1. **What is a Neural Network?**

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.

1. **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 analyze. 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-Organization: An ANN can create its own organization 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.

### 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(neurons) 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.

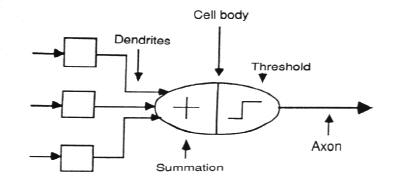
1. **How the Human Brain Learns?**

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 in the connected neurons. 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.

|  |  |
| --- | --- |
| New Picture.bmp  *Figure: Components of a neuron* | New Picture (1).bmp  *Figure: The synapse* |

### From Human Neurons to Artificial Neurons

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

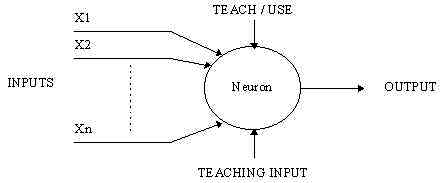


*Figure: The neuron model*

## An engineering approach

### 6.1 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.



*Figure: A simple neuron*

**6.2 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).

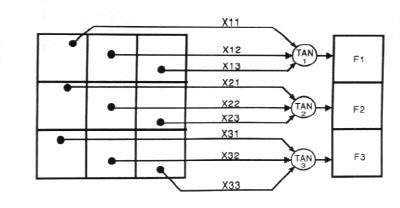
By applying the firing in every column the following truth table is obtained;

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 *generalization 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.

### 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.



For example:

The network of figure 1 is trained to recognize 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 neurons after generalization 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**

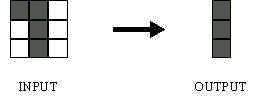
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X21: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| X22: | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| X23: | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|  |  |  |  |  |  |  |  |  |
| OUT: | 1 | 0/1 | 1 | 0/1 | 0/1 | 0 | 0/1 | 0 |

**Middle neuron**

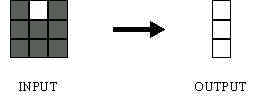
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X21: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| X22: | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| X23: | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|  |  |  |  |  |  |  |  |  |
| OUT: | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |

**Bottom neuron**

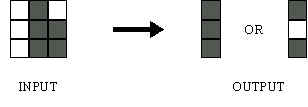
 From the tables it can be seen the following associations can be extracted:



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 favor of the T shape.

### A more complicated neuron

The previous neuron doesn't do anything that conventional 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.

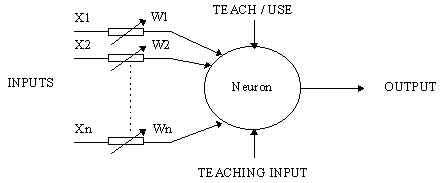


Figure 2. An MCP neuron

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

X1W1 + X2W2 + X3W3 + ... > 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.

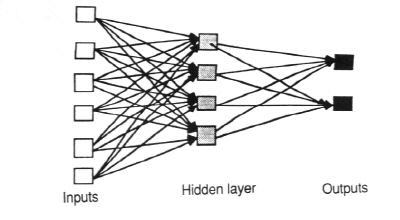
## Architecture of neural networks

### 9.1 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.

### 9.2 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 organizations.

**

*Figure: An example of a simple feedforward network*

### Network layers

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "**input**" units is connected to a layer of "**hidden**" units, which is connected to a layer of **"output**" units.

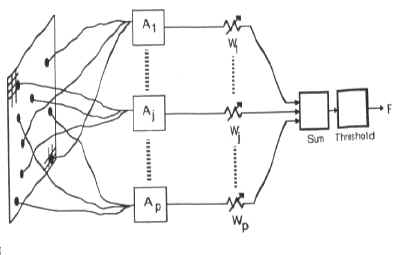
* The activity of the input units represents the raw information that is fed into the network.
* The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
* The behavior of the output units depends on 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 organization, 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 organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

### Perceptrons

The most influential work on neural nets in the 60's went under the heading of 'perceptrons' a term coined by Frank Rosenblatt. The perceptron turns out to be an MCP model ( neuron with weighted inputs ) with some additional, fixed, pre--processing. Units labeled A1, A2, Aj , Ap are called association units and their task is to extract specific, localized 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.

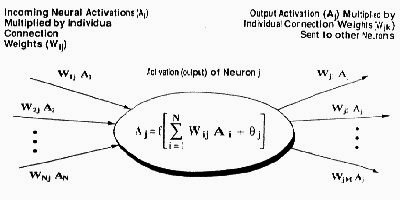


## The Learning Process

The memorization of patterns and the subsequent response of the network can be categorized into two general paradigms:

* **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 associative mapping can generally be broken down into two mechanisms:
  + Auto-association: an input pattern is associated with itself and the states of input and output units coincide. This is used to provide pattern competition, i.e. 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.
  + Hetero-association: is related to two recall mechanisms:
    - nearest-neighbor recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented, and
    - 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.
* **Regularity detection** in which units learn to respond to particular properties of the input patterns. Whereas in associative 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.

 Every neural network possesses 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:

* **Fixed networks** in which the weights cannot be changed, i.e. dW/dt=0. In such networks, the weights are fixed a priori according to the problem to solve.
* **Adaptive networks** which are able to change their weights, i.e. dW/dt not= 0.

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

* **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.  
  Important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. One well-known method, which is common to many learning paradigms, is the least mean square (LMS) convergence.
* **Unsupervised learning** uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties. Paradigms of unsupervised learning are Hibbing learning and competitive learning.

#### 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:

* Linear (or ramp)
* Threshold
* Sigmoid

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

For **threshold units**, the output are 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 neurons 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.

#### An Example to illustrate the above teaching procedure:

Assume that we want a network to recognize hand-written digits. We might use an array of, say, 256 sensors, each recording the presence or absence of ink in a small area of a single digit. The network would therefore need 256 input units (one for each sensor), 10 output units (one for each kind of digit) and a number of hidden units.

For each kind of digit 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 digit 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 we change the weight of each connection so as to reduce the error. We repeat this training process for many different images of each different images of each kind of digit until the network classifies every image correctly.

To implement this procedure we need to calculate the error derivative for the weight (EW) in order to change the weight by an amount that is proportional to the rate at which the error changes as the weight is changed. One way to calculate the EW is to perturb a weight slightly and observe how the error changes. But that method is inefficient because it requires a separate perturbation for each of the many weights.

#### 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 **EA**s of those output units and add the products. This sum equals the **EA** for the chosen hidden unit. After calculating all the **EA**s in the hidden layer just before the output layer, we can compute in like fashion the **EA**s 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.

### Neural Networks in Practice

Given this description of neural networks and how they work, what real world applications are they suited for? Neural networks have broad applicability to real world business problems. In fact, they have already been successfully applied in many industries.

Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including:

* Sales forecasting
* Industrial process control
* Customer research
* Data validation
* Risk management
* Target marketing

But to give you some more specific examples; ANN are also used in the following specific paradigms: recognition of speakers in communications; diagnosis of hepatitis; recovery of telecommunications from faulty software; interpretation of multimeaning Chinese words; undersea mine detection; texture analysis; three-dimensional object recognition; hand-written word recognition; and facial recognition.

## 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 response and computational times which are due to their parallel architecture.

Perhaps the most exciting aspect of neural networks is the possibility that some day 'conscious' networks might be produced. There is a number of scientists arguing that consciousness is a 'mechanical' property and that 'conscious' 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 integrated with computing, AI, fuzzy logic and related subjects.