**CHAPTER TWO**

**NEURAL NETWORKS**

* 1. **Overview**

Artificial Neural Network (ANN) is a mathematical function designed to mimic the basic function of a biological neuron and it has used in many application such as Prediction, Classification of inputs and Data Filtering.

The training of the network by using back propagation algorithm is produced where in the forward pass the actual output is calculated and in the backward path the weights between output layer and hidden layer and between hidden layer and input layer will be adjusted, then steps of this algorithm is repeated until the error is reduced and the importance of sigmoid transfer function is presented also in details.

* 1. **History of Artificial Neural Networks (ANNs)**

A neural network is a machine that is designed to simulate the way of a human brain works, which is composed of a large number neurons working to gather to solve a specific problem.

The history of Artificial Neural Network can be traced back to the early 1940s. The first important paper on neural network was published by physiologist, Warren McCulloch and Walter Pitts in 1943, they proposed a simple model of neuron with electronic circuit, this model consists of two input and one output, in 1949 Donald Hebb proposed a learning law that become starting point for neural network training algorithm, in the 1950 and 1960, many researchers (Block, Minsky, Papert and Rosenblatt) worked on Perceptrons, where the first type of neural network is called Perceptrons. The Perceptron is a very simple mathematical representation of the neuron where most Artificial Neural Network is based on it to this day as shown below in figure 2.1

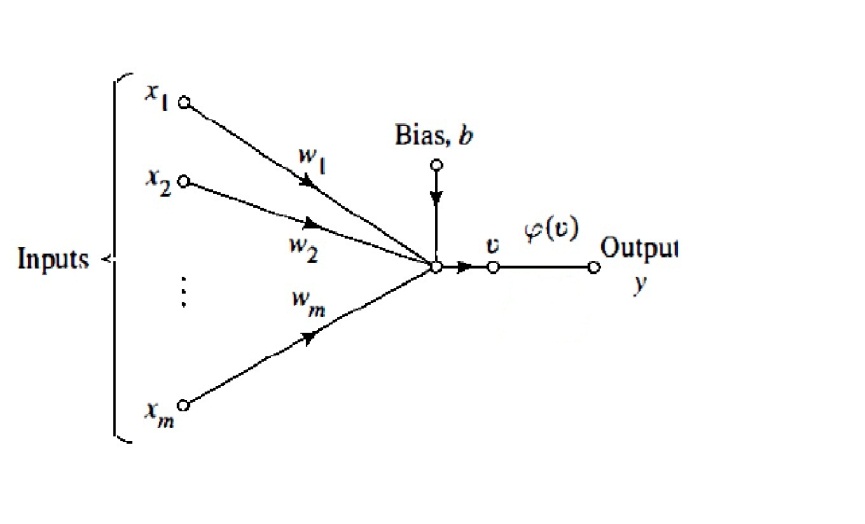


Figure 2.1:Perceptron[23].

This figure shows that the inputs of the neuron are represented by X1, X2… Xm then multiplied by corresponding weight W1, W2… Wm similar to the synaptic strength in a biological neuron, the externally applied bias is denoted by *b*. Summation of these inputs with their corresponding weights and bias ‘b’ is symbolized by V, where V is calculated by equation 2.1:

V = (2.1)

After that, the activation function is compared with value of a certain threshold. If the total summation of the inputs multiplied by their corresponding weight is more than the threshold the output (O) will be “fires” and if the total summation of the inputs multiplied by their corresponding weight is less than the threshold the output (O) will be “not fires”.

Bernard Widrow and Marcian Hoff in 1959, they developed model called "ADALINE" (Adaptive Linier Neuron) and "MADALINE" is composed of "many ADALINE" (Multilayer ADALINE).

Widrow and Hoff in 1960 developed a mathematical method for adapting the weight, where this algorithm was depended on minimizing the error squared, and then this algorithm would become called as least mean square error (LMS). In1962, Frank Rosenblatt was able to demonstrate the convergence of a learning algorithm. In 1969, Marvin Minsky and Seymour Papert published a book in which they showed that Perceptron could not learn this function which are not linearly separable.

The effect of these problems was to limit of the funding available for research into artificial neural networks therefore the neural networks research declined throughout 1970 and until mid of 1980.After a proof of the limitations of neural network in the 1970's, but much work was done on self-organizing maps by Willshaw and von der Malsburg. Hopfield presented a paper on neural networks with feedback known as Hopfield Networks.

The back propagation algorithm was first developed by Werbos in 1974; the most development happened around 1985- 1986 when Rumelhart, Hinton and Willimas invented (back-propagation), where back-propagation is a powerful tool for training multilayer neural network. Appearance of back-propagation method has spectacular the range of problems to which neural network can be applied [23, 24, 42, 43].

* 1. **Biological Neurons.**

The brain is composed of about 10 billion neurons each neurons is consists of five basic component that showed in figure 2.2

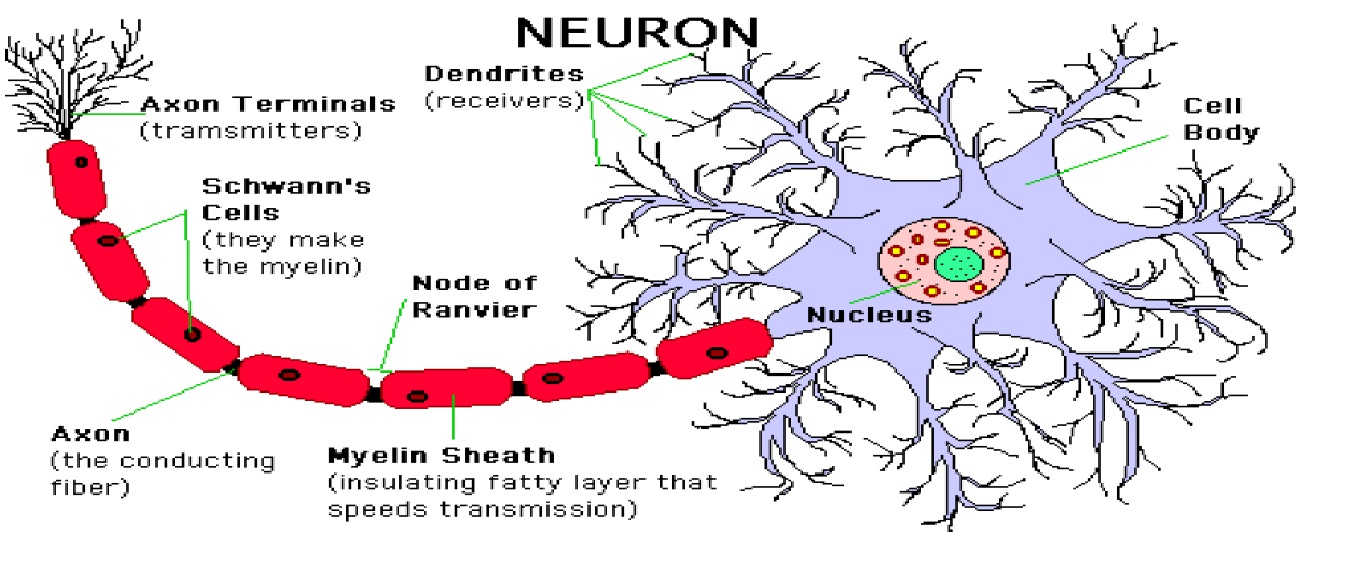


Figure 2.2: Schematic Diagram of a Biological Neuron [44].

* Dendrites: they are nerve fibre connected to cell bodies (soma), which are responsible for receiving signal from a connection point called a synapse.
* The neuron’s Cell body or soma: in which convert the incoming activations to the output activation.
* Axon: they are fibres performing as transmission lines that send activation to other neurons.
* A synaptic junction: which has both a receiving and transmitting side, when a signal is received then transmitted though chemical process in which specific transmitted substances are released from the sending side of synaptic junction, in turn changing the electrical potential inside the cell body (soma) of receiving neuron, if this potential exceed a threshold would be firing down the axon to other neurons.
* The neuron’s nucleus: where includes the genetic material in the form of DNA. This exists not just in neuron but exists in most types of cells [23, 45, 46].
  + 1. **How does the Human's Brain Work?**

The human brain has close to 100 billion nerve cells, called neurons. Each neuron is connected to thousands of others, creating a neural network that shuttles information in the form of stimuli, in and out of the brain constantly. Each of the yellow blobs in the figure 2.3 are neuronal cell bodies (soma), each neuron has long, thin nerve fibres called dendrites that bring information in and even longer fibres called axons that send information away.

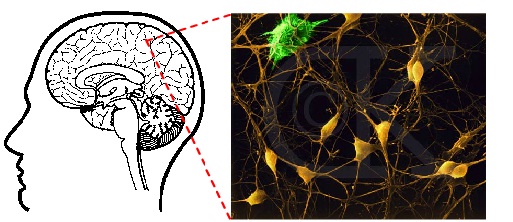
****

Figure 2.3: Biological neurons of human brain [47].

The neuron receives information in the form of electrical signals from neighboring neurons across one of thousands of synapses, small gaps that separate two neurons and act as input channels.

Once a neuron has received this charge it triggers either a "go" signal that allows the message to be passed to the next neuron or a "stop" signal that prevents the message from being forwarded, so it is important to note that a neuron fires only if the total signal received at the cell body exceeds a certain level.

For example, when a person thinks of something, sees an image, or smells a scent, that mental process or sensory stimulus excites a neuron, which fires an electrical pulse that shoots out through the axons and fires across the synapse. If enough input is received at the same time, the neuron is activated to send out a signal to be picked up by the next neuron's dendrites [23, 24, 47].

* 1. **Neural Network and their applications**

Neural network is a complex mathematical algorithm, and somewhat suitable to resolve all the issues that are not subject to the lows of mathematical constant and simulate the way of the human brain to identify the sound, word and images.

Majority applications of artificial neural network fall under three following sections:

* Classification :

Usage of the input values to assess the classification. E.g. character recognition.

* Prediction :

Usage of the input values to speculate the output. E.g. predict weather, pick the best stocks in the market.

* Filtering the data :

Make an input signal smoother such as: extraction the noise from the telephone's signal [23, 42].

* 1. **Transfer Function of Artificial Neural Networks (ANNs)**

Artificial Neural Network (ANN) was introduced by McCulloh and Pitts, where ANN is a mathematical function designed to mimic the basic function of a biological neuron, which is composed of a large number of (neurons) working together to solve a specific problems from training data that composed of inputs, weights of input and output.

Every input of that neuron is labeled X1, X2...Xn then multiplied by a corresponding weights W1,W2...Wn summation of the inputs with corresponding weights, and produces an output called (target) “NET” as shown in figure 2.4. Then the value of the result is compare with the value of the threshold.

net = (2.2)

Then the value of the result is compare with the value of the threshold,

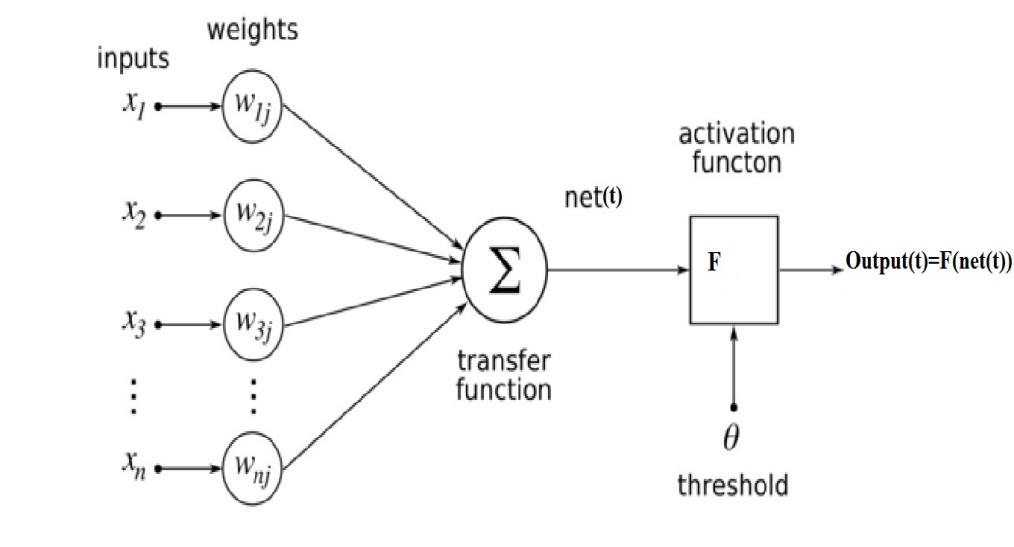


Figure 2.4: Schematic diagram of an artificial neuron [23, 25].

The Activation Function "F" is called transfer function, where the activation function "F" acts as a squashing function such that the output of a neuron in a neural network is between certain values. The transfer function translates the input signals to output signals. There are many of activation functions are such as Hard-limit Transfer Function, Linear Transfer Function and Sigmoid Transfer Function (logistic function) as shown below in figure 2.5

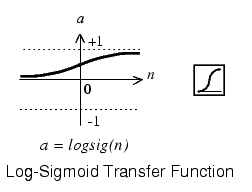
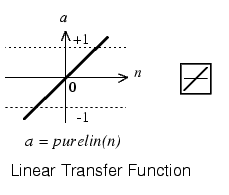
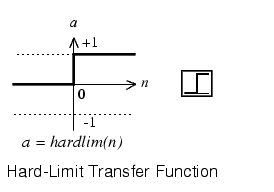
  


Figure 2.5: Types of transfer functions [48].

Where the output of Hard-limit function can be either "0" or "1" depended on the threshold. As a result of the non-continuity of this function so it has been found that it is not sufficient for multi-layer artificial neural network, so the activation function used to transform the activation level of neuron (weighted sum of inputs) to an output signal, the sigmoid function is most common type of activation function where it is an example of the logistic function therefore majority of Artificial Neural Networks (ANNs) use Sigmoid Transfer Function (logistic function) [23, 24, 25, 42].

* + 1. **Logistic Function**

A Logistic function is a common sigmoid curve, which has "s-shape”, given its name in 1844 or 1845 by Pierre François Verhulst who studied it in relation to population growth. Logistic functions are often used in neural networks to introduce nonlinearity in the model and/or to clamp signals to within a specified range. A logistic function is also known as a log-sigmoid function which the nonlinear curved s-shape function is called sigmoid function. Sigmoid function is most common type of activation function (A function used to transform the activation level of neuron (weighted sum of inputs) to an output signal) used to construct the neural network .It is mathematically well behaved, differentiable at everywhere and strictly increasing function .A sigmoid transfer function can be written in the form:

Y(x) =1 / (1+exp (-αx)) where “α” =1 (2.3)

Where Y(x): (the weighted sum of all synaptic input plus the bias) of neuron "x", and "y" is the output of the neuron. The sigmoid function is achieved by using exponential equation, and "α" is slope parameter and by varying "α" different shapes of the function can be obtained [23, 26, 49].

* + 1. **Unipolar Sigmoid Function**

Activation function of Unipolar Sigmoid Function is achieved by using logarithmic function where the output is limited between "0" and "1", the logarithmic sigmoid function is given by:

G(x) =1 / (1+exp (-x)) (2.4)

The input’s range of the unipolar transfer function is between plus infinity and minus infinity and squashes the output into the range between ("0" to "1"), as shown below infigure 2.6

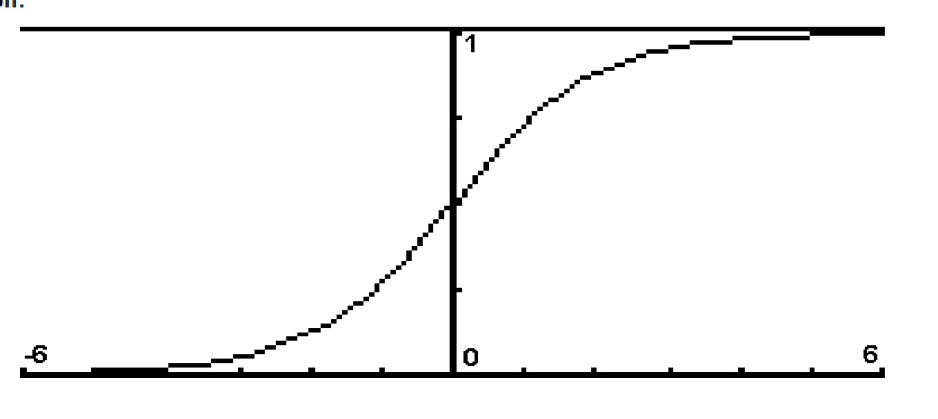
****

Figure 2.6: Unipolar Sigmoid Functions [26].

In the other word, the unipolar sigmoid is used when the range of the desired output is bounded between (zero to one) when the input has any value between (plus and minus infinity)[23, 26, 49].

* + 1. **Bipolar Sigmoid Function**

The Bipolar Sigmoid Function is similar to the sigmoid function but this activation function takes the input (which may have any value between plus infinity and minus infinity) and the output is changed into (-1 to 1) .In the other word the bipolar sigmoid is used when the range of the desired output is bounded between (- one to one) when the input has any value between (plus and minus infinity), as shown below in figure 2.7 [23, 26, 43, 49].

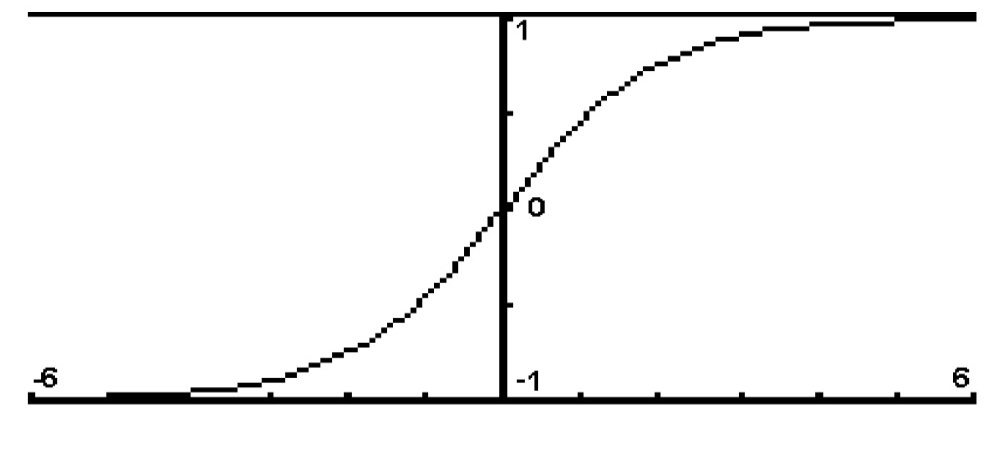


Figure 2.7: Bi-Polar Sigmoid function [26].

* 1. **Sigmoid Function in Back –Propagation Neural Network**

The artificial neural network (ANN) has been given more and more interest in the last decades and has been introduced in different application in many aspects of science.

The activation functions represent one of the most important components in the artificial neural network (ANN). They are responsible for giving the priority for the different inputs of the networks and then have a very important rule in output production.

The earlier neural networks has been constructed using hard-limit (a Threshold logic) activation functions where the output can be either zero or one, but the increasing need of the ANN in non-linear systems has imposed to use the activation functions with has non-linear properties that are not based on switching (hard-limit) functions and can give proportional output to the given input.

The use of such functions for continuous valued targets with bounded range has attracted the attention of researchers in the domain of ANN.

The sigmoid function which introduced in 1844 has been chosen to be introduced in the function of artificial neural network due to their non-linearity and continuity of the output which seems to be more effective and useful for using in the back-propagation neural network, where Back\_ propagation is an efficient and a popular method which was invented in 1986 by Rumelhart, Hinton and Williams for training multilayer neural network (the network have input layer and one or more than hidden layers and the output layer) to solve difficult problems, the training process consists of two passes through the layers of the network, the forward pass and backward pass.

In the forward pass (feed-forward):

The sigmoid function will be used to determine the output of the hidden –layer and the output of the output layer to introduce nonlinearity in the system.

In the backward pass:

The derivative of sigmoid function is used in the adjusting weights of the output layer and hidden layer to calculate the error of the back\_ propagation process therefor the derivative of sigmoid function is usually employed in learning of the network.

Because of the properties (differentiable everywhere and introducing non linearity in the system), so the logistic function (sigmoid function) is preferable to use in back-propagation neural network [23, 24, 25, 26, 27].

* + 1. **Single Layer Perceptron (SLP).**

A single-layer perceptron network (SLP) is the simplest kind of neural network. A single-layer perceptron (SLP) comprises of a number of external inputs then multiplied by corresponding weights and followed by the output layer as shown in figure 2.8.

A single-layer perceptron network can be considered the simplest kind of feed-forward network, where feed forward means that data flows from input to output layer in one direction, the output will be activated when the sum of the products of the inputs and the corresponding weights is above the threshold, where the output will be deactivated when the sum of the products of the corresponding inputs and the corresponding weights is below the threshold [50].

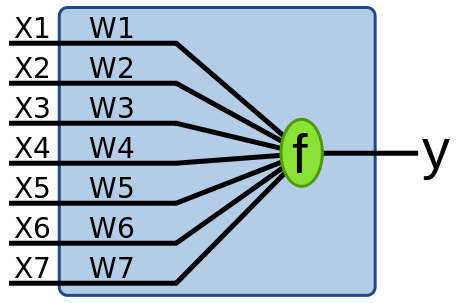


Figure 2.8: Single Layer Perceptron [50].

* + 1. **Multi-Layer Perceptron (MLP).**

Multi-Layer perceptron (MLP) is a second type of feed forward neural network, with one or more layers between input layer and output layer called hidden layers, therefore all neural networks have an input layer and an output layer as shown in figure 2.9, the number of input neurons normally corresponds to the number of independent variables which are fed the network, the number of hidden layers may vary from network to another network and the number of output neurons depends on what order the network is executing .

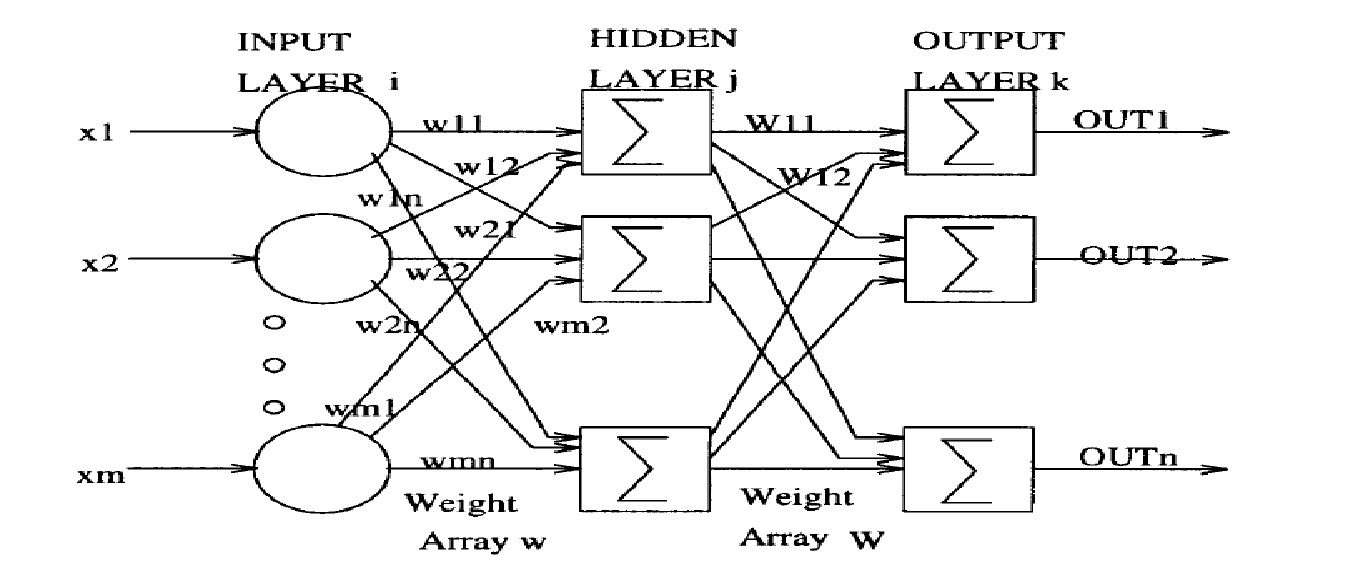


Figure 2.9: Multi-Layer Perceptron [24].

This network is consists of three layers: input layer on the left, one hidden layer in the middle and an output layer on the right.

* Input Layer :

An input layer is the first layer in a neural network that receives input data.

* Hidden layer :

There can be one or more hidden layers in feed forward neural networks with one or more neurons.

* Output layer :

There is one output layer in feed forward neural networks. The output layer is located after the input layer and the hidden layer, where the output layer is the third and last layer in artificial neural network. Multi-layer perceptron (MLP) can solve more complicated than single-layer where it can solve problems and obstacles which are not linearly separable by using Back Propagation algorithm, which can be used with any number of layers

When every node in each layer of the Artificial Neural Network is connected to every node in the neighbouring layer in this case the Artificial Neural Network is called the network of fully connected, where the Artificial Neural Network is called the network of partially connected when some of connection links are lost from the network [23, 24, 27, 28, 32, 33, 42, 43, 50]

* + 1. **Back Propagation Neural Network (BPNN)**

The Back propagation algorithm was first proposed by Paul Werbos in the 1970's. Where back propagation is a powerful tool created in 1986 when various researchers invented a systematic way for training multilayer artificial neural network to solve difficult problems them with highly algorithm called as the error back propagation algorithm.

The error back propagation consists of two basic passes through the network layer: the forward pass and the backward pass.

* In the forward pass the input is applied to the layers of the network, and its effect propagate through the network layer by layer, finally the outputs is produced as actual output of the network.
* During the backward pass the synaptic weights are adjusted (updated) by the error- correction rule. Specifically, the actual output is subtracted from a desired output called target to produce an error of the network, this error is propagated backward though the network, the synaptic weights at output layer and hidden layer are updated (adjusted) so as to make the actual output of the network closer to the desired output (target) [23, 24, 25, 28, 29, 30].

The forward pass and backward pass of the back propagation algorithm are shown below in figure 2.10.

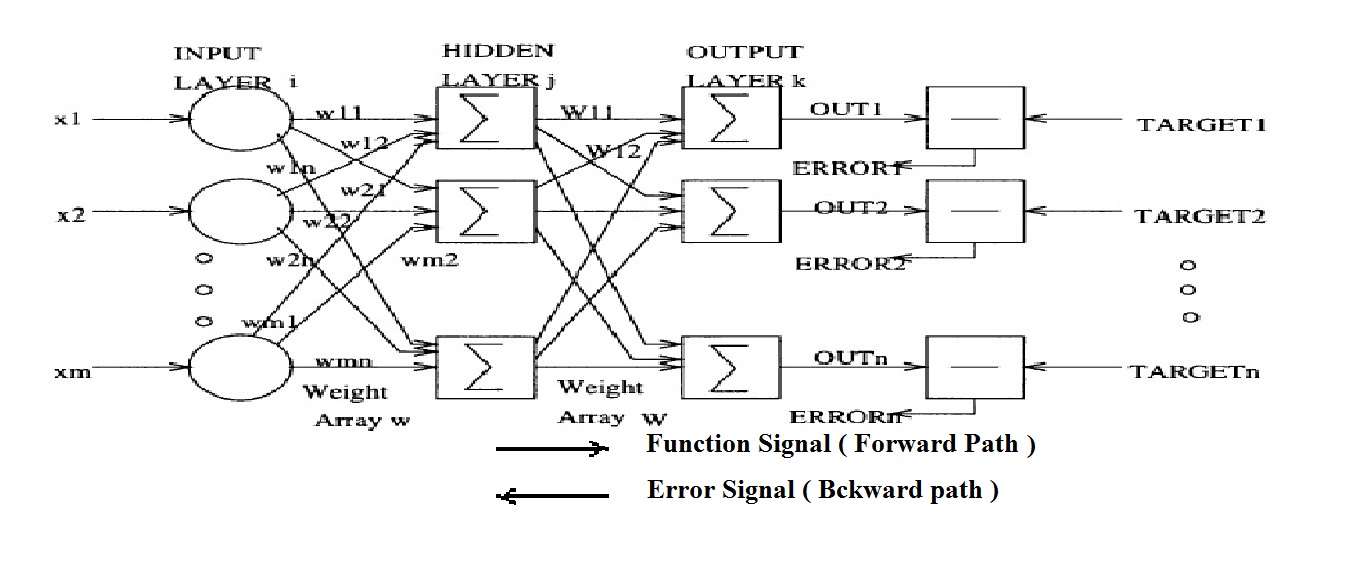


Figure 2.10: Back Propagation Neural Network Architecture [24, 51].

* + 1. **Feed Forward Path** **and Calculations**

The Feed Forward process started to learning neural network by using back propagation method where the simple three layer back propagation is shown below in figure 2.11.

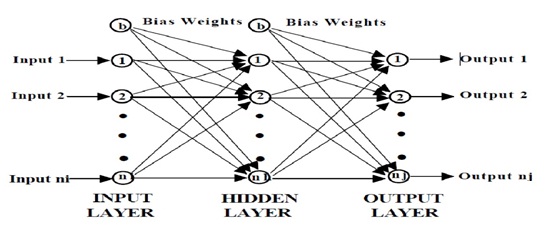


Figure 2.11: Back Propagation Network Structure [25].

Back propagation network consists of three layer: input layer (i), hidden layer (h) and output layer (j), when the inputs is passing forward through the layers the output are calculated by using a sigmoid activation function as shown in figure 2.12.

The output of any neuron (multiple inputs and signal output) in all the layers is given by using these equations:

Net = (2.5)

Out = F (net) (2.6)

F (Net) = 1 / 1 + exp (-net) (2.7)

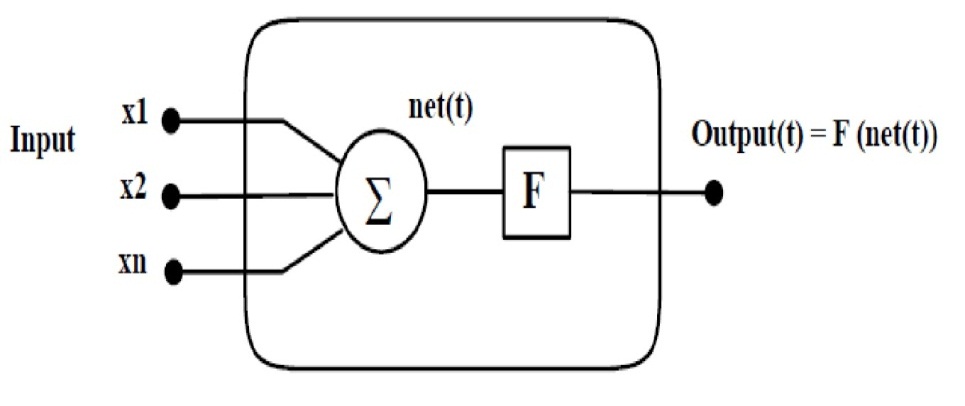


Figure 2.12: Artificial Neuron [25].

Where "F" is the sigmoid activation function, the derivative of sigmoid function is usually employed in learning of the network which is cab being obtained as follows:

∂ F (net) / ∂ net = exp (-net) / (1 + exp (-net)) ^2 = (1+ exp (-net)) \* (exp (-net) / 1 + exp (-net)) = out (1- out)

= F (net) (1- F (net)) (2.8)

Because of its properties (differentiable everywhere and introducing non linearity in the system), so the logistic function (sigmoid function) is preferable to use in back-propagation neural network [25, 32, 50, 52].

* + 1. **Input Layer (i), Hidden Layer (h) and Output Layer (j) in the Feed Forward Path.**

The feed forward path starts when the input data is passed forward through the network, where output of the input layer (Oi) is equal to input of the input layer (Ii) as written in this equation:

Input of the Input - Layer (Ii) = Output of the Input - Layer (Oi) (2.9)

Then each output of an input neuron in output of the input layer (Oi) is multiplied by their corresponding weight and summed to gather to present input of the hidden layer (Ih) as described in equation (2.10).

Input of the Hidden - Layer (Ih) = (2.10)

After that every output of a hidden neuron in output of the hidden layer (Oh) is calculated by using logistic function (sigmoid function) as written in equation (2.11).

Output of Hidden – Layer (Oh) = 1 / 1+exp (-Ih) (2.11)

Each output of an input neuron in output of the hidden layer (Oh) is multiplied by their corresponding weight and summed to gather to present input of the output layer (Ij) as described in equation (2.12).

Input of the Output - Layer (Ij) = (2.12)

Then every output of a neuron in output of the output layer (Oj) is calculated by using logistic function (sigmoid function) as written in equation (2.13).

Output of Output – Layer (Oj) = 1 / 1+exp (-Ij) (2.13)

From this equation (2.12), the output in the output layer (Oj) is a function for Input of the Output - Layer f (Ij).

These equations (2.9, 2.10, 2.11,2.12 and 2.13) that have been used above very important in the feed forward path for calculating the output which is totally defers from the desired output (Target), since all weights in all layers of the network are small random value usually between (-1 and +1) and (0 and +1) or other small values, then the error of each neuron in the output layer is calculated to be used in other layer of the network to update the weights [25, 50].

* + 1. **Backward Pass Propagation**

After the actual output was calculated in the feed forward path, the backward Pass propagation begins by calculation the Error of each neuron in the output layer, which is essentially equal (Target – Actual output that was calculated in the feed forward path).

Rumelhart and McClelland define the error in the network by the deference between the output value is supposed to have, called target and denoted by "Tj", and Actual output that was calculated in the feed forward path which is symbolized by "Oj", where the small letter "j" indicates for the output layer.

Equation (2.14) performs the error which is symbolized by "Ep".

Ep = (2.14)

So the error for each output unit "j" is based on the difference between the estimated and desired output for that unit.

Where the small letter "p" indicates what the value is for a given pattern, the purpose of training network is going to get the actual output (O) of each neuron at output layer more closer to its Target (T) subsequently the error minimized .

From equation (2.12), the output in the output layer (Oj) is a function for Input of the Output - Layer f (Ij) as described in equation (2.15):

Oj = f (Ij) (2.15)

The first derivative of this function performs backbone in error back propagation, in the output layer the error signal will be calculated by using equation (2.16) and (2.17), where the error signal is denoted by "∆j", and the derivative of this function is denoted by "Ḟ".

∆j = Ḟ (Ij) \* (Tj – Oj) (2.16)

∆j = Oj (1 – Oj) \* (Tj – Oj) (2.17)

This error value is will used to udate weights in output layer (j) and hidden layer (h) therefor the error valueis propagated back through the layers where this process is

repeated many times even this error will be decreased [7, 23].

These equations (2.9, 2.10, 2.11, 2.12, 2.13, 2.14, 2.15, 2.16 and 2.17) that have been used above, which are going to adjust the weights by using these steps:

* Feed the patterns into the layers of the network and make these patterns propagate through input's layers passing hidden's layers the to the output's layer.
* Calculating the error by doing comparison between the estimated output (actual output) and desired output (Target).
* Determine the derivative of the error for each output neuron.
* Using this derivative to update (adjust) the weights of the output layer and the hidden layers [9].

Therefore the important structure of any program that is solved by using back-propagation neural network is shown below in figure 2.13.

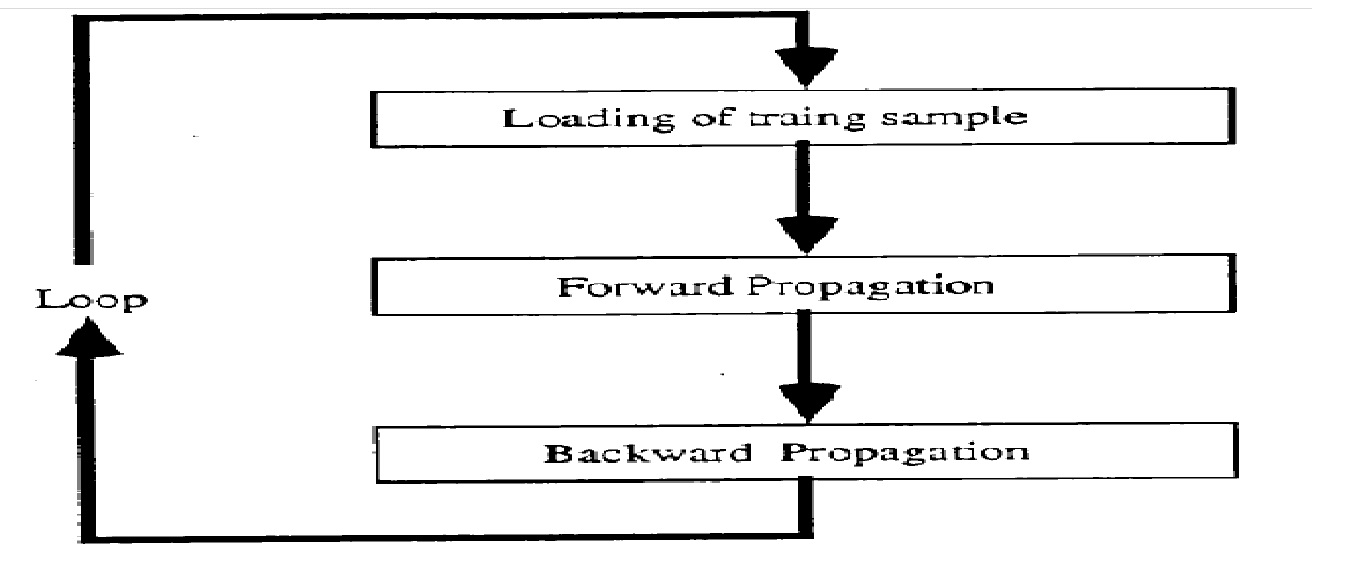


Figure 2.13: Structure of any program by using back-propagation neural network [53].

* + 1. **Learning Rate and Momentum Factor**

These two important parameters are effecting on the learning capability of the neural network.

First is the learning rate coefficient (the learning step size) which is denoted by "η", where the learning step size (η) defines how much the weights should change to decrease the error function (Mean Square Error (MSE)). If the learning step size coefficient (η) is very small then the learning process (convergence) will be very slow, if the learning rate coefficient (η) is too large then the error function is going to increase and instability probably will happen and the global minima will be missing, therefore the learning rate coefficient (η) should be chosen very carefully to accelerate the convergence and keeping the network in stability at the same time.

Second is momentum factor which is denoted by "α", where the momentum factor (α) is a method performed by Rumelhart, Hinton and Williams for improving the training time of the back propagation algorithm by solving a specific problem called "Local Minima" as shown below in figure 2.14

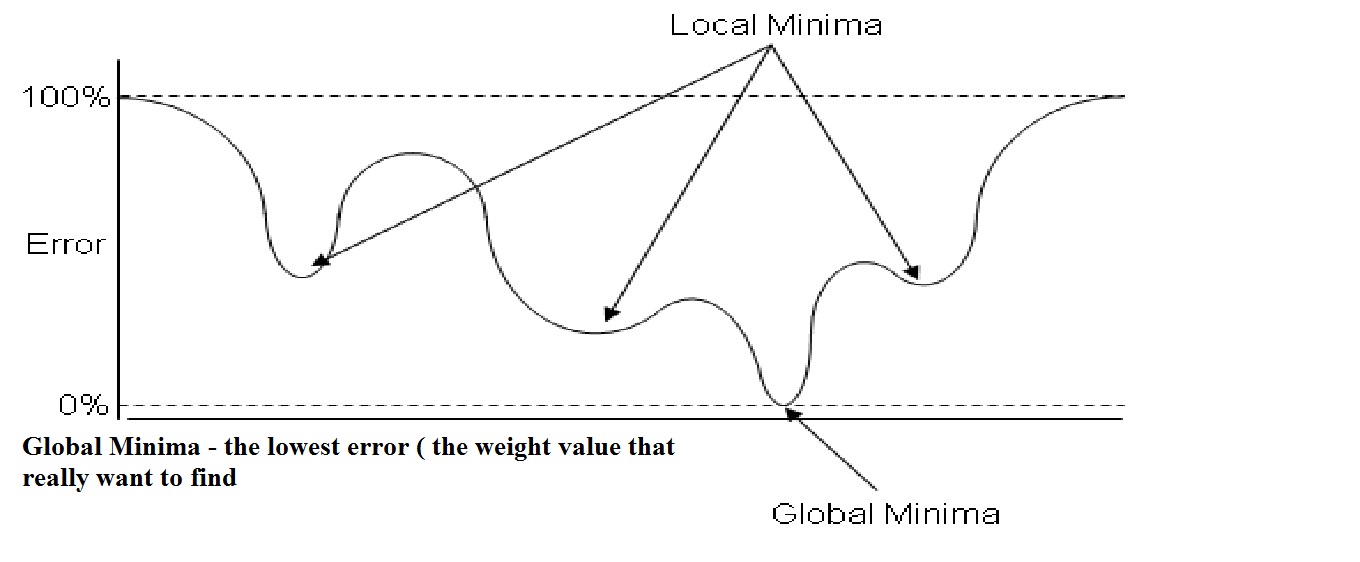


Figure 2.14: Areas of Local Minima and Global Minima [26].

The Local Minima occurs because the algorithm always changes the weights in such a way to cause the error function fall, but the error might briefly have to rise as part of more general fall. If this is the case, the algorithm will get stuck somewhere because it cannot ascend to top of that hill and in this case the error will not decrease.

The momentum factor (α) is a very important coefficient to avoid the falling in hole of the Local Minima, where in this place the error is above zero, with the aim to arrive the global minimum, where in this place the error is approximately zero.

The vocabulary "momentum" is derived from the analogy of a rolling ball with high momentum passing over a tight pit, if the ball rolls slow, the ball is going to drop and confine in the pit. If the ball rolls fast enough, the ball will not trap in the pit.

The values of the momentum factor (α) and the learning rate coefficient (η) usually range between 0 and 1 and in general these two parameters are used for accelerating back propagation process [23, 24, 25, 32, 33, 34, 35].

* + 1. **Training the Inputs data**

The best way for learning and teaching the network is to feed the first pattern and update all the weights in the all layers of the network, next apply the second pattern and change all the weights in the all layers of the network (same procedures in the first pattern), then the third pattern and so on until the last pattern, then return back to the first pattern and repeat that procedures until the error becomes very small.

One of the most popular faults in the starting of learning patterns is to feed the first pattern through layers of the network, run the algorithm after that repeat it until error will be very small, then apply the second pattern and repeat that procedures, then the third pattern and so on until the last pattern.

If this case happened, the network is finished its work with only the last pattern will be learned, that means when the next pattern applied to the network, the network will be forget the previous pattern and so on until reaching to the last pattern.

The total error of the network will be evaluated by adding up all the errors for each individual neuron and then for each pattern as shown below in figure 2.15, in other word the network continues coaching all the inputs data of the network until the total error falls down to the value of the desired objective (Target) and then the algorithm stops.When the network has been coached, in general the network is going to recognize not just the original inputs data, but also the network is going to predict another values from the inputs data, or in other cases, the network is going to recognize not only the original patterns (inputs data), but also the network is going to recognize corrupted and noisy patterns. The network is going to use the adjusted (updated) weights in the learning stage as the weights in the test stage [35]

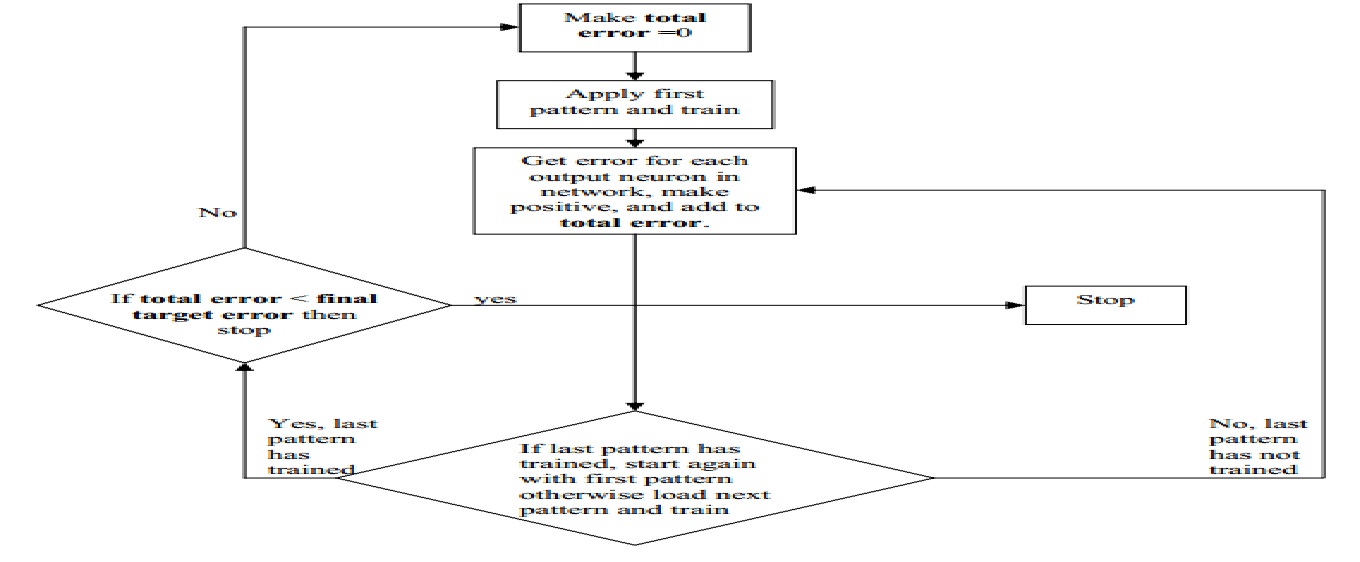


Figure 2.15: Procedure for calculating the total error [35].

* + 1. **Adjusting Weights in the Output Layer**

All the weights in the various layers are initialized to small random number, the process of updating weights begins from the end of the feed forward path, in other word, from the output layer in the feed forward path, and the error function is going to update the weight through backward to other layers of the network.

The weights (Wjh) between the hidden layers (h) and the output layers (j) are updated using equation 2.18, in order to avoid the falling in hole of Local Minima's phenomenon, where in this place the error is above zero, with the aim to arrive the global minimum, where in this place the error is approximately zero, the momentum factor (α) can be added as in equation 2.19.

Wjh (new) = Wjh (old) + η\*∆j\*Oh  (2.18)

Wjh (new) = Wjh (old) + η\*∆j\*Oh + α\*[δ\*Wjh (old)] (2.19)

Where δ\*Wjh stands for the previous weight change. Adjusting the weights for the output layer is easier than other layers because of the target value of each neuron in the output layer is available [24, 25, 32, 50].

* + 1. **Adjusting Weights in the Hidden Layer**

Unlike the output layer neurons, the target vectors are not available for neurons in the hidden layers. Rumelhart and McClelland describes the error term for a Hidden neuron as in equation (2.20) and, subsequently, in equation (2.21).

∆h = Ḟ (Ih) \* (2.20)

∆h = Oh \* (1 – Oh) \* (2.21)

Updating of the weights between the hidden layer and the input layer are calculated using equation (2.22).

Whi (new) = Whi (old) + η\*∆h\*Oi + α\*[δWhi (old)] (2.22)

This way is similar to the way that was used in adjusting weights in the output layer [24 and 25].

* 1. **Learning in Back Propagation Algorithm**

The proposed training algorithm used in the back propagation algorithm is shown below in many steps:

* Initialize the weights of the layers to small random values.
* Select a training vector (input and the corresponding output).
* Propagate the inputs data forward through the network and calculating the actual outputs in the feed forward path.
* Calculating the error from the difference between actual output and target.
* Reduce the error function by updating the weights in the output layer and the hidden layer in the backward path.
* Go to step 2 and repeat for the next pattern until the error is acceptably small or a maximum number of iterations is reached (epoch) [28, 54, 55].

Because of the simplicity in use impressive speed for training and teaching Artificial Neural Network (ANN), and because of their distinctive ability to extract meaning from the complicated data and recognize patterns beside of its massive ability to predict and data filtering, all that made the back propagation learning algorithm a powerful tool and widely used technique in the learning and training of the Artificial Neural Networks (ANNs).

* 1. **Using MATLAB for Implementing Back-Propagation**

The name MATLAB stands for matrix laboratory. It is an interactive system that provides fast matrix calculation. This is very useful feature, since most of the numerical calculations in neural computing are matrix operation. MATLAB’S excellent graphical feature can also be utilized in examining error [24].

* 1. **Summary**

This chapter presented a general overview of Artificial Neural Networks (ANNs) and their application in various aspects of the life. The importance of sigmoid transfer function at back propagation neural network was displayed in detail. Feed forward path and backward path at back propagation algorithm. Adjusting weights in the hidden and the output layers were submitted in detailed form.

This chapter gave a good background to understand usage of the back propagation neural network in assessment security of power flow that will be presented in detail later.