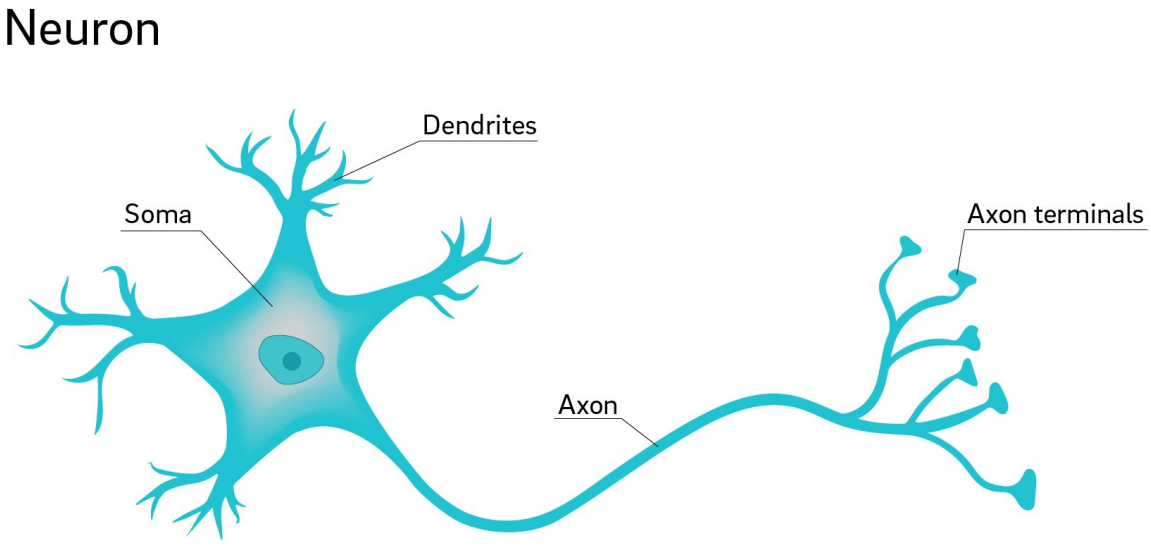
DEEP LEARNING (P-4)

techworldthink • March 11, 2022

11. (a) Describe the model of a biological neuron.

(b) Explain perceptron learning algorithm

In living organisms, the brain is the control unit of the neural network, and it has different subunits that take care of vision, senses, movement, and hearing. The brain is connected with a dense network of nerves to the rest of the body's sensors and actors. There are approximately 10^{11} neurons in the brain, and these are the building blocks of the complete central nervous system of the living body.



The neuron is the fundamental building block of neural networks. In the biological systems, a neuron is a cell just like any other cell of the body, which has a DNA code and is generated in the same way as the other cells. Though it might have different DNA, the function is similar in all the organisms. A neuron comprises three major parts: the cell body (also called Soma), the dendrites, and the axon. The dendrites

are like fibers branched in different directions and are connected to many cells in that cluster.

Dendrites receive the signals from surrounding neurons, and the axon transmits the signal to the other neurons. At the ending terminal of the axon, the contact with the dendrite is made through a synapse. Axon is a long fiber that transports the output signal as electric impulses along its length. Each neuron has one axon. Axons pass impulses from one neuron to another like a domino effect.

For creating mathematical models for artificial neural networks, theoretical analysis of biological neural networks is essential as they have a very close relationship. And this understanding of the brain's neural networks has opened horizons for the development of artificial neural network systems and adaptive systems designed to learn and adapt to the situations and inputs.

perceptron learning algorithm

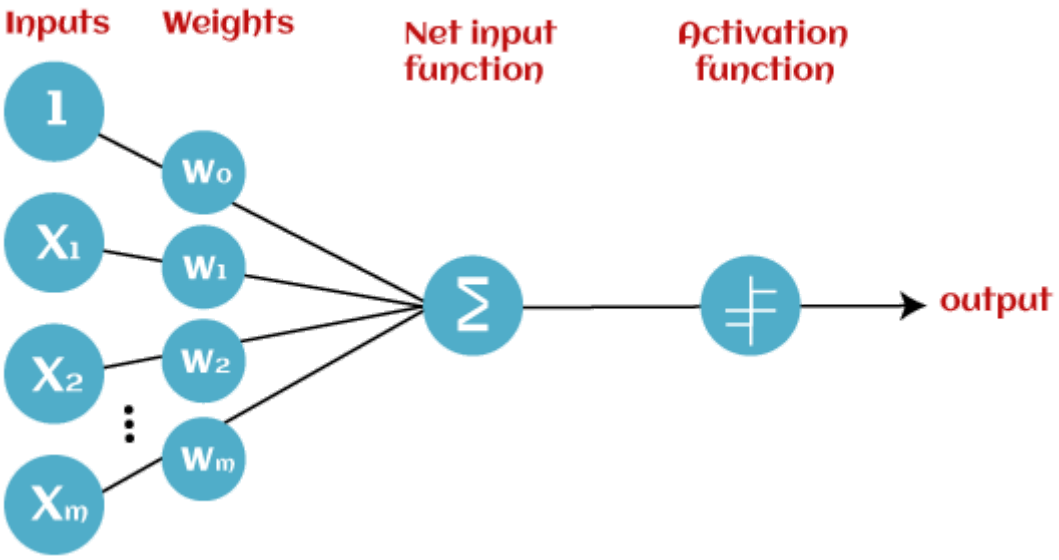
Perceptron is Machine Learning algorithm for supervised learning of various binary classification tasks. Further, *Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence*.

Perceptron model is also treated as one of the best and simplest types of Artificial Neural networks. However, it is a supervised learning algorithm of binary classifiers. Hence, we can consider it as a single-layer neural network with four main parameters, i.e., input values, weights and Bias, net sum, and an activation function.

In Machine Learning, binary classifiers are defined as the function that helps in deciding whether input data can be represented as vectors of numbers and belongs to some specific class.

Binary classifiers can be considered as linear classifiers. In simple words, we can understand it as a *classification algorithm that can predict linear predictor function in terms of weight and feature vectors.*

Mr. Frank Rosenblatt invented the perceptron model as a binary classifier which contains three main components. These are as follows:



Perceptron in Machine Learning

- Input Nodes or Input Layer:

This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.

- Wight and Bias:

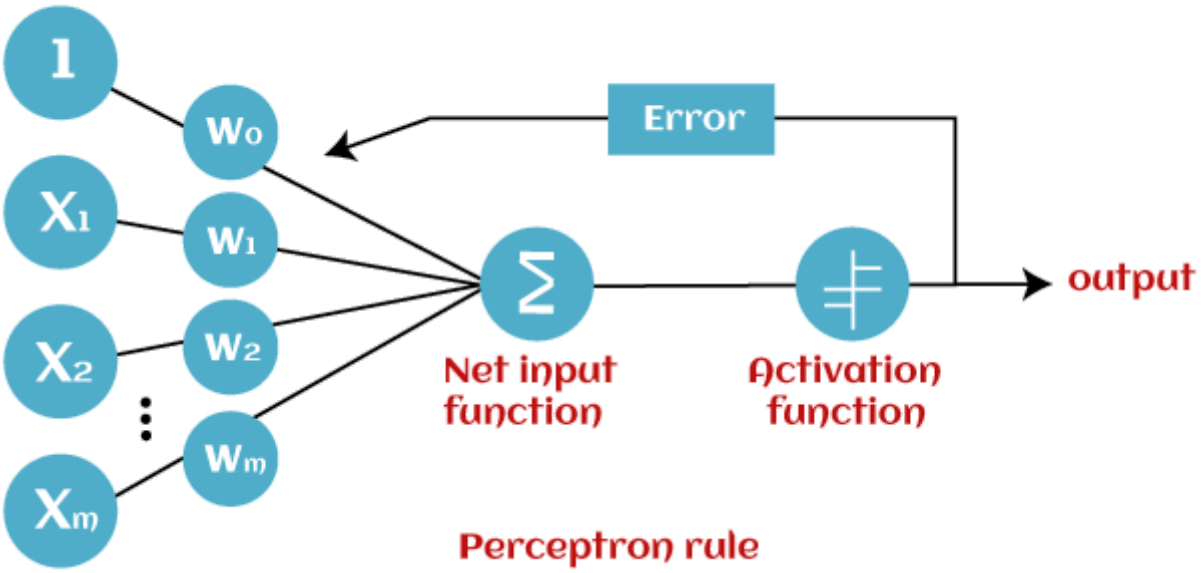
Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

- Activation Function:

These are the final and important components that help to determine whether the neuron will fire or not. Activation Function can be considered primarily as a step function.

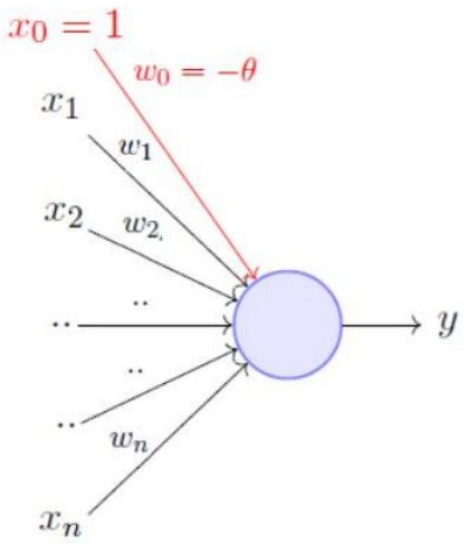
In Machine Learning, Perceptron is considered as a single-layer neural network that consists of four main parameters named input values (Input nodes), weights and Bias, net sum, and an activation function. The perceptron model begins with the multiplication of all input values and their weights, then adds these values together to create the weighted sum. Then this weighted sum is applied to the activation

function 'f' to obtain the desired output. This activation function is also known as the step function and is represented by 'f'.



Perceptron in Machine Learning

This step function or Activation function plays a vital role in ensuring that output is mapped between required values (0,1) or (-1,1). It is important to note that the weight of input is indicative of the strength of a node. Similarly, an input's bias value gives the ability to shift the activation function curve up or down.



A more accepted convention,

$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

$$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$$

Based on the layers, Perceptron models are divided into two types. These are as follows:

1. Single-layer Perceptron Model
2. Multi-layer Perceptron model

12. With a suitable example explain how backpropagation works

Backpropagation is one of the important concepts of a neural network. Our task is to classify our data best. For this, we have to update the weights of parameter and bias, but how can we do that in a deep neural network? In the linear regression model, we use gradient descent to optimize the parameter. Similarly here we also use gradient descent algorithm using Backpropagation.

For a single training example, Backpropagation algorithm calculates the gradient of the error function. Backpropagation can be written as a function of the neural network. Backpropagation algorithms are a set of methods used to efficiently train artificial neural networks following a gradient descent approach which exploits the chain rule.

The main features of Backpropagation are the iterative, recursive and efficient method through which it calculates the updated weight to improve the network until it is not able to perform the task for which it is being trained. Derivatives of the activation function to be known at network design time is required to Backpropagation

Backpropagation is the essence of neural network training. It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and make the model reliable by increasing its generalization.

Backpropagation in neural network is a short form for “backward propagation of errors.” It is a standard method of training artificial neural networks. This method helps calculate the gradient of a loss function with respect to all the weights in the network.

Back propagation is an algorithm for supervised learning of artificial neural networks using gradient descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network weights.

1) Initially the weights are assigned at random.

2) The algorithm iterates through main cycles of to process called

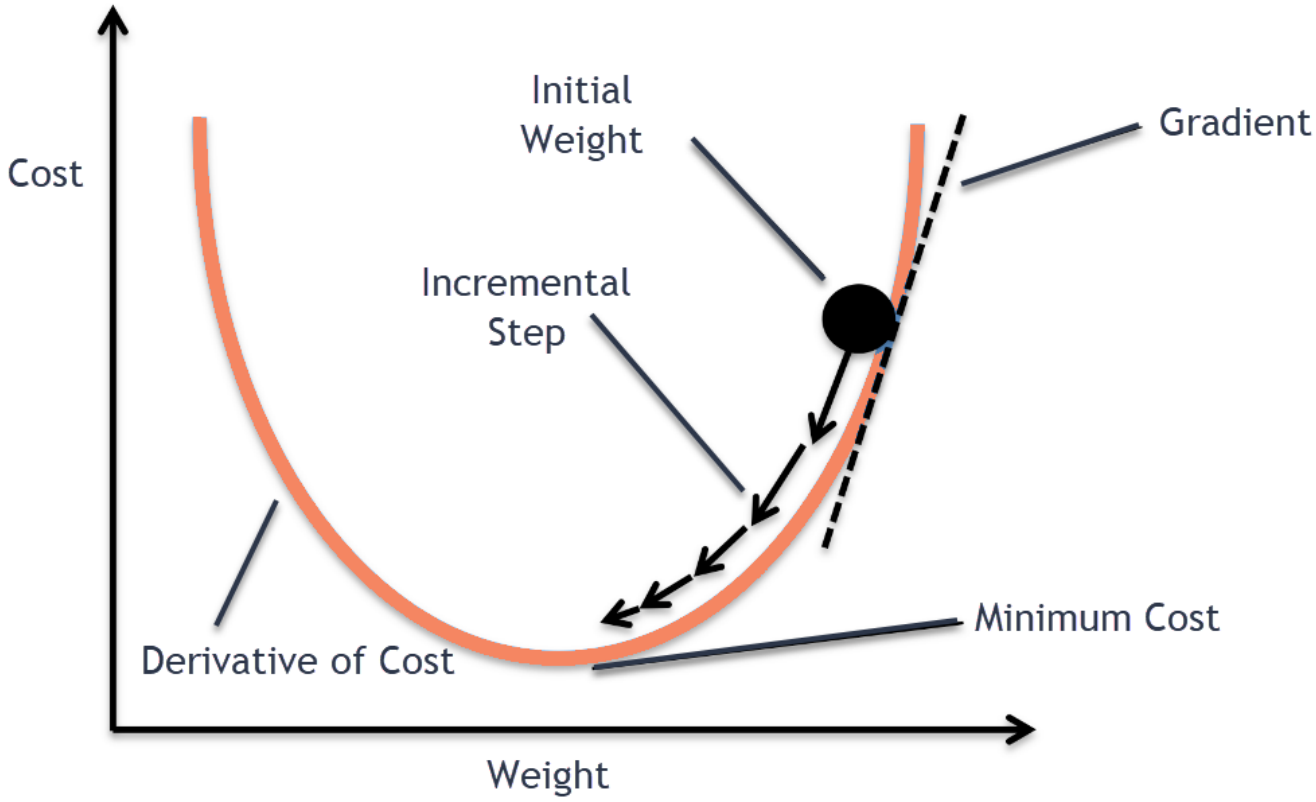
a) Forward phase

b) Backward phase

A Forward Phase : Neurons are activated in sequence from the input layer to the output layers. Applying neuron's weights and activation function along the way an output signal is produced from the final layer.

A Backward phase : Output signal combined with the true target value. Difference in results i.e.. error is propagated backward in the network to modify the connection weights

3) The technique used to determine how much a weight can be changed is known as Gradient Descent method.



At any point on the surface of the curve the gradient suggest how steeply the error will be reduced or increased for a change in the weight. Algorithm helps to change the weights that result in the greatest reduction in error .

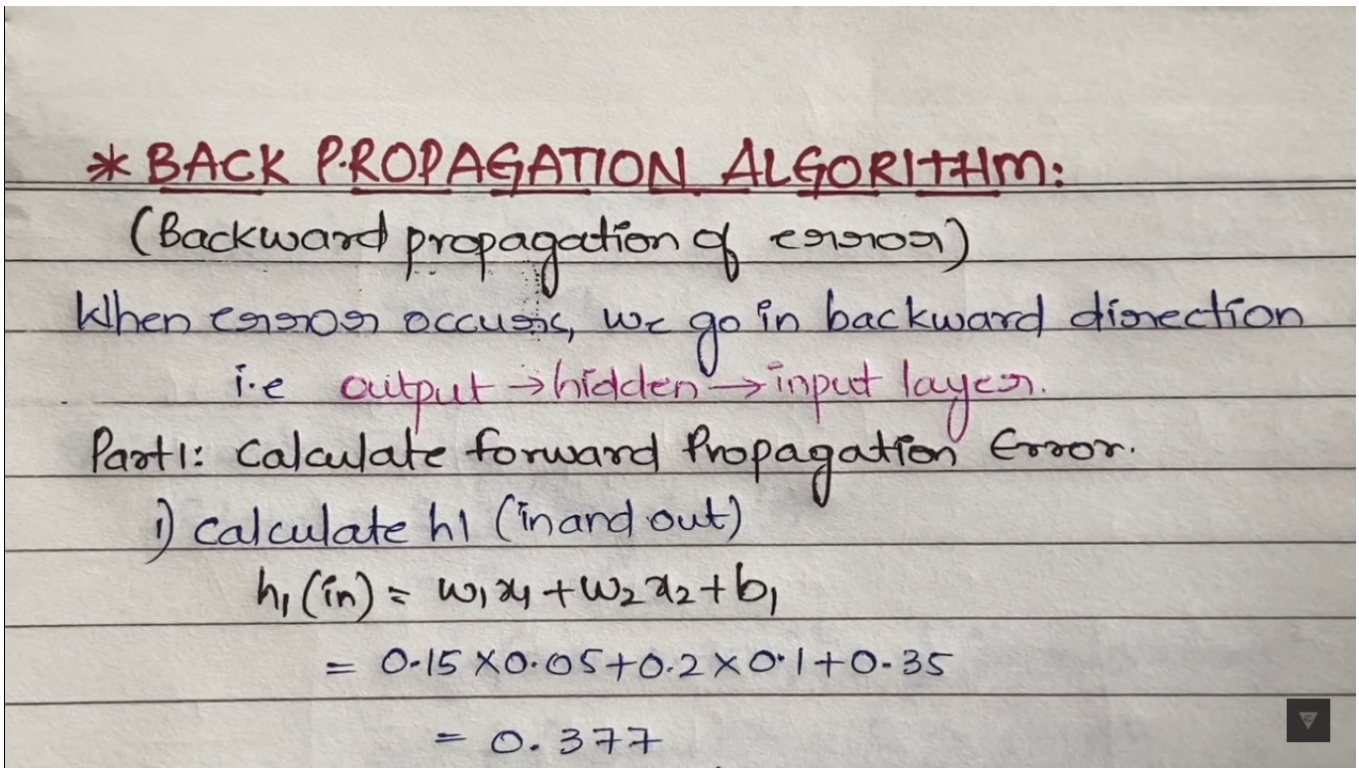
WHAT IS A GRADIENT?

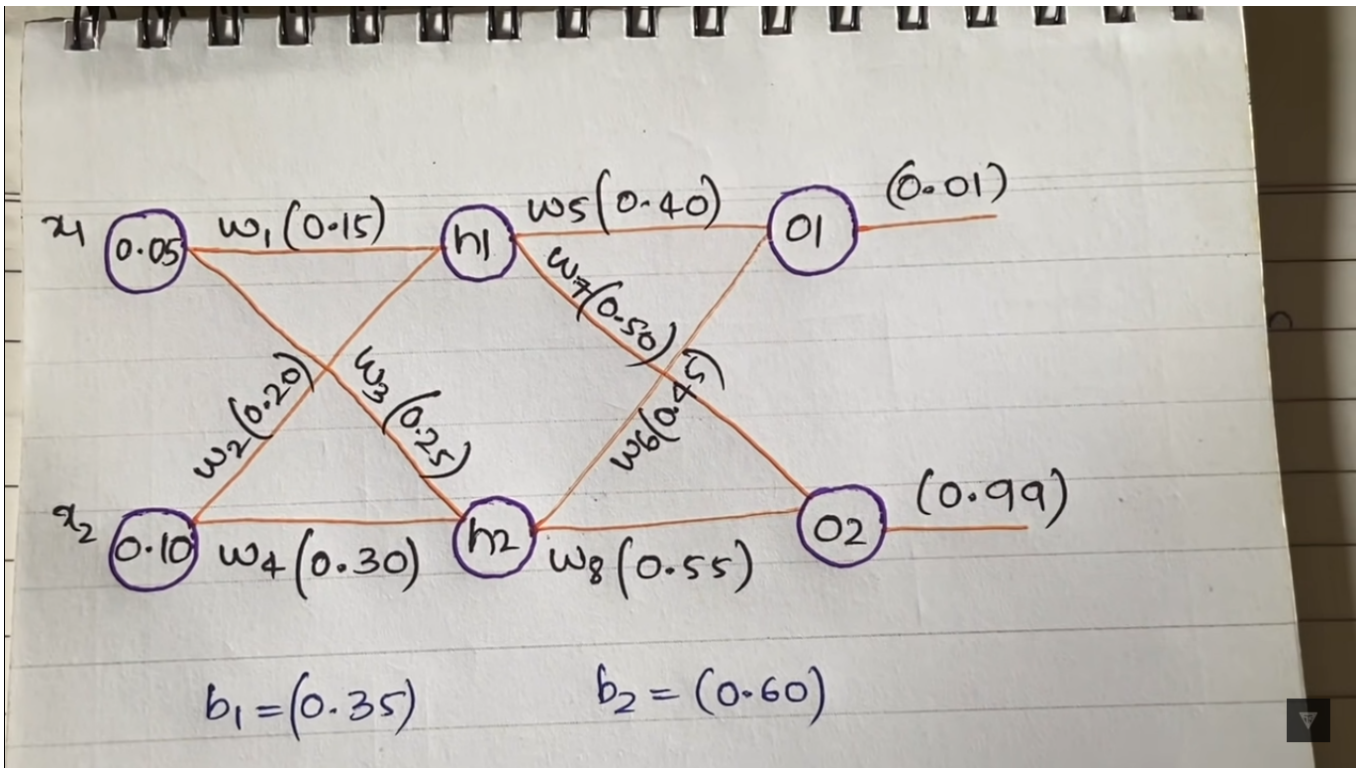
In machine learning, a gradient is a derivative of a function that has more than one input variable known as the slope of a function.In mathematical terms, the gradient simply measures the change in all weights with regard to the change in error.

Gradient Descent is an iterative optimization algorithm for finding the local minimum of a function.

To find the local minimum of a function using Gradient Descent, take steps proportional to the negative of the gradient(move away from the gradient) of the function at the current point.

It takes steps proportional to the positive of the gradient (moving towards the gradient), we will approach a local maximum of the function. This procedure is called gradient ascent.





(backward propagation of error)

When error occurs, we go in backward direction

i.e output → hidden → input layer.

Part1: Calculate forward Propagation Error.

1) Calculate h1 (in and out)

$$h_1(in) = w_1 x_1 + w_2 x_2 + b_1$$

$$= 0.15 \times 0.05 + 0.2 \times 0.1 + 0.35$$

$$= 0.377$$

$$h_1(out) = 1 / (1 + e^{-h_1(in)})$$

$$= \frac{1}{1 + e^{-(0.377)}} = 0.5932$$

2) Calculate h_2 (in and out)

$h_2(in) = x_1w_3 + x_2w_4 + b_1$

$= 0.05 \times 0.25 + 0.10 \times 0.3 + 0.35$ $\quad h_2(out) = 0.596$

$= 0.0125 + 0.03 + 0.35 = 0.3925$

3) Calculate o_1 (in and out)

$o_1(in) = h_1(out) \times w_5 + h_2(out) \times w_6 + b_2$

$= 0.593 \times 0.4 + 0.596 \times 0.45 + 0.6$

$= 1.105$

$o_1(out) = 1/(1+e^{-o_1(in)}) = 0.7513$

3) Calculate o_1 (in and out)

$o_1(in) = h_1(out) \times w_5 + h_2(out) \times w_6 + b_2$

$= 0.593 \times 0.4 + 0.596 \times 0.45 + 0.6$

$= 1.105$

$o_1(out) = 1/(1+e^{-o_1(in)}) = 0.7513$

4) Calculate o_2 (in and out)

$o_2(in) = h_1(out) \times w_7 + h_2(out) \times w_8 + b_2$

$= 0.5932 \times 0.50 + 0.5968 \times 0.55 + 0.6$

$= 1.22484$

$$O_2(out) = 1/(1+e^{-O_2(in)})$$
$$= 0.7729$$

5) calculate E_{Total}

$$E_{total} = \Sigma \frac{1}{2}(target - o(p))^2$$
$$= E_{01} + E_{02}$$
$$= \frac{1}{2}(0.01 - 0.7513)^2 + \frac{1}{2}(0.99 - 0.7729)^2$$
$$= \frac{1}{2}(-0.7413)^2 + \frac{1}{2}(0.2171)^2$$

Part: 2: Calculating backward propagation error (output layer → hidden layer)

w_5, w_6, w_7 and w_8.

first let us adjust w_5 (0.6)

$$w_5^* = w_5 - \eta \frac{\partial E_{total}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{01}} \times \frac{\partial out_{01}}{\partial net_{01}} \times \frac{\partial net_{01}}{\partial w_5}$$

$$\frac{\partial total}{\partial out_{01}} = out_{01} - target_{01} = 0.751365 - 0.01$$
$$= 0.7413565$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial total}{\partial out_{o1}} = out_{o1} - target_{o1} = 0.751365 - 0.01$$

$$= 0.7413565$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1})$$

$$= 0.751365(1 - 0.751365) = 0.186815602$$

$$\frac{\partial net_{o1}}{\partial w_5} = out_{h_1} = 0.59326992$$

likewsie update all the weight.....