BACK PROPAGATION LEARNING ALGORITHM

CO-2 SESSION 12

AIM

To familiarise students with the basic concept of Feed forward neural network based on Learning algorithms and to process of neural network in various Applications.

INSTRUCTIONAL OBJECTIVES

This unit is designed to:

1. Demonstrate Back Propagation overview
2. List out the activation functions of BPNN
3. Describe accuracy and loss functions

LEARNING OUTCOMES

At the end of this unit, you should be able to:

1. Define the functions BPNN algorithm
2. Summarise the development Back propagation neural network
3. Describe the various activation functions with loss values.

(DEEMED TO BE UNIVERSITY)

TOPICS TO BE COVERED

1. Back Propagation Overview
2. Back Propagation Algorithm
3. A Step By Step Back Propagation Example
   1. Gradient Descent Rule
   2. Delta learning Rule

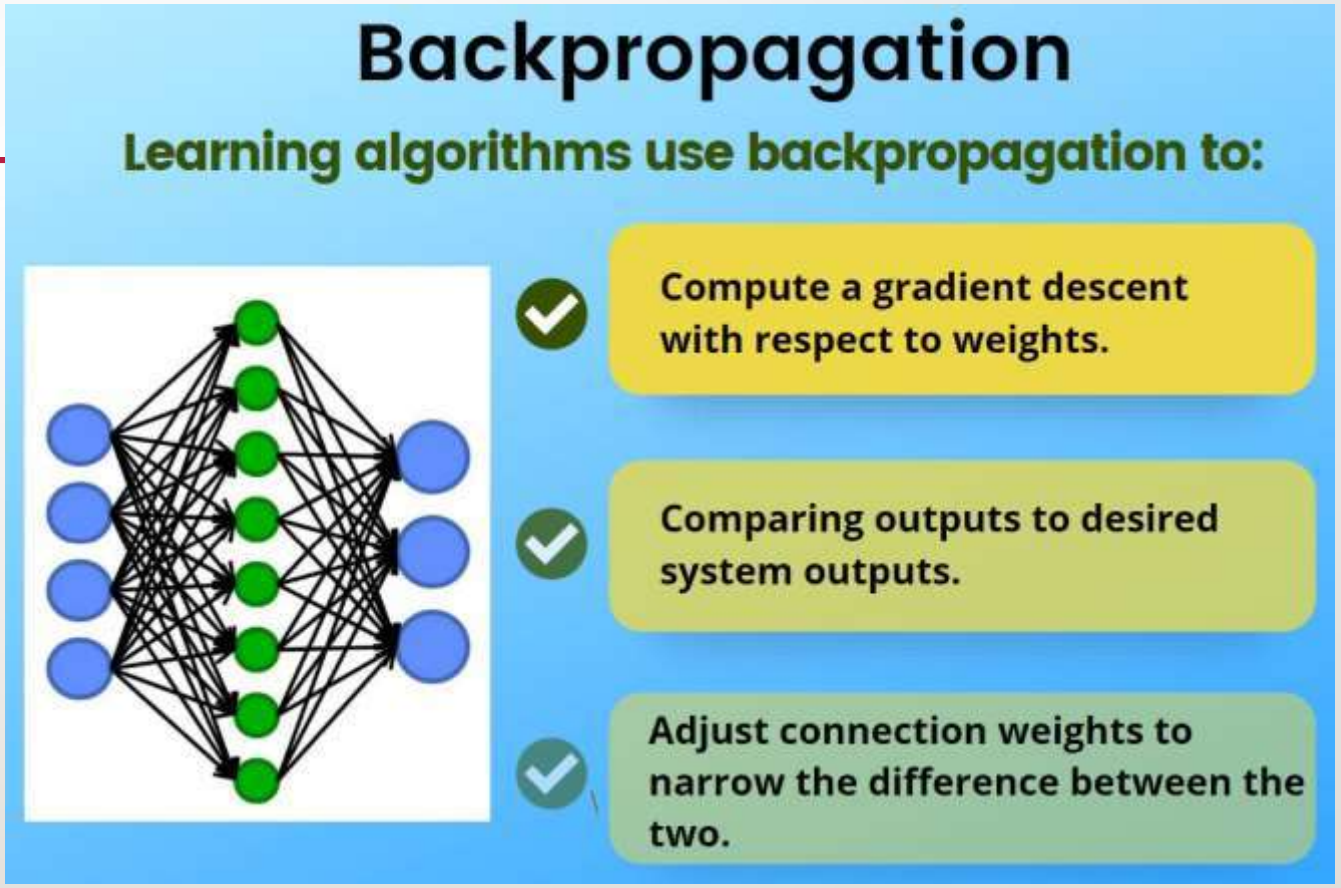
WHAT IS BACK PROPAGATION?

- Backpropagation is an algorithm used in artificial intelligence (AI) to fine-tune mathematical weight functions and improve the accuracy of an artificial neural network's outputs. Backpropagation is the process of tuning a neural network's weights to better the prediction accuracy.
- There are two directions in which information flows in a neural network. Forward propagation — also called inference — is when data goes into the neural network and out pops a prediction.
- A neural network can be thought of as a group of connected input/output (I/O) nodes. The level of accuracy each node produces is expressed as a loss function (error rate).

BACK PROPAGATION OVERVIEW

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.


Backpropagation
Learning algorithms use backpropagation to:
Compute a gradient descent with respect to weights.
Comparing outputs to desired system outputs.
Adjust connection weights to narrow the difference between the two.

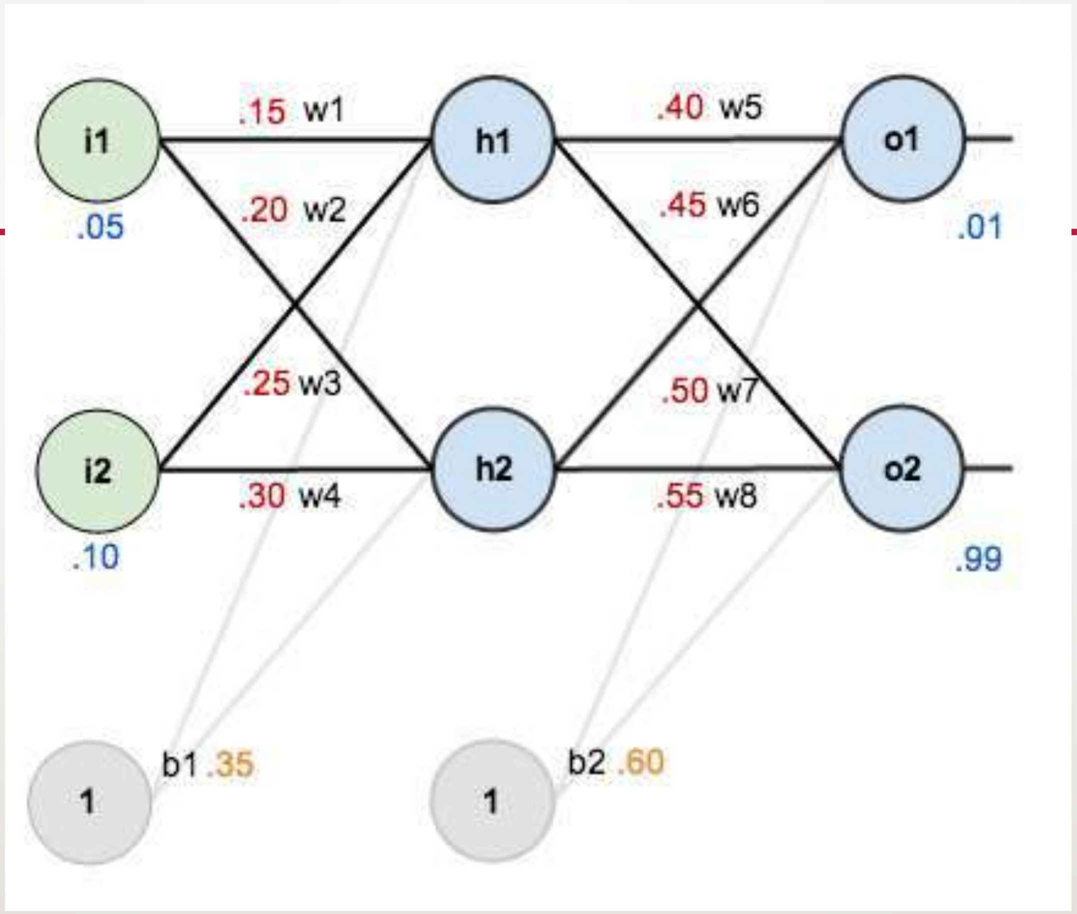
HOW BACKPROPAGATION WORKS: SIMPLE ALGORITHM

- Inputs X, arrive through the preconnected path
- Input is modeled using real weights W. The weights are usually randomly selected.
- Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
- Calculate the error in the outputs

  $\text{Error}_B = \text{Actual Output} - \text{Desired Output}$

- Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.
- Keep repeating the process until the desired output is achieved

A Step by Step BACK PROPAGATION Example



The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.
To work with a single training set: given inputs 0.05 and 0.10, we want the neural network to output 0.01 and 0.99.

BACK PROPAGATION WORKING METHOD

Calculating the Total Error

We can now calculate the error for each output neuron using the squared error function and sum them to get the total error:

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

For example, the target output for o_1 is 0.01 but the neural network output 0.75136507, therefore its error is:

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for o_2 (remembering that the target is 0.99) we get:

$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

The Backwards Pass

Our goal with backpropagation is to update each of the weights in the network so that they cause the actual output to be closer the target output, thereby minimizing the error for each output neuron and the network as a whole.

Output Layer

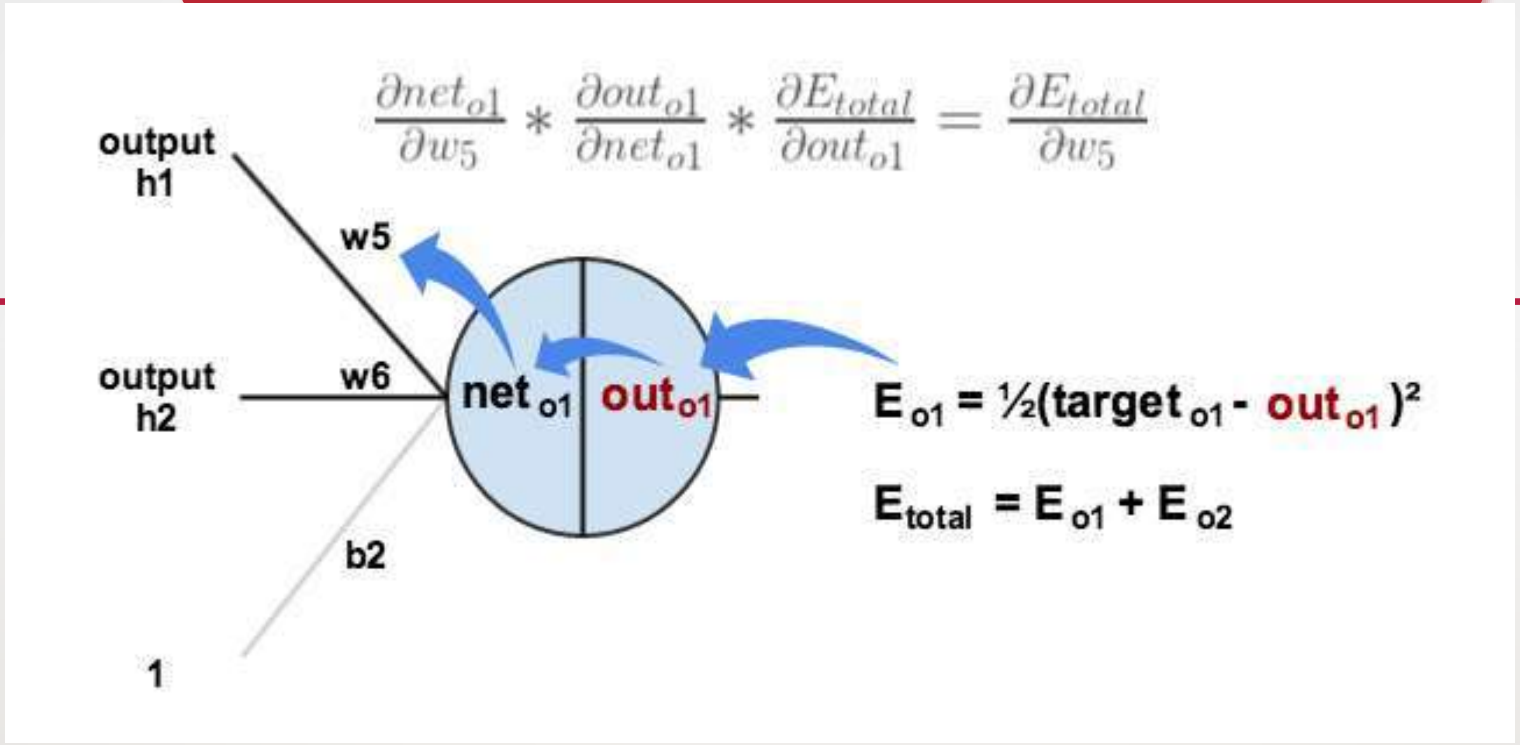
Consider w_5 we want to know how much a change in w_5 affects the total error, $\frac{\partial E_{total}}{\partial w_5}$

$\frac{\partial E_{total}}{\partial w_5}$ is read as "the partial derivative of E_{total} with respect to w_5". You can also say "the gradient with respect to w_5".

By applying the chain rule we know that:

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

BACK PROPAGATION



$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$
output h1
w5
output h2
w6
net o1
out o1
b2
1
$E_{o1} = ½(target_{o1} - out_{o1})^2$
$E_{total} = E_{o1} + E_{o2}$

1

2

3

We need to figure out each piece in this equation.
First, how much does the total error change with respect to the output?

$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

BACK PROPAGATION

When we take the partial derivative of the total error with respect to out_{o1}, the quantity $\frac{1}{2}(target_{o2} - out_{o2})^2$ becomes zero because out_{o1} does not affect it which means we're taking the derivative of a constant which is zero.

Next, how much does the output of o_1 change with respect to its total net input?
The partial derivative of the logistic function is the output multiplied by 1 minus the output:

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$$

Finally, how much does the total net input of $o1$ change with respect to w_5?

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$$

Putting it all together: $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

You'll often see this calculation combined in the form of the delta rule:

The Delta rule in machine learning and neural network environments is a specific type of backpropagation that helps to refine connectionist ML/AI networks, making connections between inputs and outputs with layers of artificial neurons.

In general, backpropagation has to do with recalculating input weights for artificial neurons using a gradient method. Delta learning does this using the difference between a target activation and an actual obtained activation. Using a linear activation function, network connections are adjusted.

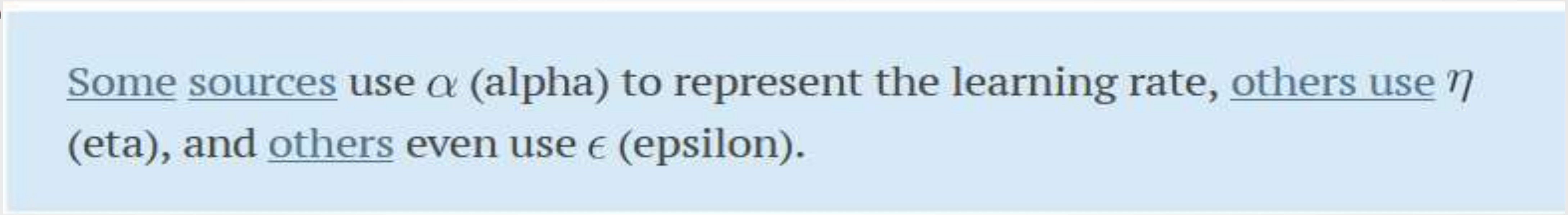
Another way to explain the Delta rule is that it uses an error function to perform gradient descent learning.

Apply the Delta rule

$$\frac{\partial E_{total}}{\partial w_5} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

To decrease the error, we then subtract this value from the current weight (optionally multiplied by some learning rate, eta, which we'll set to 0.5):

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$


Some sources use α (alpha) to represent the learning rate, others use η (eta), and others even use ϵ (epsilon).

We can repeat this process to get the new weights w_6 , w_7 and w_8

$$w_6^+ = 0.408666186$$

$$w_7^+ = 0.511301270$$

$$w_8^+ = 0.561370121$$

WEB REFERENCES

[1] https://www.guru99.com/backpropogation-neural-network.html

[2] https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

[3] http://neuralnetworksanddeeplearning.com/chap2.html

Self-Assessment Questions

1. What do the gradients of backpropagation compute?

(a) Profit Function
(b) Loss function
(c) Negative Function
(d) Positive Function

2. Which rule is followed by the Backpropagation algorithm?

(a) Static Rule
(b) Dynamic Rule
(c) Chain Rule
(d) None

THANK YOU

KL (DEEMED TO BE UNIVERSITY) | CSE COMPUTER SCIENCE & ENGINEERING

OUR TEAM