1 Transfer Function

The behaviour of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories:

1. linear (or ramp)
2. threshold
3. sigmoid

For linear units, the output activity is proportional to the total weighted output.

For threshold units, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurones than do linear or threshold units, but all three must be considered rough approximations.

To make a neural network that performs some specific task, we must choose how the units are connected to one another (see figure 4.1), and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

We can teach a three-layer network to perform a particular task by using the following procedure:

1. We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.

2. We determine how closely the actual output of the network matches the desired output.

3. We change the weight of each connection so that the network produces a better approximation of the desired output.

5.2 The Back-Propagation Algorithm

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (EW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the EW.

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each EW by first computing the EA, the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EA has been computed for a unit, it is straight forward to compute the EW for each incoming connection of the unit. The EW is the product of the EA and the activity through the incoming connection.

Note that for non-linear units, (see Appendix C) the back-propagation algorithm includes an extra step. Before back-propagating, the EA must be converted into the EI, the rate at which the error changes as the total input received by a unit is changed.

6. Project work procedure of character recognition.

It has two parts :-

➔ Training phase.

➔ Testing phase.

Training phase

➔ We consider a set of English alphabetical characters such as A,B,C,....... which are to be recognized.

➔ The characters are engraved in 14 x 14 grid.

➔ The gride patterns are represented as a bipolar vector of 196 components using input matrix X → 1x196

➔ If pixel in the grid is shaded, the vector component is 1 otherwise it is - 1

➔ These characters are to be associated with their ASCII equivalent.

➔ ASCII numbers of the characters have been represented using bipolar equivalents using a output matrix Y → 1x7.

Associative memory calculation.

■ Associative memory for character A

M = XTY

X → 1 x 196

XT → 196 x 1

Y → 1 x 7

Association of A

MA = (196 x 1) X (1 x 7)
= 196 x 7
Association of B
MB = 196 x 7
Total associative memory for all al phabets (A to Z)
= ΣXTi Yi
Where i = 1 to 26

Thus, the (X, Y) pattern pairs which are to be associated using the associative memory model are the grid patterns.

And their ASCII equivalents.

Testing phase

❑ In the refinement of the pattern or removal of noise to retrieve the closest matching stored pattern we have used the hamming distances.

❑ Hamming distance of a vector X from Y
HD (x,y) =Σ Xi - Yi

An Example to illustrate the avbove teaching procedure :

➔ We want a Network to recognise Character. We might use an arry of, say, 196 sensors, each recording the presence or absence of Ink in a small area of a single character.

➔ The Network would therefore need 196 Input units (One for each sensor), 26 output units (One for each kind of character).

➔ For each kind of Character recorded by the sensors, the network should produce high activity in the appropriate output unit and low activity in the other output units.

➔ To train the network, we present an image of a Character and compare the actual activity of the 10 output units with the desired activity. We then calculate the error, which is defined as the square of the difference between the actual and the desired activities. Next wechange the weight of each connection so as to reduce the error. We repeat this training process for many different image of each different image of each kind of Character until the network classifies every image correctly.

➔ To calculate the Error derivative for the weight (EW) is to use the Back-propagation algorithm which is described above, and has become now a days one of the most important tools for training neural networks. It was developed independently by two teams, one (Forelman-Soulie, Gallinari and Le Cun) in France, the other (Rumelhart, Hinton and Williams) in U.S.

7. Conclusion

The computing world has a lot to gain fron neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast responseand computational times which are due to their parallel architecture.

Neural networks also contribute to other areas of research such as neurology and psychology. They are regularly used to model parts of living organisms and to investigate the internal mechanisms of the brain.

Perhaps the most exciting aspect of neural networks is the possibility that some day 'consious' networks might be produced. There is a number of scientists arguing that conciousness is a 'mechanical' property and that 'consious' neural networks are a realistic possibility.

Finally, I would like to state that even though neural networks have a huge potential we will only get the best of them when they are intergrated with computing, AI, fuzzy logic and related subjects.

References:

1. An introduction to neural computing. Aleksander, I. and Morton, H. 2nd edition
2. Neural Networks at Pacific Northwest National Laboratory http://www.emsl.pnl.gov:2080/docs/cie/neural/neural.homepage.html
3. Industrial Applications of Neural Networks (research reports Esprit, I.F.Croall, J.P.Mason)
4. A Novel Approach to Modelling and Diagnosing the Cardiovascular System http://www.emsl.pnl.gov:2080/docs/cie/neural/papers2/keller.wcnn95.abs.html
5. Artificial Neural Networks in Medicine http://www.emsl.pnl.gov:2080/docs/cie/techbrief/NN.techbrief.ht
6. Neural Networks by Eric Davalo and Patrick Naim
7. Learning internal representations by error propagation by Rumelhart, Hinton and Williams (1986).
8. Klimasauskas, CC. (1989). The 1989 Neuro Computing Bibliography. Hammerstrom, D. (1986). A Connectionist/Neural Network Bibliography.
9. DARPA Neural Network Study (October, 1987-February, 1989). MIT Lincoln Lab. Neural Networks, Eric Davalo and Patrick Naim
10. Assimov, I (1984, 1950), Robot, Ballatine, New York.
11. Electronic Noses for Telemedicine http://www.emsl.pnl.gov:2080/docs/cie/neural/papers2/keller.ccc95.abs.html
12. Pattern Recognition of Pathology Images http://kopernik-eth.npac.syr.edu:1200/Task4/pattern.html