Chapter 2

Artificial Neural Network (ANN): An Overview

Computers are becoming powerful day by day. The capabilities of computer can be explored to invent new technologies and ideas that will led to the development of human beings. A human being is a very powerful, capable and intelligent creature of nature. Due to his natural intelligence human can do many complicated tasks with ease. For example, human beings can learn things very easily by day to day experience and apply that in their life. But, they are also bounded by natural constraints. They get tired and forget easily.

Computers neither get tired nor forget anything that is stored in their memory. But, it is very challenging task to incorporate human intelligence in computers. Actually, human brain is a network of millions of well interconnected biological neurons. To form such types of networks of artificial neurons is not a soft nut to crack. Human beings can easily differentiate between patterns and even correlate them. For example, human beings can be trained to identify any English alphabet and later asked to identify by providing a totally different pattern of the same alphabet. In such a situation human beings can easily identify the alphabet. But, it is not easy to write a program in any programming language to do so. The traditional way of programming is not actually a perfect tool to accomplish such tasks. As the traditional computer programs are suitable for pattern matching but not pattern learning and identification.

The development of Artificial Neural Networks (ANNs) started approximately sixty years ago. But, computers were not developed at that time. So, it was very difficult to test and develop such ANNs using the computers. For the next few decades a rapid development in the computer technology both hardware and software overshadowed the development in the ANN technology. Actually, the lack of proper computer technology at the early stage of the development of ANNs had casted doubts on the capability of this technology.

Now-a-days the researchers are getting modern hardware and software technologies to identify the shortcomings of the previously developed technologies. High speed and efficient computers help the researchers to develop more developed ANN models that are comparatively much efficient and better than the previously developed models. Actually, ANN technology is of much interest because ANN models can be developed to help the researchers who are working in many domains like handwriting pattern recognition, speech pattern recognition, natural language processing, electrical circuit designing and in numerous fields where the basis of training and identification is the identification of the patterns. ANNs can be implemented in hardware as well as software. It is expensive to develop ANN hardware. Most of the researchers opt to develop ANN models by software implementation.

2.1Neural Networks

A neural network is a network of interconnected processing units called the neurons. The links between the neurons are the weighted links. The weights on the links either exaggerate or inhibit the processed outputs of the neurons. Actually, neural networks are designed based on the idea of the complicated networks of biological neurons that makes the human brain, which makes human beings such a superb creature, having the power of learning by experience.

2.1.1 Biological Neural Networks

A biological neural network [63] is a network of interconnected neurons present inside the human brain. A biological neuron consists of three components: its dendrite, soma cell and axon. A biological neuron takes inputs from some sense organs or some other neurons with the help of dendrites. Axon of the neuron is the component responsible for producing output. Soma cell does the actual information processing. Soma cells adds the incoming signal and after receiving sufficient amount of input the soma cell fires or does not fire at all and thus produces binary output. Firing of soma cell is considered as binary 1 and non firing of soma cell is considered as binary 0. There is a synaptic gap between the axon of a neuron and dendrite of another neuron. The signals transmitted across the links are actually electrical pulses. The pulses are transmitted

across the synaptic gaps by means of some chemical reactions. The chemical reactions are either exhibitory or inhibitory in nature. There are differential concentrations of ions made up of potassium, sodium and chloride. The 'ions' are present on either side of the neuron's axon sheath which is the white matter of the human brain. The differential concentrations of ions produce an action potential which led to transmission from a particular neuron. A generic biological neuron is illustrated in Figure 2.1.

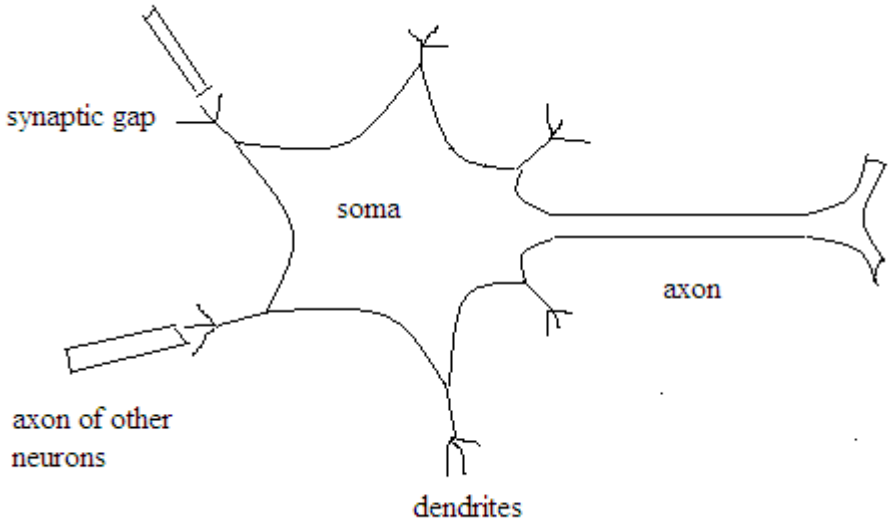

Figure 2.1: A Biological Neuron

A biological neuron consists of a soma cell. In the above figure there are six spots on the soma cell that represents the dendrites. Dendrites at two spots receive inputs from the axons of other neurons. There is one axon in the soma cell which produces the output. In a soma cell inputs can be given at many places but output is produced at only one place. Synaptic gaps can be clearly seen between the axon of other neurons and the dendrites of the soma cell shown in the above figure. Neurons present in the human brain are fault tolerant. Human brain is able to tolerate damage. Human beings are born with billions of neurons. Most of the neurons are present in the human brain which never gets replaced in case of damage. In spite of continuous loss of neurons human beings continue to learn. Actually, human neurons have the capability to take over the work of dying neurons [63].

2.1.2 Artificial Neural Networks

Artificial neural networks (ANNs) are information processing systems that have some features in common with biological neural networks. ANNs are generalizations of mathematical models of human cognition. Information processing in an ANN occurs at many simple elements called neurons. These neurons are interconnected and signals are passed between neurons over the connection links. The connection links are having associated weights. These weights are multiplied by the input signals. These weights exaggerate or degrade the net output produced by the output neuron. Depending upon a threshold value the net output decides that the output neuron will fire or not. Each neuron applies an activation function to the net output to determine the output produced by the output neuron [63].

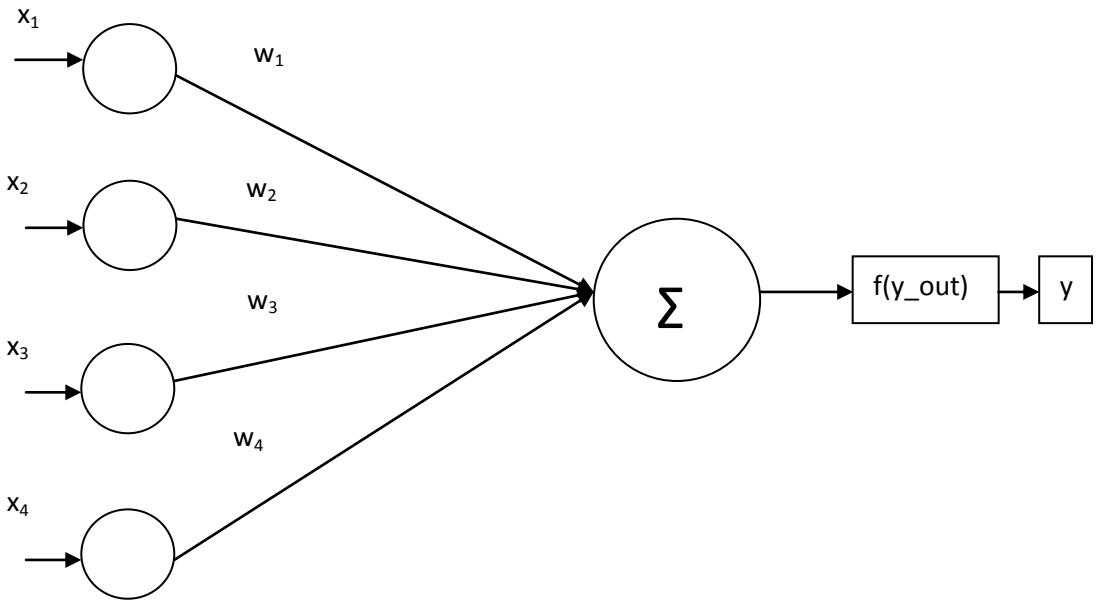

Figure 2.2: An Artificial Neural Network

Figure 2.2 shows an ANN consisting of five neurons. There are two layers of neurons and one layer of weights. An ANN consisting of a layer of weights connecting two layers of neurons is called a single layer ANN. The neurons present in the first neuron layer are the input neurons which are actually not doing any kind of processing but simply accepting the inputs from some outside source and transmitting the inputs to

the output neurons over the connection links. The signals transmitted over the connection links are multiplied by the associated weights on the connection links and summed at the output neuron in the following way to produce the net output as shown in Equation 2.1.

$$y\_out = x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4 \quad \text{.....Equation 2.1}$$

The net output is supplied to the activation function 'f' to produce the final output which is either binary '1' or '0'. Binary '1' indicates the firing of the neuron and binary '0' indicates that neuron didn't fire at all.

$$y = f(y\_out) \quad \text{.....Equation 2.2}$$

'y' is the final output produced by the ANN [63] as shown in Equation 2.2.

2.1.3 Types of Neural Networks

ANNs differ in mainly architectures and way of learning. Based on their architectures and ways of learning ANNs can be categorized into various types like single layer, multi layer, competitive ANNs etc.

2.1.3.1 Single Layer Neural Networks

Single layer ANNs consists of only a single layer of weights sandwiched between two layers of neurons viz. input neuron layer and the output neuron layer. Figure 2.3 shows a single layer ANN consisting of two layers of neurons, three in each layer. A single layer of weights is sandwiched between two layers of neurons. First layer displays the input layer of neurons and the second layer displays the output layer of neurons. All the input neurons are completely interconnected with all the output neurons. So, for the layers of three input and three output neurons that are completely interconnected, the weights required are nine. In Figure 2.3, 'x_1', 'x_2' and 'x_3' represents the input neurons and 'y_1', 'y_2' and 'y_3' represents the output neurons. And, 'w_{11}', 'w_{12}', 'w_{13}', 'w_{21}', 'w_{22}', 'w_{23}', 'w_{31}', 'w_{32}' and 'w_{33}' represent the weights of the links connecting input and output neuron layers [63].

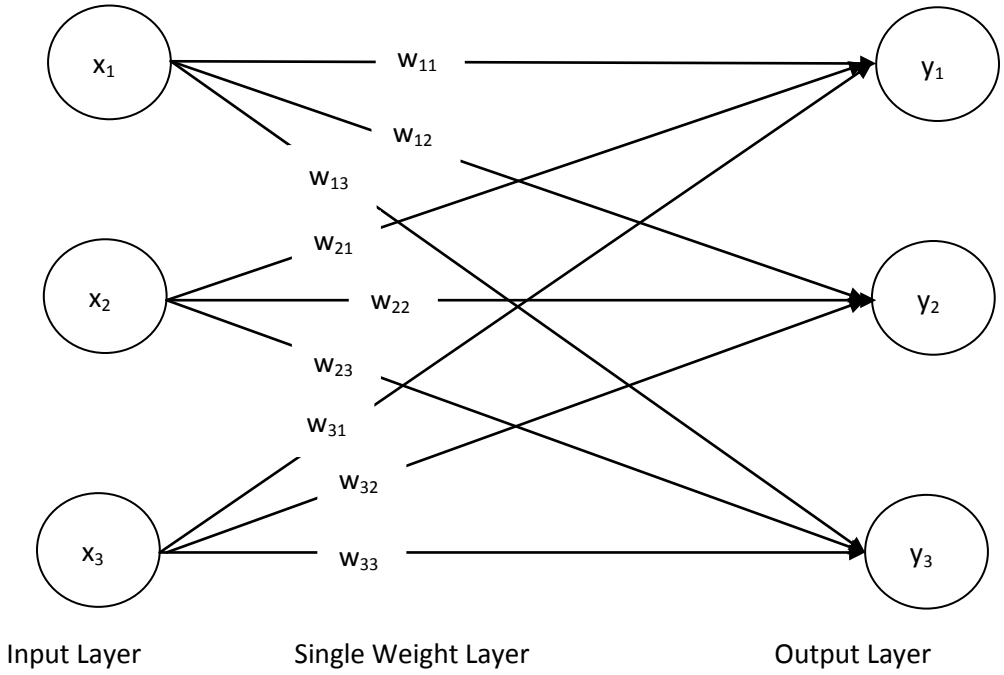

Figure 2.3: A Single Layer Neural Network

2.1.3.2 Multiple Layer Neural Networks

Multiple layer ANNs consist of two or more number of layers of weights and one or more layer of neurons sandwiched between two layers of neurons viz. input neuron layer and the output neuron layer. Figure 2.4 shows a multiple layer ANN which consists of three sets of neurons, three neurons in each set. Two sets of weights are sandwiched between three layers of neurons. First neuron layer displays the input layer of neurons, second neuron layer displays the middle layer of neurons which is also called the hidden layer and third neuron layer displays the output layer of neurons. All the input neurons are completely interconnected with all the neurons of the hidden neuron layer which in turn is interconnected completely with all the neurons of the output neuron layer. So, for the input layer consisting of three neurons, three hidden layer neurons and three output neurons that are completely interconnected, the weights required are eighteen. In Figure 2.4, 'x_1', 'x_2' and 'x_3' represents the input neurons, 'z_1', 'z_2' and 'z_3' represents neurons of the hidden layer and 'y_1', 'y_2' and 'y_3' represents the output neurons. 'u_{11}', 'u_{12}', 'u_{13}', 'u_{21}', 'u_{22}', 'u_{23}', 'u_{31}', 'u_{32}' and 'u_{33}' represent the weights of the links connecting input

and hidden neuron layers and 'v_{11}', 'v_{12}', 'v_{13}', 'v_{21}', 'v_{22}', 'v_{23}', 'v_{31}', 'v_{32}' and 'v_{33}' represent the weights of the links connecting hidden and output neuron layers.

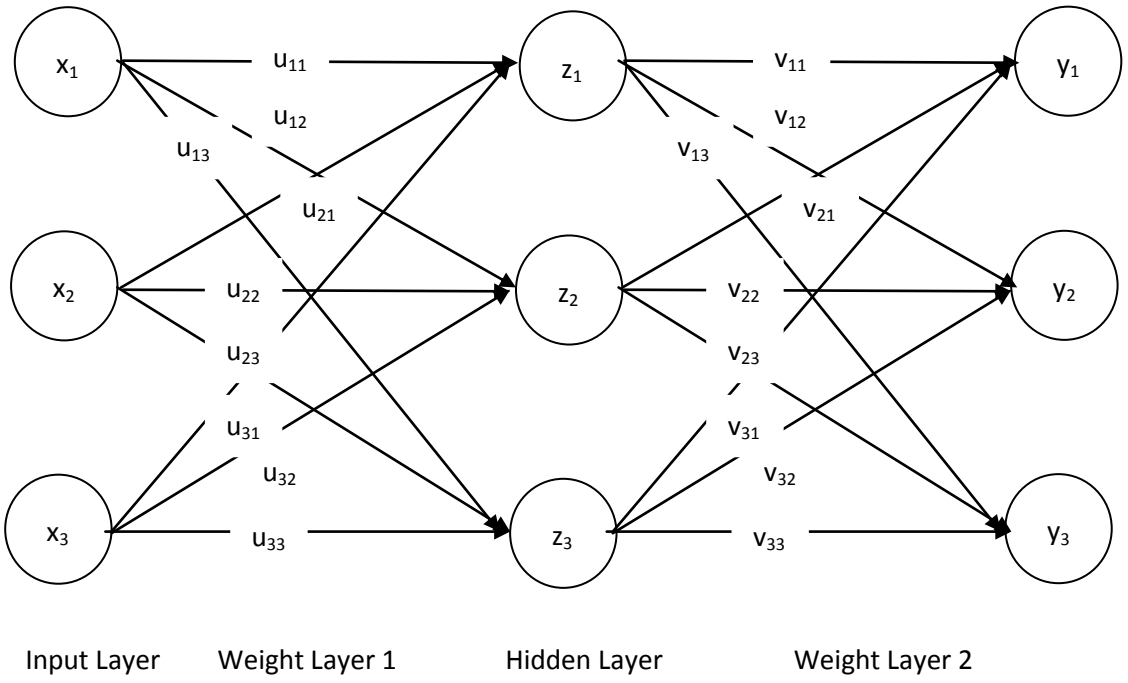

Figure 2.4: A Multiple Layer Neural Network

Number of hidden layers in a multiple layer ANN varies network to network. It actually depends upon the requirements and complexity of the application. A single layer ANN is sufficient to train an ANN to identify the printed characters of same dimension, but in case of handwritten character recognition multiple layer ANNs are more efficient and reliable [63].

2.1.3.3 Competitive Neural Networks

Sometimes more than one neuron in a neural network tries to fire at a time producing ambiguous results. Competitive ANNs are special type of ANNs which solve such problems. Preference is given to that neuron among the competitive neurons that produces the maximum output. This situation is sometimes referred to as winner take all situations.

Figure 2.5 shows a competitive ANN with five neurons. Neurons 'A_1', 'A_2', 'A_3', 'A_4' and 'A_5' are completely interconnected with weights '–ε' [63].

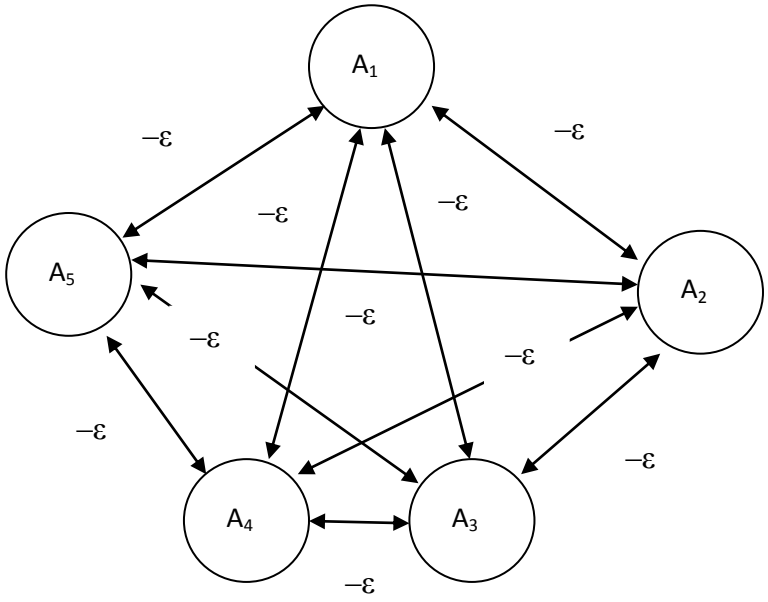

Figure 2.5: A Competitive Neural Network

2.1.4 Selecting the Weights in Neural Networks

Selecting the weights in ANNs is a vital task. Ways of selecting weights of ANN vary network to network depending upon the requirements. Weights can be fixed or dynamic. In case of dynamic weight setting, weights can be initialized by assigning small values selected on random basis and go on changing until standard weights are not determined. Modifying weights is also called training the net. Based upon these properties learning in ANNs can be classified as supervised and unsupervised learning [63].

2.1.4.1 Supervised Learning

Supervised learning means training with the help of a teacher. Actually, teachers know the facts but the students learning under teachers start from the nil. Under the supervision of teachers students learn step-wise. At each step students commit some mistakes that are rectified and acknowledged by the teachers. Finally, the students get

trained and commit fewer or no mistakes at all and at that point of time a student is said to be trained. This training comes under the supervision of someone who already knows the answers and hence is called supervised learning. In case of ANNs, the neural network weights are initialized to small weights that are selected at random. ANNs can be trained for the problems for which the answers are already known by the ANN designer. But, ANNs commit mistakes when inputs are provided to them and produces false results. At that point of time weights are modified by using a suitable formula, which differs for different ANNs. The weights converge towards standard weights and produce correct results [63].

2.1.4.2 Unsupervised Learning

Unsupervised learning means learning without the help of a teacher. These types of ANNs are also called ANNs which use unsupervised type of learning or self-organizing maps. The self-organizing maps group similar input vectors together without the use of training data or sometimes using a training data which is kept fixed. A number of input vectors are presented to the net without setting any weights or using fixed weights. The ANN sometimes modifies the weights in order to assign the most similar input vectors to same output cluster. A vector is produced by the ANN that represents each cluster which is called an exemplar. Actually, unsupervised training is used in those situations where answer is not known in advance [63].

2.1.5 Bias and Threshold

A bias can be included by adding a special neuron, the input of which is always '1'. The weight included on that link is also '1'. If bias is included, the activation function is given by Equation 2.3.

$$f(y\_out) = \{1 \text{ if } y\_out >= 0 \text{ else } -1\} \quad \text{.....Equation 2.3}$$

Where, 'y_out' is the net output given in Equation 2.4.

$$y\_out = b + \sum_i x_i * w_i \quad \text{.....Equation 2.4}$$

And, 'x_i' is the 'i^{th}' input of the 'i^{th}' input neuron and 'w_i' is the related weight.

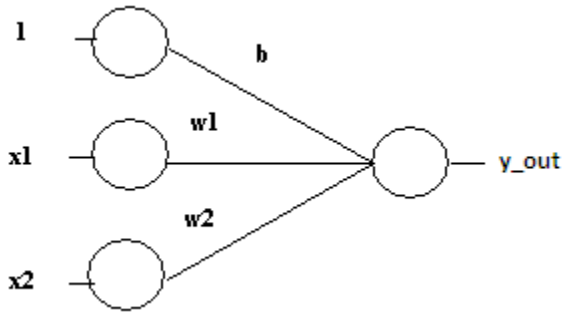

Figure 2.6: A Simple ANN with a Bias

Figure 2.6 represents a simple ANN with a bias '1' where the related weight is 'b'. Bias can be further increased to enhance the net output of the ANN.

Instead of using a bias a fixed value called threshold can also be used. The threshold decides when the neuron will fire. If threshold value θ is included then the activation function is given by Equation 2.5, [63].

$$f(y\_out) = \{ 1 \text{ if } y\_out >= \theta \text{ else } -1\} \quad \text{.....Equation 2.5}$$

Equation 2.6, which is shown below, gives the value of net output.

$$y\_out = \sum_i x_i * w_i \quad \text{.....Equation 2.6}$$

2.1.6 Simple Single Layer Neural Networks

Some of the simple ANNs that were developed earlier are discussed in the sub sections. These ANNs are single layered and always converge to give correct results for the patterns for which these are already trained. These ANNs can be used to develop multiple layer ANNs to solve more complex problems.

2.1.6.1 Hebb

Hebb is the earliest and simplest ANN. It was proposed by Hebb that an ANN can be trained by modifying the synapse weights if the ANN gives unexpected results. The weights are modified by the following formula as shown in Equation 2.7.

$$w_i(new) = w_i(old) + x_i y \quad \text{.....Equation 2.7}$$

Where, 'w_i(new)' is the modified weight, 'w_i(old)' is the existing weight, 'x_i' is the input at the 'i^{th}' neuron and 'y' is the activation produced by the neural net.

The above equation works very well for the bipolar data. But for the binary data, sometimes this equation may not produce the desired results [63].

Algorithm 2.1 (Hebb): Hebb Learning [63]

STEP 1. Initialize the weights

$w_i = 0$ for i= 1 to n

STEP 2. Repeat Step 3 to Step 5 for each training and target vector pair, s: t

STEP 3. Set the activations for the input units

$x_i = s_i$ for i= 1 to n

STEP 4. Set the activation for the output unit

y = t

STEP 5. Modify the weights in case of mismatch of target vector and the output vector

$w_i(new) = w_i(old) + x_i y$ for i= 1 to n

Modify the bias

b(new) = b(old) + y

STEP 6. STOP.

2.1.6.2 Perceptron

Perceptron [63] is a simple and single layer ANN. Perceptron is far better than some other single layer ANNs like Hebb and Sigmoid. In a simple peceptron one or more than one neurons are connected to a single neuron with the help of weights as shown in Figure 2.7. Weights may be initialized to '0' or very small values taken at random. An iterative procedure is used to find out the standard weights. When standard weights are generated, the net generates correct outputs. The output of the 'j^{th}' perceptron is 'y_j'. Output is calculated by applying the activation function 'f', which is a function of the net output, 'y_out' as shown in Equation 2.8.

$y_j = f(y\_out_j)$

Where

$y\_out_j = \sum_i x_i w_{ij}$ .....Equation 2.8

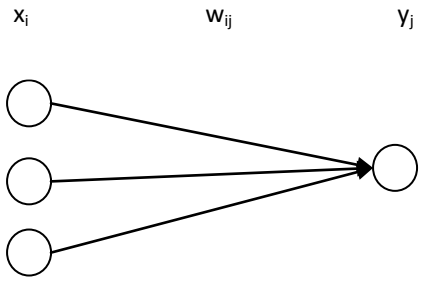

Figure 2.7: A Simple Perceptron

Where, 'x_i' is the 'i^{th}' element of input vector and 'w_{ij}' is the weight between the 'i^{th}' element of the input vector and 'j^{th}' element of the output vector. The function '$f(y\_out_j)$' takes the following values depending upon the values of 'y_out_j' as shown in Equation 2.9.

$$f(y\_out_j) = \begin{cases} 1 \text{ if } y\_out_j > \theta \\ 0 \text{ if } -\theta <= y\_out_j <= \theta \\ -1 \text{ if } y\_out_j < -\theta \end{cases} \quad \text{.....Equation 2.9}$$

Here, 'θ' is the threshold value, taken at random. For each training input, the ANN calculates the response of the output unit. The ANN determines whether an error occurred for this pattern by comparing the calculated output with the target value. If an error occurs for a particular training input pattern, the weights change as given in Equation 2.10.

$$w_{ij}(new) = w_{ij}(old) + \alpha t_j x_i \quad \text{.....Equation 2.10}$$

'α' is the learning rate, the value of which is taken at random, 't_j' is the target value, which is the output expected from the net. The output 'y_j' produced by the net is compared with 't_j' and the difference leads to the modification in weight given by Equation 2.10. The process continues until 'y_j' becomes equal to 't_j'. Weights obtained at that point are the final and standard weights [63].

Algorithm 2.2 (Perceptron): Perceptron Learning [63]

STEP 1. Initialize the weights and bias

[bias may or may not be used as threshold is taken]

$w_i = 0$ for i= 1 to n

b=0

Set learning rate α ($0 < \alpha <= 1$)

(For simplicity α can be set to 1)

STEP 2. While stopping condition is false, do Steps 3-7

STEP 3. For each training pair s:t, do Steps 4-6

STEP 4. Set the activations for the input units

$x_i = s_i$ for i= 1 to n

STEP 5. Compute response of output unit

$y\_out_j = b + \sum_i x_i w_{ij}$

$$y = f(y\_out_j) = \begin{cases} 1 \text{ if } y\_out_j > \theta \\ 0 \text{ if } -\theta <= y\_out_j <= \theta \\ -1 \text{ if } y\_out_j < -\theta \end{cases}$$

STEP 6. Update weights and bias if an error occurred for this pattern

If y<>t

$w_i(new) = w_i(old) + \alpha t x_i$ for i= 1 to n

$b(new) = b(old) + \alpha t$

else

$w_i(new) = w_i(old)$ for i= 1 to n

b(new) = b(old)

STEP 7. Test stopping condition:

If no weights change in Step 3, Stop

Else

Continue

STEP 8. STOP.

2.1.6.3 Adaline

An ADALINE is a single layer ANN which contains a single unit neuron that receives inputs from several input neurons. It also receives an input from a neuron whose signal is always one which is used as bias. Several ADALINEs that receive signals from

the same input units can be combined in a single layer ANN. Multilayer ADALINEs are called MADALINE [63].

Algorithm 2.3 (Adaline): Adaline Learning [63]

STEP 1. Initialize the weights and learning rate α

$w_i = 0$ for i= 1 to n

STEP 2. While stopping condition is false, do Steps 3-7

STEP 3. Repeat Steps 4 to Step 6 for each bipolar training and target vector pair,

s : t

STEP 4. Set the activations for the input units

$x_i = s_i$ for i= 1 to n

STEP 5. Compute response of output unit

$y\_out_j = b + \sum_i x_i w_{ij}$

STEP 6. Modify the weights in case of mismatch of target vector and the output vector

$w_i(new) = w_i(old) + \alpha(t - y\_out_j)x_i$ for i= 1 to n

Modify the bias

$b(new) = b(old) + \alpha(t - y\_out_j)$

STEP 7. Test for stopping condition

If the largest weight change that occurred in Step 3 is smaller than a specified tolerance, then stop otherwise continue

STEP 8. STOP.

2.1.7 Multiple Layer Neural Networks

Multiple layer ANNs are used in order to solve complex problems. Sometimes it becomes necessary for the user, to control the degree of similarity of patterns which are placed on the same cluster. Adaptive Resonance Theory nets are developed to allow the user to control the degree of similarity of patterns placed on the same cluster [63].

2.1.7.1 Architecture of ART1

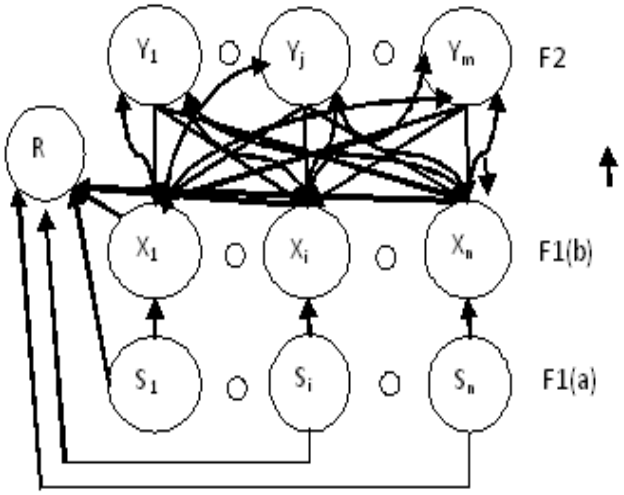

Figure 2.8: Structure of ART1

ART1-net consists of three layers of neurons, 'F1(a)' layer called input layer, 'F1(b)' layer called interface layer and 'F2' layer called cluster layer. 'F1(b)' and 'F2' layers are connected by two sets of weights, bottom up weights 'b_{ji}' from each of the neurons of 'F2' layer to each neuron of the 'F1(b)' layer and top down weights 't_{ij}' from each of the neurons of 'F1(b)' layer to each neuron of the 'F2' layer as shown in Figure 2.8 [4,63]. 'R' is the reset unit which is connected to all the neurons of 'F1(a)' as well as 'F1(b)'. Each neuron of 'F1(a)' layer is connected to the corresponding neuron in 'F1(b)' layer.

2.1.7.2 Training of ART1

The vector pattern is presented to ART1-net. 'S' is the sample pattern which is presented to 'F1(a)' layer and the signals are sent to the corresponding 'X' units of 'F1(b)' layer. 'F1(b)' units are broadcasted to the 'F2' layer with the bottom up weights 'b_{ij}'. 'F2' units compute its net outputs and the units compete for the right to be active. The unit with the largest net output sets its activation to '1', all the other units have activation equal to '0'. The index of the winning unit is set to 'J'. The winning unit becomes the candidate to learn the input pattern. A signal is sent down from 'F2' to 'F1 (b)' layer multiplied by the top down weights 't_{ji}'. The 'X' units of 'F1(b)' layer remain "on" only if they receive non zero signals from both 'F1(a)' and 'F2' units. ||x|| is the norm of the vector 'x' and it gives the number of components in which the top down weight vector for the winning 'F2' unit 't_J' and the input vector 's' are both '1'. It is also called a match. ||s|| is the norm of the vector s which is represented by $\sum_i s_i$. If the ratio of norm 'x' to norm 's' is greater than or equal to the vigilance parameter, the weights (top down and bottom up) for the winning character is adjusted as shown in Equation 2.11 and Equation 2.12.

$$b_{iJ}(new) = Lx_i/(L-1+||x||) \quad \text{.....Equation 2.11}$$

$$t_{Ji}(new) = x_i \quad \text{.....Equation 2.12}$$

'L' is the parameter used to update 'b_{ij}'. If the ratio is less than vigilance parameter 'ρ', the candidate unit is rejected, and another candidate is chosen. The current winning cluster unit becomes inhibited, so it cannot be chosen again as a candidate on this learning trial, and activations of the 'F1' units are reset to '0'. The same input vector again sends its signal to the interface units, which again send this as the bottom up signal to the 'F2' layer and the competition is repeated without the participation of the inhibited units. The process is continued until either a satisfactory match is found which is a candidate is accepted or all the units are inhibited. The user decides the action if all the units are inhibited. The value of the vigilance parameter may be reduced which allow less

similar patterns to be placed on the same cluster, or to increase the number of cluster units or to designate the current input pattern as an outlier that could not be clustered [63].

Algorithm 2.4 (ART1): Adaptive Resonance Theory (ART1) [63]

STEP 1. Initialize parameters

$L > 1$, $0 < \rho <= 1$, [L = parameter used to update bottom up weights, ρ = vigilance parameter]

Initialize weights

$0 < b_{ij}(0) < L/(l - 1 + n)$, $t_{ij}(0) = 1$,

[b_{ij} = Bottom up weights from F_1(b) unit X_i to F_2 unit Y_j , t_{ij} = Top down weights from F_2 unit Y_j to F_1(b) unit X_i , n = number of elements in the input vector]

STEP 2. While stopping condition is false, repeat Steps 3 – 14

STEP 3. For each training input, repeat Steps 4-13

STEP 4. Set activations of all F2 units to zero

Set activations of all F1(a) units to input vector s

STEP 5. Compute the norm of s

$||s|| = \sum_i^n (s_i)$

STEP 6. Send input signal from F_1(a) to the F_1(b) layer

$x_i = s_i$

STEP 7. For each non-inhibited F_2 node

If $y_j <> -1$ then

$y_j = \sum_i^n (b_{ij})\,(x_i)$

STEP 8. While reset is true repeat steps 9-12

STEP 9. Find J such that $y_J >= y_j$ for all j

If y_J = -1, then all nodes are inhibited and this pattern cannot be clustered [J = index of winning unit]

STEP 10. Recompute activation x of $F_1(b)$

$x_i = s_j t_{Ji}$

STEP 11. Compute the norm of vector x

$||x|| = \sum_i^n (x_i)$

STEP 12. Test for reset

If ||x||/||s|| < ρ, then

yJ = -1 (inhibit node J) (and GOTO STEP 8)

If ||x||/||s|| >= ρ, then GOTO STEP 13

STEP 13. Update the weights for node J

b_{iJ} (new) = $Lx_i/(L - 1 + ||x||)$

t_{Ji} (new) = x_i

STEP 14. Test for stopping condition:

STEP 15. STOP.

2.1.8 Applications of ANNs

ANNs can be used in almost all the fields. Research is going on to use ANNs in most of the areas. ANNs have become a topic of multidisciplinary research now a day. Some of the fields where ANNs are used actively are discussed below [63].

2.1.8.1 Signal Processing

Signal processing is an engineering discipline which deals with the implementation of filters to remove or reduce unwanted frequency components from an

information bearing signal [65]. In this field ANNs are used for many applications. One of the first of its kind was that in suppressing noise in telephone lines. The ANN used here is ADALINE. Adaptive echo cancellers are required in transcontinental satellite links for long distance telephone connections. The two-way round trip time delay in order to transmit radio signals is in the order of half a second. The switching technology involved in the old echo suppression is very much disruptive with path delays of the length. The repeater amplifiers of wire based telephone connection also produce echoes. At the end of the long distance telephone line, the incoming signal is applied to both the telephone system component called the hybrid and the adaptive filter which is an ADALINE type of ANN. The difference between the output of the hybrid and the output of the ADALINE is the error. The error is used to correct the weights on the ADALINE. Finally, the ADALINE is trained to remove the echo from the hybrid's output signal [63].

2.1.8.2 Control

ANNs are also used in solving the control problems. An application of this type is truck backer-upper to provide steering directions to a trailer truck attempting to back up to a loading dock. Information is available describing the position of the cab of the truck, position of the rear of the trailer, position of the loading dock which is fixed and the angles that the truck and the trailer make with the loading dock. The ANN learns to steer the truck so that the trailer can reach the dock [63]. There are many practical problems of control where ANNs can be used to improve the performance. One can be thought in automatic parking system and others can be applied in the robotic applications for auto adjustment problems.

2.1.8.3 Pattern Recognition

Pattern recognition can be defined as the classification of data based on knowledge already gained or on statistical information extracted out of the patterns [57]. It is one of the different applications of artificial neural networking. This is applied in the automatic recognition of handwritten characters. The handwritten characters of human beings produce a large number of patterns for a single alphabet. So, it becomes very challenging task to recognize such patterns with ease and accuracy. The large variations

in sizes, positions and styles of writing make it a very difficult task for the traditional techniques. The training is accomplished by presenting a sequence of training vectors, or patterns, each with an associated target output vector. The weights are adjusted using a suitable learning algorithm.

2.1.8.4 Medicine

ANNs can be used in the field of medicines. Expert systems are designed using ANNs which are performing very well as a good advisor to a doctor or can even be implemented in far flung areas where mobile phones and internet can reach but doctors cannot. Actually, different diagnosis patterns are designed for different types of diseases by human experts like experienced doctors and ANNs are trained to learn the patterns. Later these ANNs can be used as expert systems that can be used by the doctors having less experience in the field as well as may be provided to far flung areas with the help of technologies like mobile phones and internet.

2.1.8.5 Speech Production

Reading English text is not an easy task because the appropriate phonetic pronunciation of an alphabet depends upon the context of its appearance. Traditional approach to the problem is constructing a set of rules for the standard pronunciation of various groups of letters with a look-up table for the exceptions. NETtalk is an artificial neural network approach to speech production. NETtalk requires a set of examples of written input with the correct pronunciation of it. The input consists of the letter that is to be correctly spoken and three letters before and after it. Some additional symbols are used to indicate the end of the word or punctuation. The ANN is trained using 1000 common English words. After training, the ANN can read new words with correct pronunciations, with few errors [63].

2.1.8.6 Speech Recognition

ANNs are also used in the field of speech recognition. Systems are there that are very useful in this field but have limitations in vocabulary or grammar. Many multilayer ANNs are designed to recognize speaker independent speech. One such ANN was

developed by Kohonen using the self organizing map. This ANN is called a phonetic typewriter. The output units of self organizing map are arranged in two dimensional arrays. The inputs of the net are short segments of the speech waveform. As the net groups similar inputs, the clusters formed are positioned in such a way that different samples of the same phoneme occur on output units that are close together in the output array. The speech input signals are mapped to the phoneme regions and the output units are connected to the appropriate typewriter key in order to construct the phonetic typewriter [63].

2.1.8.7 Banking and Insurance Sector

ANNs can also be used in some of the financial sectors like banking and insurance companies. The financial companies provides loan to its customers and need the recovery of the loan in time to run the company properly. But market is full of customers who, after taking the loans, fail to repay the EMIs, either intentionally or unintentionally. Similarly, in case of the insurance companies, most of the time company gives the insurance to fraud customers. So, before providing the insurance policies or loans to the customers, it becomes very much important for the companies to analyze the behavior of the customers, ANNs help in doing so. ANNs can be developed that are trained to learn the behavioral patterns of different types of customers and helps the companies to take decisions to approve the loans and policies.

2.1.9 History of ANN

History of ANNs is a combination of different experiments with the biological neuron systems, computer modeling of the biological neuron systems, the development of many mathematical models and the applications in a wide range of domains and also the hardware implementation of the ANNs. The history of ANNs can be divided in different periods.

2.1.9.1 Beginning of the Neural Networks (1940s)

2.1.9.1.1 McCulloch-Pitts

These ANNs are regarded as the one of the ANNs developed at an early stage of ANN development. The researchers considered that if many simple neurons are combined to form a neuron system then the computational power of the network is increased. Weights on these neurons are set so that the neurons perform a particular simple logic function, with different neurons performing different functions. The neurons can be arranged into a network to produce an output that can be represented as a combination of logic functions. The flow of information through the network assumes a unit time step for a signal to travel from one neuron to the next. The idea of threshold is also used in this ANN [63].

2.1.9.1.2 Hebb

Donald Hebb, who was a psychologist at McGill University, designed the first learning rule for ANNs. According to him, if two neurons are active simultaneously, then the strength of the connection between them should be increased. Refinements are made to this general statement to allow computer simulations [63].

2.1.9.2 First Golden Age of Neural Networks (1950s and 1960s)

The period of 1950s and 1960s is considered as the first golden age of ANNs. During that period some of the efficient ANNs were designed such as Perceptron and Adaline. John Von Neumann was very much interested in modeling the brain [63].

2.1.9.2.1 Perceptron

Frank Rosenblatt introduced an ANN called perceptron together with many other researchers. A typical perceptron consists of an input layer connected by links with fixed weights. The weights are adjustable. The perceptron learning rule uses an iterative weight adjustment which is more powerful than Hebb learning rule. Rosenblatt described many types of perceptrons like the one that uses a threshold value to fire. The early success of perceptron led to enthusiastic claims. The mathematical proof of the convergence of

iterative learning under suitable assumptions was followed by the demonstration of the limitations of the perceptrons [63].

2.1.9.2.2 Adaline

Bernard Widrow and his student, Marcian (Ted) Hoff, developed a learning rule which is closely related to perceptron learning rule. The perceptron learning rule adjusts the weight on the links by a unit whenever the response of the net is not correct. The rule that is used in ADALINE to modify the weights in case of an error is called the delta rule. The delta rule adjusts the weights to reduce the difference between the net input to the output unit and the desired output which results in the smallest mean square error [63].

2.1.9.3 Quite Years (1970s)

The period of 1970s is considered as the quite years of ANNs. Minsky and Papert demonstrated for the limitations of perceptrons but research on ANNs were still carried out. During that period some of the ANNs were designed by Kohonen, Anderson, Grossberg and Carpenter [63].

2.1.9.3.1 Kohonen

Teuvo Kohnen of Helsnki University of Technology in 1972 worked with associated memory ANNs [63].

2.1.9.3.2 Anderson

James Anderson of Brown University also started his work on associated memory ANNs. He developed these ideas into his “Brain-State-in-a-Box”, which truncates the linear output of earlier models to prevent the output from becoming too large as the ANN iterates to find a standard solution. These nets were applied in the field of medical diagnosis [63].

2.1.9.3.3 Grossberg

Stephen Grossberg published a number of papers in this field during the period. His work is very mathematical and very biological [63].

2.1.9.3.4 Carpenter

Together with Stephen Grossberg, Gail Carpenter has developed a theory of self organizing neural networks which is also called Adaptive Resonance Theory. Adaptive Resonance Theory is developed in two forms. ART1 is one form which is developed for clustering binary vectors. ART2 is another form which accepts continuous-valued vectors. These nets use unsupervised learning [63].

2.1.9.4 Renewed Enthusiasm (1980s)

The period of 1980s is called the renewed enthusiasm period in the development of ANNs. During that period many advanced concepts of ANNs were developed like Backpropagation, Hopfield nets, Neocognitron and Boltzmann machine.

2.1.9.4.1 Backpropagation

Back propagation method propagates the error back to the hidden layers. A back propagation net consists of three phases. In the first phase input pattern is presented to the net for training. This phase is also called the feed forward phase of the input pattern. In the second phase the error is calculated and propagated back to the hidden layer. In the third and final phase the weights are adjusted [63].

2.1.9.4.2 Hopfield Nets

John Hopfield, of the California Institute of Technology and David Tank, a researcher at AT&T developed many ANNs based on fixed weights and adaptive activations. These ANNs served as associative memory nets and were used to solve many problems like “Travelling Salesman Problem” [63].

2.1.9.4.3 Neocognitron

Kunihiko Fukushima and his colleagues at NHK laboratories in Tokyo have developed a series of specialized ANNs to recognize characters. One of those ANNs is neocognitron. An earlier version of neocognitron is cognitron which failed to recognize distorted characters position-wise or rotation-wise [63].

2.1.9.4.4 Boltzmann Machine

Many researchers developed ANNs which are non-deterministic in nature. In these types of ANNs the weights or activations are changed on the basis of a probability density function [63].

2.1.9.4.5 Hardware Implementation

ANNs are also implemented in hardware. Carver Mead, of California Institute of Technology co invented the software to design microchips. Nobel laureate Leon Cooper of Brown University introduced one of the first multilayer nets, the reduced coulomb energy network [63].

Example 1.1 Training a neural network to solve two input OR-Gate problem

A two input OR-Gate problem can be easily solved with the help of a single layer ANN [63]. A simple OR-Gate consists of two inputs and an output as shown in Figure 2.9.

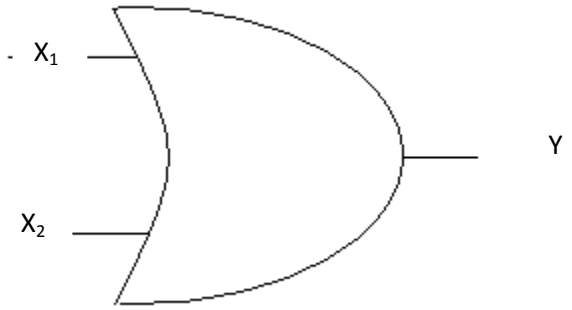

Figure 2.9: Logic Symbol of an OR-Gate

'X_1' and 'X_2' are the two inputs and 'Y' is the single output of an OR-Gate. Figure 2.10 shows the ANN implementation of the OR-Gate.

Table 2.1: Truth Table of the OR-Gate

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

The OR-Gate produces '0' for the first combination of binary inputs and produces one for the other combinations of the binary inputs as shown in Table 2.1. This is known to the users who are familiar with different features of different gates. But, it is not known to the ANN and the layman. So, if they want to learn about the gates with the help of some expert systems which is using ANNs, the ANN has to be trained for the purpose.

Simple perceptron can be used to solve the above problem. To, implement the above problem using ANNs, an ANN is needed with two inputs and a single output. This can be achieved by considering an ANN with one layer of input neurons and one layer of output neuron and a single layer of weights as shown in Figure 2.10.

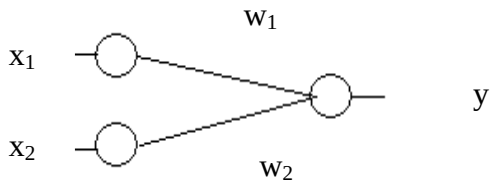

Figure 2.10: ANN Implementation of OR-Gate

'x_1' and 'x_2' are the two inputs of the input layer, 'w_1' and 'w_2' are the weights of the single weight layer and 'y' is the single output of the output layer.

Epoch 1

Table 2.2: Initial Values of the OR-Gate ANN

x1	x2	w1	w2	y_out	y	t
0	0	0	0	0	0	0
0	1	0	0	0	0	1
1	0	0	1	0	0	1
1	1	1	1	2	1	1

Initially, weights are set to zero as shown in Table 2.2 because the standard weights that produce the correct outputs are not known in advance. The, problem takes some epochs to converge to the standard weights.

Threshold value 'Θ' is taken as '0.2' which is selected at random. Learning parameter 'α' is taken as '1'. First combination of the input that is $x_1 = x_2 = 0$ is presented to the net.

The net output of the input vector, 'y_out' can be calculated using Eq. 2.8

$y\_out = x_1*w_1 + x_2*w_2$

$y\_out = 0*0 + 0*0$

y_out = 0, which is less than threshold.

Using, Eq. 2.9, the activation produced by the net 'y' is calculated.

y = 0, which is equal to the target. So, weights are not modified.

Second combination of the input that is $x_1 = 0$ and $x_2 = 1$ is presented to the net.

Again, the following values are calculated using perceptron.

$y\_out = 0*0 + 1*0$

y_out = 0, which is less than threshold.

So, y = 0, which is not equal to the target. So, weights need to be modified in this case.

Eq. 2.10 is used to modify the weights

$w_1(new) = w_1(old) + \alpha t_1 x_1$

$w_1(new) = 0 + 1*1*0$

So, $w_1(new) = 0$

And

$w_2(new) = w_2(old) + \alpha t_2 x_2$

$w_2(new) = 0 + 1*1*1$

So, $w_2(new) = 1$

Third combination of the input that is $x_1 = 1$ and $x_2 = 0$ is presented to the net.

y_out = 1*0 + 0*1

y_out = 0, which is less than threshold.

So, y = 0, which is not equal to the target. So, weights need to be modified in this case.

$w_1(new) = w_1(old) + \alpha t_1 x_1$

$w_1(new) = 0 + 1*1*1$

So, $w_1(new) = 1$

And

$w_2(new) = w_2(old) + \alpha t_2 x_2$

$w_2(new) = 1 + 1*1*0$

So, $w_2(new) = 1$

Fourth combination of the input that is $x_1 = 1$ and $x_2 = 1$ is presented to the net.

y_out = 1*1 + 1*1

y_out = 2, which is greater than threshold.

So, y = 1, which is equal to the target. So, weights need not to be modified in this case.

First epoch could not produce correct results for all the vectors. So, the ANN is trained for another epoch.

Epoch 2

Table 2.3: Values of the OR-Gate ANN for Epoch 2.

x1	x2	w1	w2	y_out	y	t
0	0	1	1	0	0	0
0	1	1	1	1	1	1
1	0	1	1	1	1	1
1	1	1	1	2	1	1

First combination of the input that is $x_1 = x_2 = 0$ is presented to the net as shown in Table 2.3.

y_out = 0*1 + 0*1

y_out = 0, which is less than threshold.

y = 0, which is equal to the target. So, weights are not modified.

Second combination of the input that is $x_1 = 0$ and $x_2 = 1$ is presented to the net.

y_out = 0*1 + 1*1

y_out = 1, which is greater than threshold.

So, y = 1, which is equal to the target. So, weights are not modified in this case.

Third combination of the input that is $x_1 = 1$ and $x_2 = 0$ is presented to the net.

y_out = 1*1 + 0*1

y_out = 1, which is greater than threshold.

So, y = 1, which is equal to the target. So, weights are not modified in this case.

Fourth combination of the input that is $x_1 = 1$ and $x_2 = 1$ is presented to the net.

y_out = 1*1 + 1*1

y_out = 2, which is greater than threshold.

So, y = 1, which is equal to the target. So, weights are not modified in this case.

Second epoch produced correct results for all the vectors. So, the ANN is not required to be trained for another epoch because further epochs produce the same results. The weights obtained in the second epoch becomes the standard weights which can be

used to calculate the output of the OR-GATE, if an input vector of two inputs is presented to the net.

2.2Conclusion

This chapter discussed the role of Artificial Neural Networks (ANNs) to develop intelligence in computers. The need to develop intelligence in machines is also discussed. The idea of developing ANN emerged from the capability of human beings to learn from experiences and human brain plays a great role in this. Human brain is a network of biological neurons interconnected to perform these tasks. Artificial Neurons can be developed using computer programs and ANNs can be designed to perform the same functions. Different types of ANNs with their applications in different fields are also discussed.