MALLA REDDY COLLEGE OF ENGINEERING

UNIT – II

Multi-layer Perceptron– Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

MALLA REDDY COLLEGE OF ENGINEERING

UNIT – II Multi-layer Perceptron— Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

Multi-layer Perceptron :

The multilayer perceptron is an artificial neural network structure and is a nonparametric estimator that can be used for classification and regression. We discuss the back propagation algorithm to train a multilayer perceptron for a variety of applications.

We have pretty much decided that the learning in the neural network happens in the weights. So, to perform more computation it seems sensible to add more weights. There are two things that we can do:

add some backwards connections, so that the output neurons connect to the inputs again, or add more

neurons.

The **first approach** leads into recurrent networks. These have been studied, but are not that

commonly used.We will instead consider the **second approach**.

MALLA REDDY COLLEGE OF ENGINEERING

UNIT – II Multi-layer Perceptron¯ Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

Multi-layer Perceptron :

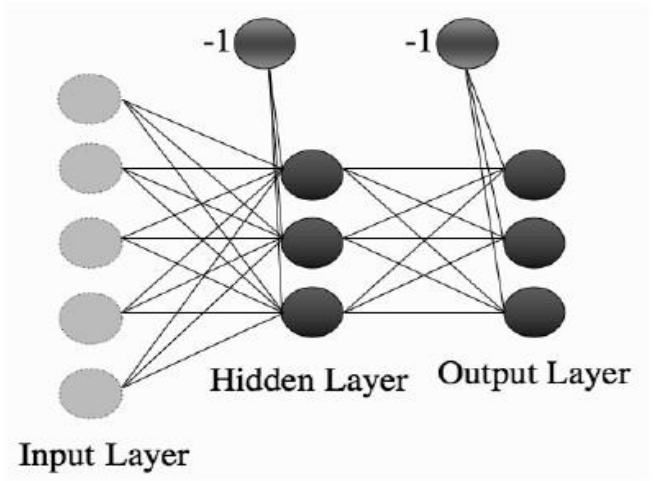


FIGURE 4.1 The Multi-layer Perceptron network, consisting of multiple layers of connected neurons.

MALLA REDDY COLLEGE OF ENGINEERING

UNIT – II Multi-layer Perceptron¯ Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

Multi-layer Perceptron :.

we can check that a prepared network can solve the two-dimensional XOR problem, something that we have seen is not possible for a linear model like the Perceptron.

A suitable network is shown in Figure 4.2. To check that it gives the correct answers, all that is required is to put in each input and work through the network, treating it as **two different Perceptrons**, first computing the activations of the neurons in the middle layer (labelled as C and D in Figure 4.2) and then using those activations as the inputs to the single neuron at the output. As an example, I'll work out what happens when you put in (1, 0) as an input;

MALLA REDDY COLLEGE OF ENGINEERING

UNIT – II Multi-layer Perceptron¯ Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

Multi-layer Perceptron :.

A suitable network is shown in Figure 4.2. To check that it gives the correct answers, all that is required is to put in each input and work through the network, treating it as **two different Perceptrons**, first computing the activations of the neurons in the middle layer (labelled as C and D in Figure 4.2) and then using those activations as the inputs to the single neuron at the output. As an example, I'll work out what happens when you put in (1, 0) as an input;

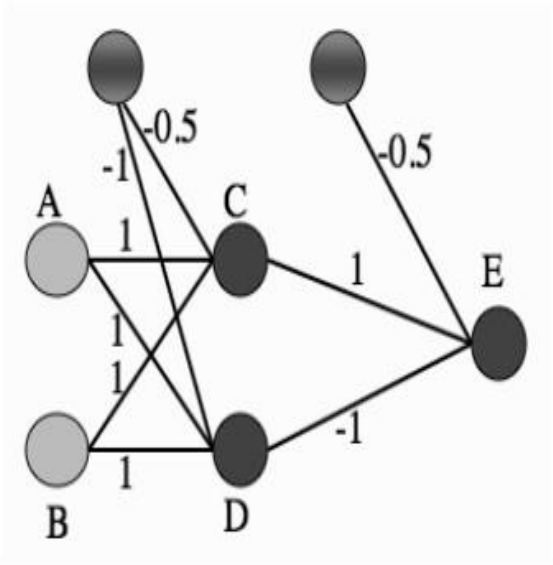


FIGURE 4.2 A Multi-layer Perceptron network showing a set of weights that solve the XOR problem.

MALLA REDDY COLLEGE OF ENGINEERING

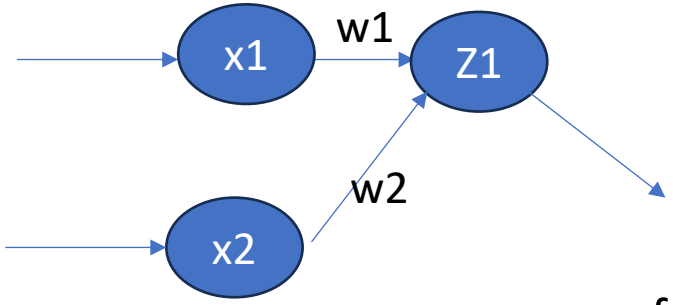
UNIT – II Multi-layer Perceptron— Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

Multilayer Perceptron with XOR example

y= z1+z2

Where z1= $x1\overline{x2}$ z2 = $\overline{x1}$x2

Z1= $x1\overline{x2}$



X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$y = x1\overline{x2} + \overline{x1}x2$

f(Yn) = {1 if Yin≥0

{0 Yin<0

Activation Function

MALLA REDDY COLLEGE OF ENGINEERING

UNIT – II Multi-layer Perceptron— Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

X1	X2	Z
0	0	0
0	1	0
1	0	1
1	1	0

Multi-layer Perceptron with XOR

Let us consider the weights for w11 = 1 w21 =1

Threshold = 1 learning rate = 1.5

(0,0) ⟶ Zin = Wij*Xi = *1*0+1*0 = 0 (out = 0)*

(0,1) ⟶ Zin = Wij*Xi = *1*0+1*1 = 1 (out = 1)*

*- wij =wij +n(t-o)xi ∴ -w11 = 1+1.5 *(0-1)*0=0 w21 = 1+1.5 * (0-1) * 1 = -0.5*

MALLA REDDY COLLEGE OF ENGINEERING

UNIT – II Multi-layer Perceptron— Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

Multi-layer Perceptron with XOR

First function = Z1= $x1\overline{x2}$

Let us consider the weights for w11 = 1 w21 =1 updated weights:w11= 1, w21= -0.5

Threshold = 1 learning rate = 1.5

X1	X2	Z1
0	0	0
0	1	0
1	0	1
1	1	0

(0,0) ⟶ Zin = Wij*Xi = *1*0+(-0.5)*0 = 0 (out = 0)*

(0,1) ⟶ Zin = Wij*Xi = *1*0+(-0.5)*1 = -0.5 (out = 0)*

*(1,0) ⟶ Zin = Wij*Xi = 1*1+(-0.5)*0 = 1 (out=1)*

*(1,1) ⟶ Zin = Wij*Xi = 1*1+(-0.5) * 1 = 0.5 (out =0)*

*- wij =wij +n(t-o)xi ∴ -w11 = 1+1.5 *(0-1)*0=1 w21 = 1+1.5 * (0-1) * 1 = -0.5*

MALLA REDDY COLLEGE OF ENGINEERING

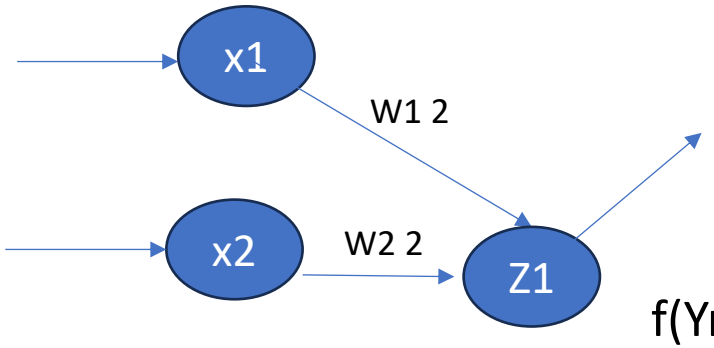
UNIT – II Multi-layer Perceptron— Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

Multilayer Perceptron with XOR example

y= z1+z2

Where z1= $x1\overline{x2}$ z2 = $\overline{x1}$x2

z2 = $\overline{x1}$x2



X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$y = x1\overline{x2} + \overline{x1}$x2

f(Yn) = {1 if Yin≥0

{0 Yin<0

Activation Function

MALLA REDDY COLLEGE OF ENGINEERING

UNIT – II Multi-layer Perceptron— Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

Multi-layer Perceptron with XOR

First function = z2 = $\overline{x1}$x2

Let us consider the weights for w12 = 1 w2 2 = 1 updated weights:w12= -0.5, w2 2=1

X1	X2	Z2
0	0	0
0	1	**1**
1	0	0
1	1	0

Threshold = 1 learning rate = 1.5

(0,0) ⟶ Z2in = Wij*Xi = *1*0+(1)*0 = 0 (out = 0)*

(0,1) ⟶ Z2in = Wij*Xi = *1* 0 +(1)*1 =* ***1*** *(out =* ***1****)*

(1,0) ⟶ *Zin = Wij*Xi = 1*1+(1)*0 = 1 (out=0)*

Fyin = {1 if yin ≥ 0
{0 if yin ≤ 0

Activation function

*- wij =wij +n(t-o)xi ∴ -w12 = 1+1.5 *(0-1)* 1=-0.5 w22 = 1+1.5 * (0-1) * 0 = 1*

MALLA REDDY COLLEGE OF ENGINEERING

UNIT – II Multi-layer Perceptron— Going Forwards – Going Backwards: Back Propagation Error – Multi-layer Perceptron in Practice – Examples of using the MLP – Overview – Deriving Back-Propagation – Radial Basis Functions and Splines – Concepts – RBF Network – Curse of Dimensionality – Interpolations and Basis Functions – Support Vector Machines

Multi-layer Perceptron with XOR

First function = z2 = $\bar{x}1x2$

Let us consider the weights for w12 = 1 w2 2 = 1 updated weights:w12= -0.5, w2 2=1

Threshold = 1 learning rate = 1.5

X1	X2	Z2
0	0	0
0	1	**1**
1	0	0
1	1	0

(0,0) ⟶ Z2in = Wij*Xi = *-0.5* (0) + 1 * 0 = 0 (out = 0)*

(0,1) ⟶ Z2in = Wij*Xi = *-0.5* 0 +(1)*1 = **1** (out = **1**)*

(1,0) ⟶ *Z2in = Wij*Xi = -0.5*1+(1)*0 = -0.5 (out=0)*

(1,1) ⟶ *Z2in = Wij*xi = -0.5 * 1 + 1 * 1 =0.5 (out = 0)*

Fyin = {1 if yin ≥ 0

Activation function {0 if yin ≤ 0

*- wij =wij +n(t-o)xi ∴ -w12 = 1+1.5 *(0-1)* 1=-0.5 w22 = 1+1.5 * (0-1) * 0 = 1*

MALLA REDDY COLLEGE OF ENGINEERING

- Y = Z1 OR Z2
- => yin = Z1v1 +z2v2
- => v1 = 1 ; v2 = 1

Threshold = 1 and learning rate = 1.5

X1	X2	Z1	Z2	Y
0	0	0	0	0
0	1	0	**1**	**1**
1	0	1	0	1
1	1	0	0	0

✓ (0,0) -> yin = vi * xi = 1*0+1*0 = 0 (out =0)

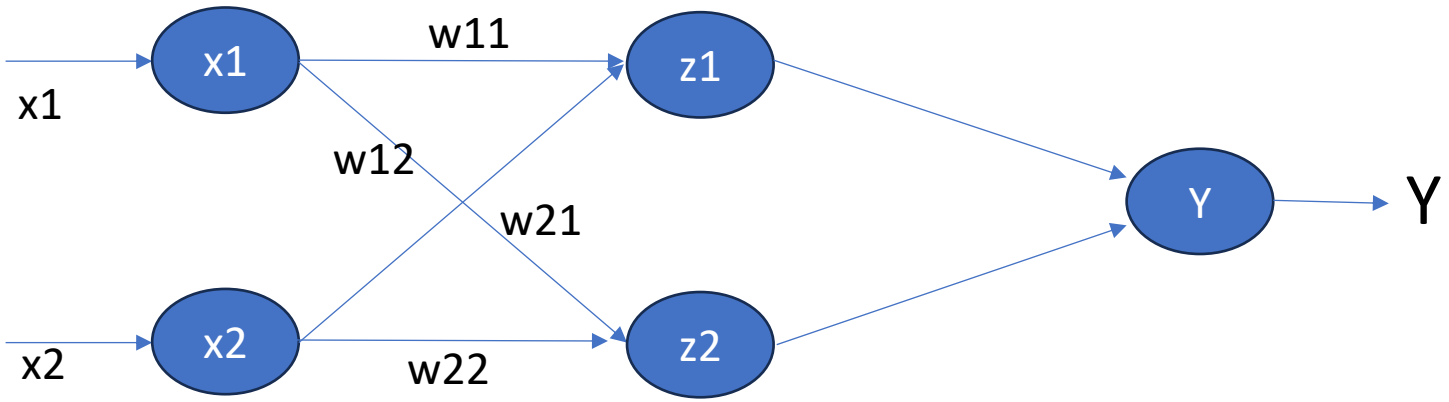
✓ (0,1) -> yin = vi*xi = 1*0+1*1 = 1 (out = 1)

✓ (1,0)-> yin = vi*xi = 1*1+1*0 = 1 (out = 1)

✓ (0,0)-> yin = vi*xi = 0*1 + 0*1 = 0 (out =0)

MALLA REDDY COLLEGE OF ENGINEERING

- Final weights with output Y



MALLA REDDY COLLEGE OF ENGINEERING

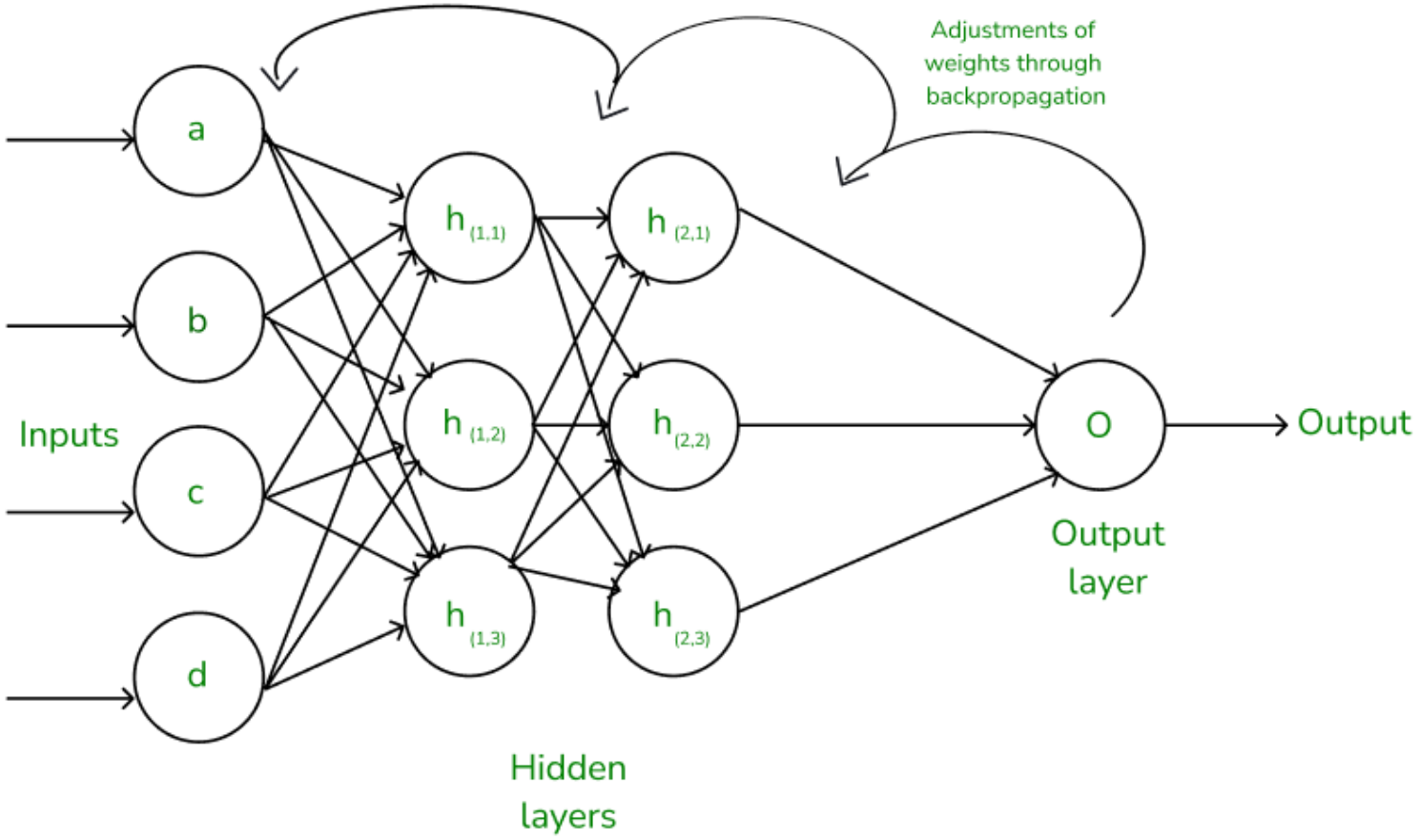
•In machine learning, backpropagation is an effective algorithm used to train artificial neural networks, especially in feed-forward neural networks.

•Backpropagation is an iterative algorithm, that helps to minimize the cost function by determining which weights and biases should be adjusted. During every epoch, the model learns by adapting the weights and biases to minimize the loss by moving down toward the gradient of the error. Thus, it involves the two most popular optimization algorithms,

•such as gradient descent or stochastic gradient descent.

•Computing the gradient in the backpropagation algorithm helps to minimize the cost function and it can be implemented by using the mathematical rule called chain rule from calculus to navigate through complex layers of the neural network.

MALLA REDDY COLLEGE OF ENGINEERING



MALLA REDDY COLLEGE OF ENGINEERING

Advantages of Using the Backpropagation Algorithm in Neural Networks

Backpropagation, a fundamental algorithm in training neural networks, offers several advantages that make it a preferred choice for many machine learning tasks. Here, we discuss some key advantages of using the backpropagation algorithm:

1. ***Ease of Implementation*:** Backpropagation does not require prior knowledge of neural networks, making it accessible to beginners. Its straightforward nature simplifies the programming process, as it primarily involves adjusting weights based on error derivatives.
2. ***Simplicity and Flexibility:*** The algorithm's simplicity allows it to be applied to a wide range of problems and network architectures. Its flexibility makes it suitable for various scenarios, from simple feedforward networks to complex recurrent or convolutional neural networks.
3. ***Efficiency*:** Backpropagation accelerates the learning process by directly updating weights based on the calculated error derivatives. This efficiency is particularly advantageous in training deep neural networks, where learning features of a function can be time-consuming.
4. ***Generalization:*** Backpropagation enables neural networks to generalize well to unseen data by iteratively adjusting weights during training. This generalization ability is crucial for developing models that can make accurate predictions on new, unseen examples.
5. ***Scalability*:** Backpropagation scales well with the size of the dataset and the complexity of the network. This scalability makes it suitable for large-scale machine learning tasks, where training data and network size are significant factors.

MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm

The Backpropagation algorithm works by two different passes, they are:

Forward pass Backward pass

In forward pass, initially the input is fed into the input layer. Since the inputs are raw data, they can be used for training our neural network.

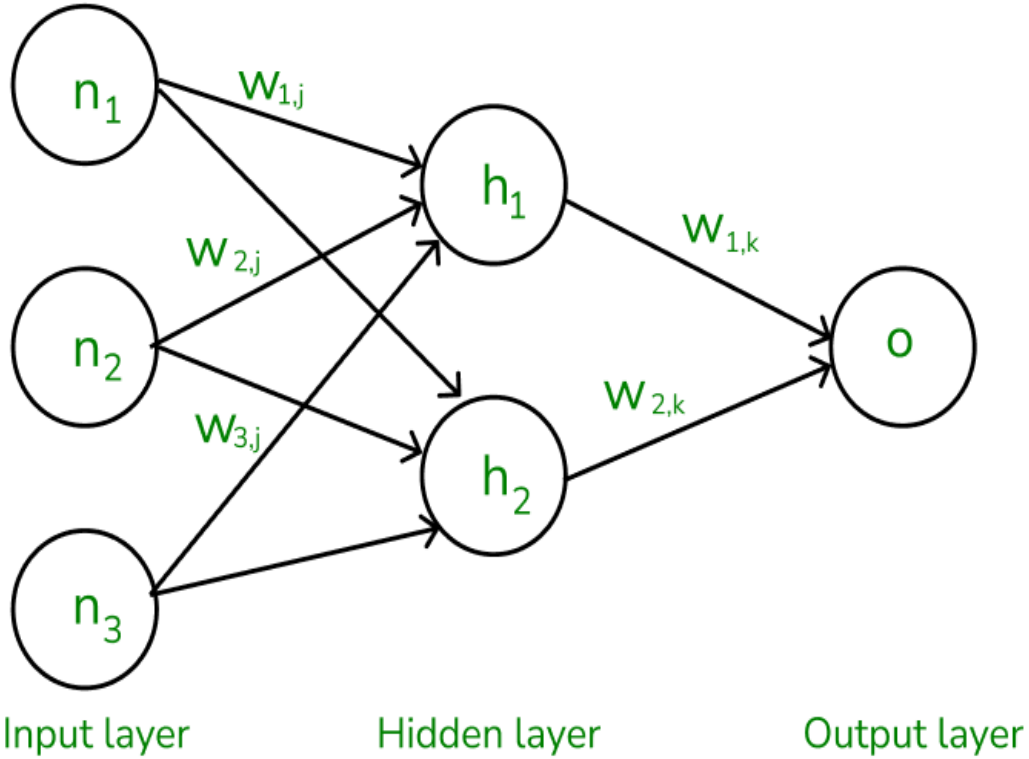
The inputs and their corresponding weights are passed to the hidden layer. The hidden layer performs the computation on the data it receives. If there are two hidden layers in the neural network, for instance, consider the illustration fig(a), h_1 and h_2 are the two hidden layers, and the output of h_1 can be used as an input of h_2. Before applying it to the activation function, the bias is added.

To the weighted sum of inputs, the activation function is applied in the hidden layer to each of its neurons. One such activation function that is commonly used is ReLU can also be used, which is responsible for returning the input if it is positive otherwise it returns zero. By doing this so, it introduces the non-linearity to our model, which enables the network to learn the complex relationships in the data. And finally, the weighted outputs from the last hidden layer are fed into the output to compute the final prediction, this layer can also use the activation function called the **softmax function** which is responsible for converting the weighted outputs into probabilities for each class.

MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm

The Backpropagation algorithm works by two different passes, they are:
Forward pass



MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm

The Backpropagation algorithm works by two different passes, they are: Backward pass

•In the backward pass process shows, the error is transmitted back to the network which helps the network, to improve its performance by learning and adjusting the internal weights.

•To find the error generated through the process of forward pass, we can use one of the most commonly used methods called mean squared error which calculates the difference between the predicted output and desired output. The formula for mean squared error is:

Mean squared error = (predicted output – actual output)^2

•Once we have done the calculation at the output layer, we then propagate the error backward through the network, layer by layer.

•The key calculation during the backward pass is determining the gradients for each weight and bias in the network. This gradient is responsible for telling us how much each weight/bias should be adjusted to minimize the error in the next forward pass. The chain rule is used iteratively to calculate this gradient efficiently.

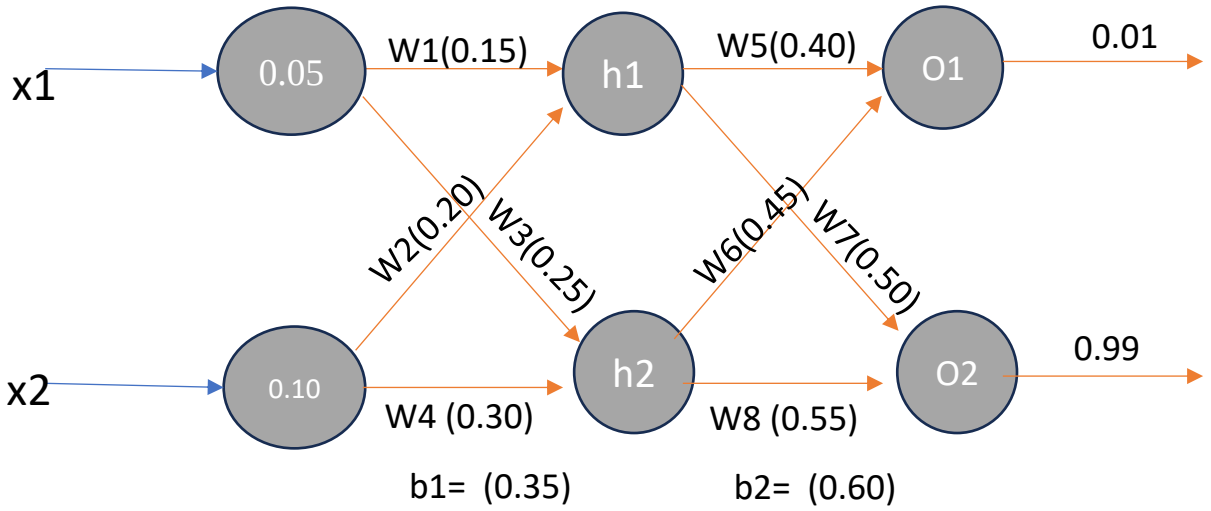
•In addition to gradient calculation, the activation function also plays a crucial role in backpropagation, it works by calculating the gradients with the help of the derivative of the activation function.

MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 1:
calculate forward propagation error
calculate h1(in and out)
where h1 = w1x1+w2x2+b1

Here h1 and h2 hidden layers
⇒ w1 to w8 are weights
⇒ b1 and b2 are bias factor
where b1 between input and hidden layer
where b2 between hidden and output layer



MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 1:
calculate forward propagation error
calculate h1(in and out)

where h1 = w1x1+w2x2+b1

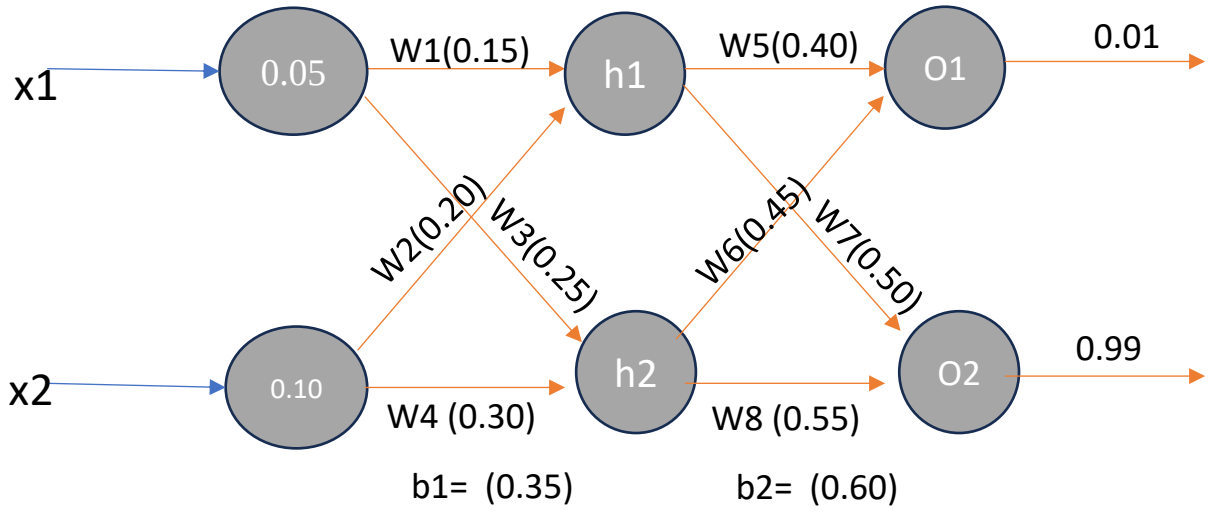
h1(in)

= 0.15*0.05+0.20*0.10+0.35

= 0.377

H2 (out) = $\frac{1}{1+e^{-h1(in)}}$

H2 (out) = $\frac{1}{1+e^{-0.3777}}$ **= 0.5932**



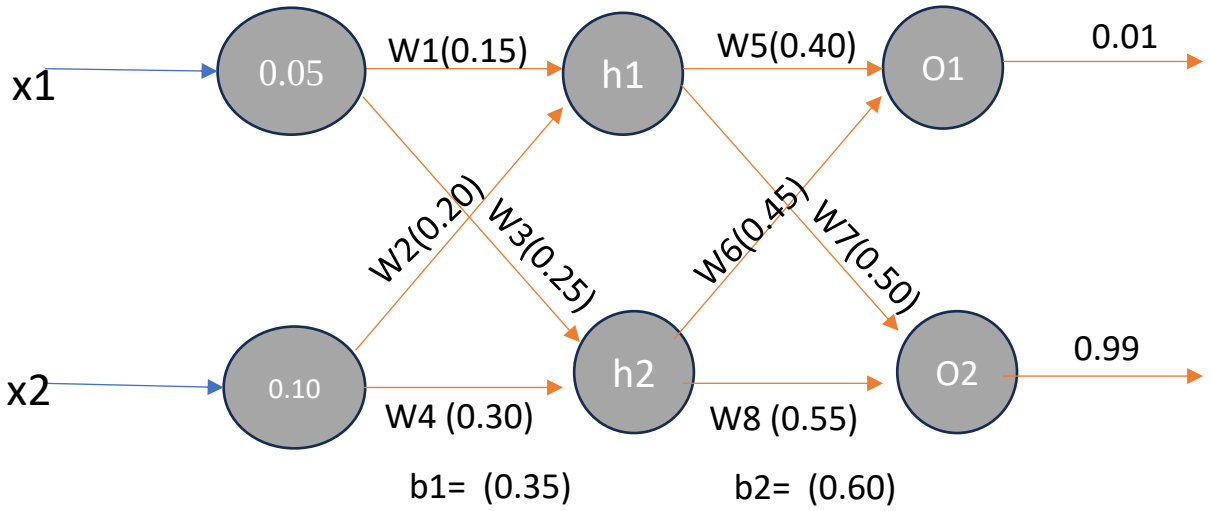
MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 2:

calculate forward propagation error

calculate h2(in and out)



where h2 = x1w3+x2w4+b1

h2(in)

= 0.05*0.25+0.10*0.3+0.35

= 0.0125+0.03+0.35 = 0.3925

h2 (out) = $\frac{1}{1+e^{-h2(in)}}$

H2 (out) = $\frac{1}{1+e^{-0.3925}}$ **= 0.5968**

MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate forward propagation error

calculate O1(in and out)

where O1 = h1(out)*w5+h2(out)*w6+b2

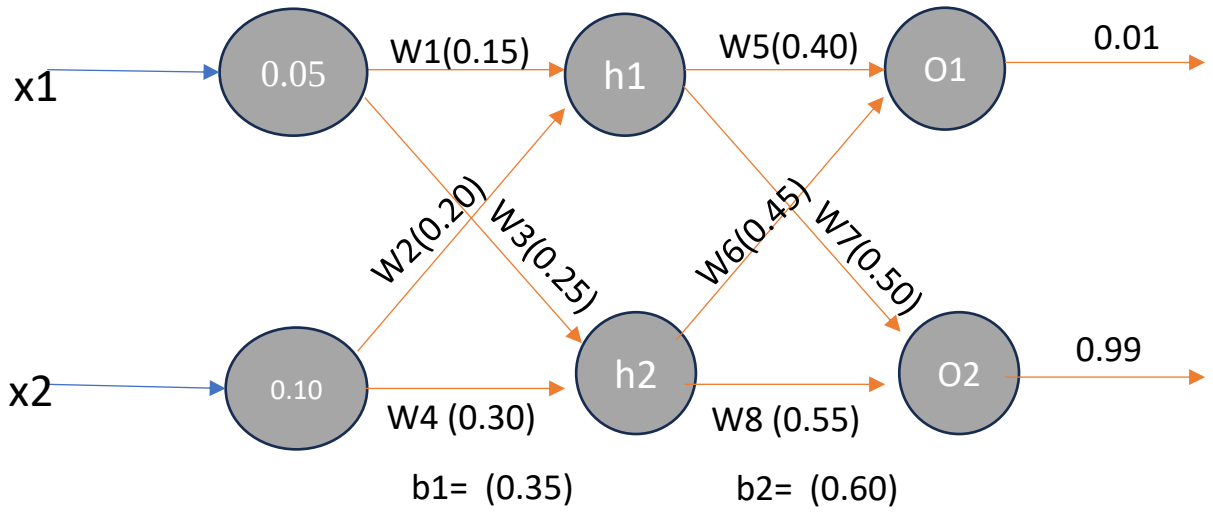
O1(in)

= 0.593*0.4+0.596*0.45+0.6

= 1.105

O1 (out) = $\frac{1}{1+e^{-O1(in)}}$

H2 (out) = $\frac{1}{1+e^{-1.105}}$ **= 0.7513**



MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate forward propagation error

calculate O2(in and out)

where O2 = h1(out)*w7+h2(out)*w8+b2

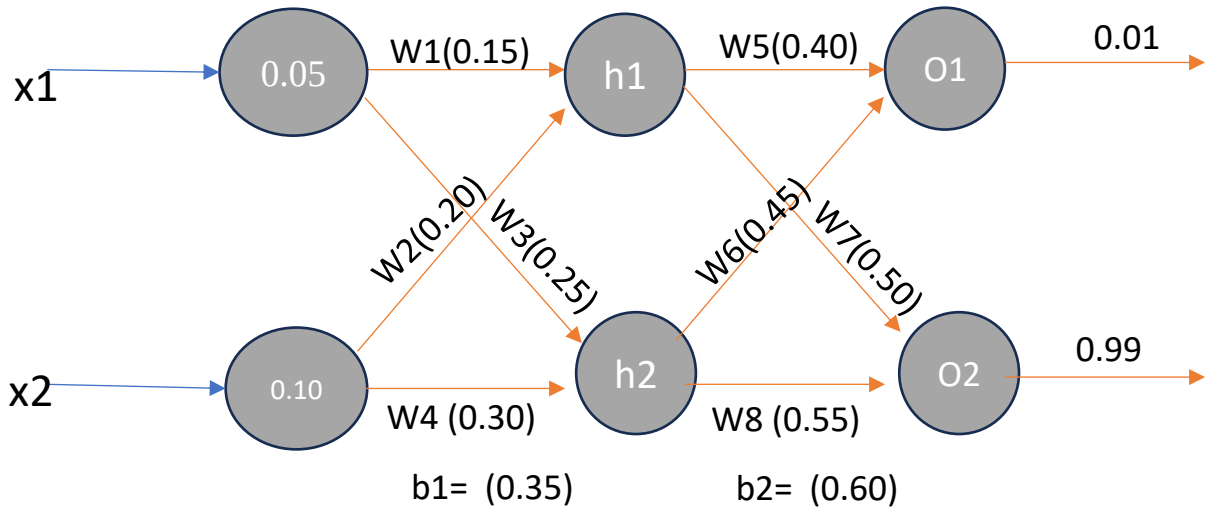
O2(in)

= 0.5932*0.50+0.5968*0.55+0.6

= 1.22484

O2 (out) = $\frac{1}{1+e^{-O2(in)}}$

H2 (out) = $\frac{1}{1+e^{-1.22484}}$ **= 0.7729**



MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate forward propagation error

calculate E total where E represents Error

Etotal = $\sum \frac{1}{2}(target - Output)^2$

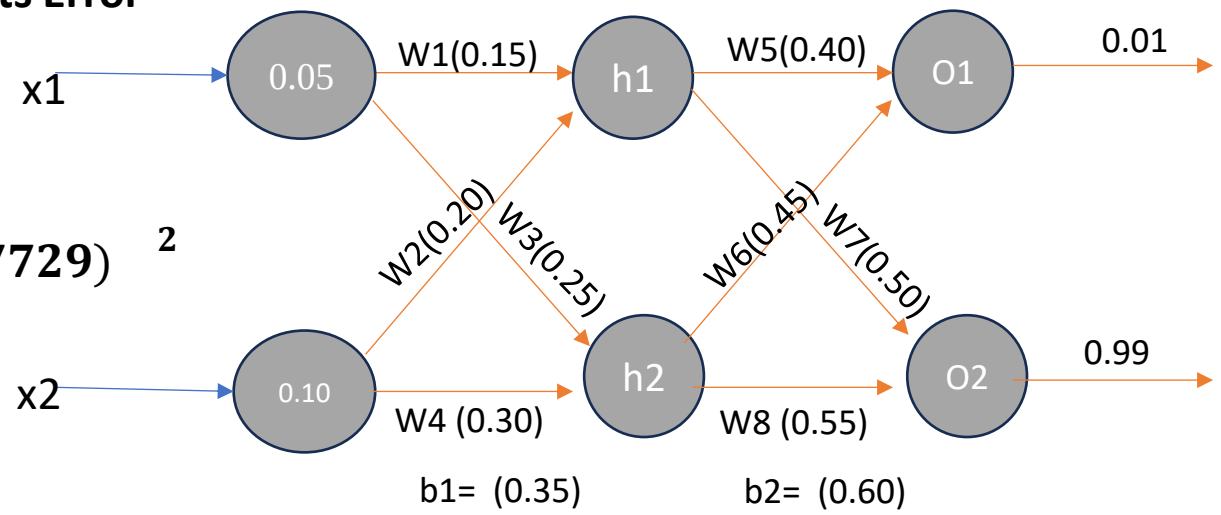
= $E_{O1} + E_{O2}$ **where error at output O1**

where error at output O2

$= \frac{1}{2}(0.01 - 0.7513)^2 + \frac{1}{2}(0.99 - 0.7729)^2$

$= \frac{1}{2}(-0.7413)^2 + \frac{1}{2}(0.2171)^2$

= 0.274+ 0.0235 = 0.29837 (appprx)



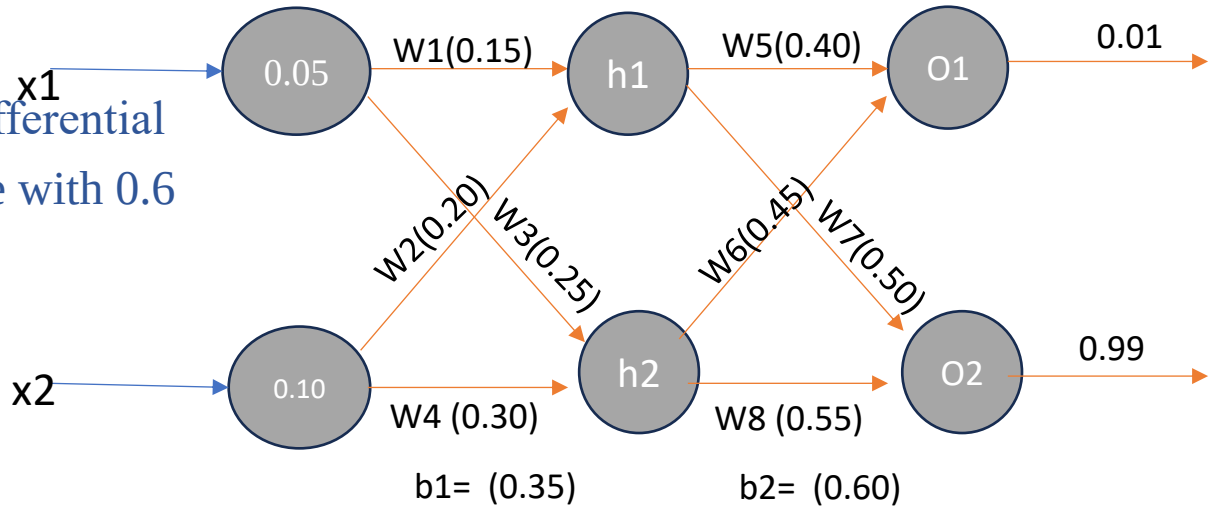
MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate Backward propagation error (output layer to hidden layer)

W5,W6,W7,W8



*$W5 = W5 - n\frac{\partial Etotal}{\partial W5}$ where ∂ Partial Differential

Where n is Learning Rate with 0.6

$$\frac{\partial Etotal}{\partial W5} = \frac{\partial Etotal}{\partial out\ O1} * \frac{\partial Out\ O1}{\partial net\ O1} * \frac{\partial net\ O1}{\partial W5}$$

$\frac{\partial Etotal}{\partial out\ O1}$ =Out O1 – Target O1

= 0.751365-0.01 = 0.7413565

$\frac{\partial Out\ O1}{\partial net\ O1}$ = Out O1 (1-Out O1)

= 0.751365(1-0.751365)

= 0.186815602

MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate Backward propagation error (output layer to hidden layer)

W5,W6,W7,W8

*W5 = W5-n$\frac{\partial Etotal}{\partial W5}$ where ∂ Partial Differential

Where n is Learning Rate with 0.6

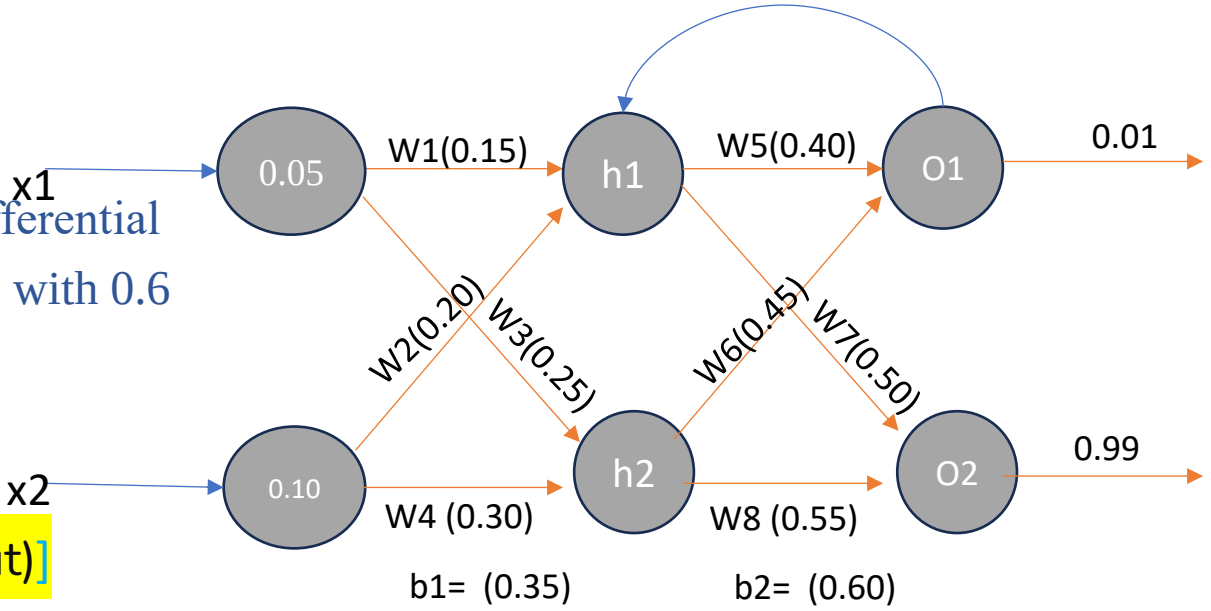
$\frac{\partial Etotal}{\partial W5} = \frac{\partial Etotal}{\partial out\ O1} * \frac{\partial Out\ O1}{\partial net\ O1} * \frac{\partial net\ O1}{\partial W5}$

$\frac{\partial net\ O1}{\partial W5}$ = out h1 = 0.59326992 [h1 (out)]

$\frac{\partial Out\ O1}{\partial net\ O1}$ = Out O1 (1-Out O1)

= 0.751365(1-0.751365)

= 0.186815602



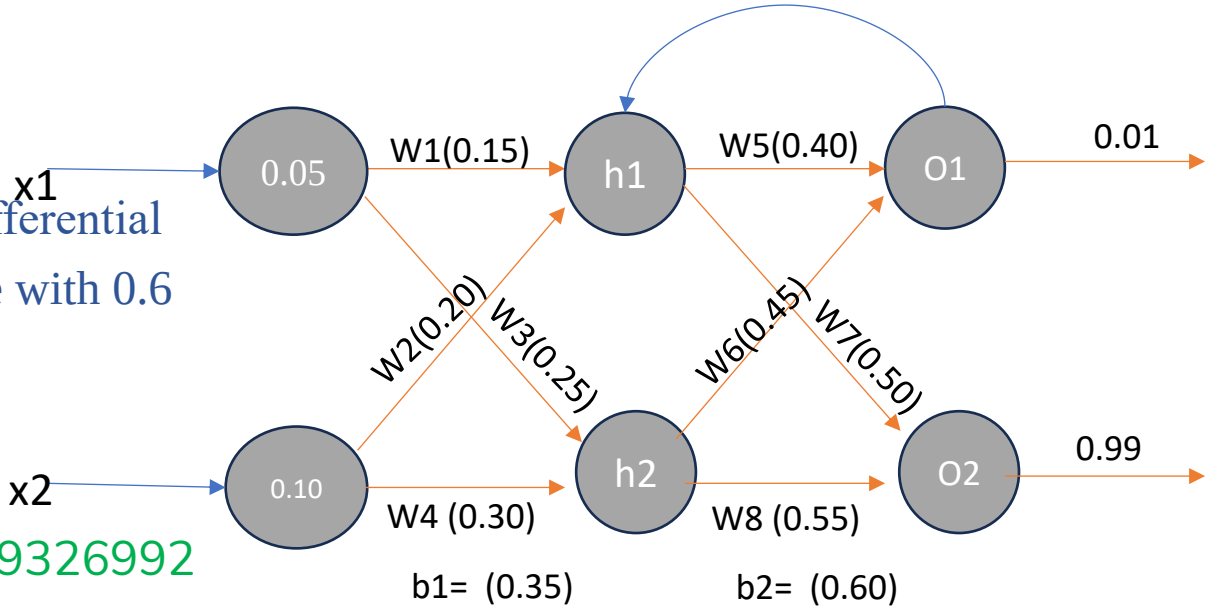
MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate Backward propagation error (output layer to hidden layer)

W5,W6,W7,W8



*$W5 = W5 - n\frac{\partial Etotal}{\partial W5}$ where ∂ Partial Differential

Where n is Learning Rate with 0.6

$$\frac{\partial Etotal}{\partial W5} = \frac{\partial Etotal}{\partial out\ O1} * \frac{\partial\ Out\ O1}{\partial\ net\ O1} * \frac{\partial\ net\ O1}{\partial W5}$$

$$\frac{\partial Etotal}{\partial W5} = 0.7413565*0.186815602*0.59326992$$

$$= 0.08216704$$

$$W5^* = W5 - n\frac{\partial Etotal}{\partial W5} = 0.4-(0.6*0.08216704)$$

$$= 0.350699776$$

MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate Backward propagation error (hidden layer to input layer)

W1,W2,W3,W4

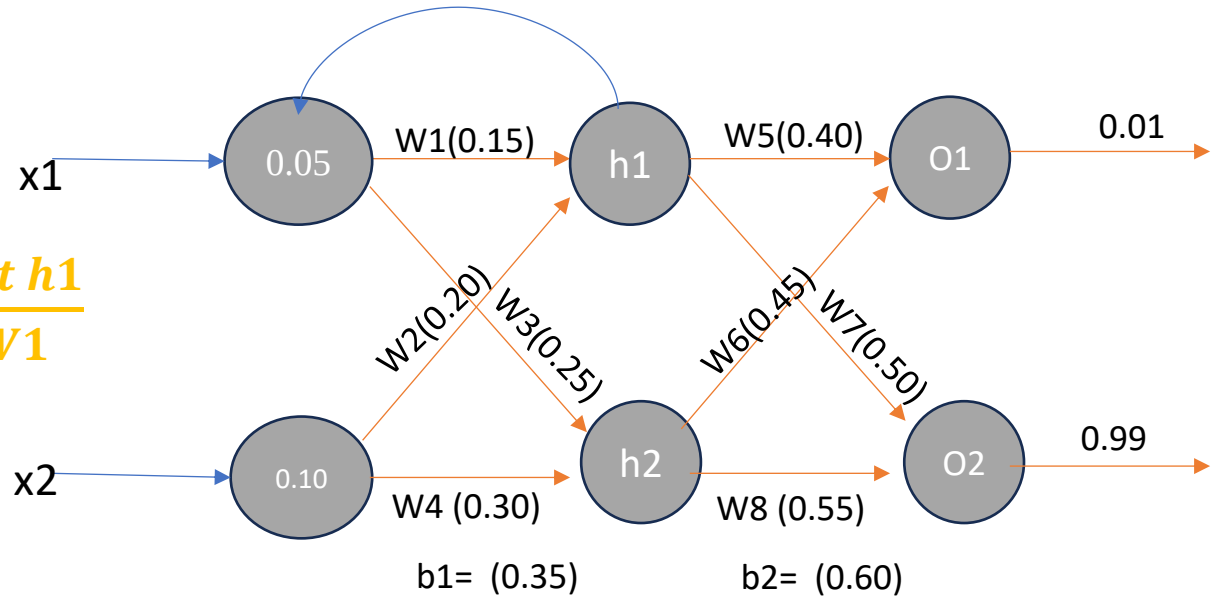
$$*W1 = W1 - n\frac{\partial Etotal}{\partial W1}$$

$$\frac{\partial Etotal}{\partial W1} = \frac{\partial Etotal}{\partial out\ h1} * \frac{\partial out\ h1}{\partial net\ h1} * \frac{\partial net\ h1}{\partial W1}$$

$$\frac{\partial Etotal}{\partial out\ h1} = \frac{\partial E\ O1}{\partial out\ h1} + \frac{\partial E\ O2}{\partial out\ h1}$$

$$\frac{\partial E\ O1}{\partial out\ h1} = \frac{\partial E\ O1}{\partial net\ O1} * \frac{\partial net\ O1}{\partial out\ O1}$$

$$\frac{\partial E\ O2}{\partial out\ h1} = \frac{\partial E\ O2}{\partial net\ O2} * \frac{\partial net\ O2}{\partial out\ h1}$$



MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate Backward propagation error (hidden layer to input layer)

W1,W2,W3,W4

$$\frac{\partial E\ O1}{\partial\ out\ h1} = \frac{\partial E\ O1}{\partial\ net\ O1} * \frac{\partial net\ O1}{\partial\ out\ h1}$$

$$\frac{\partial E\ O2}{\partial\ out\ h1} = \frac{\partial E\ O2}{\partial\ net\ O2} * \frac{\partial net\ O2}{\partial\ out\ h1}$$

$$\frac{\partial E\ O2}{\partial\ net\ O2} = \frac{\partial E\ O2}{\partial\ out\ O2} * \frac{\partial out\ O2}{\partial\ net\ O2}$$

$$\frac{\partial E\ O2}{\partial\ out\ O2} = (\text{out O2- target O2})$$

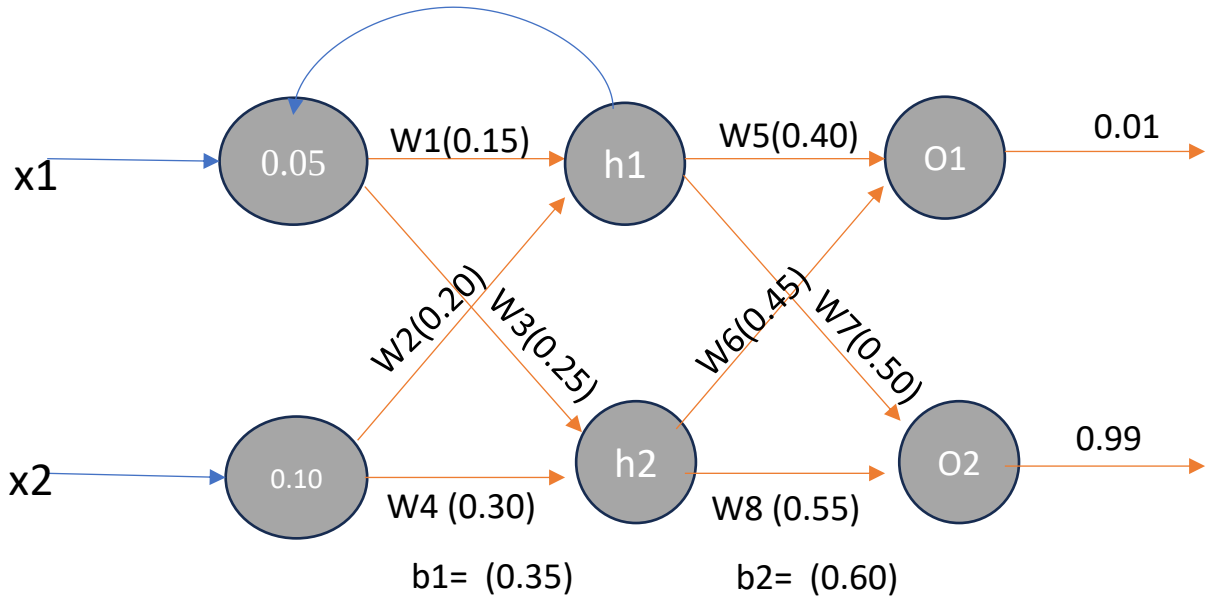
$$= 0.772928465\text{-}0.99$$

$$= -0.2170771535$$

$$\frac{\partial out\ O2}{\partial\ net\ O2} = \text{out O2} - (1\text{-out O2})$$

$$= 0.772928465 - (1- 0.772928465)$$

$$= 0.175510052$$



MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate Backward propagation error (hidden layer to input layer)

W1,W2,W3,W4

$$\frac{\partial E\ O1}{\partial\ out\ h1} = \frac{\partial E\ O1}{\partial\ net\ O1} * \frac{\partial net\ O1}{\partial\ out\ O1}$$

$$\frac{\partial E\ O2}{\partial\ out\ h1} = \frac{\partial E\ O2}{\partial\ net\ O2} * \frac{\partial net\ O2}{\partial\ out\ h1}$$

$$\frac{\partial E\ O2}{\partial\ net\ O2} = \frac{\partial E\ O2}{\partial\ out\ O2} * \frac{\partial out\ O2}{\partial\ net\ O2}$$

$$\frac{\partial E\ O2}{\partial\ out\ O2} = (\text{out O2- target O2})$$

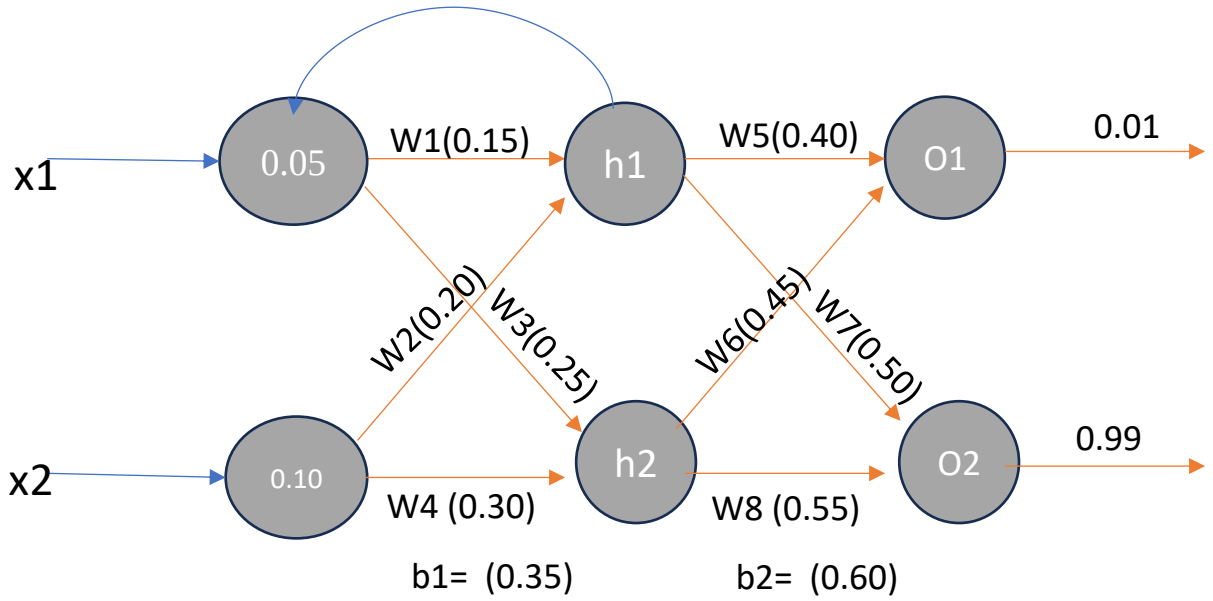
$$= 0.772928465-0.99$$

$$= -0.2170771535$$

$$\frac{\partial out\ O2}{\partial\ net\ O2} = \text{out O2} - (1-\text{out O2})$$

$$= 0.772928465 - (1- 0.772928465)$$

$$= 0.175510052$$



$$\frac{\partial E\ O2}{\partial\ net\ O2} = -0.2170771535 * 0.175510052$$

$$=$$

$$\frac{\partial net\ O2}{\partial\ out\ h1} = \text{on total of O2 from h1 => W7}$$

$$= 0.50$$

MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

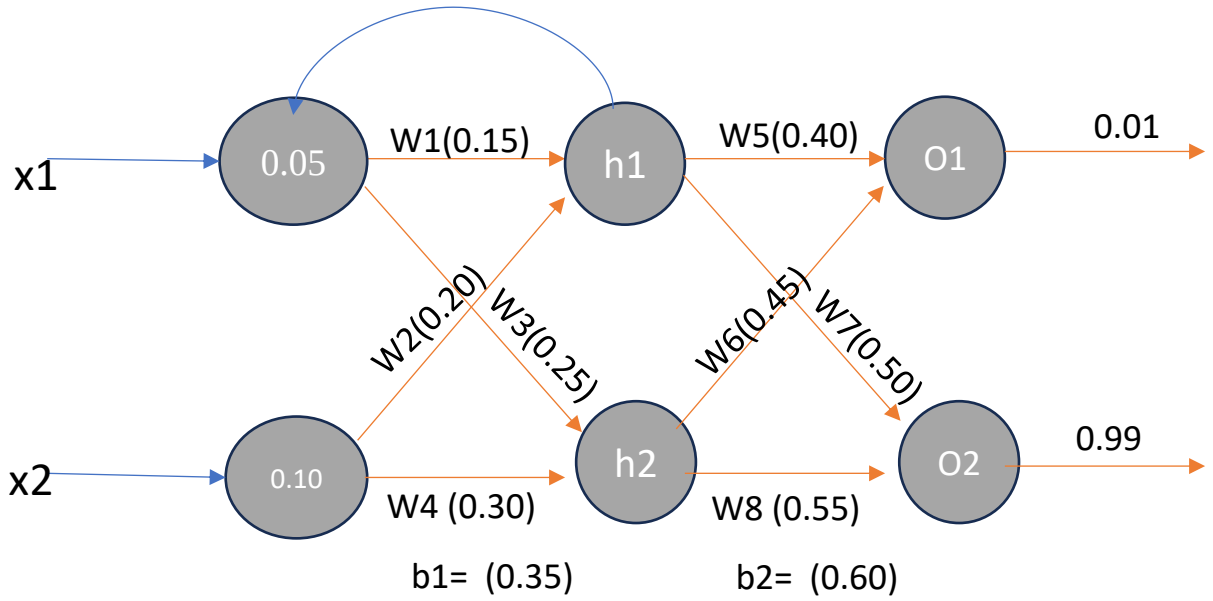
calculate Backward propagation error (hidden layer to input layer)

W1,W2,W3,W4

$$\frac{\partial E\ O1}{\partial\ out\ h1} = \frac{\partial E\ O1}{\partial\ net\ O1} * \frac{\partial net\ O1}{\partial\ out\ h1}$$

$$\frac{\partial E\ O2}{\partial\ out\ h1} = \frac{\partial E\ O2}{\partial\ net\ O2} * \frac{\partial net\ O2}{\partial\ out\ h1}$$

= -0.03809823 * 0.50



$$\frac{\partial E\ O2}{\partial\ net\ O2} = \frac{\partial E\ O2}{\partial\ out\ O2} * \frac{\partial out\ O2}{\partial\ net\ O2}$$

$$\frac{\partial E\ O2}{\partial\ out\ O2} = (\text{out O2- target O2})$$

= 0.772928465-0.99

= -0.2170771535

$$\frac{\partial out\ O2}{\partial\ net\ O2} = \text{out O2} - (1\text{-out O2})$$

= 0.772928465 – (1- 0.772928465)

= 0.175510052

$$\frac{\partial E\ O2}{\partial\ net\ O2} = -0.2170771535 * 0.175510052$$

= -0.03809823

$$\frac{\partial net\ O2}{\partial\ out\ h1} = \text{on total of O2 from h1 => W7}$$

= 0.50

MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate Backward propagation error (hidden layer to input layer)

W1,W2,W3,W4

$$\frac{\partial E\ O1}{\partial\ out\ h1} = \frac{\partial E\ O1}{\partial\ net\ O1} * \frac{\partial net\ O1}{\partial\ out\ h1}$$

$$= 0.37257738 * 0.4$$

$$\frac{\partial E\ O2}{\partial\ out\ h1} = \frac{\partial E\ O2}{\partial\ net\ O2} * \frac{\partial net\ O2}{\partial\ out\ h1}$$

$$\frac{\partial E\ O1}{\partial\ net\ O1} = \frac{\partial E\ O1}{\partial\ out\ O1} * \frac{\partial out\ O1}{\partial\ net\ O1}$$

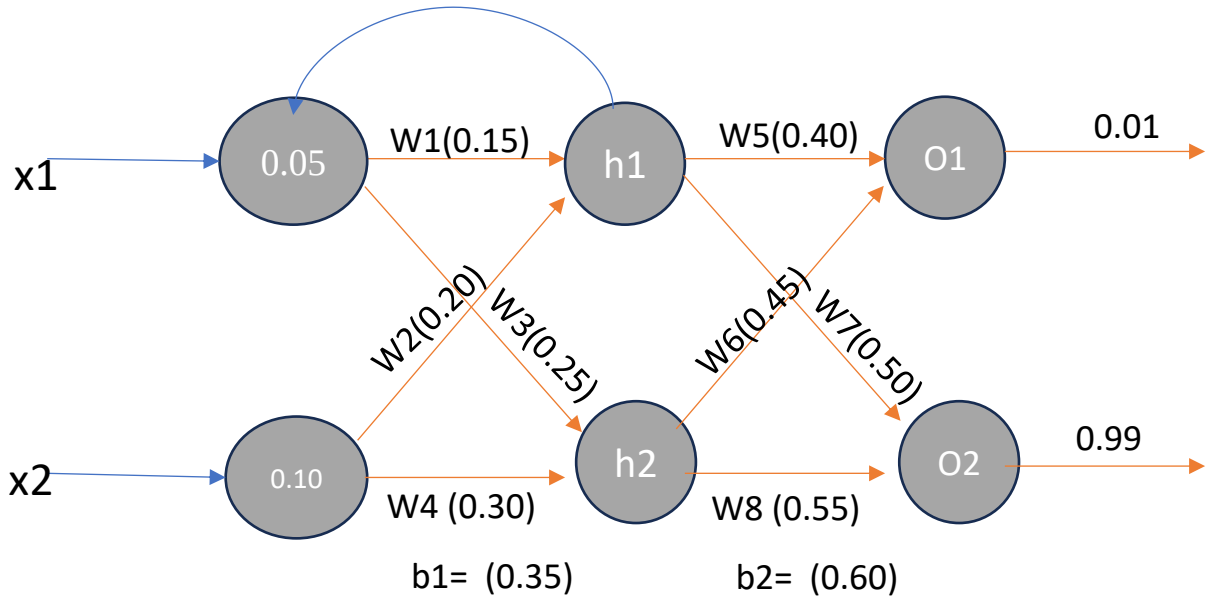
$$\frac{\partial E\ O1}{\partial\ out\ O1} = (\text{out O1} - \text{target O1})$$

$$= 0.7513 - 0.01 \qquad = 0.7413$$

$$\frac{\partial out\ O1}{\partial\ net\ O1} = \text{out O1} - (1 - \text{out O1})$$

$$= 0.7513 - (1 - 0.7513)$$

$$= 0.5026$$



$$\frac{\partial E\ O1}{\partial\ net\ O1} = 0.7413 * 0.5026$$

$$= 0.37257738$$

$$\frac{\partial net\ O1}{\partial\ out\ h1} = \text{on total of O1 from h1 => W5}$$

$$= 0.4$$

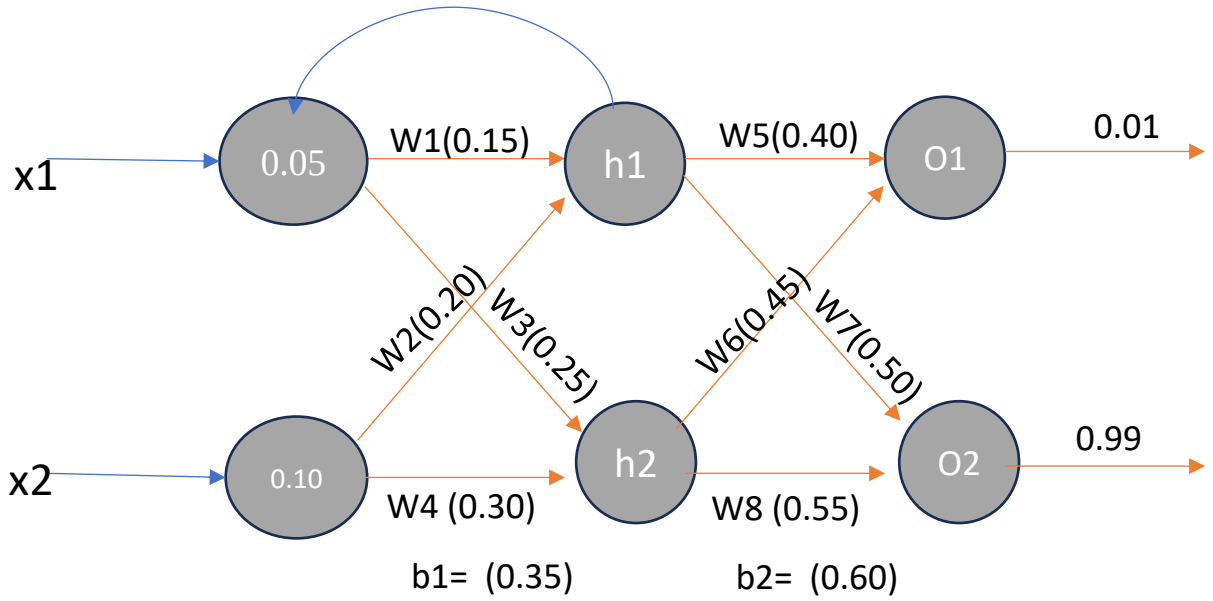
MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate Backward propagation error (hidden layer to input layer)

W1,W2,W3,W4



$$\frac{\partial E\ O1}{\partial\ out\ h1} = \frac{\partial E\ O1}{\partial\ net\ O1} * \frac{\partial net\ O1}{\partial\ out\ O1}$$

=0.37257738*0.4

$$\frac{\partial E\ O2}{\partial\ out\ h1} = \frac{\partial E\ O2}{\partial\ net\ O2} * \frac{\partial net\ O2}{\partial\ out\ h1}$$

= -0.03809823 * 0.50

$$\frac{\partial Etotal}{\partial\ W1} = \frac{\partial Etotal}{\partial\ out\ h1} * \frac{\partial\ out\ h1}{\partial\ net\ h1} * \frac{\partial\ net\ h1}{\partial\ W1}$$

$$\frac{\partial Etotal}{\partial\ out\ h1} = \frac{\partial E\ O1}{\partial\ out\ h1} + \frac{\partial E\ O2}{\partial\ out\ h1}$$

= (0.37257738*0.4) + (-0.03809823 * 0.50)

= 0.019049115+0.149030952

=0.168080067

$$\frac{\partial\ out\ h1}{\partial\ net\ h1}$$= out h1(1-out h1) =0.5932(1-0.5932)

=0.5932*0.4077 =0.24184764

$$\frac{\partial\ net\ h1}{\partial\ W1} = \frac{\partial}{\partial w1}(w1x1+w2x2+b1) = (x1+0+0)=0.05$$

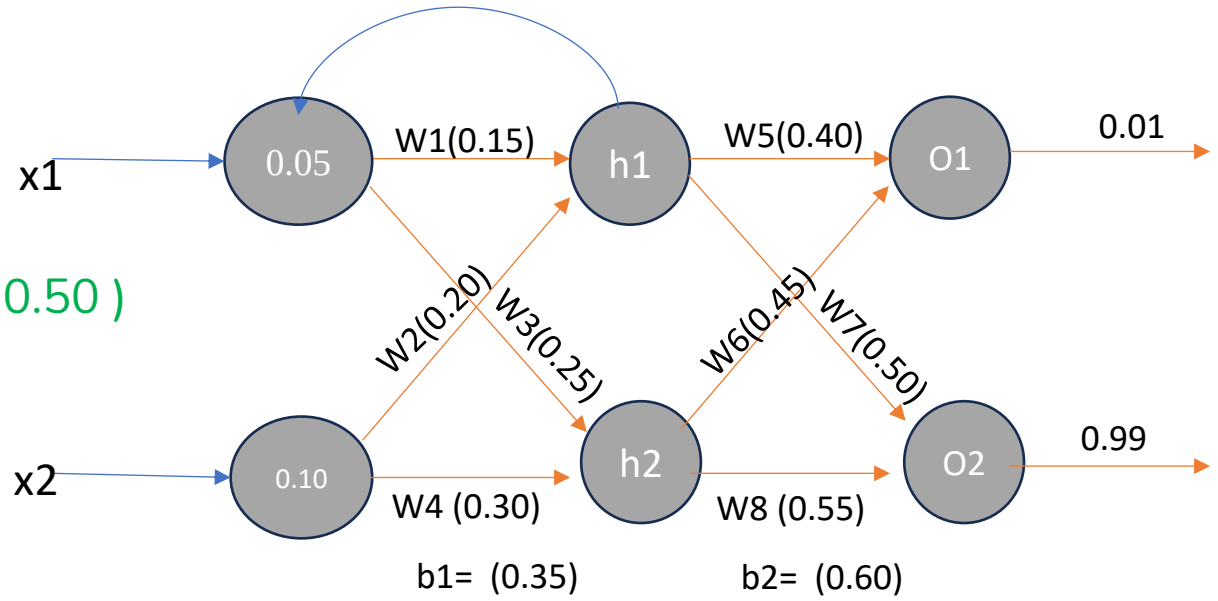
MALLA REDDY COLLEGE OF ENGINEERING

Working of Backpropagation Algorithm with an example

part 3:

calculate Backward propagation error (hidden layer to input layer)

W1,W2,W3,W4



$$\frac{\partial Etotal}{\partial out\ h1} = \frac{\partial E\ O1}{\partial out\ h1} + \frac{\partial E\ O2}{\partial out\ h1}$$

= (0.37257738*0.4) + (-0.03809823 * 0.50)

= 0.019049115+0.149030952

=0.168080067

$$\frac{\partial Etotal}{\partial W1} = \frac{\partial Etotal}{\partial out\ h1} * \frac{\partial out\ h1}{\partial net\ h1} * \frac{\partial net\ h1}{\partial W1}$$

= 0.168080067 * 0.24184764*0.05

= 0.00203248837

*w1 = $W1-n\frac{\partial Etotal}{\partial W1}$

$= 0.15 - (0.6 * 0.00203248837)$

=0.15-0.000304873=0.149695163

$\frac{\partial out\ h1}{\partial net\ h1}$= out h1(1-out h1) =0.5932(1-0.5932)

=0.5932*0.4077 =0.24184764

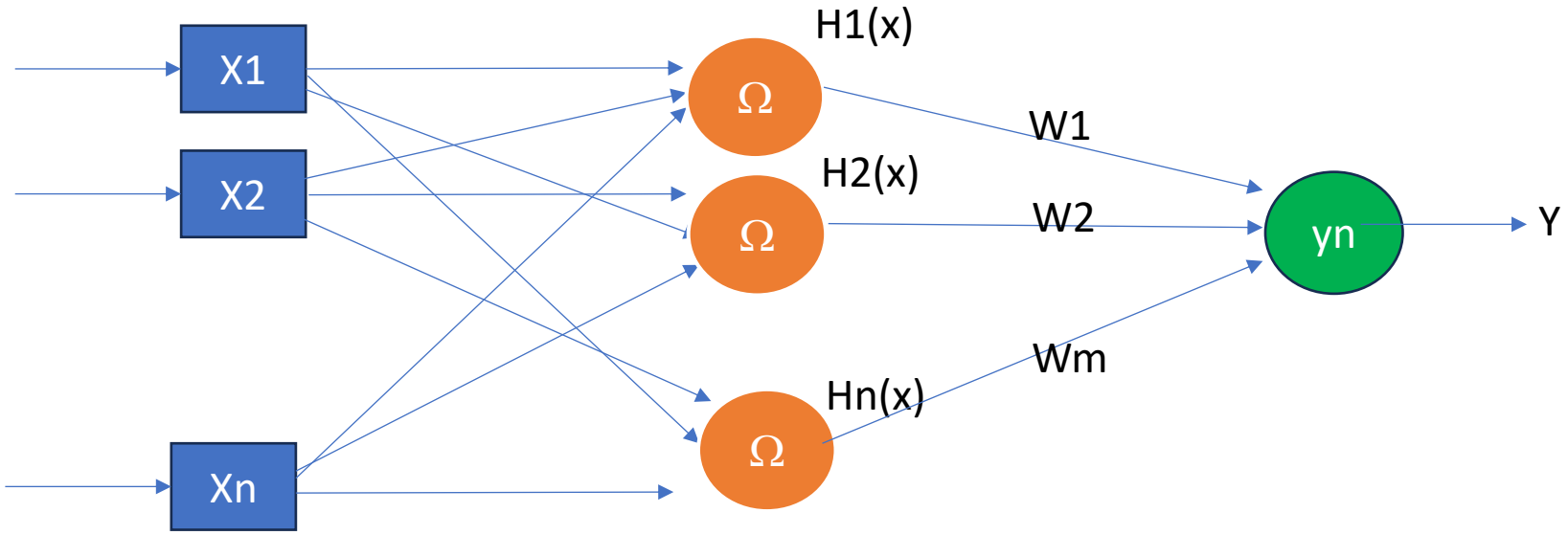
$\frac{\partial net\ h1}{\partial W1} = \frac{\partial}{\partial w1}$(w1x1+w2x2+b1) = (x1+0+0)=0.05

MALLA REDDY COLLEGE OF ENGINEERING

Radial Basis Functions and Splines – Concepts – RBF Network

Single layer perceptron can be used for classifying **linearly Separable** data
Multilayer perceptron can be used for classifying **non- linearly Separable** data

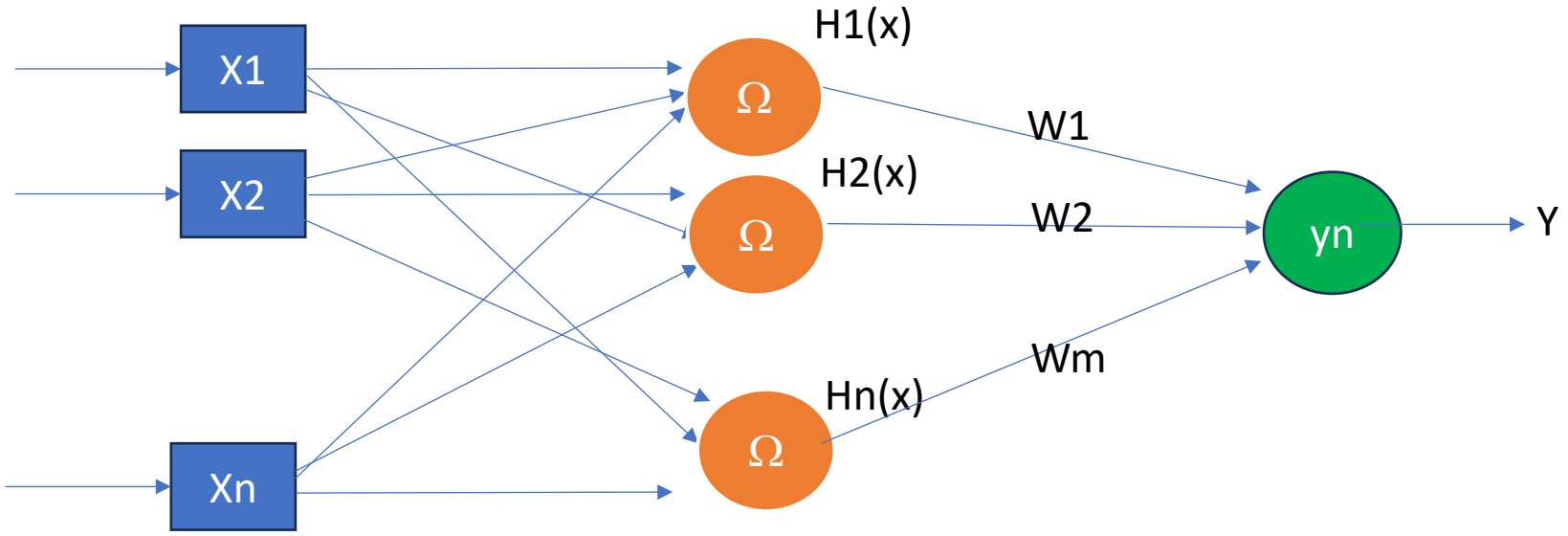
Radial Basis Functions is a type of Multilayer perceptron
which has one input layer one output layer with strictly one hidden layer



MALLA REDDY COLLEGE OF ENGINEERING

Radial Basis Functions and Splines – Concepts – RBF Network

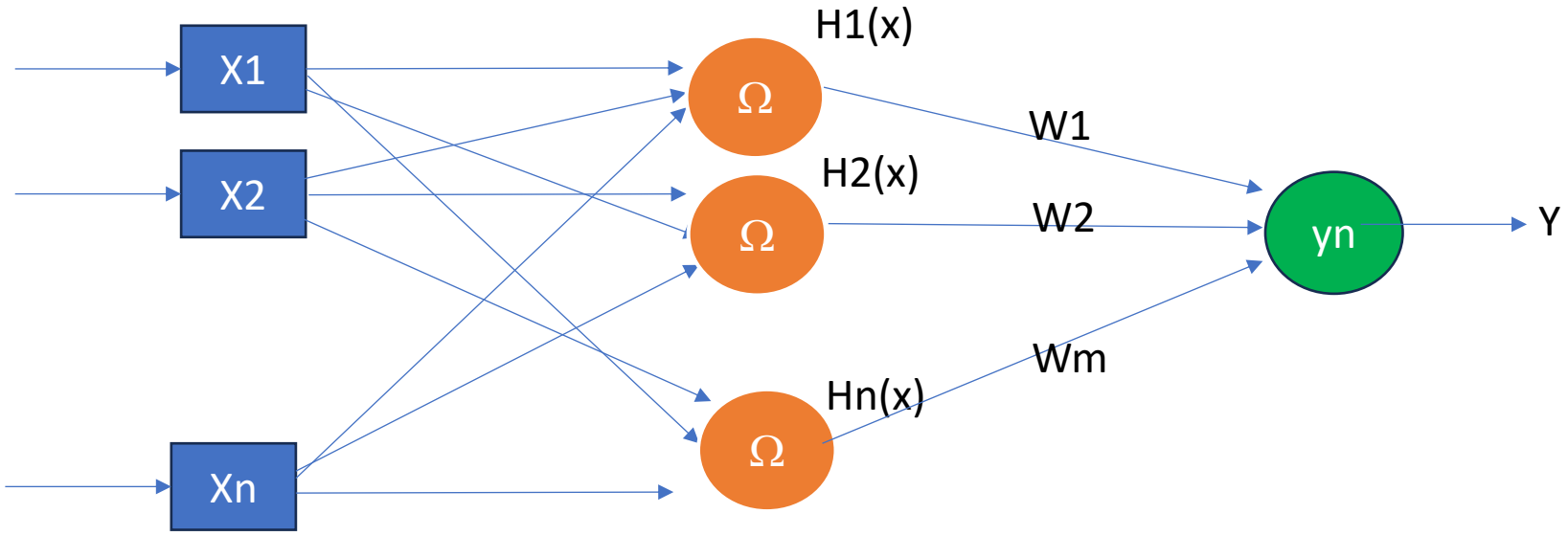
The hidden layer use a non – linear radial basis function as the **activation function**. Which converts the input parameter into high dimension space which is then fed into the network to linearly separate the problem.



MALLA REDDY COLLEGE OF ENGINEERING

Radial Basis Functions and Splines – Concepts – RBF Network

This cab be used in 1. Classification 2. interpolation 3. function approximation 4. Time series prediction 5. System control



MALLA REDDY COLLEGE OF ENGINEERINGA

- The Gaussian Radial basis function which monotonically decrease with distance from the centre.

$$H(x) = e^{\frac{-(x-r)^2}{r^2}}$$ where x = input

R = radius

C = centre

A multiquadratic RBF which monotonically increase with distance from the centre

$$H(x) = \sqrt{\frac{r^2+(x-c)^2}{r}}$$

MALLA REDDY COLLEGE OF ENGINEERINGA

- Algorithm for RBF

where input : input vector : x1,x2…….xn

Output : yn

Assign random weights for every connection from the hidden layer to the output layer in the network in the range [-1 to +1]

Forward phase :

Calculate input and output in the input layer (input layer as direct transfer function)

where the output of the nodes equals the input

Input at node I in the Input layer is **Ii=xi** where xi is the input received at node I

Output at node I “Oi” in the input layer is **Oi = Ii**

MALLA REDDY COLLEGE OF ENGINEERINGA

- Step2: for each node j in the hidden layer, find the center (c) and the variance r
  define hidden layer neurons with gaussian RB

$Hi(x) = e^{\frac{-(xi-Cj)^2}{r^2}}$ compute (x-cj) applying Euclidian distance measure between x and cj

Step 3 : for each node k in the output layer, compute linear weighted sum of the output of each neuron k from the hidden layer neurons j.

$f_k(x) = \sum_{j=1}^{m} Wij\ Hj(x)$ where Wij = weight in the link from the hidden layer neuron j to the output layer

H(x) is the **output of a hidden layer** neuron j for an input vector x

Backward phase :

1. Train the hidden layer using Back Propagation
2. update the weights between the hidden layer and output layer

MALLA REDDY COLLEGE OF ENGINEERING

RBF Networks are conceptually similar to K-Nearest Neighbor (k-NN) models, though their implementation is distinct. The fundamental idea is that an item's predicted target value is influenced by nearby items with similar predictor variable values. Here's how RBF Networks operate:

1.Input Vector: The network receives an n-dimensional input vector that needs classification or regression.

2.RBF Neurons: Each neuron in the hidden layer represents a prototype vector from the training set. The network computes the Euclidean distance between the input vector and each neuron's center.

3.Activation Function: The Euclidean distance is transformed using a Radial Basis Function (typically a Gaussian function) to compute the neuron's activation value. This value decreases exponentially as the distance increases.

4.Output Nodes: Each output node calculates a score based on a weighted sum of the activation values from all RBF neurons. For classification, the category with the highest score is chosen.

Key Characteristics of RBFs

•**Radial Basis Functions**: These are real-valued functions dependent solely on the distance from a central point. The Gaussian function is the most commonly used type.

•**Dimensionality**: The network's dimensions correspond to the number of predictor variables.

•**Center and Radius**: Each RBF neuron has a center and a radius (spread). The radius affects how broadly each neuron influences the input space.

Architecture of RBF Networks

The architecture of an RBF Network typically consists of three layers:

Input Layer

•**Function:** After receiving the input features, the input layer sends them straight to the hidden layer.

•**Components:** It is made up of the same number of neurons as the characteristics in the input data. One feature of the input vector corresponds to each neuron in the input layer.

Hidden Layer

•**Function:** This layer uses radial basis functions (RBFs) to conduct the non-linear transformation of the input data.

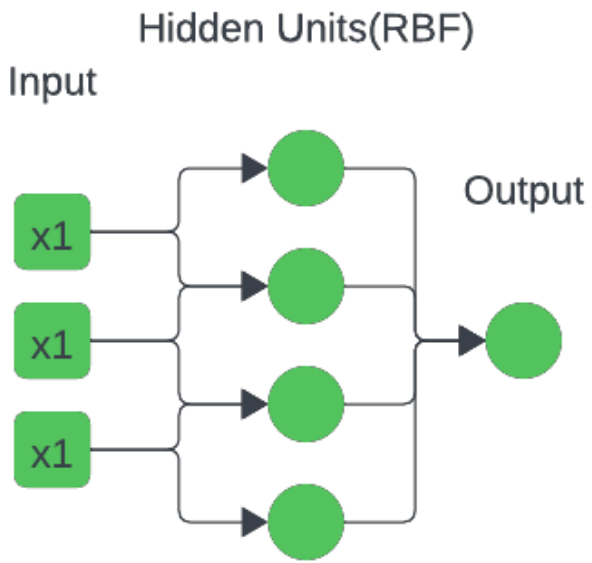
•**Components:** Neurons in the buried layer apply the RBF to the incoming data. The Gaussian function is the RBF that is most frequently utilized.

MALLA REDDY COLLEGE OF ENGINEERING

- **RBF Neurons:** Every neuron in the hidden layer has a spread parameter (σ) and a center, which are also referred to as prototype vectors. The spread parameter modulates the distance between the center of an RBF neuron and the input vector, which in turn determines the neuron's output.

Output Layer

- **Function:** The output layer uses weighted sums to integrate the hidden layer neurons' outputs to create the network's final output.
- **Components:** It is made up of neurons that combine the outputs of the hidden layer in a linear fashion. To reduce the error between the network's predictions and the actual target values, the weights of these combinations are changed during training.



MALLA REDDY COLLEGE OF ENGINEERING

Training Process of radial basis function neural network

An RBF neural network must be trained in three stages: choosing the center's, figuring out the spread parameters, and training the output weights.

Step 1: Selecting the Centers

- **Techniques for Centre Selection:** Centre's can be picked at random from the training set of data or by applying techniques such as k-means clustering.
- **K-Means Clustering:** The center's of these clusters are employed as the center's for the RBF neurons in this widely used center selection technique, which groups the input data into k groups.

Step 2: Determining the Spread Parameters

- **The spread parameter (σ)** governs each RBF neuron's area of effect and establishes the width of the RBF.
- **Calculation:** The spread parameter can be manually adjusted for each neuron or set as a constant for all neurons. Setting σ based on the separation between the center's is a popular method, frequently accomplished with the help of a heuristic like dividing the greatest distance between canters by the square root of twice the number of center's

Step 3: Training the Output Weights

- **Linear Regression:** The objective of linear regression techniques, which are commonly used to estimate the output layer weights, is to minimize the error between the anticipated output and the actual target values.
- **Pseudo-Inverse Method:** One popular technique for figuring out the weights is to utilize the pseudo-inverse of the hidden layer outputs matrix

Advantages of RBF Networks

MALLA REDDY COLLEGE OF ENGINEERING

1. **Universal Approximation**: RBF Networks can approximate any continuous function with arbitrary accuracy given enough neurons.
2. **Faster Learning**: The training process is generally faster compared to other neural network architectures.
3. **Simple Architecture**: The straightforward, three-layer architecture makes RBF Networks easier to implement and understand.

Applications of RBF Networks

- **Classification**: RBF Networks are used in pattern recognition and classification tasks, such as speech recognition and image classification.
- **Regression**: These networks can model complex relationships in data for prediction tasks.
- **Function Approximation**: RBF Networks are effective in approximating non-linear functions.

Example of RBF Network

Consider a dataset with two-dimensional data points from two classes. An RBF Network trained with 20 neurons will have each neuron representing a prototype in the input space. The network computes category scores, which can be visualized using 3-D mesh or contour plots. Positive weights are assigned to neurons belonging to the same category and negative weights to those from different categories. The decision boundary can be plotted by evaluating scores over a grid.

The **Curse of Dimensionality in Machine Learning** arises when working with high-dimensional data, leading to increased computational complexity, overfitting, and spurious correlations. Techniques like dimensionality reduction, feature selection, and careful model design are essential for mitigating its effects and improving algorithm performance. Navigating this challenge is crucial for unlocking the potential of high-dimensional datasets and ensuring robust machine-learning solutions.

What is the Curse of Dimensionality?

- The Curse of Dimensionality refers to the phenomenon where the efficiency and effectiveness of algorithms deteriorate as the dimensionality of the data increases exponentially.

MALLA REDDY COLLEGE OF ENGINEERING

- In high-dimensional spaces, data points become sparse, making it challenging to discern meaningful patterns or relationships due to the vast amount of data required to adequately sample the space.
- The Curse of Dimensionality significantly impacts machine learning algorithms in various ways. It leads to increased computational complexity, longer training times, and higher resource requirements. Moreover, it escalates the risk of overfitting and spurious correlations, hindering the algorithms' ability to generalize well to unseen data.

How to Overcome the Curse of Dimensionality?

To overcome the curse of dimensionality, you can consider the following strategies:

Dimensionality Reduction Techniques:

- Feature Selection: Identify and select the most relevant features from the original dataset while discarding irrelevant or redundant ones. This reduces the dimensionality of the data, simplifying the model and improving its efficiency.
- Feature Extraction: Transform the original high-dimensional data into a lower-dimensional space by creating new features that capture the essential information. Techniques such as Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) are commonly used for feature extraction.

Data Preprocessing:

- Normalization: Scale the features to a similar range to prevent certain features from dominating others, especially in distance-based algorithms.
- Handling Missing Values: Address missing data appropriately through imputation or deletion to ensure robustness in the model training process.

Feature Selection and Dimensionality Reduction

1. **Feature Selection:** `SelectKBest` is used to select the top k features based on a specified scoring function (`f_classif` in this case). It selects the features that are most likely to be related to the target variable.
2. **Dimensionality Reduction:** `PCA` (Principal Component Analysis) is then used to further reduce the dimensionality of the selected features. It transforms the data into a lower-dimensional space while retaining as much variance as possible.

Training the classifiers

1. **Training Before Dimensionality Reduction:** Train a Random Forest classifier (`clf_before`) on the original scaled features (`X_train_scaled`) without dimensionality reduction.

MALLA REDDY COLLEGE OF ENGINEERING

2. **Evaluation Before Dimensionality Reduction:** Make predictions (`y_pred_before`) on the test set (`X_test_scaled`) using the classifier trained before dimensionality reduction, and calculate the accuracy (`accuracy_before`) of the model.
3. **Training After Dimensionality Reduction:** Train a new Random Forest classifier (`clf_after`) on the reduced feature set (`X_train_pca`) after dimensionality reduction.
4. **Evaluation After Dimensionality Reduction:** Make predictions (`y_pred_after`) on the test set (`X_test_pca`) using the classifier trained after dimensionality reduction, and calculate the accuracy (`accuracy_after`) of the model.

Interpolation in Machine Learning

The practice of guessing unknown values based on available data points is known as interpolation in the context of machine learning. In tasks like regression and classification, where the objective is to predict outcomes based on input features, it is important. Machine learning algorithms are capable of producing well-informed predictions for unknown or intermediate values by interpolating between known data points.

Interpolation Types

The intricacy and applicability of interpolation techniques varied for various kinds of data. Typical forms of interpolation include the following:

- **Interpolation in Linear Form**: By assuming a linear relationship between neighboring data points, linear interpolation calculates values along a straight line that connects them.
- **Equation-Based Interpolation:** By fitting a polynomial function to the data points, polynomial interpolation produces a more flexible approximation that is capable of capturing nonlinear relationships.
- **Interpolation of Splines**: By building piece wise polynomial functions that connect data points gradually, spline interpolation prevents abrupt changes in the interpolated function.
- **Interpolation of Radial Basis Function**: Values based on the separations between data points are interpolated using radial basis functions in radial basis function interpolation.

Interpolation in Linear Form

A straightforward but efficient technique for guessing values between two known data points is linear interpolation.

MALLA REDDY COLLEGE OF ENGINEERING

The value of y at any intermediate point x can be approximated using the following formula, given two data points: ((1, 1) (x 1 ,y 1) and (2 , 2) (x 2 ,y 2). i.e y=y_1+(x−x_1)·(y_2−y_1)/x_2−x_1

Implementation

- This code snippet illustrates linear interpolation using **LinearNDInterpolator** from SciPy.
- It randomly generates 10 data points in 2D space with corresponding values.
- The **LinearNDInterpolator function** constructs an interpolation function based on these points. It then interpolates the value at a specified point and visualizes both the data points and the interpolated point on a scatter plot.
- Finally, the interpolated value at the specified point is printed.

Polynomial Interpolation

- Polynomial interpolation is a method of estimating values between known data points by fitting a polynomial function to the data. The goal is to find a polynomial that passes through all the given points.
- This method is useful for approximating functions that may not have a simple analytical form. One common approach to polynomial interpolation is to use the Lagrange polynomial or Newton's divided differences method to construct the interpolating polynomial.

Implementation

- This article demonstrates polynomial interpolation using the **interp1d function** from SciPy.
- It begins by generating sample data representing points along a sine curve. The interp1d function is then applied with a cubic spline interpolation method to approximate the curve between the data points.
- Finally, the original data points and the interpolated curve are visualized using matplotlib, showcasing the effectiveness of polynomial interpolation in approximating the underlying function from sparse data points.

MALLA REDDY COLLEGE OF ENGINEERING

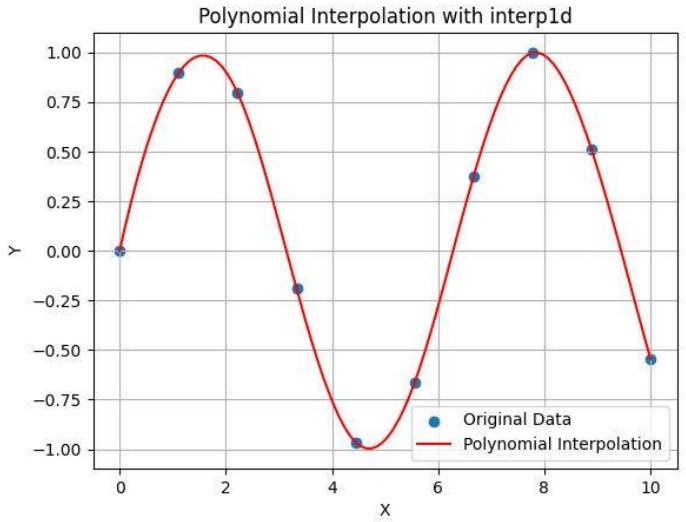
```
import numpy as np
from scipy.interpolate import interp1d import matplotlib.pyplot as plt

# Generate some sample data x = np.linspace(0, 10, 10)
y = np.sin(x)

# Perform polynomial interpolation poly_interp = interp1d(x, y,
kind='cubic')

# Generate points for plotting the interpolated curve x_interp =
np.linspace(0, 10, 100)
y_interp = poly_interp(x_interp)

# Plot the original data and the interpolated curve plt.scatter(x, y,
label='Original Data')
plt.plot(x_interp, y_interp, color='red', label='Polynomial Interpolation')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Polynomial Interpolation with interp1d') plt.legend()
plt.grid(True) plt.show()
```



MALLA REDDY COLLEGE OF ENGINEERING

Applications Of Interpolation in Machine Learning

Interpolation is a method used in various fields for estimating values between known data points. Some common applications of interpolation include:

- **Image Processing:** Interpolation is used to resize images by estimating the values of pixels in the resized image based on the values of neighboring pixels in the original image.
- **Computer Graphics:** In computer graphics, interpolation is used to generate smooth curves and surfaces, such as Bezier curves and surfaces, which are used to create shapes and animations.
- **Numerical Analysis:** Interpolation is used in numerical analysis to approximate the value of a function between two known data points. This is useful in areas such as finite element analysis and computational fluid dynamics.
- **Signal Processing:** In signal processing, interpolation is used to upsample signals, which increases the number of samples in a signal without changing its frequency content.
- **Mathematical Modeling:** Interpolation is used in mathematical modeling to estimate unknown values based on known data points, such as in the construction of mathematical models for physical systems.
- **Geographic Information Systems (GIS):** Interpolation is used in GIS to estimate values of geographical features, such as elevation or temperature, at locations where data is not available.
- **Audio Processing:** In audio processing, interpolation is used to resample audio signals

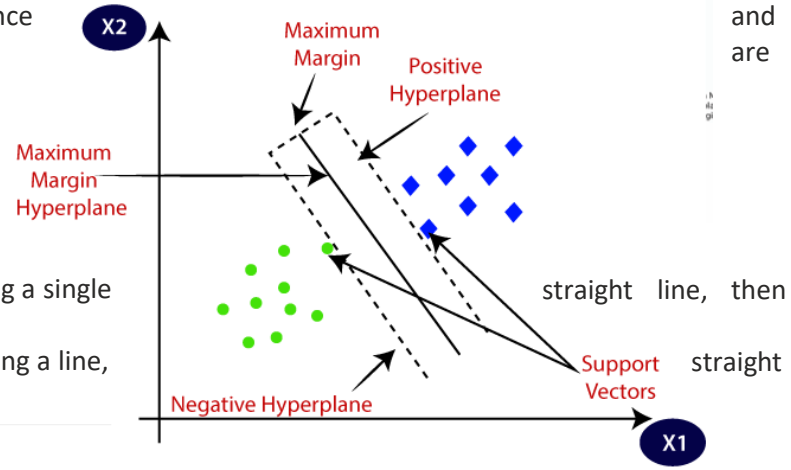
Support Vector Machine Algorithm

MALLA REDDY COLLEGE OF ENGINEERING

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, hence and algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



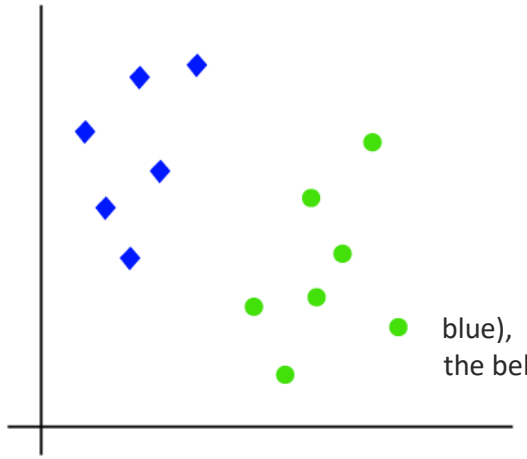
Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a line, straight then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Linear SVM:

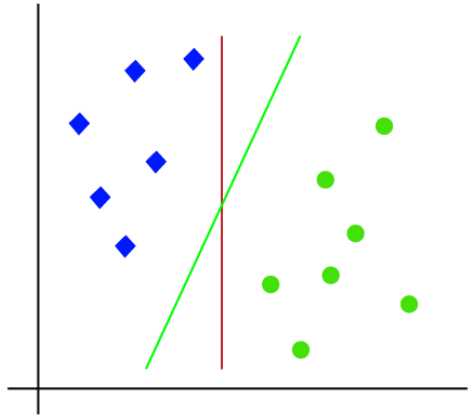
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and dataset blue), and has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the the below image:



MALLA REDDY COLLEGE OF ENGINEERING

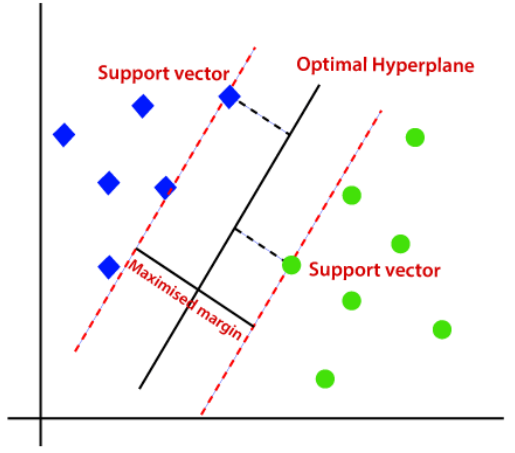
So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be

multiple lines that can separate these classes. Consider the below image:



the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin.

The **hyperplane** with maximum margin is called the **optimal hyperplane**.

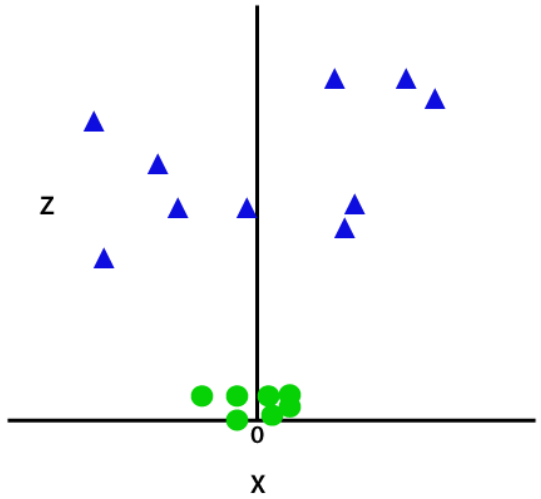


MALLA REDDY COLLEGE OF ENGINEERING

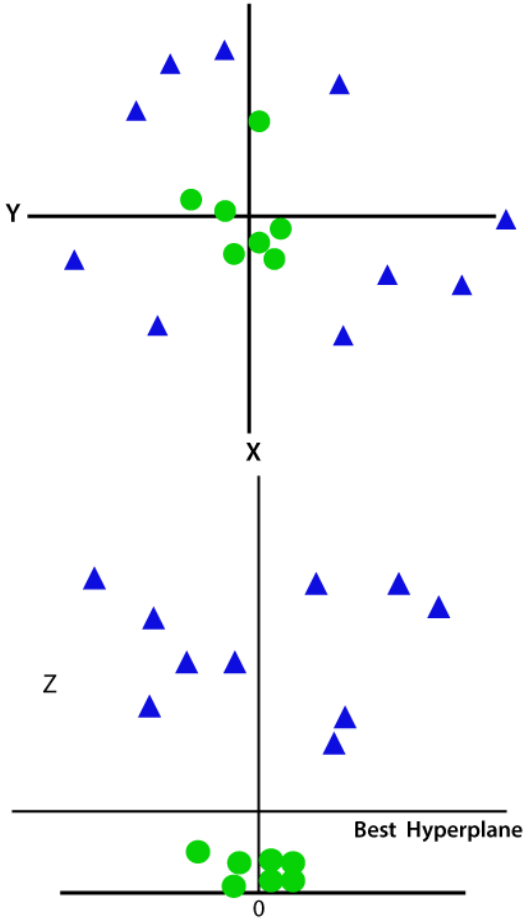
Non-Linear SVM:
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as: $z= x^2 + y^2$

By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will

MALLA REDDY COLLEGE OF ENGINEERING

become as:

Hence we get a circumference of radius 1 in case of non-linear data.