UNIT 12 NEURAL NETWORKS AND DEEP LEARNING

Structure

12.1 INTRODUCTION

Jain et al. in 1996 mentioned in their work that a neuron is a unique biological cell that has the capability of information processing. Figure 1 describes a biological neuron's structure, consisting of a cell body and tree-like branches called axons and dendrites. The working of neurons is based on receiving the signals from other neurons through their dendrites, processing the alerts through their body, and finally passing the signals to other neurons via its axon. The synapse is responsible for connecting two neurons through an axon for the first neuron while the dendrite for the second neuron. A synapse can either enhance or reduce the learning capabilities' signal value. If the signals exceed a particular value, called a threshold, then the neuron fires, otherwise not fire.

Biological Neuron	**Artificial Neuron**
Cell Nucleus (Soma)	Node
Dendrites	Input
Synapse	Weights or interconnections
Axon	Output

Table1: Biological Neuron and Artificial Neuron

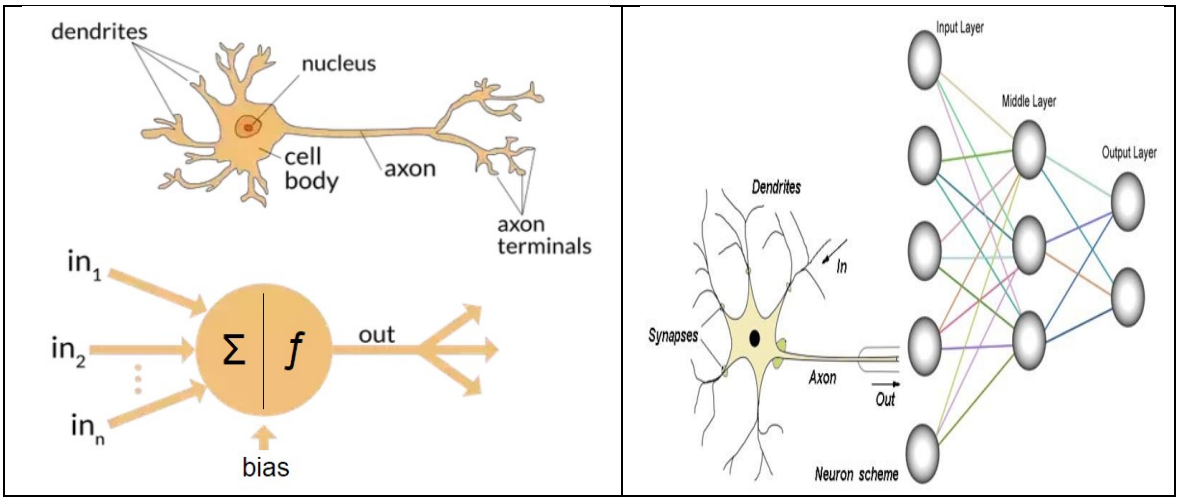


Figure 1: Biological Neuron and Artificial Neuron

12.2 OBJECTIVES

After completing this unit, you will be able to:

- Understand the concept of Neural Networks
- Understand Feed forward Neural networks
- Understand Back propagation Algorithm
- Understand the concept of Deep Learning

12.3 OVERVIEW OF ARTIFICIAL NEURAL NETWORKS

An artificial neural network (ANN) is like a computing system that simulates how the human brain analyzes information and processes it. It is the branch of artificial intelligence (AI) and solves problems that may be difficult or impossible to solve such issues for humans. In addition, ANNs have the potential for self-learning that provide better results if more data becomes available.

Artificial neurons consist of the following things:

- **Interconnecting model of neurons**. The neuron is the elementary component connected with other neurons to form the network.
- **Learning algorithm** to train the network. Various learning algorithms are available in the literature to train the model. Each layer consists of neurons, and

these neurons are connected to other layers. Weight is also assigned to each layer, and these weights are changed at each iteration for training purpose.

An artificial neural network (ANN) consists of an interconnected group of artificial neuronsthat process information through input, hidden and output layers and use a connectionist approach to computation. Neural networks use nonlinearstatisticaldata modeling tools to solve complex problems by finding the complex relationships between inputs and outputs. After getting this relation, we can predict the outcome or classify our problems.

ANN is similar to the biological neural networks as both perform the functions collectively and in parallel. Artificial Neural Network (ANN) is a general term used in various applications, such as weather predictions, pattern recognitions, recommendation systems, and regression problems.

Figure 2 describes three neurons that perform "AND" logical operations. In this case, the output neuron will fire if both input neurons are fired. The output neurons use a threshold value (T), T=3/2 in this case. If none or only one input neuron is fired, then the total input to the output becomes less than 1.5 and firing for output is not possible. Take another scenario where both input neurons are firing, and the total input becomes 1+1=2, which is greater than the threshold value of 1.5, then output neurons will fire. Similarly, we can perform the "OR" logical operation with the help of the same architecture but set the new threshold to 0.5. In this case, the output neurons will be fired if at least one input is fired.

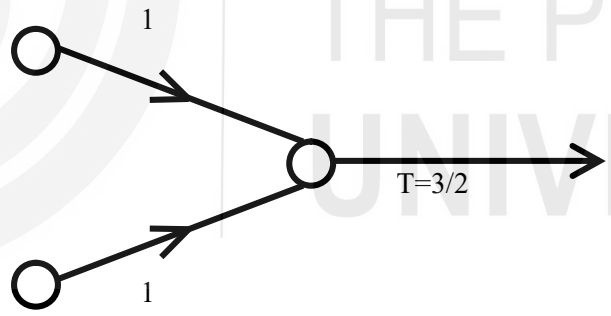


Figure 2: Three neurons diagram

Example-1 : Below is a diagram if a single artificial neuron (unit):

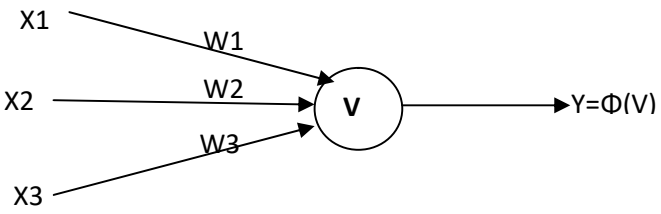


Figure A: Single unit with three inputs.

The node has three inputs x = (x1, x2, x3) that receive only binary signals (either 0 or 1). How many different input patterns this node can receive? What if the node had four inputs? Or Five inputs? Can you give a formula that computes the number of binary input patterns for a given number of inputs?

Answer - 1: For three inputs the number of combinations of 0 and 1 is 8:

x1 : 0 1 0 1 0 1 0 1
x2 : 0 0 1 1 0 0 1 1
x3 : 0 0 0 0 1 1 1 1

and for four inputs the number of combinations is 16:

x1 : 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
x2 : 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
x3 : 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
x4 : 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

You may check that for five inputs the number of combinations will be 32. Note that $8 = 2^3$, $16 = 2^4$ and $32 = 2^5$ (for three, four and five inputs).

Thus, the formula for the number of binary input patterns is: 2^n, where n in the number of inputs.

☞ **Check Your Progress 1**

Question -1: Below is a diagram if a single artificial neuron (unit):

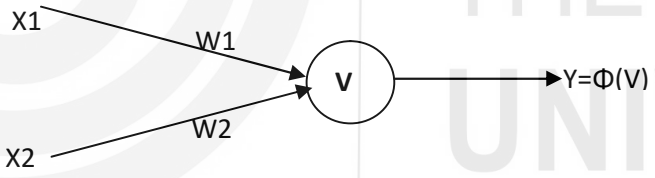


Figure A-1: Single unit with three inputs.

The node has three inputs x = (x1, x2) that receive only binary signals (either 0 or 1). How many different input patterns this node can receive?

……

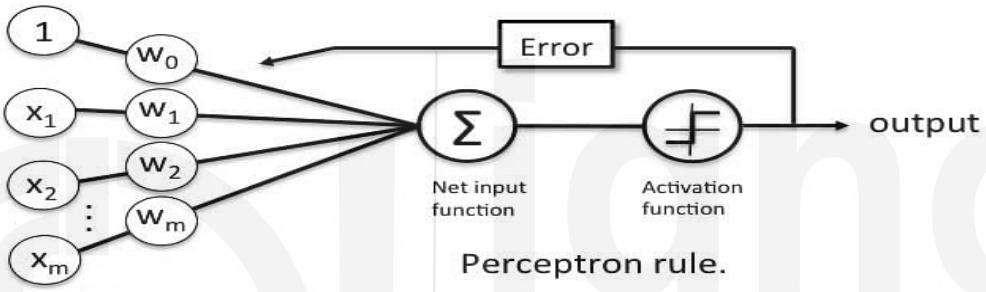
……

12.4 MULTILAYER FEEDFORWARD NEURAL NETWORKS WITH SIGMOID ACTIVATION FUNCTIONS

A multilayer feed forward neural network consists of the interconnection of various layers, named input, hidden layer, and output layer. The number of hidden layers is not fixed. It depends

upon the requirements and complexity of the problem. The simple neural network is one with a single input layer and an output layer is known as perceptrons. A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result. The word perceptron is used here because every connection has a certain weight, and through these connections, one layer is connected to the next layer.

The model's working is defined as follows: All inputs usually are multiplied by the weight, and this weighted sum is calculated. After it, this sum is applied to the activation function, and it is the output of an individual layer. This output becomes the input to the next layer. We have various activation functions, such as sigmoid, tanh, and Relu. After getting the output, the predicted output is compared with the actual output.



12.4.1 Neural Networks with Hidden Layers

Figure 3 describes the hidden layers of a neural network by adding more neurons in between the input and output layers. There may be a single hidden layer or multiple hidden layers.

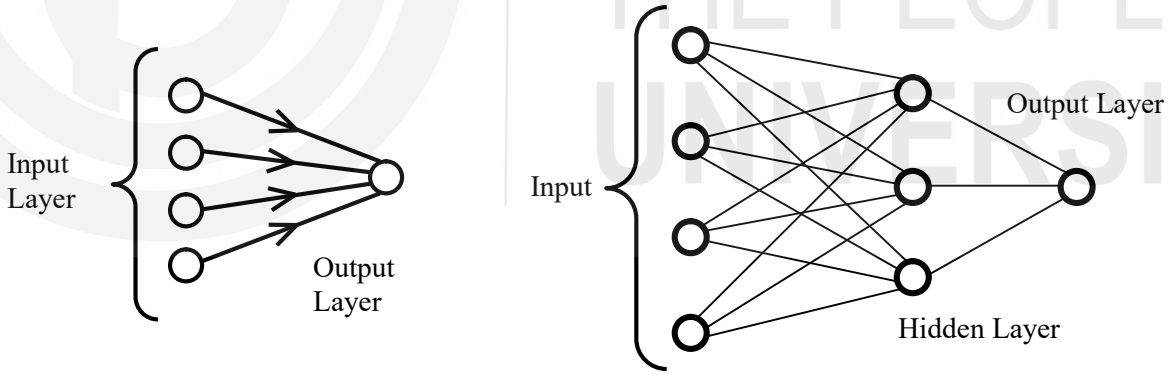


Figure 3: Neural network with a hidden layer

Data/ input is labeled in the input layer using *x value*with 1*, 2, 3, …, m as the subscript*, while neurons in the hidden layer are labeled as *h* with subscripts *1, 2, 3, …, n*... this n and m may be different as the hidden layer neurons, and the input neurons may have different values. Also, as several hidden layers may be multiple, the *first hidden layer has superscript 1, while the second hidden layer has superscript 2, and so on*. Output is labeled as *y* with a *hat i.e.,*$\hat{y}$.

The input data/ features with m dimension represented as ($x_1, x_2, \ldots, x_m$). You may say that a feature is nothing, but it is only a dependent variable that significantly influences a specific outcome/ dependent variable.

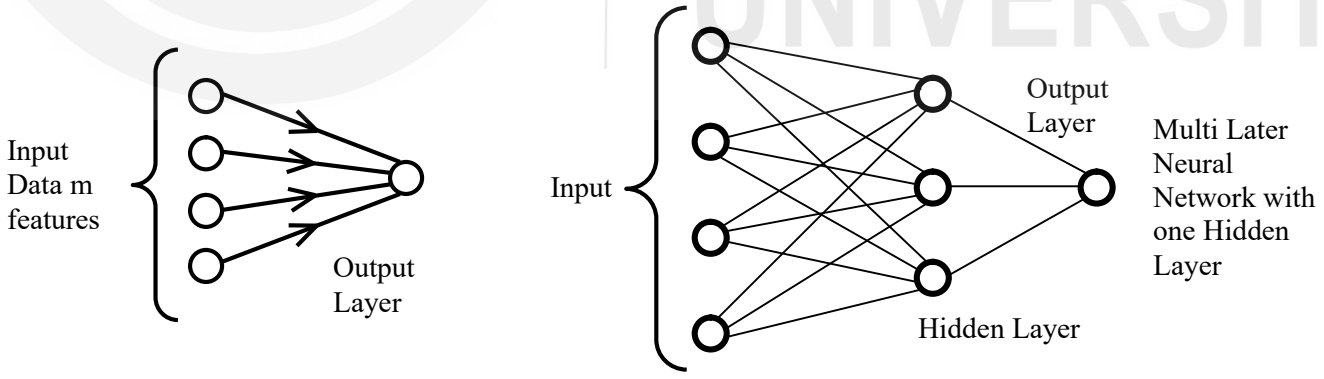
Now, we multiply m features ($x_1, x_2, \ldots, x_m$) with ($w_1, w_2, \ldots, w_m$) as a weight matrix, and then the sum is computed by adding these multiplicative terms. Finally, we define it as a **dot product**:

$$\mathbf{w.x} = w_1 x_1 + w_2 x_2 + \ldots + w_m x_m = \sum_{i=1}^{m} w_i x_i$$

There is the following important observation:

1. For ***m features***, we need precisely ***m*** weights to compute a dot product
2. Further, the exact computation is performed at the hidden layer. With n hidden neurons at the hidden layer, you need n number of consequences ($w_1, w_2, \ldots, w_n$) to find out the dot products
3. With one hidden layer, the output is defined as h: ($h_1, h_2, \ldots, h_n$)
4. Now imagine that you are working on a single-layer perceptron where you may consider hidden output h: ($h_1, h_2, \ldots, h_n$) as input data and perform dot product with weights ($w_1, w_2, \ldots, w_n$) to get your final output y_hat i.e.,$\hat{y}$.

Now, refereeing the above steps, you can understand the working of the multiple layers model and how it works; When you train the model networks on more extensive datasets with many input features, this process will consume a lot of computing resources. Therefore, deep learning was not popular in the early days as limited computing resources were available. However, when better configuration hardware is available, deep learning takes the attention of researchers. The procedure for forwarding the input features to the hidden layer and the hidden layer to the output layer is shown below in Figure 4.



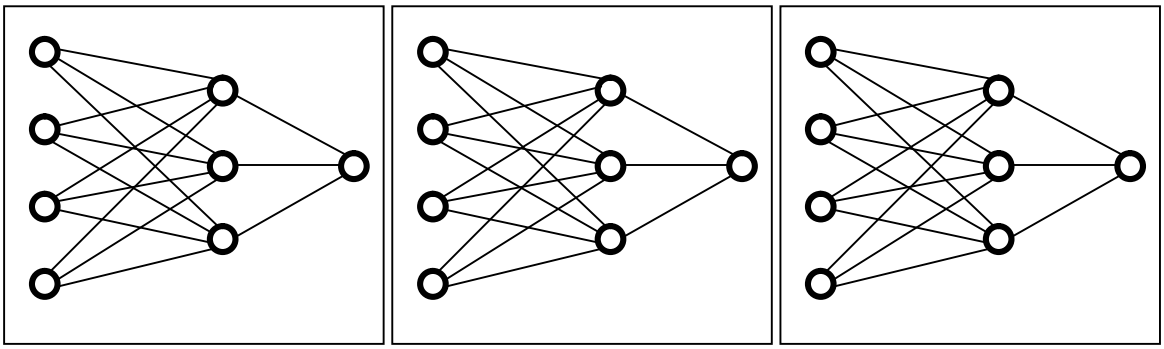


Figure 4: Neural network with different weights

Now you can understand the exact working how multiple layers work.

Example – 2 Consider the unit shown below.

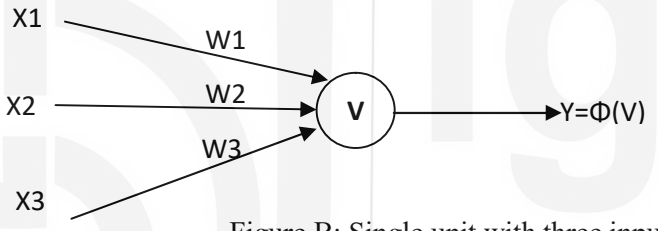


Figure B: Single unit with three inputs.

Suppose that the weights corresponding to the three inputs have the following values:

$w_1 = 2$; $w_2 = -4$; $w_3 = 1$

and the activation of the unit is given by the step-function:

$\Phi(V) = 1$ for $V >= 0$ and $\Phi(V) = 0$ Otherwise

Calculate what will be the output value y of the unit for each of the following input patterns:

Pattern	P_1	P_2	P_3	P_4
X_1	1	0	1	1
X_2	0	1	0	1
X_3	0	1	1	1

Answer:

To find the output value y for each pattern we have to:

a) Calculate the weighted sum: $v = \sum_i wi\ xi = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3$

b) Apply the activation function to v, the calculations for each input pattern are:

$P1 : v = 2 \cdot 1 - 4 \cdot 0 + 1 \cdot 0 = 2 , (2 > 0) , y = \phi(2) = 1$

$P2 : v = 2 \cdot 0 - 4 \cdot 1 + 1 \cdot 1 = -3 , (-3 < 0) , y = \phi(-3) = 0$

$P3 : v = 2 \cdot 1 - 4 \cdot 0 + 1 \cdot 1 = 3 , (3 > 0) , y = \phi(3) = 1$

$P4 : v = 2 \cdot 1 - 4 \cdot 1 + 1 \cdot 1 = -1 , (-1 < 0) , y = \phi(-1) = 0$

Example - 3: Logical operators (i.e. NOT, AND, OR, XOR, etc) are the building blocks of any computational device. Logical functions return only two possible values, true or false, based on the truth or false values of their arguments. For example, operator AND returns true only when all its arguments are true, otherwise (if any of the arguments is false) it returns false. If we denote truth by 1 and false by 0, then logical function AND can be represented by the following table:

x1 :	**0**	**0**	**1**	**1**
x2 :	**0**	**1**	**0**	**1**
x1 AND x2 :	**0**	**0**	**0**	**1**

This function can be implemented by a single-unit with two inputs:

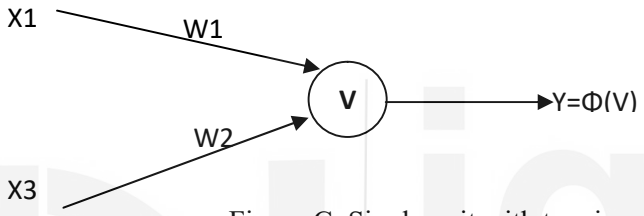


Figure C: Single unit with two inputs.

if the weights are w1 = 1 and w2 = 1 and the activation of the unit is given by the step-function:

$\Phi(V) = 1$ for $V>=2$ and $\Phi(V) = 0$ Otherwise

Note that the threshold level is 2 ($v \geq 2$).

a) Test how the neural AND function works.

Answer (a):

P1 : $v = 1 \cdot 0 + 1 \cdot 0 = 0$, $(0 < 2)$, $y = \phi(0) = 0$

P2 : $v = 1 \cdot 1 + 1 \cdot 0 = 1$, $(1 < 2)$, $y = \phi(1) = 0$

P3 : $v = 1 \cdot 0 + 1 \cdot 1 = 1$, $(1 < 2)$, $y = \phi(1) = 0$

P4 : $v = 1 \cdot 1 + 1 \cdot 1 = 2$, $(2 = 2)$, $y = \phi(2) = 1$

b) Suggest how to change either the weights or the threshold level of this single unitto implement the logical OR function (true when at least one of the arguments is true):

x1 :	**0**	**0**	**1**	**1**
x2 :	**0**	**1**	**0**	**1**
x1 OR x2 :	**0**	**1**	**1**	**1**

Answer(b): One solution is to increase the weights of the unit: w1 = 2 and w2 = 2:

P1 : $v = 2 \cdot 0 + 2 \cdot 0 = 0$, $(0 < 2)$, $y = \phi(0) = 0$

P2 : $v = 2 \cdot 1 + 2 \cdot 0 = 2$, $(2 = 2)$, $y = \phi(2) = 1$

P3 : $v = 2 \cdot 0 + 2 \cdot 1 = 2$, $(2 = 2)$, $y = \phi(2) = 1$

P4 : $v = 2 \cdot 1 + 2 \cdot 1 = 4$, $(4 > 2)$, $y = \phi(4) = 1$

Alternatively, we could reduce the threshold to 1:

$\Phi(V) = 1$ for $V>=1$ and $\Phi(V) = 0$ Otherwise

c) The XOR function (exclusive or) returns true only when one of the arguments is true and another is false. Otherwise, it returns always false. This can be represented by the following table:

x1 :	**0**	**0**	**1**	**1**
x2 :	**0**	**1**	**0**	**1**
x1 XOR x2 :	**0**	**1**	**1**	**0**

Do you think it is possible to implement this function using a single unit? A network of several units?

Answer(c): This is a difficult question, and it puzzled scientists for some time because it is impossible to implement the XOR function neither by a single unit nor by a single-layer feed-forward network (single-layer perceptron). This was known as the XOR problem. The solution was found using a feed-forward network with a hidden layer. The XOR network uses two hidden nodes and one output node.

☞ Check Your Progress 2

Question-2 : Consider the unit shown below.

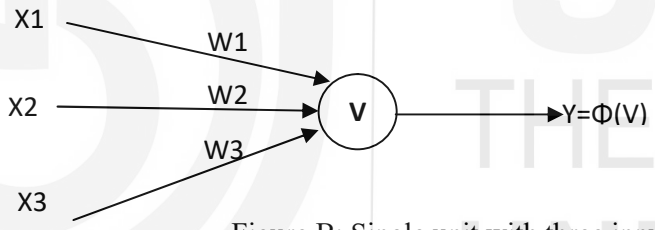


Figure B: Single unit with three inputs.

Suppose that the weights corresponding to the three inputs have the following values:

$w_1 = 1$; $w_2 = -1$; $w_3 = 2$

and the activation of the unit is given by the step-function:

$\Phi(V) = 1$ for $V>=1$ and $\Phi(V) = 0$ Otherwise

Calculate what will be the output value y of the unit for each of the following input patterns:

Pattern	**P_1**	**P_2**	**P_3**	**P_4**
X_1	1	0	1	1
X_2	0	1	0	1
X_3	0	1	1	1

……

……

Question - 3: NAND, NOR) are the universal building blocks of any computational device. Logical functions return only two possible values, true or false, based on the truth or false values of their arguments. For example, operator NAND returns False only when all its arguments are True, otherwise (if

any of the arguments is false) it returns false. If we denote truth by 1 and false by 0, then logical function NAND can be represented by the following table:

x1 :	0	0	1	1
x2 :	0	1	0	1
x1 NAND x2 :	1	1	1	0

This function can be implemented by a single unit with two inputs:

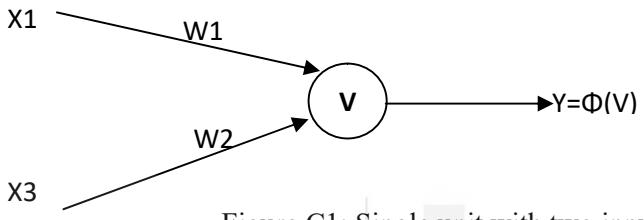


Figure C1: Single unit with two inputs.

if the weights are w1 = 1 and w2 = 1 and the activation of the unit is given by the step-function:

$\Phi(V) = 1$ for $V >= 2$ and $\Phi(V) = 0$ Otherwise

Note that the threshold level is 2 ($v \geq 2$).

a) Test how the neural NAND function works.
b) Suggest how to change either the weights or the threshold level of this single unit in order to implement the logical NOR function (true when at least one of the arguments is true):

x1 :	0	0	1	1
x2 :	0	1	0	1
x1 NOR x2 :	1	0	0	0

……

……

12.5 SIGMOID NEURONS: AN INTRODUCTION

So far, we have paid attention to a neural network model, how it works and what is the role of hidden layers. But now, we are required to emphasize on activation functions and their role in neural networks. The activation function is a mathematical function that decides the threshold value for a neuron, it may be linear or nonlinear.The purpose of an activation function is to add non-linearity to the neural network. If you have a linear activation function, then the number of hidden layers does matter, and the final output remains a linear combination of the input data. However, this linearity cannot help solving complex problems like patterns separated by curves where nonlinear activation is required.

Moreover, the activation function does not have a helpful derivative as its derivative is 0 everywhere. Therefore, it doesn't work for **Backpropagation**, a fundamental and valuable concept in multilayer perceptron.

Now, as we've covered the essential concepts, let's go over the most popular neural networks activation functions.

Binary Step Function: Binary step function depends on a threshold value that decides whether a neuron should be activated or not. The input fed to the activation function is compared to a certain threshold; if the input is greater than it, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.

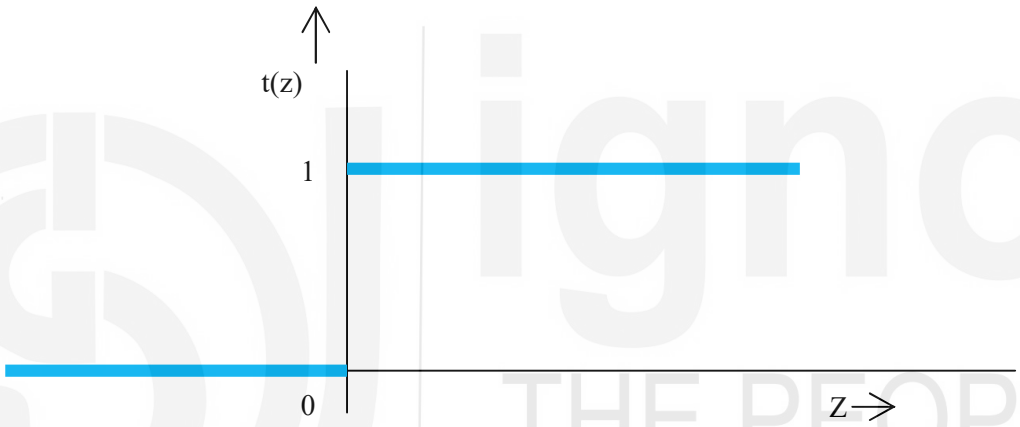


Figure 6: Sigmoidal Function

Mathematically it can be represented as:

$$F(x) = 0 \text{ for all } x<0$$
$$F(x) = 1 \text{ for all } x >=0$$

Here are some of the limitations of binary step function:

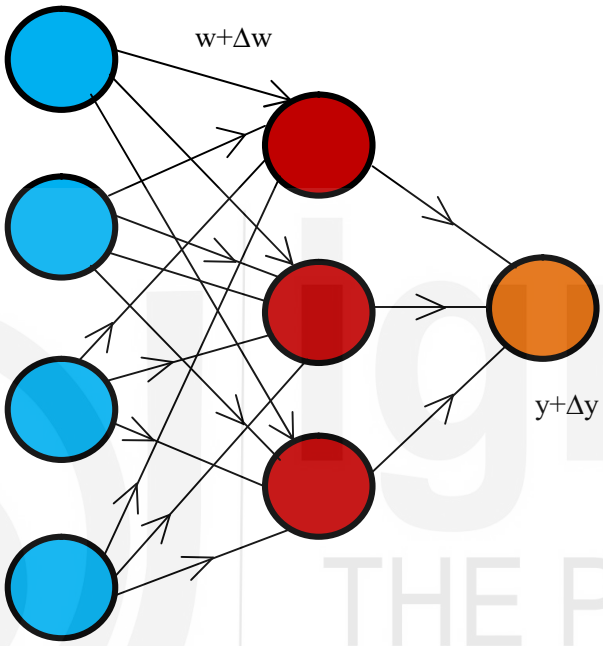
- It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.
- The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

The idea of step function/Activation will be clear from this paragraph. For example, we have a perceptron with an activation function that isn't very "stable" as a relationship candidate.

For example, say some person has bipolar issues. One day ($z < 0$), s/he behaves with no responses as s/he is quiet, and on the second day ($z \geq 0$), s/he changes the mood and becomes very talkative, and speaks non-stop in front of you. There is s no transition for the spirit, and you

don't know the behavior when s/he will be quiet or talking. In such cases, we have a nonlinear step function that helps.

So, minor changes in the weight of the input layer of our model may activate the neuron by flipping from 0 to 1, which impacts the working of the hidden layer's working, and then the outcome may affect. Therefore, we want a model that enhances our exiting neural network by adjusting the weights. However, it is not possible by a linear activation function. If we don't have such activation functions, this task cannot be accomplished by simply changing the weights.



So, we need to say goodbye to the perceptron model with this linear activation function.

We are finding a new activation function that accomplishes our task for our neural network through the sigmoid function. We are changing only one thing: the activation function, and it meets our recruitments, which are sudden changes in the mood. Now, we define the learning Function by

$$Z = \sum_{i-1}^{m} w_i x_i + bias$$

$$\text{Sigmoidal function is: } \sigma(z) = \frac{1}{1 + e^{-z}}$$

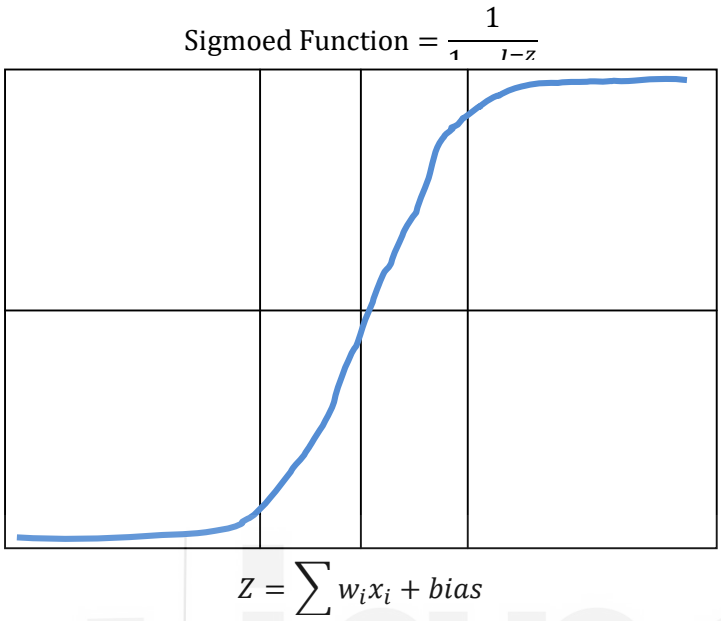


Figure 7: Sigmoidal Function

$\sigma(z)$ The function is called the sigmoid function. First, the value, Z, is computed then the sigmoid function is applied to$Z$. However, it looks very abstract or strange to you how it works. Those who don't have good knowledge of mathematics need not worry. **Figure 7** explains its curve and its derivative. Here are some observations mentioned:

1. The output of the Sigmoid Function produces the same results as produced by the linear step function;the output remains between 0 and 1. The curve marks 0.5 at *z=0*, for which we can make a straightforward rule that if the sigmoid neuron's output becomes more than or equal to 0.5, then its output one; otherwise, output o given for smaller values.
2. The sigmoid function should be continuous. It means that partial derivative, that is, *σ(z) / (1-σ(z)),* which is differentiable everywhere on the curve.

3. If z is a significant negative value, then the output is approximately 0; if z is a significant positive value, the output is given by around 1

The sigmoid activation function introduces non-linearity, which is the essential part, into our model. The meaning of this non-linearity is that the output is found out by the dot product of some inputs x (x_1, x_2, …, x_m), weights w (w_1, w_2, …, wm) plus bias, and then apply sigmoid function, cannot be represented linearly. The idea is that the nonlinear activation function allows us to classify nonlinear decision boundaries in our data.

We use hidden layers in our model by replacing perceptron with sigmoid activation function neurons. Now, the question arises what the requirement for hidden layers is? Are these useful? The answer is in yes. Hidden layers help us handle complex problems that single-layer neurons cannot solve.

Hidden layers twist the problem so that it can rewrite the problem and provide easy solutions to complex problems, pattern recognition problems. For example, figure 8 explains a classic textbook problem, recognition of handwritten digits, that can help you understand the workings of hidden layers and how they work.

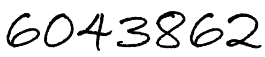


Figure 8: Digits in dataset MNIST

The digits in figure 8 is taken from a well-known dataset called MNIST. It has 70,000 examples of numbers that were written by a human. A picture of 28x28 pixels represents every digit. Therefore, this value is 28*28=784 pixels. Every pixel takes a deal between 0 and 255 (RGB color code). Zero manners the coloration is white and 255 manners the shade black.

Now, think about that computer that can really "see" a digit like a human see—the answer is no. Therefore, we need proper training to recognize these digits. The computer can't understand an image as a human can see. For this purpose, it can be interpreted to analyze how the pixel numbers are working to represent an image. Here, we dissect an image into an array defined by 784 numbers as appearing in each collection [0, 0, 180, …, … 77, 0, 0, 0], and after that, we need to feed the array into our model.

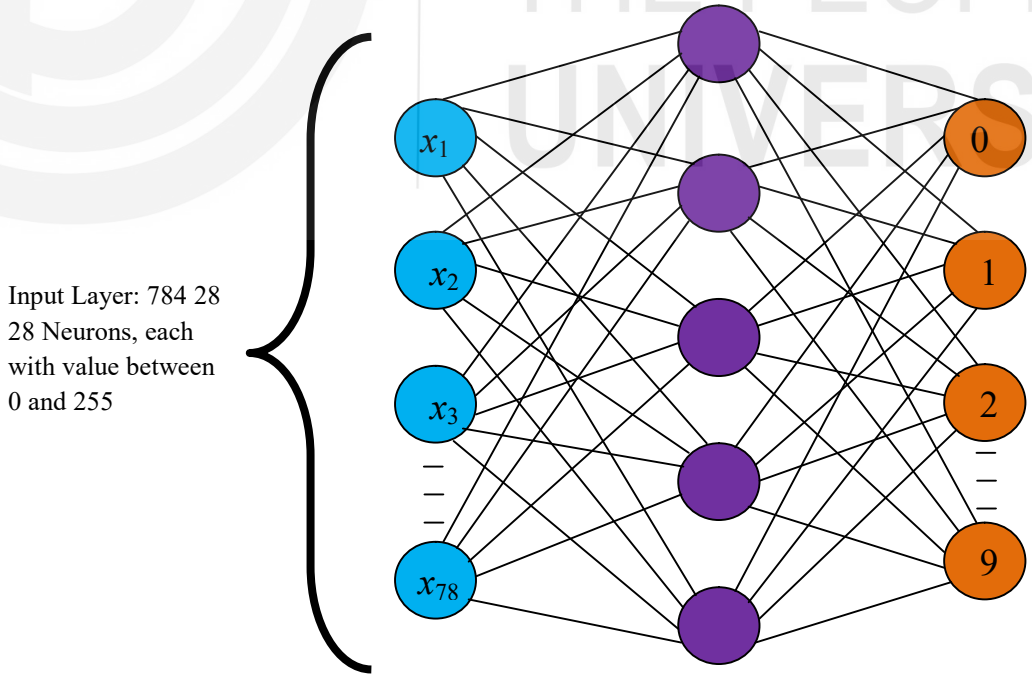


Figure 9: Neural Network with 28 * 28 pixel values

We set up a neural network, figure 9, for the problem mentioned above. It consists of 784 neurons for input layers with 28x28 pixel values. So, you may consider a total of 16 hidden neurons and ten output neurons. The ten output neurons returning in the form of an array will have different values to classify any digit from 0 to 9. So, for example, if the neural network finds the handwritten number is a zero, then the output array of [1, 0, 0, 0, 0, 0, 0, 0, 0, 0] would be returned, the first output of the array would be fired a zero, while rest of neurons at output layer would be set at 0. Similarly, take another example, If the neural network gets that the handwritten digit is a 5, then the array sequence would [0, 0, 0, 0, 0, 1, 0, 0, 0, 0] with six digits one while rest of the values will be 0's. Now, you can easily find out the sequence for any other number.

☞ **Check Your Progress 3**

Question-4 Discuss the utility of Sigmoid function in neural networks. Compare Sigmoid function with the Binary Step function.

12.6 BACK PROPAGATION ALGORITHM

The Backpropagation algorithm is a supervised learning algorithm for training the neural network model. **This algorithm** was first introduced in the 1960s, it was not popular, and in 1989 it gets popularized by Rumelhart, Hinton, and Williams, who have used this concept in a paper titled *"Learning representations by back-propagating errors.". It* is one of the most fundamental building blocks of any neural network.**, if you have multiple layers in the** neural network. Then**it is used to adjust the weight in the backward direction.**

When designing a neural network, we initially need to initialize the weights and biases with some random values. We initially gave some random values for weight and bias, but our model, through the backpropagation algorithm, will adjust these values and get the output if the difference between our actual output and predicted output is a large, more significant error.

This algorithm trains the neural network model based on chain rule method. In simple terms, you can say that after every forward pass through a network, the backpropagation algorithm works to perform a backward pass to adjust the weights and biased parameters of the model. It repeatedly adjusts the weights and biases of all the edges among all the layers so that Error i.e., the difference between predicted output and real output, should be minimum.

In other words, you can conclude that Backpropagation is **used to minimize the error/ cost function by repeatedly adjusting the network's weights and biases.** The level of adjustment is calculated by a method called the gradients of the error function concerning weight and biased parameters. In short, we are changing the weight and bias parameters so that Error becomes very small. Below Figure 10 explains the working of Backpropagation algorithm.

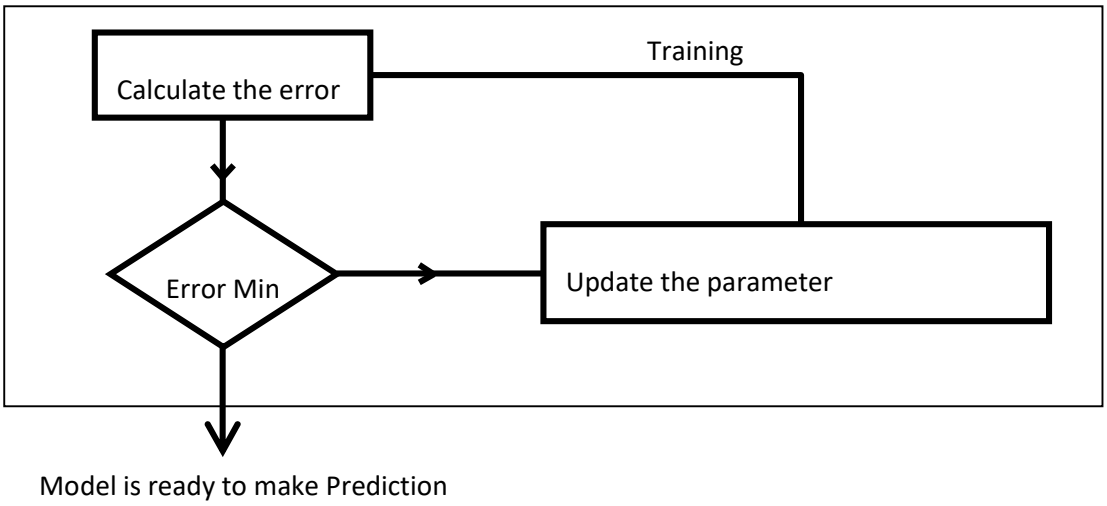


Figure 10: Backpropagation Model

The steps of backpropagation model are given below:

1. First, some random value 'W'is initialized as the weights and propagated forward accordingly.
2. Then, find out the Error after reducing that Error by propagating backward and increasing the value weight 'W.'
3. After that, observe the Error and whether it has been increased. If supplementing, then don't increase the value of 'W.'
4. Once again propagated backward and, at this time, decreased the value of 'W'.
5. Now, notice the Error, and check it has been reduced or not.

The weights that minimize the cost/ error reported. The detailed working is given by:

- Calculate the Error – The difference between the output produced by the model and the actual.
- Minimize the Error – need to check the Error, whether it is minimum or not.
- Tune the parameters – If the error value is substantial, the weights and biases must be updated. After reporting that significant Error again, this process will be repeated until the Error becomes very small.
- Check whether the model is ready for prediction – if an Error becomes significantly less, you can give some inputs to your model to get the output.

Now we learned about the need of backpropagation model and the meaning of training the model.

Now, we understand how the weight values are adjusted to reduce the Error. We are to determine whether an increment or decrement in the weight is required. After knowing it, we can keep updating the weights in that direction to reduce the Error. After some time, you will get to the

exact point where the Error is also increased if weights are updated further. We need to stop at that moment, and it becomes the final weight value.

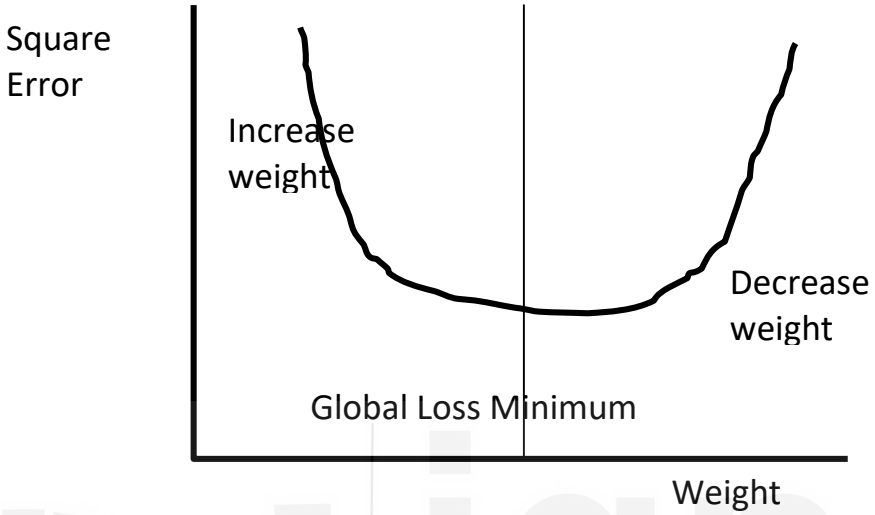


Figure 12: Error Calculation

Backpropagation Algorithm:

Initially, initialize the network weights and take small random values for it

do

for every training example, say we are terming as ex

prediction_ouput = neural_net_output_produced (network, ex) /*//it is used for forward pass*

actual output = teacher output(ex)

compute the Error part by (prediction output-actual output)

compute Delta w{h}} for all the mentioned weights, from the hidden layer to the output layer /*// it is used for the backward pass*

compute Delta w{i}} for all weights, from input layer to hidden layer *// It is backward pass continue step*

need to update network weights accordingly *// input layer does not modify*

till all examples are correctly classified or after meeting another stopping criterion

12.6.1 How does Backpropagation work?

Now you may consider below Neural Network for a better understanding:

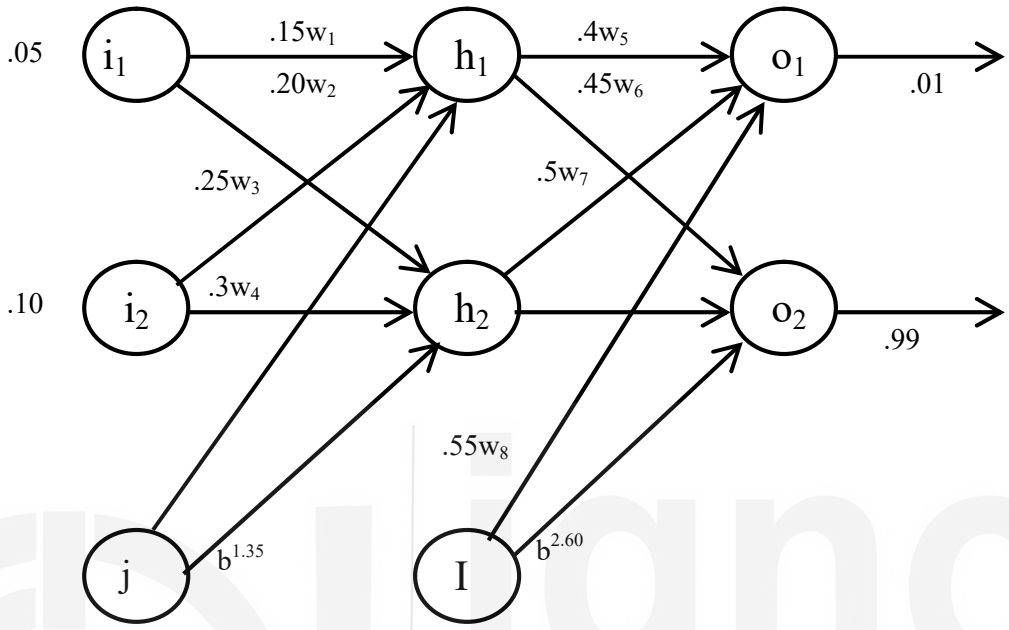


Figure 13: Neural Network Example

This network contains:

1. Three input layers
2. Two layersof hidden neurons
3. Two neurons at the output layer.

The following steps are used in the Backpropagation:

Step1: We need to use forward propagation

Step 2: After that, we have to follow backward propagation

Step 3: We put all the values to calculate the updated weight

Step 1: We use forward propagation

We start the working with forwarding propagation

$O_{1\text{Output}}$

$$net\ 0_1 = w_5 \times out\ h_1 + w_6 \times out\ h_2 + b_2 \times 1 \rightarrow .4 \times .5932 + .45 \times .5968 + .6 \times 1 = 1.1019$$

$$Out\ 0_1 = \frac{1}{1 + l^{-net\ 0_1}} \rightarrow \frac{1}{1 + l^{-1.1059}} = 0.7523$$

$$Out\ 0_2 = 0.7629$$

We also repeat the process for the output layer neurons. Hidden layer neurons outputs become the inputs.

Error for $0^1 : E0_1 = \sum \frac{1}{2}(target - output)^2 = \frac{1}{2}(.01 - 0.7513)^2 = 0.2748$

Error for $0_2 : E0_2 = 0.02356$

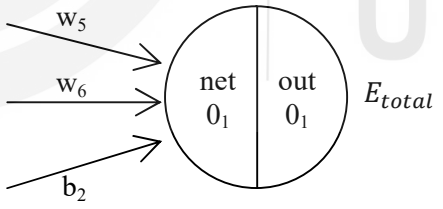
$$Total\ Error : E\ total = E0_1 + E0_2 = 0.2748 + 0.02356$$

Step – 2: Follow backward propagation

Now, we use Backpropagation to reduce the errors by adjusting weight and biases.

Now, we are checking for W_5

$$\frac{SEtotal}{Sw_5} = \frac{SEtotal}{Sout\ 0_1} \times \frac{Sout\ 0_1}{Snet\ 0_1} \times \frac{Snet\ 0_1}{Sw_5}$$



After applying the Backpropagation, we find a total change in errors regarding output-1 : O_1 and output-2 : O_2.

$$E_{total} = \frac{1}{2}(target\ 0_1 - out\ 0_2)^2 + \frac{1}{2}(target\ 0_2 - out\ o_2)^2$$

$$\frac{SE_{total}}{Sout\ 0_1} = -(target\ 0_1 - out\ 0_1) = -(0.01 - .7513) = 0.74136$$

Now, we need to propagate backward to find the changes in O1 concerning its total net input.

$$out\ o1 = 1/1 + e^{-net}$$

$$\frac{\delta out\ 01}{\delta net\ 01} = out\ o1\ (1 - out\ o1) = 0.75136507\ (1 - 0.75136507) = 0.186815602$$

Now, we check the total changes in O1 concerning weight W5.

$$net\ 0_1 = w_5 \times out\ h_1 + w_6 \times out\ h_2 + b_2 \times 1$$

$$\frac{\delta net\ 0_1}{\delta w_5} = 1 * out\ h_1 w_5^{(1-1)} + 0 + 0 = .593269$$

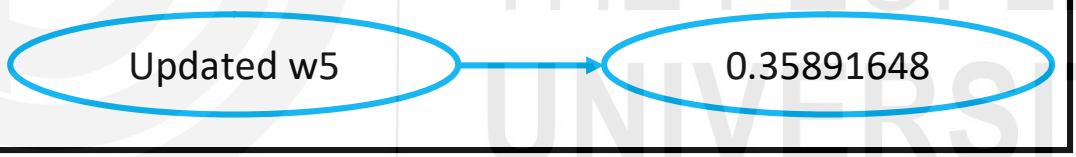
Step 3: We need to put all the values for calculating the updated weight
If we put all the values together, then:

$$\frac{\boldsymbol{\delta Etotal}}{\boldsymbol{\delta w_5}} = \frac{\boldsymbol{\delta Etotal}}{\boldsymbol{\delta out\ o1}} * \frac{\boldsymbol{\delta out\ o1}}{\boldsymbol{\delta net\ o1}} * \frac{\boldsymbol{\delta net\ o1}}{\boldsymbol{\delta w_5}} \rightarrow \mathbf{0.082167041}$$

Now, find out the updated value of weight W5:

$$W_5^+ = W_5 - n\frac{\delta Etotal}{\delta w_5}$$

$$W_5^+ = .4 - .5 * 0.082167041$$



- Similarly, we calculate the weight for others also.
- After that, we need to propagate forward and compute the output. After that, recalculate the Error.
- If a computed error is tiny, we need to stop. Otherwise, we further need to propagate backward and adjust the weight accordingly.
- We keep this process will continue till the Error is significantly less quantity.

☞ Check Your Progress 4

Question 5: Write Back Propagation algorithm, and showcase its execution on a neural network of your choice (make suitable assumptions if any)

……
……..

12.7 FEED FORWARD NETWORKS FOR CLASSIFICATION AND REGRESSION

Feed forward neural network is used for various problems, including classification , regression, and pattern encoding. In the first case, the web returns a value called z=f(w,x), which is very close to the target value y. While in the second case, the target becomes the input itself $v(x, y, f(w, x))$.To deal with multiclassification, we can use either of the techniques.

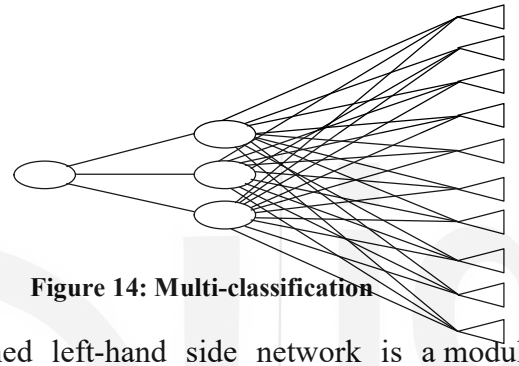
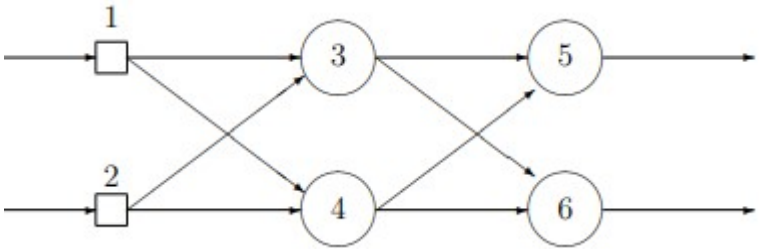


Figure 14: Multi-classification

The above-mentioned left-hand side network is a modular architecture. Here, every class connects with three distinct hidden neurons. While mentioned right-hand side network defines a fully connected network, which is used for a richer classification process. The left-side network is advantageous as it is modular and supports the classifiers' gradual construction. Whenever we feel to add a new class, the fully connected network requires further training, while the modular network only involves training for a new module. The same issue also holds for regression. However, it is worth mentioning that the output neurons are typically linear in regression tasks since there is no need to approximate any code.

As we have mentioned, a Neural network (NN) has several hidden layers; every layer consists of multiple neurons/ nodes. Each node connects with input layer connections and output layer connections. Also, we have mentioned that every connection is assigned a different weight, and finally output layer. Before giving the data into the NN, the dataset should be normalized and then processed. Training a neural network means adjusting the weights so that errors should be minimum. After introducing the NN, we can apply new data for classification or regression purposes.

Example - 4: The following diagram represents a feed-forward neural network with one hidden layer:



A weight on connection between nodes i and j is denoted by wij , such as w13 is the weight on the connection between nodes 1 and 3. The following table lists all the weights in the network:

w13 = -2, w23 = 3 ; w14=4,w24=-1 ; w35=1,w45=-1 ; w36 = -1, w46 = 1

Each of the nodes 3, 4, 5 and 6 uses the following activation function:

$\Phi(V) = 1$ for $V>=0$ and $\Phi(V) = 0$ Otherwise

Where, v denotes the weighted sum of a node. Each of the input nodes (1 and 2) can only receive binary values (either 0 or 1). Calculate the output of the network (y5 and y6) for each of the input patterns:

Pattern :	P1	P2	P3	P4
Node 1 :	0	1	0	1
Node 2 :	0	0	1	1

Answer: In order to find the output of the network it is necessary to calculate weighted sums of hidden nodes 3 and 4:

$v_3 = w_{13}x_1 + w_{23}x_2$, $v_4 = w_{14}x_1 + w_{24}x_2$

Then find the outputs from hidden nodes using activation function ϕ:

$y_3 = \phi(v_3)$, $y4 = \phi(v_4)$.

Use the outputs of the hidden nodes y3 and y4 as the input values to the output layer (nodes 5 and 6), and find weighted sums of output nodes 5 and 6:

v5 = w35y3 + w45y4 , v6 = w36y3 + w46y4 .

Finally, find the outputs from nodes 5 and 6 (also using ϕ):

y5 = ϕ(v5) , y6 = ϕ(v6) .

The output pattern will be (y5, y6). Perform this calculation for each input pattern:

P1: Input pattern (0, 0)

v3 = $-2 \cdot 0 + 3 \cdot 0 = 0$, y3 = $\phi(0) = 1$
v4 = $4 \cdot 0 - 1 \cdot 0 = 0$, y4 = $\phi(0) = 1$
v5 = $1 \cdot 1 - 1 \cdot 1 = 0$, y5 = $\phi(0) = 1$
v6 = $-1 \cdot 1 + 1 \cdot 1 = 0$, y6 = $\phi(0) = 1$

The output of the network is (1, 1)

P2: Input pattern (1, 0)

$v3 = -2 \cdot 1 + 3 \cdot 0 = -2,\ y3 = \phi(-2) = 0$

$v4 = 4 \cdot 1 - 1 \cdot 0 = 4,\ y4 = \phi(4) = 1$

$v5 = 1 \cdot 0 - 1 \cdot 1 = -1,\ y5 = \phi(-1) = 0$

$v6 = -1 \cdot 0 + 1 \cdot 1 = 1,\ y6 = \phi(1) = 1$

The output of the network is (0, 1).

P3: Input pattern (0, 1)

$v3 = -2 \cdot 0 + 3 \cdot 1 = 3,\ y3 = \phi(3) = 1$

$v4 = 4 \cdot 0 - 1 \cdot 1 = -1,\ y4 = \phi(-1) = 0$

$v5 = 1 \cdot 1 - 1 \cdot 0 = 1,\ y5 = \phi(1) = 1$

$v6 = -1 \cdot 1 + 1 \cdot 0 = -1,\ y6 = \phi(-1) = 0$

The output of the network is (1, 0).

P4: Input pattern (1, 1)

$v3 = -2 \cdot 1 + 3 \cdot 1 = 1,\ y3 = \phi(1) = 1$

$v4 = 4 \cdot 1 - 1 \cdot 1 = 3,\ y4 = \phi(3) = 1$

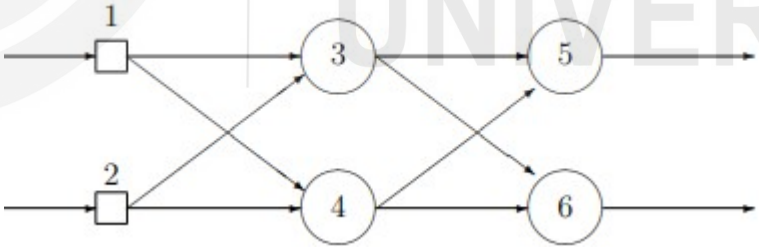
$v5 = 1 \cdot 1 - 1 \cdot 1 = 0,\ y5 = \phi(0) = 1$

$v6 = -1 \cdot 1 + 1 \cdot 1 = 0,\ y6 = \phi(0) = 1$

The output of the network is (1, 1).

☞ **Check Your Progress 5**

Question-6 The following diagram represents a feed-forward neural network with one hidden layer:



A weight on connection between nodes i and j is denoted by wij , such as w23 is the weight on the connection between nodes 2 and 3. The following table lists all the weights in the network:

w13 = -3, w23 = 2 ; w14=3,w24= - 2 ; w35=4,w45= - 3 ; w36 = -2, w46 = 2

Each of the nodes 3, 4, 5 and 6 uses the following activation function:

$\Phi(V) = 1$ for $V >= 1$ and $\Phi(V) = 0$ Otherwise

Where, v denotes the weighted sum of a node. Each of the input nodes (1 and 2) can only receive binary values (either 0 or 1). Calculate the output of the network (y5 and y6) for each of the input patterns:

Pattern :	P1	P2	P3	P4
Node 1 :	0	1	0	1
Node 2 :	0	0	1	1

.....

.....

12.8 DEEP LEARNING

Deep learning is a subset of artificial intelligence, commonly called AI, that tells us the workings of the human brain to process data and patterns defining for decision making. Deep learning has capable of learning unsupervised from unstructured data or unlabeled data. Deep learning is further classified as an AI function that is used to simulate the workings of the human brain in processing data to detect objects, recognize speech, translate languages, and make decisions.

12.8.1 How Deep Learning Works

Initially, there was a limitation of computing resources, and the concept of deep learning was not so popular. Once these resources were available, deep learning took the attention of the researchers. Deep Learning can handle all forms of data from all world regions. This data is available in massive amounts, termed big data, and is taken from various sources, including social media, search engines, different e-platforms, and others multimedia sources. Big data is accessible through multiple fintech applications such as cloud computing.

However, this data is so vast and primarily considered unstructured that it could take decades or centuries for humans to understand or find meaningful decisions. As mentioned earlier, the deep learning model's work is similar to the multilayer perceptron models. We have various models, such as convolution neural network (CNN) and long short term Model (LSTM). The exact working of CNN and LSTM is out of scope, but you can refer to the working of multilayer perception to understand the working of the deep learning model.

12.8.2 Deep Learning vs. Machine Learning

One of the popular AI techniques available for processing big data is machine learning.

For example, if a digital payments company wants to find out the occurrence of fraud in their system, such a company might use machine learning tools. The used algorithm, built on the machine learning technique, will process all transactions on the digital platform, try to find out the patterns, and mention the detection of anomalies by the pattern.

Deep learning, termed a subset of machine learning algorithms, uses a hierarchical structure of neural networks model to forward the same process as the machine learning algorithm used. The neural networks, like the human brain, connect like a web. The program built for machine learning linearly uses data, while deep learning systems enable a nonlinear approach to process the data.

12.8.3 A Deep Learning Example

For example, the fraud detection system mentioned above can be solved through deep learning. However, as a machine learning system starts working with parameters, such as transactions of dollars an account holder sends or receives, the deep-learning method can also use the same parameters, but it works on a neural network.

As we have mentioned earlier, each layer of the neural network takes the inputs from the previous layer; for example, the input layer has the parameters like sender information, data from social media, a credit score of the customer, using IP address, and others and passed the output to next layer for decision making. The final layer, the output layer, decides whether fraud has been detected or not.

Deep learning, a prevalent technology, is used across all major industries for various tasks, including decision making. Other examples may include commercial apps for image recognition, apps for a recommendation system, and medical research tools for exploring the possibility of reusing drugs.

☞ Check Your Progress 6

Question-7 Compare between Deep Learning and Machine Learning

……

……

12.9 SUMMARY

In this unit we learned about the fundamental concepts of Neural networks, and various concepts related to area of neural networks and deep learning, this includes the understanding of the activation function, back propagation algorithm, feed forward networks and many more. In this unit the concepts are simplified with the help of the numerical, which will help you map the theoretical concepts of neural networks with that of their implementation part.

12.10 SOLUTIONS/ANSWERS

☞ Check Your Progress 1

Question -1 : Below is a diagram if a single artificial neuron (unit):

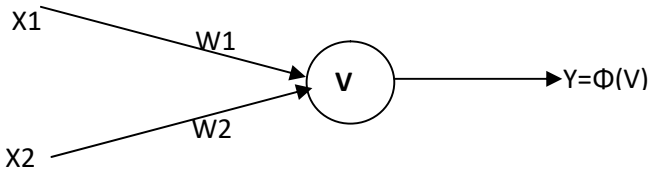


Figure A-1: Single unit with three inputs.

The node has three inputs x = (x1, x2) that receive only binary signals (either 0 or 1). How many different input patterns this node can receive?

Solution:**Refer to Section 12.3**

☞ Check Your Progress 2

Question-2 : Consider the unit shown below.

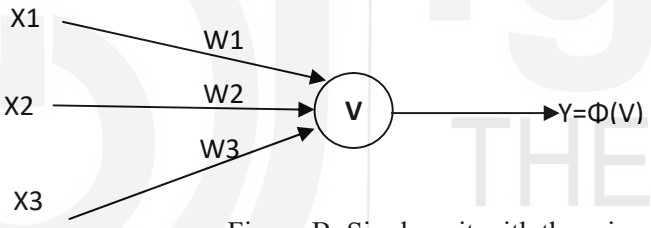


Figure B: Single unit with three inputs.

Suppose that the weights corresponding to the three inputs have the following values:

$w_1 = 1$; $w_2 = -1$; $w_3 = 2$

and the activation of the unit is given by the step-function:

$\Phi(V) = 1$ for $V >= 1$ and $\Phi(V) = 0$ Otherwise

Calculate what will be the output value y of the unit for each of the following input patterns:

Pattern	P_1	P_2	P_3	P_4
X_1	1	0	1	1
X_2	0	1	0	1
X_3	0	1	1	1

Solution: Refer to section 12.4

Question - 3: NAND, NOR) are the universal building blocks of any computational device. Logical functions return only two possible values, true or false, based on the truth or false values of their arguments. For example, operator NAND returns False only when all its arguments are True, otherwise (if any of the arguments is false) it returns false. If we denote truth by 1 and false by 0, then logical function NAND can be represented by the following table:

x1 :	**0**	**0**	**1**	**1**
x2 :	**0**	**1**	**0**	**1**
x1 NAND x2 :	**1**	**1**	**1**	**0**

This function can be implemented by a single unit with two inputs:

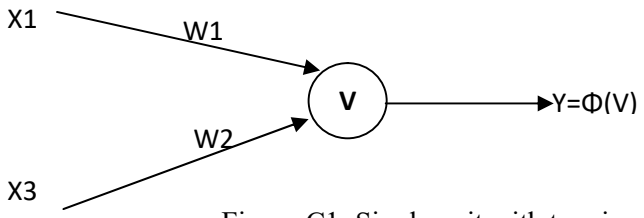


Figure C1: Single unit with two inputs.

if the weights are w1 = 1 and w2 = 1 and the activation of the unit is given by the step-function:

$\Phi(V) = 1$ for $V >= 2$ and $\Phi(V) = 0$ Otherwise

Note that the threshold level is 2 ($v \geq 2$).

a) Test how the neural NAND function works.
b) Suggest how to change either the weights or the threshold level of this single unit in order to implement the logical NOR function (true when at least one of the arguments is true):

x1 :	**0**	**0**	**1**	**1**
x2 :	**0**	**1**	**0**	**1**
x1 NOR x2 :	**1**	**0**	**0**	**0**

Solution: Refer to section 12.4

☞ Check Your Progress 3

Question-4 Discuss the utility of Sigmoid function in neural networks. Compare Sigmoid function with the Binary Step function.

Solution: Refer to Section 12.5

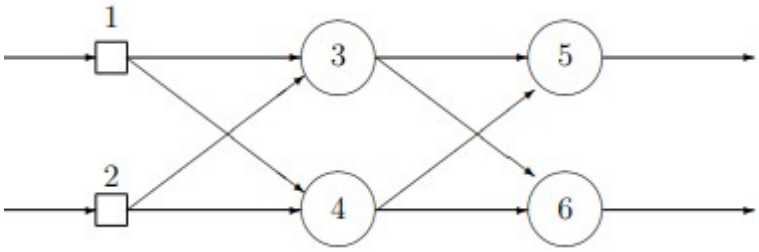
☞ Check Your Progress 4

Question 5: Write Back Propagation algorithm, and showcase its execution on a neural network of your choice (make suitable assumptions if any)

Solution: Refer to Section 12.6

☞ Check Your Progress 5

Question-6 The following diagram represents a feed-forward neural network with one hidden layer:



A weight on connection between nodes i and j is denoted by wij , such as w23 is the weight on the connection between nodes 2 and 3. The following table lists all the weights in the network:

w13 = -3, w23 = 2 ; w14=3,w24= - 2 ; w35=4,w45= - 3 ; w36 = -2, w46 = 2

Each of the nodes 3, 4, 5 and 6 uses the following activation function:

$\Phi(V) = 1$ for $V>=1$ and $\Phi(V) = 0$ Otherwise

Where, v denotes the weighted sum of a node. Each of the input nodes (1 and 2) can only receive binary values (either 0 or 1). Calculate the output of the network (y5 and y6) for each of the input patterns:

Pattern :	P1	P2	P3	P4
Node 1 :	0	1	0	1
Node 2 :	0	0	1	1

Solution: Refer to Section 12.7

☞ Check Your Progress 6

Question-7 Compare between Deep Learning and Machine Learning

Solution: Refer to Section 12.8

12.11 FURTHER READINGS

1) Dr K Uma Rao, "Artificial Intelligence and Neural Networks", Pearson Education (January 2011)

2) Tariq Rashid, "Make Your Own Neural Network: A Gentle Journey Through the Mathematics of Neural Networks, and Making Your Own Using the Python Computer Language",

3) Russell J. Stuart, Norvig Peter," Artificial Intelligence" , Pearson: A Modern Approach Paperback – January 2015.

4) F. AcarSavaci, Artificial Intelligence and Neural Networks, Springer; 2006th edition (18 July 2006).

5) Vladimir Golovko (Editor), Akira Imada (Editor), "Neural Networks and Artificial Intelligence: 8th International Conference, ICNNAI 2014"

6) ToshinoriMunakata, "Fundamentals of the New Artificial Intelligence" Springer; 2nd ed. 2008 edition (February 2008)