

# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900

Open access books available

186,000

International authors and editors

200M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index  
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?  
Contact [book.department@intechopen.com](mailto:book.department@intechopen.com)

Numbers displayed above are based on latest data collected.  
For more information visit [www.intechopen.com](http://www.intechopen.com)



# Stochastic Artificial Intelligence: Review Article

*T.D. Raheni and P. Thirumoorthi*

## Abstract

Artificial intelligence (AI) is a region of computer techniques that deals with the design of intelligent machines that respond like humans. It has the skill to operate as a machine and simulate various human intelligent algorithms according to the user's choice. It has the ability to solve problems, act like humans, and perceive information. In the current scenario, intelligent techniques minimize human effort especially in industrial fields. Human beings create machines through these intelligent techniques and perform various processes in different fields. Artificial intelligence deals with real-time insights where decisions are made by connecting the data to various resources. To solve real-time problems, powerful machine learning-based techniques such as artificial intelligence, neural networks, fuzzy logic, genetic algorithms, and particle swarm optimization have been used in recent years. This chapter explains artificial neural network-based adaptive linear neuron networks, back-propagation networks, and radial basis networks.

**Keywords:** artificial intelligence, artificial neural network, functions, weights, bias, Adaline network, back-propagation network, radial basis network

## 1. Introduction

In day-to-day life, artificial intelligence (AI) has brought further advantages to pattern features and human expert systems. Based on experience and through learning, it continues to gain further potential in industrial growth. The primary elements for a neural network are the neurons, which are special types of brain cells. The neuron has the ability to retain, realize, and execute the previous existence of every action.

A neural network is an analytical model that is inspired directly by biological neural networks. An artificial neural network (ANN) is an information processing system and is capable of processing nonlinear relationships between inputs and outputs. The network consists of interconnected neurons and functions to produce an output pattern of a given input pattern [1]. The learning process of a neural network takes place by itself, which means the network learns by examples that make it more powerful, so there is no need to devise an algorithm to perform a particular task. Because of the above reasons, a neural network has no internal mechanism to perform a specific task [2].

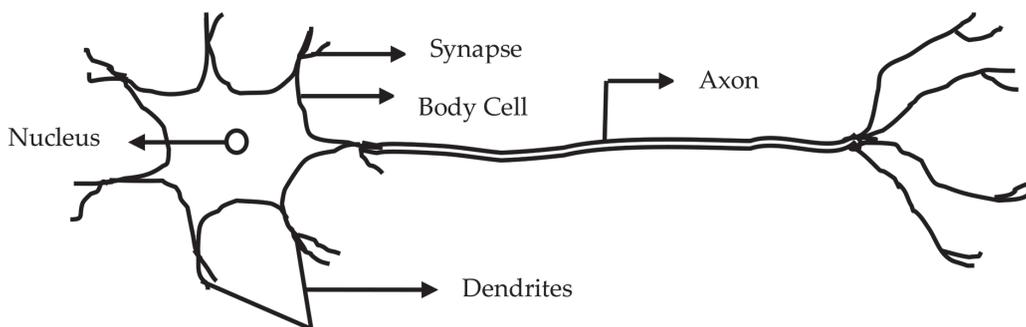
The network consists of nodes that are connected by weights and obtains knowledge through variations in the node weights that are being exposed as samples. Every neuron is linked to other neurons by a network link, and the network

link is associated with the weights that contain instructions in the input signal. In addition, each neuron has a centralized state of its own. The centralized state is the activation level of the neuron that serves as input to the neurons. The activation level of the neuron is imparted to the other neurons. To make the neural process more beneficial, mainframe computers are used. Various computational tasks are developed using ANNs at a more rapid rate than traditional systems.

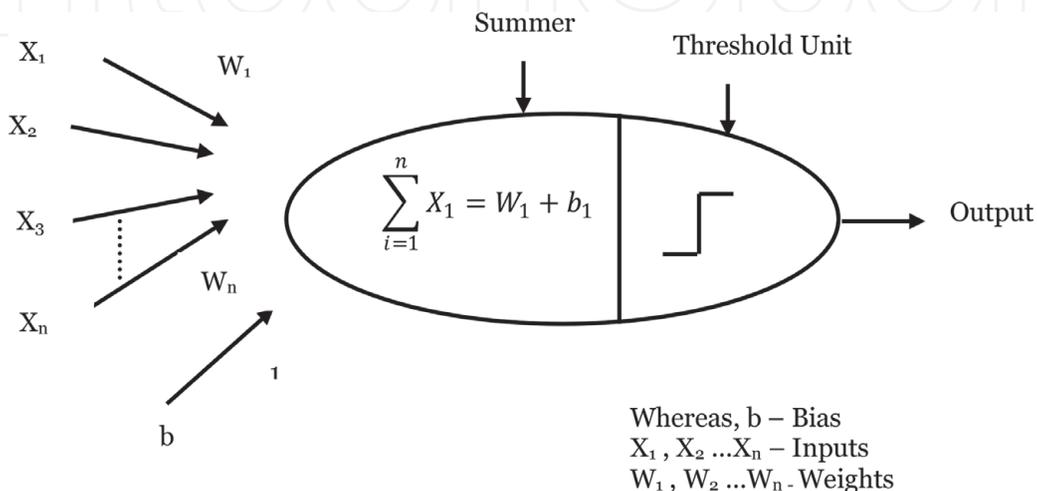
## 2. Biological neural network

Human brains consist of neurons with a number of connections. The basic element of a neural network is called a neuron. The biological neural network consists of axons, dendrites, and a cell body (soma). Each cell performs relatively simple computations, whose nature is indistinct from slow-style networks. Dendrites are tree-like structures (dendrite trees) that accept signals from the neighboring neurons, and each branch is connected to one neuron. The tree-like dendrite structure is connected to the main body of the neuron called the soma (cell body). The cell body is a cylindrical shape that sums the incoming signal. Dendrites are connected by a synapse. A synapse is a structure that allows a nerve cell to pass electric signals to another nerve cell. An axon is a thin cylindrical cell that carries the impulse of the nerve cell. A single nerve cell has 1000 to 10,000 synapses, while 100 billion neurons are present in our brain, and every neuron has 1000 dendrites. The processing of a biological neural network is a slow process and the learning process is uncertain [3].

The simple biological neural network architecture is shown in **Figure 1**.



**Figure 1.**  
*Biological neural network—Simple biological neuron architecture.*



**Figure 2.**  
*ANN model.*

## 2.1 Model and elements of an artificial neural network

The network is observed as weighted directed graphs in which the directed edges of corresponding weights are connected to input and output neurons. The network receives the input in the form of a pattern and image in point (vector) form. The inputs are mathematically assigned by the notation  $x(n)$  for “ $n$ ” number of inputs. An ANN model is shown in **Figure 2**.

Every input is multiplied by its corresponding weights. To solve a problem with the network, weight is used and correspondingly weight is represented as the strength of connections between the neurons. If the weighted sum is zero, then bias is added to make the output not zero. Bias has the weight in which the input is always equal to 1 [4].

## 3. Classification of a neural network

Neural networks are classified on the basis of patterns to determine the weights correspondingly. The neurons are arranged in the form of layers and have the same activation function. Neural network processing depends on the following segments:

- i. Network topology
- ii. Learning methods
- iii. Adjustment of weights and activation functions

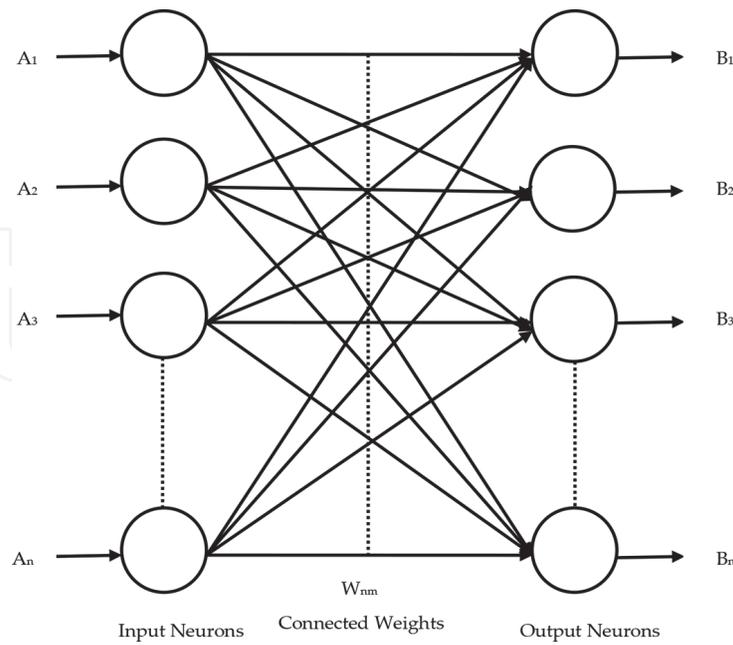
The networks are arranged by connecting the points or with connecting lines. Depending on the topology of the network, it is classified as follows:

- i. Single-layer feed-forward network
- ii. Multilayer feed-forward network

### 3.1 Single-layer feed-forward network

In this network, the signals or the information move only in a forward (one) direction from the input nodes, through the hidden nodes and to the output nodes. A single-layer network does not require cycles or loops. The network consists of output nodes in a single layer, while the inputs are directly fed to the outputs through a series of interconnected weights. Every node is calculated by its sum of the product of the weights. If the value is above threshold (above zero), then the activated value is 1 (positive), and if the value is below threshold (below zero), then the activated value is  $-1$  (negative) [5]. The above network functions such that input nodes are connected to the corresponding hidden nodes with different weights and result in a series of output per node. It consists of multiple neurons that are interconnected to form a single-layer network. The two layers, namely input and output layers, are used. In the input layer, the neurons pass data from one node to the other node and the inputs are scattered and perform no calculation. Each input layer  $a_1, a_2, a_3, \dots, a_n$  is linked to each neuron in the output layer through the connection weight. Each output neuron value such as  $b_1, b_2, b_3, \dots, b_n$  is calculated according to the set of input values. Based on the connection weights the values of the output layer are varied accordingly. This type of network is widely used in

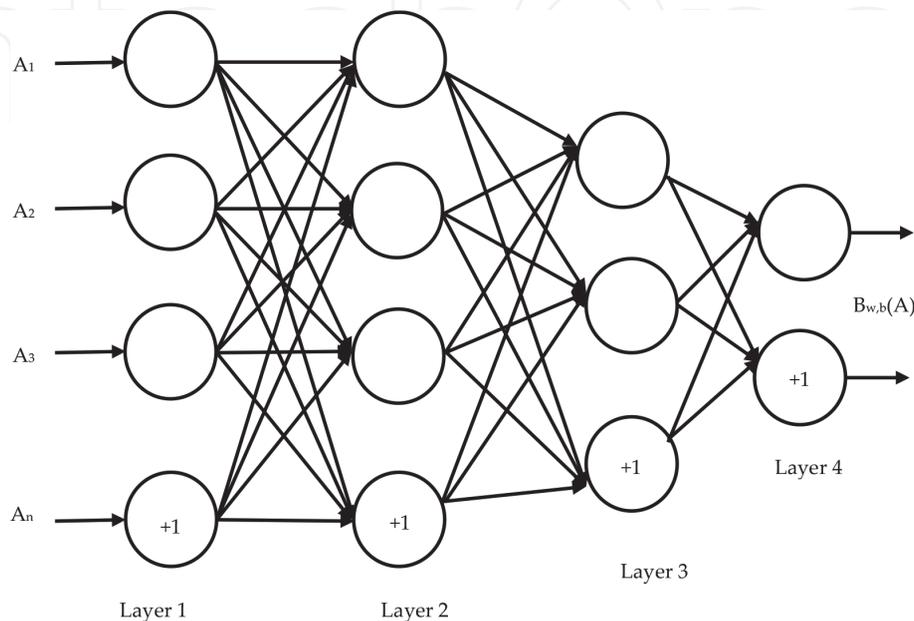
applications like computer vision, speech recognition, and pattern classification problems. A single-layer neural network is shown in **Figure 3**.



**Figure 3.**  
*Single-layer neural network.*

### 3.2 Multilayer neural network

A multilayer neural network is an interconnection of signals in which the inputs and calculations flow forward from the input nodes to the output nodes. The number of signals in a neural network is the number of layers in the network. It consists of more layers for estimated units, and usually the connections are interdependent in the forward path. Every neuron in a single layer is interconnected with neurons of the consequent layer. Multilayer networks use other learning algorithms such as back propagation, Hopfield, Adaline (adaptive linear neuron), etc. In this network, if a layer is connected to the input, then the layer is a hidden layer. The multilayer network is shown in **Figure 4**.



**Figure 4.**  
*Multilayer neural network.*

The network consists of input layers  $A_1, A_2, \dots, A_n$ . To train the network we require training layers, i.e., input layers  $\{(a(1), b(1)) \dots (a(m), b(m))\}$  of  $m$  training sets. To train the network, the gradient descent method is one of the best methods. This method seeks to find the minimum function in the network set.

### 3.3 Learning methods of a neural network

The key aspect of a neural network is the ability to learn by itself. Training or learning methods help the neural network adapt itself by making appropriate adjustments and good responses. To train the network according to custom needs, there are three types of learning. Once a network has been designed for a precise application, then the network is equipped to be trained. To begin these processes, the initial weights are chosen randomly. Then, the training or learning process begins with the following techniques:

- i. Supervised learning
- ii. Unsupervised learning
- iii. Reinforcement learning

#### 3.3.1 Supervised learning

Supervised learning is one of the learning methods in which the data, observations, and measurements are defined with predefined classes. It is similar to how a “teacher” explains content to students. The pairing of each input vector with the target vector determines the desired output. Training pairs mainly deal with the input vector and the corresponding target vector. The training process takes place when the input vector is applied, resulting in an output vector. If the actual response differs from the target response, the network will obtain an error signal. The corresponding error signal is used to calculate the adjustment of weights so that both the actual and target outputs match.

A supervised algorithm in a neural network encompasses classification and regression types for learning processes [6]. In the classification type, outputs are confined to a finite set of data, whereas in the regression type, the output may contain analytical data within the given limits. The best execution process for error minimization is the supervised learning algorithm. Input and output data are used to train the mapping function of the network and are given by the following relation:

$$B = f(a) \quad (1)$$

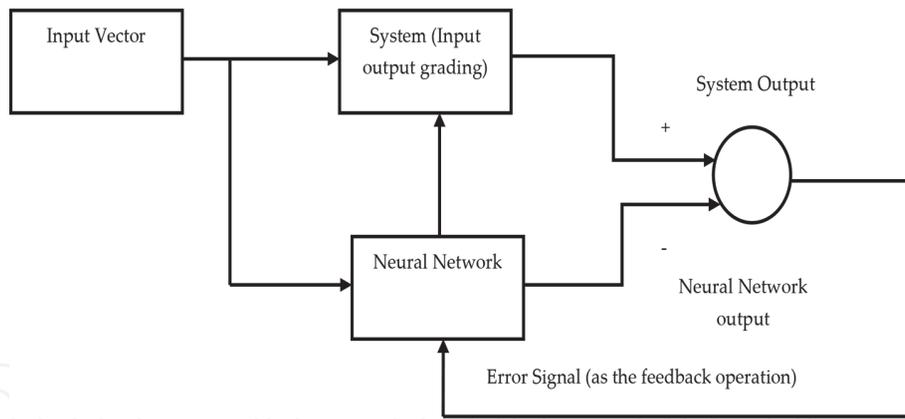
where  $f(a)$  is the function of input  $a$ .

The aim is to provide an approximate mapping function so that new input data ( $a$ ) help to predict the output ( $B$ ). The supervised learning algorithm is shown in **Figure 5**.

The purpose of supervised learning is to vary its weights according to the input/output samples. After executing this network, input-to-output mapping with minimum error has been achieved. Without proper training sets, performance is no longer determined, while it seems stochastic in either case.

#### 3.3.2 Unsupervised learning

Unsupervised learning is the type of machine learning function that describes the hidden layer from the unviabile data. The unviabile data explain the classification and measurements that are not included in the observations. Because of unviabile data, there can be no calculation to reach an accuracy level.

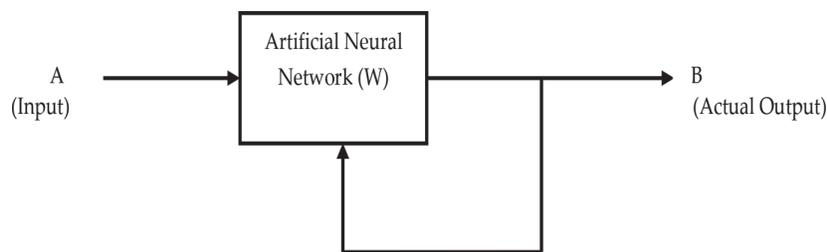


**Figure 5.**  
*Supervised learning.*

Unsupervised learning is utilized in self-organizing neural networks, and this type of learning does not require a teacher to teach the network [7]. To train the network, data sets used in the supervised model are used along with the synaptic weights, which are assigned as:

$$\text{Unsupervised training} = \frac{1}{\sqrt{\text{Number of input attributes}}} \quad (2)$$

It has the ability to solve complex problems and analyze the changes that occur in undefined data. It is widely used for preprocesses in the network of a supervised learning algorithm. A block diagram of unsupervised learning is shown in **Figure 6**.



**Figure 6.**  
*Unsupervised learning.*

Unsupervised learning methods can be further grouped into clustering and association problems.

Clustering is a basic method to analyze the data that occur in the network. It spots some inherent structures present in a set of objects based on a similarity measure. The clustering technique is based on statistical model identification or competitive learning. It is widely used for feature extraction, vector quantization, image segmentation, function approximation, and data mining [8]. The association learning rule is a machine learning method to create relations between variables in large databases, and the approach to unsupervised learning is of two types, namely:

- Anomaly detection
- Neural networks learning—Hebbian learning method

Anomaly detection is used for analyzing events and observations. The system is broadly classified into three types such as unsupervised anomaly detection, supervised anomaly detection, and semisupervised anomaly detection. It preprocesses the data sets and detects the faults that occur in the system.

The Hebbian method is a learning rule that determines the weight of the two different units either to increase or to decrease the weight to activate the function. Learning is performed by varying the synaptic gap between the weights. The weight of the vector increases gradually with respect to the input. The Hebbian rule for updating the weight is given by:

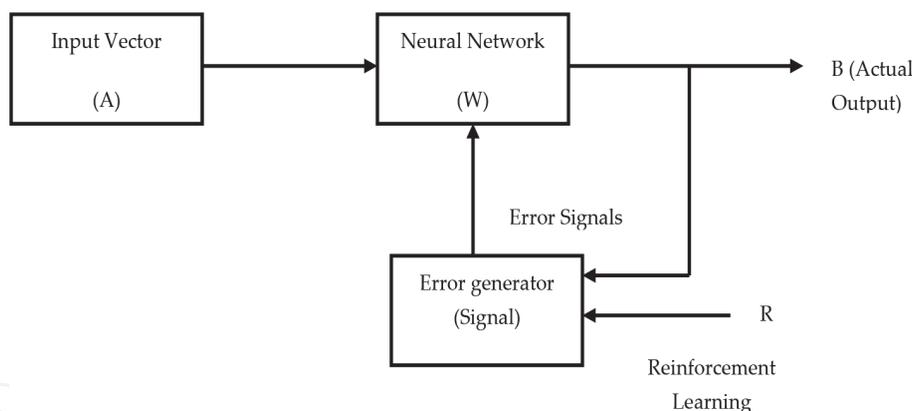
$$w_{a(\text{new})} = w_{a(\text{old})} + C_a * B \quad (3)$$

where  $w_{a(\text{new})}$  is the new weight equal to the sum of the old weight and the learning method  $C_a * B$ .

### 3.3.3 Reinforcement learning

This learning is identical to supervised learning and the difference in operating the network from its actual output for about 50%. In supervised learning, for each output data, the simultaneous input data are known when compared to reinforcement learning. Due to the absence of a training data set, reinforcement learning learns from its experience.

The trial and error process is designed to maximize the expected value of a criterion of functions and actions followed by an improvement, so it is referred to as reinforced learning. The reinforcement signal and the corresponding input patterns depend on the previous data of the stochastic unit. Gaussian processes combine the neural networks for model-based reinforcement learning [9]. A block diagram of reinforcement learning is shown in **Figure 7**.



**Figure 7.**  
 Block diagram of reinforcement learning.

## 3.4 Activation functions of a neural network

### 3.4.1 Weights

$W_1, W_2, \dots, W_n$  are the factors of the weight that are associated with each node to determine the quality of input row vector  $Y = [Y_1, Y_2, \dots, Y_n]T$ . Every single input is multiplied by a related weight by connecting the activation function. **Figure 8** shows the basic elements of a neural network.

### 3.4.2 Threshold

The internal threshold is the offset that marks the activation function of the output node  $Z$  and is given by:

$$Z = \sum_{i=1}^n (X_i W_i) - \theta_k \quad (4)$$

Threshold function may be either binary type or bipolar type, respectively. The output of a binary threshold function is given by:

$$Z = f(p) \quad (5)$$

condition is "0" if  $p < 0$ s

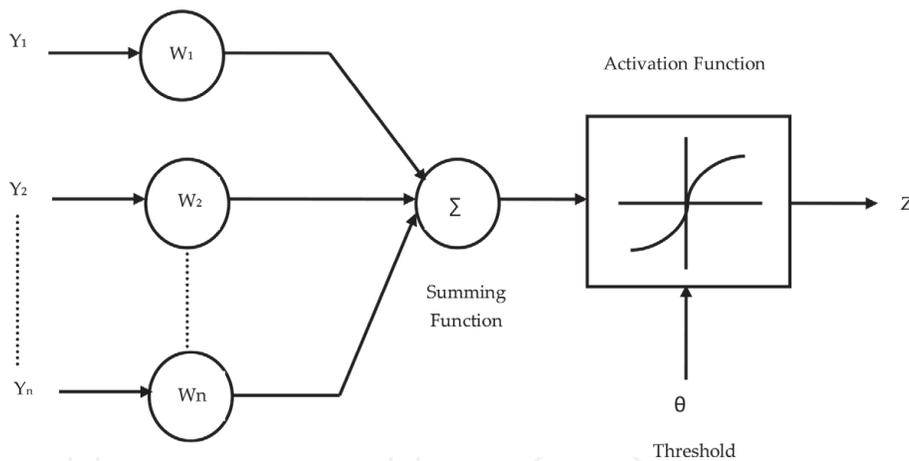
condition is "1" if  $p \geq 0$

### 3.4.3 Linear activation function

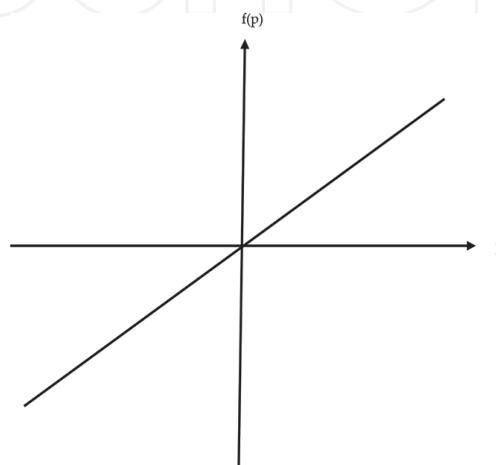
Linear function fulfills the superposition concept. The activation function performs mathematical operations on a signal output, and the equation for linear activation function is given by:

$$Z = f(p) = \alpha.p \quad (6)$$

where  $\alpha$  is the slope of the linear activation function. The linear activation curve is shown in **Figure 9**.



**Figure 8.**  
Basic elements of a neural network.



**Figure 9.**  
Linear activation curve.

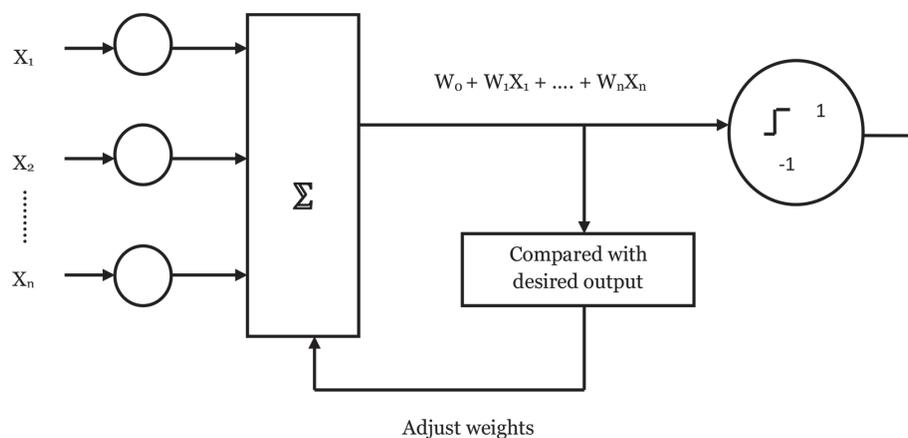
Slope 1 is the identity function. The output  $Z$  of the identity function is equal to the input function ( $p$ ).

#### 4. Methods of implementing a neural network

The methods of implementing a neural network are the adaline method, back-propagation method, and radial basis method.

##### 4.1 Adaptive linear neuron network

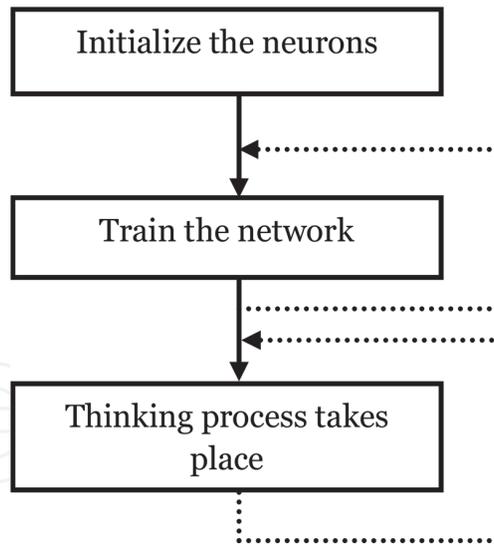
The Adaline model was developed by Widrow Hoff. It is a single-layer network consisting of other nodes. The network is classified by varying the weights in such a manner that it diminishes mean square error for every iteration. An Adaline network is shown in **Figure 10**.



**Figure 10.**  
*Adaline network.*

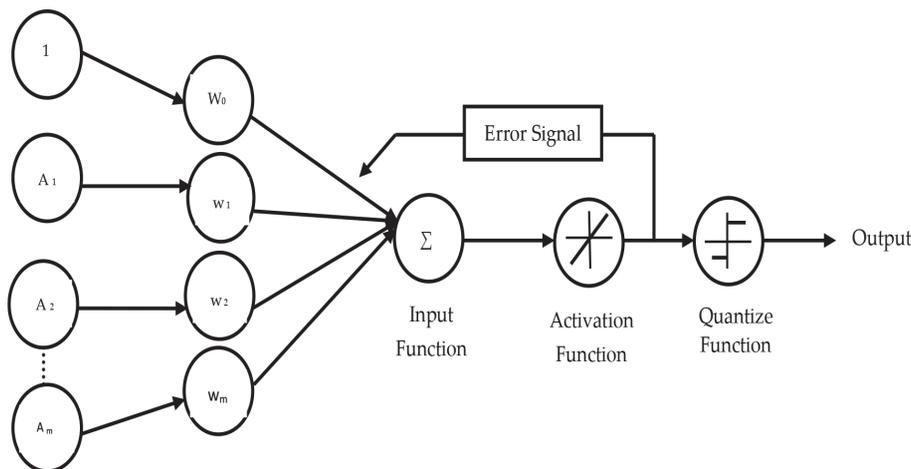
Each node acquires a number of inputs and propagates a single output. The input and output signals to the Adaline network use bipolar activation function. When compared to other types of networks, the input and output function of the Adaline network is linear. The weights are bounded by the input and varied accordingly to the user's choice. The bias in the network acts as an adjustable weight where the activation function is always 1. The output function of the Adaline network has one output unit and the network is a trained delta rule. This rule is otherwise known as the least mean square rule. This type of learning rule is used to decrease the mean squared error between the output and activation function [10]. An Adaline network is implemented by using the three steps, which are shown in **Figure 11**.

- i. **Initialize:** assume random weights to all the links that are connected to the network.
- ii. **Training:** initialize the input weights and arrange the known inputs in a random sequence. Compare errors between input and output by simulating the network. This forms an error function and by adjusting the weights, learning function takes place. Repeat the process until the total error is less than  $\Sigma$ .
- iii. **Thinking:** in the thinking process, the network will respond to input nodes. Even for trained inputs, it does not provide a good result. By defining an error



**Figure 11.**  
Steps—Adaline network.

function, it measures the performance in terms of weights. The derivative of the function with respect to weights is obtained by varying the weights, and error in the system is decreased. A block diagram of an Adaline network is shown in **Figure 12**.



**Figure 12.**  
Block diagram of an Adaline network.

The input to the neural network is represented as  $A$ .

$A = [1 A_1, A_2, \dots, A_m]$ , whereas  $1 = A_0 = \text{bias}$ .

$W = [W_0, W_1, W_2, \dots, W_m]$  represents the weight in the network.

Initially, weights are chosen in a random manner. The value of  $-1$  and  $1$  is taken. The weighted sum of input neurons, including a bias term, is calculated by comparing with output neurons. Based on the delta rule, the weights are adjusted. The output equation for the network is given by:

$$C = \sum_{k=1}^n a_k w_k + \alpha \tag{7}$$

where  $a$  is the input vector,  $w$  is the weight of the vector,  $n$  is the number of inputs,  $\alpha$  is a constant, and  $C$  is the output vector.

By assuming  $a_0 = 1$  and  $w_0 = \alpha$ , the output equation is further minimized to:

$$C = \sum_{k=0}^n a_k w_k \quad (8)$$

#### 4.1.1 Adaline learning algorithm

The learning algorithm of a network is a delta rule but its base is different. To reduce variation between the input and output, the delta rule is preferred and improves the weight between the connections. The main objective is to reduce error that occurs in overall training networks [11, 12]. The updated weight in the network is given by the following equation:

$$w \leftarrow w + \delta(t - C)a \quad (9)$$

where  $\delta$  is the learning rate of the network,  $C$  is the model output, and  $t$  is the desired target output.

The Adaline network merges with the least square error and the equation is given by:

$$E = (t - k)^2 \quad (10)$$

where  $E$  is the least mean square error.

According to the input response, the system is activated and training is started. Consider  $Y$  as an input and  $W$  is the weight. Let us assume  $x_{n+1} = 1$  and  $x_{n+1}$  as the bias weight. Therefore, the weighted sum “ $S$ ” is a dot product of the function given in the equation:

$$S = W.Y = \sum_i w_i y_i \quad (11)$$

The identity function of the network is chosen as  $I = S$  and considered as an activation function. The squared error  $E = (O - I)^2$  is the error function. The network defines error function and determines performance of input, weight, and desired output. Adaline networks are used in net input values and noise correction. The following are the steps for learning the Adaline algorithm:

Step 1: Assume the synaptic weight values in the range from  $-1$  to  $+1$ .

Step 2: Set activation functions of the input units:

$$A_0 = 1 \text{ and } A_i = S(i = 1, 2, 3, \dots, n)$$

Step 3: Compute the net input to the neuron as:

$$S = W.Y = \sum_i w_i y_i$$

Step 4: Update the corresponding bias and weights:

$$W_0 (\text{New}) = W_0 (\text{Old}) + \alpha(t - y_{in}) \quad (12)$$

$$W_1 (\text{New}) = W_i (\text{Old}) + \alpha(t - y_{in})x_i \quad (13)$$

where  $i = 1, 2, 3, \dots, n$ .

Step 5: If the following conditions are satisfied, then the network is stopped or else:

The steps from the initial conditions are repeated.

Adaline network example:

By using an Adaline network, train the AND, NOT gate function with bipolar inputs and targets performs one epoch of training. The input values are given as:

$X_1$	$X_2$	$t$
1	1	-1
1	-1	1

Solution:

The initial weights are taken to be  $W_1 = 0.1$ ,  $W_2 = 0.2$ ,  $b = 0.4$ , learning rate  $\alpha = 0.6$ .

The weights are calculated until the least mean square is obtained.

First input:

$$X_1 = 1; X_2 = 1; t = -1; b = 0.4; W_1 = 0.1; W_2 = 0.2; \alpha = 0.6.$$

$$Y_{in} = b + W_1X_1 + W_2X_2 = 0.7.$$

$(t - Y_{in}) = -1.7$  not equal to zero, then update the weights,

$$W_1 (\text{New}) = W_1 (\text{Old}) + \alpha(t - Y_{in})X_1 = -0.42.$$

$$W_2 (\text{New}) = W_2 (\text{Old}) + \alpha(t - Y_{in})X_2 = -0.82.$$

$$b (\text{New}) = b (\text{Old}) + \alpha(t - Y_{in}) = -0.62.$$

$$\Delta W_1 = \alpha(t - Y_{in})X_1 = -0.102; \Delta W_2 = \alpha(t - Y_{in})X_2 = -0.204;$$

$$\Delta b = \alpha(t - Y_{in}) = -1.02.$$

To compute the error,  $E = (t - Y_{in})^2 = 2.89$ .

Similarly, for the second input:

$$X_1 = 1; X_2 = -1; t = 1; W_1 = -0.41; W_2 = -0.82; b = -0.62.$$

$$Y_{in} = b + W_1X_1 + W_2X_2 = -0.24.$$

$(t - Y_{in}) = 1.24$  not equal to zero.

Update the weights:

$$W_1 (\text{New}) = W_1 (\text{Old}) + \alpha(t - Y_{in})X_1 = 0.324.$$

$$W_2 (\text{New}) = W_2 (\text{Old}) + \alpha(t - Y_{in})X_2 = -1.564.$$

$$b (\text{New}) = b (\text{Old}) + \alpha(t - Y_{in}) = 0.124; E = (t - Y_{in})^2 = 1.5376.$$

Epoch 1: For the first input,  $Y_{in} = b + W_1X_1 + W_2X_2 = -1.116$ .

$(t - Y_{in}) = 0.116$ ; update the weights  $W_1 (\text{New}) = W_1 (\text{Old}) + \alpha(t - Y_{in})X_1 = 0.3936$ .

$W_2 (\text{New}) = W_2 (\text{Old}) + \alpha(t - Y_{in})X_2 = -1.4944$ ;  $b (\text{New}) = b (\text{Old}) + \alpha(t - Y_{in}) = 0.1936$ ;  $E = (t - Y_{in})^2 = 0.01345$ ;  $\Delta W_1 = \alpha(t - Y_{in})X_1 = 0.0696$ .

$$\Delta W_2 = \alpha(t - Y_{in})X_2 = 0.0696; \Delta b = \alpha(t - Y_{in}) = 0.0696.$$

So now the error for two inputs varies from 2.89 to 0.013435.

$$\text{Mean error} = 2.89 + 0.01345 = 3.0245.$$

## 4.2 Back-propagation network

A back-propagation network is a common method of training a neural network. The training method is used for a multilayer neural network. The network consists of processing elements with continuous differentiable activation function. In this network, a gradient descent method is used for minimizing the total squared error of the network. Training the network of a given set of input/output pairs is identified and the network has a procedure for changing the weights to classify given input patterns correctly. This is the network where the error is propagated back to the hidden unit [13]. A back-propagation network is a sensitive approach for

dividing the contribution to each weight. The two differences between updating the rule are as follows:

- i. Activation of the hidden unit/neurons is used instead of activation of the input value or input neuron.
- ii. The rule contains a gradient descent for the activation function to operate.

The back-propagation network is the reformation of the least mean square algorithm and varies the network weights to minimize mean squared error between the actual and desired outputs of the network. The network is trained and exerted using training samples of respective inputs and desired outputs are fetched. The algorithm consists of input and output layers to vary weights and analyze the arrangements of input in an acceptable manner. This algorithm takes a unique set compared to other techniques—during the learning period itself the weights are calculated [14]. The error signal is calculated by taking the difference between the calculated and target output. The result is measured in the output layer.

In the back-propagation network, the testing of data is implemented in the feed-forward path. While executing the network, it has the ability to operate in other hidden layers and is more efficient than operating with one hidden layer. The training process requires further time to train the network but the net result of the network during the training process produces a better result. The network is disintegrated into three categories, namely: (i) computation of the feed-forward network, (ii) back propagation to the output and hidden layer, and (iii) updating of weights. The algorithm will be terminated as the error value approaches a negligible numerical value.

The feed-forward computation network undergoes two processes. The first process receives the values of the hidden layer nodes, and in the second process the value from the hidden layer is used to compute the values of the output layer. Once the hidden layer values are determined, the network produces values from the hidden layer to the output layer. The hidden layer is observed once when the error from the output layer is propagated to the hidden layer. Weights are updated only if all the errors in the network are calculated. Further iterations help the network to train and produce a good training result. Block diagram of back propagation network is shown in **Figure 13**.

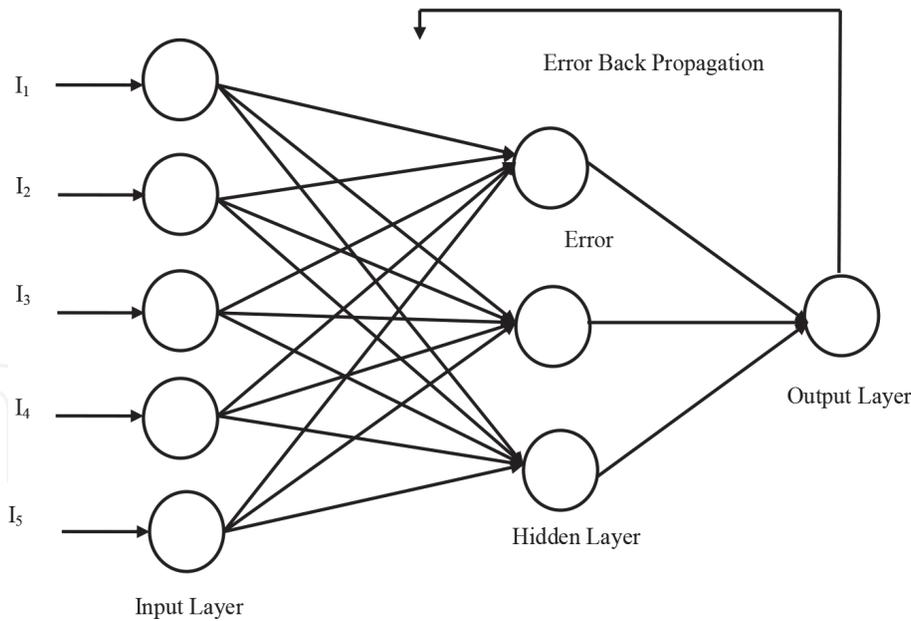
To calculate the derivative function for the squared error with respect to the weights of the network, the gradient descent method is used in the back-propagation network. The squared error function is defined by:

$$E = \frac{1}{2}(t - c)^2 \quad (14)$$

where  $E$  is the squared error,  $t$  is the target output for a given sample, and  $c$  is the actual output of the output neuron.

The constant (1/2) is included, while the differentiating constant is canceled. A limitation of using the back-propagation algorithm is that the input vectors are not normalized and because of that, its performance is not improved. The network identifies only the local minimum values not the global minimum function to determine the errors.

There are two types of back-propagation networks, namely static and recurrent neural networks. The static network produces a mapping of static input for static output. It helps to solve static classification issues like optical character recognition. The recurrent type is a feed-forward network, until a fixed value is obtained.

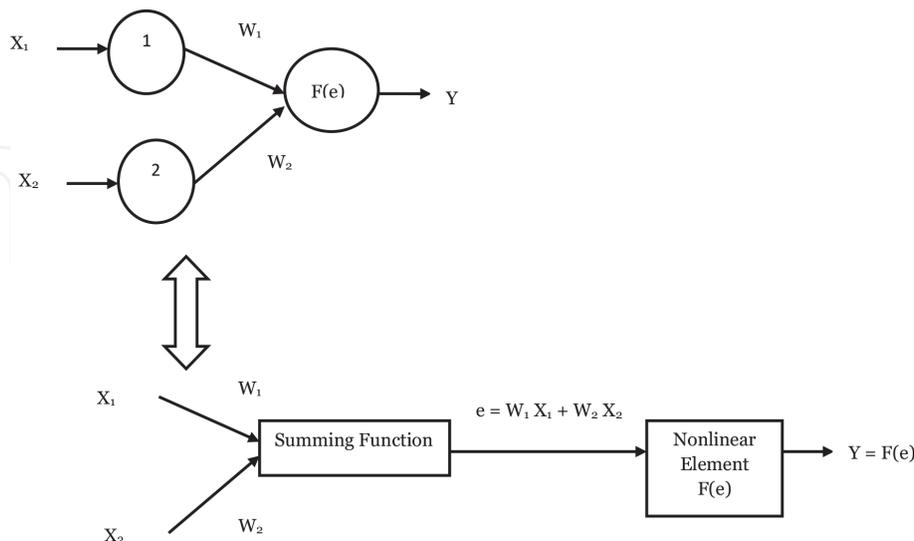


**Figure 13.**  
Block diagram—Back propagation network.

The error is computed and propagated backward. Mapping of the recurrent network is nonstatic.

#### 4.2.1 Learning process of the back-propagation network

Each neuron is composed of two units. The primary unit sums the product of weight coefficients and input signals. The secondary unit realizes the nonlinear function, in which the neurons are transferred to the activation function. Signal  $e$  is the adder output signal, and  $Y = F(e)$  is the output signal of the nonlinear element. Signal  $y$  is also the output signal of the neuron. **Figure 14** shows the learning process of the network.



**Figure 14.**  
Learning process of the back-propagation network.

A training data set is required to instruct the neural network. The training data set consists of input signals ( $X_1$  and  $X_2$ ) assigned to corresponding target output. The training network is an iterative process, and for each iteration, weight coefficients of nodes are changed using new data from the training data set. Each training

step starts by forcing both input signals from the training set. It is possible to determine the output signal values for each neuron in every network layer.

Training steps of the back-propagation algorithm:

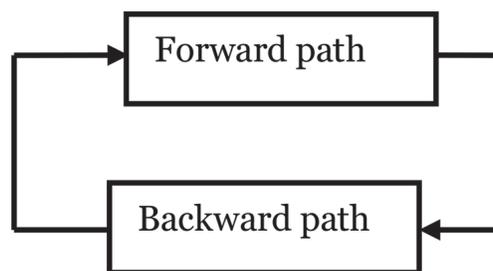
Step 1: The network of random weights is initialized.

Step 2: The training process takes place using the following steps:

- i. Initially, training values are given as input to network and calculate the output of the network.
- ii. The training process (i.e., starting with the output layer, back to the input layer):
  - a. Compares the network output with the correct output (an error function).
  - b. Adapts the weights in the current layer.

Step 3: By using the gradient descent method, the error is minimized.

Step 4: The propagating delta rule is used to adjust the error backward from the output to the hidden layer to the inputs. The back-propagation network is shown in **Figure 15**.



**Figure 15.**  
 Layer of back-propagation network.

Back-propagation neural network example problem:

By using the back-propagation network, train the input vectors for the following functions:  $X_1 = 0.15$ ;  $X_2 = 0.10$ ;  $b_1 = 0.45$ ;  $b_2 = 0.80$ ;  $t_1 = 0.10$ ;  $t_2 = 0.85$ . The example of back propagation network is shown in **Figure 16**.

Solution:

Initialize the weights as  $W_1 = 0.35$ ;  $W_2 = 0.50$ ;  $W_3 = 0.75$ ;  $W_4 = 1.25$ ;  $W_5 = 0.80$ ;  $W_6 = 0.56$ ;  $W_7 = 0.45$ ;  $W_8 = 0.56$ .

Activation function,  $H_1 = \frac{1}{1+e^{-H_1}}$

Forward pass:

$$H_1 = b_1 + W_1X_1 + W_2X_2 = 0.55215$$

$$\text{Out } H_1 = \frac{1}{1+e^{-H_1}} = 0.6346$$

$$H_2 = b_1 + W_3X_1 + W_4X_2 = 0.6875$$

$$\text{Out } H_2 = \frac{1}{1+e^{-H_2}} = 0.66541$$

Now, for calculating  $Y_1$ :

$$Y_1 = \text{Out } H_1 * W_5 + \text{Out } H_2 * W_6 + b_2 = 1.6803096$$

$$\text{Out } Y_1 = \frac{1}{1+e^{-Y_1}} = 0.842945$$

In the same way:

$$Y_2 = \text{Out } H_2 * W_8 + \text{Out } H_1 * W_7 + b_2 = 1.45819$$

$$\text{Out } Y_2 = \frac{1}{1+e^{-y_2}} = 0.811255$$

Calculating total error:

$$E_{\text{Total}} = \sum \frac{1}{2} (\text{Target} - \text{Output})^2 = \frac{1}{2} (T_1 - \text{Out } Y_1)^2 + \frac{1}{2} (T_2 - \text{Out } Y_2)^2 = 0.27665$$

Backward pass:

To update weights, consider  $W_5$

$$\text{Error at } W_5 = \frac{\Delta E_{\text{Total}}}{\Delta W_5} = \frac{\Delta E_{\text{Total}}}{\Delta \text{Out}_{Y_1}} * \frac{\Delta \text{Out}_{Y_1}}{\Delta Y_1} * \frac{\Delta Y_1}{\Delta W_5} = 0.07035$$

$$\text{Updating } W_5, W_5 = W_5 - \eta * \frac{\Delta E_{\text{Total}}}{\Delta W_5}$$

$\eta$  is the learning rate = 0.1;  $W_5 = 0.7929$

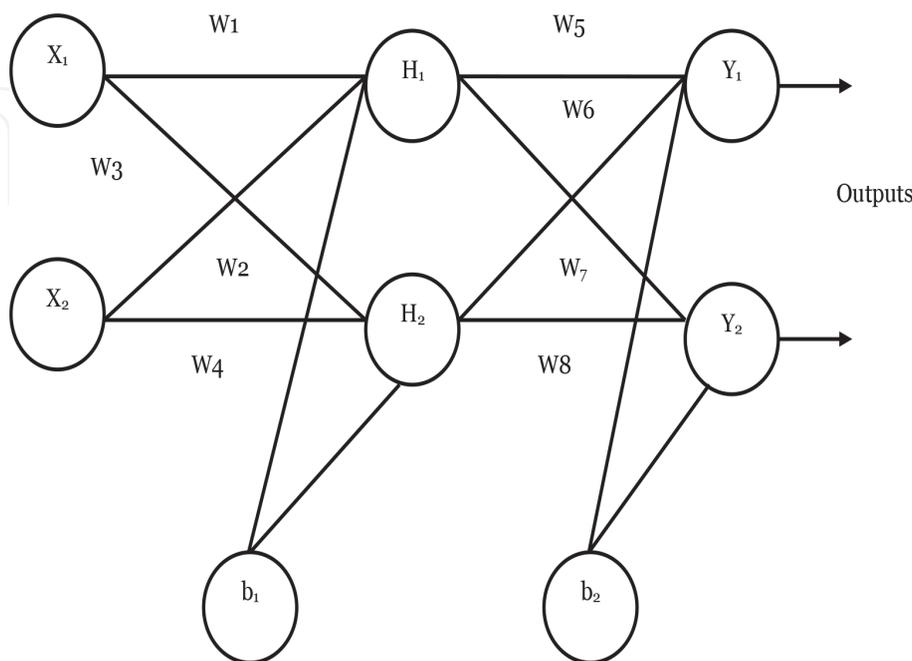
In the same way, calculations are done for updating the weights for  $W_6, W_7,$  and  $W_8$ . In a similar manner, the weights are updated for  $W_1, W_2, W_3,$  and  $W_4$  by using hidden layers in the network.

Advantages of the back-propagation network:

- i. The network is fast, simple, and programming code is easy when compared to other networks. It supports high-speed applications.
- ii. It does not require any parameters to tune except for the number of inputs, and the network does not require prior knowledge to implement.

Disadvantages of the back-propagation network:

- i. The network consumes more time for training and is stuck in local minima resulting in suboptimal solutions.
- ii. A broad amount of input and output data is required, so there exists a complexity when solving a problem. The network is quite sensitive for noisy data.
- iii. A major drawback occurs in a single-layer signal and the network cannot learn the process. It approximates nonlinear separable tasks and functions.



**Figure 16.**  
Back-propagation network.

### Applications of the back-propagation network:

The network is especially useful for machine learning processes, face recognition systems, image or speech recognition, classification, function approximation, time series prediction, etc.

### 4.3 Radial basis function

The radial basis function is a three-layer feed-forward neural network. The transfer function of the hidden layer is the radial basis function. It is derived from function approximation theory. In a neural network, the radial basis function is modeled by the narrow-tuned feedback that is viewed in biological neurons [15]. This type of tuned response is found in several parts of nervous systems. In a feed-forward network, one hidden layer is required for the design of simple structures of lower computational cost. A radial basis network is a nonlinear type that makes the bias function change. The network is used to create regression-type problems.

The radial basis function is composed of three layers, namely input layer, hidden layer, and output layer. The sigmoid type of activation function is not used as in the case of the back-propagation algorithm, whereas the radial basis network uses Gaussian function as an activation function. The input layer consists of neurons with a linear activation function given to the hidden layer. The connection between input and hidden layers is not observed, which means that input neurons received from each hidden neuron remain the same in the network [16]. The Gaussian activation function is determined by:

$$F(d) = \exp(-d^2/\mu^2) \quad (15)$$

where  $\mu$  is the real parameter value and  $d$  is the distance between the input and intermediate vector (the distance is usually measured in terms of Euclidean norm).

Consider the input vector for a period of “ $m$ ” time denoted by:

$$Y(m) = [y_1(m), y_2(m), y_3(m) \dots y_n(m)]^T \quad (16)$$

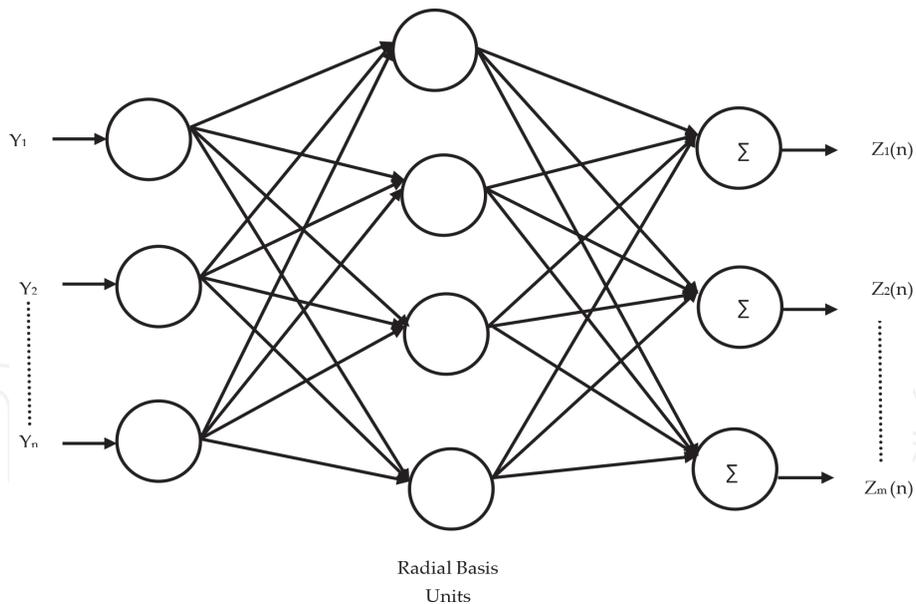
The intermediate vector for each hidden neuron is denoted by  $B_i$  (for  $i = 1, 2, 3, \dots, k$ ), where “ $k$ ” is the number of neurons in the hidden layer. The output of each neuron in the radial basis function is given by:

$$h_i(n) = F_i(\|Y(m) - B_i\|), \text{ for } i = 1, 2, \dots, k \quad (17)$$

Operation of the radial basis network is based on a least mean square algorithm and the local minima values are used for training the neural network. The training process requires a longer computation time but the learning period is less in the network [17]. Schematic representation of the radial basis network is shown in **Figure 17**.

Every neuron in the radial basis function stores a sample vector from the training set. The neuron in the network compares the input vector with its sample vector, and outputs a value between 0 and 1. If the input is equal to the sample vector, then the output of that neuron will be 1. The neuron’s response value is called the activation value.

Every neuron in the radial basis function computes a measure of the similarity between input and sample vector. The values are obtained from the training set. Input vectors are similar to sample vectors and return a value closer to 1. There are different possible choices of similarity functions, but the most popular is based on



**Figure 17.**  
*Radial basis network.*

the Gaussian function. The equation for a Gaussian function with a one-dimensional input is given by:

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (18)$$

where  $x$  is the input,  $\mu$  is the mean, and  $\sigma$  is the standard deviation.

Training steps of the radial basis function:

Step 1: Initialize the input vector  $Y$  from the obtained training set.

Step 2: Determine the output of the hidden layer.

Step 3: Compute the output  $Z$  and compare with the desired value. Adjust each weight  $W$  accordingly:

$$Z = W_{ij}(n + 1) = W_{ij}(n) + \eta(y_j - z_j)y_i \quad (19)$$

Step 4: Repeat the steps from 1 to 3 for each vector in the training set.

Step 5: Repeat the steps from 1 to 4 unless the error is smaller than the maximum acceptable limit.

Applications of the radial basis function:

Applications of the radial basis function are function approximation type, classification, interpolation, and time series prediction. These applications provide various industrial uses like stock price prediction, fraud detection in financial transactions, and anomaly detection of data.

## 5. Conclusion

This chapter encompassed the learning algorithms of neural networks such as adaline, back-propagation, and the radial basis network. Of all the learning methods, the back-propagation network is effective in training because of its mature back-propagating mechanism. The training process of the radial basis function is rapid and almost matches the ability of the back-propagation network. The radial basis function is a good substitute for the back-propagation network. When

the selected features are clear enough, then the back-propagation network produces satisfactory results. The study of neural network has been slow, but now computers have better processing power. The back-propagation network effectively solves the exclusive-OR problem.

IntechOpen

IntechOpen

### **Author details**

T.D. Raheni\* and P. Thirumoorthi  
Kumaraguru College of Technology, Coimbatore, Tamilnadu, India

\*Address all correspondence to: [raheni92@gmail.com](mailto:raheni92@gmail.com)

### **IntechOpen**

---

© 2020 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms Commons Attribution - NonCommercial 4.0 License (<https://creativecommons.org/licenses/by-nc/4.0/>), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited. 

## References

- [1] Sivanandam SN, Paulraj M. Introduction to Artificial Neural Networks. Delhi, India: Vikas Publishing House Pvt. Ltd.; 2011
- [2] Deepa SN, Sivanandam SN. Principles of Soft Computing. 2nd ed. Delhi, India: Wiley India Pvt. Ltd.; 2011. ISBN 10: 8126527412/ISBN 13: 9788126527410
- [3] Zurada JM. Introduction to Artificial Neural Systems. 1st ed. Eagan, USA: Jaico Publishing House; 1992. ISBN 10: 0314933913/ISBN 13: 978-0314933911
- [4] Eluyode OS, Akomolafe DT. Comparative study of biological and artificial neural networks. European Journal of Applied Engineering and Scientific Research. 2013;2(1):36-46. ISSN: 2278-0041
- [5] Fausett L. Fundamentals of Neural Network Architectures, Algorithms and Applications. Florida Institute of Technology. Singapore: Pearson Education; 1994. ISBN: 978-81-317-0053-2
- [6] Sathya R, Abraham A. Comparison of supervised and unsupervised learning algorithms for pattern classification. International Journal of Advanced Research in Artificial Intelligence (IJARAI). 2013;2(2):34-38
- [7] Khanum M, Mahoob T. A survey on unsupervised learning algorithms for automation, classification and maintenance. International Journal of Computer Applications. 2015;119(13): 34-39
- [8] Du K. Clustering: A neural network approach. Neural Networks. 2010;23(1): 89-107
- [9] Nagabandi A, Kahn G. Model-Based Reinforcement Learning with Neural Network Dynamics. California, USA: Berkeley Artificial Intelligence Research; 2017
- [10] Ali Zilouchian and Mo Jamshidi. Intelligent Control Systems Using Soft Computing Methodologies. CRC Press LLC; 2001. ISBN: 0-8493-1875-0
- [11] Widrow B, Lehr MA. Artificial Neural Networks of Perceptron, Madaline and Back Propagation Family. Neurobionics. Germany: Elsevier Science Publishers; 1993:133-205
- [12] Maind SB, Wankar P. Research paper on the basics of artificial neural networks. International Journal on Recent and Innovation Trends in Computing and Communication. 2014; 2(1):96-100. ISSN: 2321-816 9
- [13] Kishore R, Kaur T. Backpropagation algorithm: An artificial neural network approach for pattern recognition. International Journal of Scientific & Engineering Research. 2012;3(6):1-4. ISSN: 2229-5518
- [14] Joshi SC, Cheeran AN. MATLAB based back-propagation neural network for automatic speech recognition. 2014; 3(7):10498-10504. DOI: 10.15662/ijareeie.2014.0307016. ISSN (Print): 2320-3765; ISSN (Online): 2278-8875
- [15] Karkalos NE, Markopoulos AP. Surface roughness prediction during grinding: A comparison of ANN and RBFNN models. WSEAS Transactions on System and Control. 2016;11:384-389. E-ISSN: 2224-2856
- [16] Wong KP. Artificial intelligence and neural network applications in power systems. In: IEEE 2nd International Conference on Advances in Power System Control, Operation and Management, Hongkong. 1993
- [17] Mohamad H. Hassoun. Fundamentals of Artificial Neural Networks. 1st ed. MA, USA: MIT Press Cambridge; 2010. ISBN: 978-81-203-1356-9