



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

AY: 2023-24

<b>Class:</b>	TE	<b>Semester:</b>	VI
<b>Course Code:</b>	CSL604	<b>Course Name:</b>	Machine Learning Lab

<b>Name of Student:</b>	Parth Manoj Raut
<b>Roll No.:</b>	44
<b>Experiment No.:</b>	8
<b>Title of the Experiment:</b>	Implementation of Error Backpropagation Perceptron Training Algorithm
<b>Date of Performance:</b>	
<b>Date of Submission:</b>	

### Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

### Checked by

Name of Faculty : Mr Raunak Joshi

Signature :

Date :



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

**Aim:** Implementation of Error Backpropagation Perceptron Training Algorithm

**Objective:** Able to design a neural network and use activation function as learning rule that converges using a backpropagation algorithm.

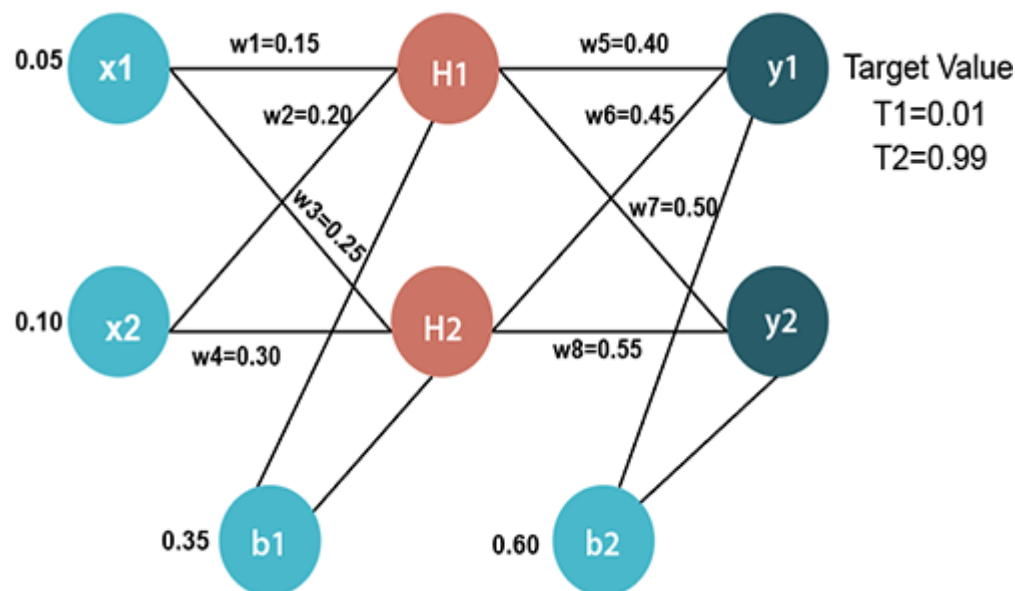
### Theory:

**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.

Now, how error function is used in Backpropagation and how Backpropagation works? Let start with an example and do it mathematically to understand how exactly updates the weight using Backpropagation.



Input values

$X_1=0.05$

$X_2=0.10$



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

### Initial weight

W1=0.15	w5=0.40
W2=0.20	w6=0.45
W3=0.25	w7=0.50
W4=0.30	w8=0.55

### Bias Values

b1=0.35    b2=0.60

### Target Values

T1=0.01

T2=0.99

Now, we first calculate the values of H1 and H2 by a forward pass.

### Forward Pass

To find the value of H1 we first multiply the input value from the weights as

$$\begin{aligned} H1 &= x1 \times w1 + x2 \times w2 + b1 \\ H1 &= 0.05 \times 0.15 + 0.10 \times 0.20 + 0.35 \\ \mathbf{H1} &= \mathbf{0.3775} \end{aligned}$$

To calculate the final result of H1, we performed the sigmoid function as

$$\begin{aligned} H1_{\text{final}} &= \frac{1}{1 + \frac{1}{e^{H1}}} \\ H1_{\text{final}} &= \frac{1}{1 + \frac{1}{e^{0.3775}}} \\ \mathbf{H1_{\text{final}} = 0.593269992} \end{aligned}$$

We will calculate the value of H2 in the same way as H1

$$\begin{aligned} H2 &= x1 \times w3 + x2 \times w4 + b1 \\ H2 &= 0.05 \times 0.25 + 0.10 \times 0.30 + 0.35 \\ \mathbf{H2} &= \mathbf{0.3925} \end{aligned}$$

To calculate the final result of H1, we performed the sigmoid function as



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

$$H2_{final} = \frac{1}{1 + \frac{1}{e^{H2}}}$$

$$H2_{final} = \frac{1}{1 + \frac{1}{e^{0.3925}}}$$

$$H2_{final} = 0.596884378$$

Now, we calculate the values of y1 and y2 in the same way as we calculate the H1 and H2.

To find the value of y1, we first multiply the input value i.e., the outcome of H1 and H2 from the weights as

$$y1 = H1 \times w_5 + H2 \times w_6 + b2$$
$$y1 = 0.593269992 \times 0.40 + 0.596884378 \times 0.45 + 0.60$$
$$y1 = 1.10590597$$

To calculate the final result of y1 we performed the sigmoid function as

$$y1_{final} = \frac{1}{1 + \frac{1}{e^{y1}}}$$
$$y1_{final} = \frac{1}{1 + \frac{1}{e^{1.10590597}}}$$
$$y1_{final} = 0.75136507$$

We will calculate the value of y2 in the same way as y1

$$y2 = H1 \times w_7 + H2 \times w_8 + b2$$
$$y2 = 0.593269992 \times 0.50 + 0.596884378 \times 0.55 + 0.60$$
$$y2 = 1.2249214$$

To calculate the final result of H1, we performed the sigmoid function as



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

$$y2_{\text{final}} = \frac{1}{1 + \frac{1}{e^{y2}}}$$

$$y2_{\text{final}} = \frac{1}{1 + \frac{1}{e^{1.2249214}}}$$

$$y2_{\text{final}} = \mathbf{0.772928465}$$

Our target values are 0.01 and 0.99. Our  $y1$  and  $y2$  value is not matched with our target values  $T1$  and  $T2$ .

Now, we will find the **total error**, which is simply the difference between the outputs from the target outputs. The total error is calculated as

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

So, the total error is

$$\begin{aligned} &= \frac{1}{2} (t1 - y1_{\text{final}})^2 + \frac{1}{2} (T2 - y2_{\text{final}})^2 \\ &= \frac{1}{2} (0.01 - 0.75136507)^2 + \frac{1}{2} (0.99 - 0.772928465)^2 \\ &= 0.274811084 + 0.0235600257 \\ &\mathbf{E_{\text{total}} = 0.29837111} \end{aligned}$$

Now, we will backpropagate this error to update the weights using a backward pass.

### Backward pass at the output layer

To update the weight, we calculate the error correspond to each weight with the help of a total error. The error on weight  $w$  is calculated by differentiating total error with respect to  $w$ .

$$\text{Error}_w = \frac{\partial E_{\text{total}}}{\partial w}$$

We perform backward process so first consider the last weight  $w5$  as



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

$$\text{Error}_{w5} = \frac{\partial E_{\text{total}}}{\partial w5} \dots \dots \dots (1)$$

$$E_{\text{total}} = \frac{1}{2} (T1 - y1_{\text{final}})^2 + \frac{1}{2} (T2 - y2_{\text{final}})^2 \dots \dots \dots (2)$$

From equation two, it is clear that we cannot partially differentiate it with respect to w5 because there is no any w5. We split equation one into multiple terms so that we can easily differentiate it with respect to w5 as

$$\frac{\partial E_{\text{total}}}{\partial w5} = \frac{\partial E_{\text{total}}}{\partial y1_{\text{final}}} \times \frac{\partial y1_{\text{final}}}{\partial y1} \times \frac{\partial y1}{\partial w5} \dots \dots \dots (3)$$

Now, we calculate each term one by one to differentiate  $E_{\text{total}}$  with respect to w5 as

$$\frac{\partial E_{\text{total}}}{\partial y1_{\text{final}}} = \frac{\partial (\frac{1}{2} (T1 - y1_{\text{final}})^2 + \frac{1}{2} (T2 - y2_{\text{final}})^2)}{\partial y1_{\text{final}}}$$

$$= 2 \times \frac{1}{2} \times (T1 - y1_{\text{final}})^{2-1} \times (-1) + 0$$

$$= -(T1 - y1_{\text{final}})$$

$$= -(0.01 - 0.75136507)$$

$$\frac{\partial E_{\text{total}}}{\partial y1_{\text{final}}} = 0.74136507 \dots \dots \dots (4)$$

$$y1_{\text{final}} = \frac{1}{1 + e^{-y1}} \dots \dots \dots (5)$$

$$\frac{\partial y1_{\text{final}}}{\partial y1} = \frac{\partial (\frac{1}{1 + e^{-y1}})}{\partial y1}$$

$$= \frac{e^{-y1}}{(1 + e^{-y1})^2}$$

$$= e^{-y1} \times (y1_{\text{final}})^2 \dots \dots \dots (6)$$

$$y1_{\text{final}} = \frac{1}{1 + e^{-y1}}$$

$$e^{-y1} = \frac{1 - y1_{\text{final}}}{y1_{\text{final}}} \dots \dots \dots (7)$$

Putting the value of  $e^{-y}$  in equation (5)



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

$$\begin{aligned}
 &= \frac{1 - y1_{final}}{y1_{final}} \times (y1_{final})^2 \\
 &= y1_{final} \times (1 - y1_{final}) \\
 &= 0.75136507 \times (1 - 0.75136507) \\
 \frac{\partial y1_{final}}{\partial y1} &= 0.186815602 \dots \dots \dots (8)
 \end{aligned}$$

$$y1 = H1_{final} \times w5 + H2_{final} \times w6 + b2 \dots \dots \dots (9)$$

$$\begin{aligned}
 \frac{\partial y1}{\partial w5} &= \frac{\partial (H1_{final} \times w5 + H2_{final} \times w6 + b2)}{\partial w5} \\
 &= H1_{final}
 \end{aligned}$$

$$\frac{\partial y1}{\partial w5} = 0.596884378 \dots \dots \dots (10)$$

So, we put the values of  $\frac{\partial E_{total}}{\partial y1_{final}}$ ,  $\frac{\partial y1_{final}}{\partial y1}$ , and  $\frac{\partial y1}{\partial w5}$  in equation no (3) to find the final result.

$$\begin{aligned}
 \frac{\partial E_{total}}{\partial w5} &= \frac{\partial E_{total}}{\partial y1_{final}} \times \frac{\partial y1_{final}}{\partial y1} \times \frac{\partial y1}{\partial w5} \\
 &= 0.74136507 \times 0.186815602 \times 0.593269992 \\
 \text{Error}_{w5} &= \frac{\partial E_{total}}{\partial w5} = 0.0821670407 \dots \dots \dots (11)
 \end{aligned}$$

Now, we will calculate the updated weight  $w5_{new}$  with the help of the following formula

$$\begin{aligned}
 w5_{new} &= w5 - \eta \times \frac{\partial E_{total}}{\partial w5} \text{ Here, } \eta = \text{learning rate} = 0.5 \\
 &= 0.4 - 0.5 \times 0.0821670407 \\
 w5_{new} &= 0.35891648 \dots \dots \dots (12)
 \end{aligned}$$

In the same way, we calculate  $w6_{new}$ ,  $w7_{new}$ , and  $w8_{new}$  and this will give us the following values

$$\begin{aligned}
 w5_{new} &= 0.35891648 \\
 w6_{new} &= 408666186 \\
 w7_{new} &= 0.511301270 \\
 w8_{new} &= 0.561370121
 \end{aligned}$$

### Backward pass at Hidden layer

Now, we will backpropagate to our hidden layer and update the weight  $w1$ ,  $w2$ ,  $w3$ , and  $w4$  as we have done with  $w5$ ,  $w6$ ,  $w7$ , and  $w8$  weights.



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

We will calculate the error at  $w_1$  as

$$\text{Error}_{w_1} = \frac{\partial E_{\text{total}}}{\partial w_1}$$

$$E_{\text{total}} = \frac{1}{2}(T_1 - y_{1\text{final}})^2 + \frac{1}{2}(T_2 - y_{2\text{final}})^2$$

From equation (2), it is clear that we cannot partially differentiate it with respect to  $w_1$  because there is no any  $w_1$ . We split equation (1) into multiple terms so that we can easily differentiate it with respect to  $w_1$  as

$$\frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial H_{1\text{final}}} \times \frac{\partial H_{1\text{final}}}{\partial H_1} \times \frac{\partial H_1}{\partial w_1} \dots \dots \dots (13)$$

Now, we calculate each term one by one to differentiate  $E_{\text{total}}$  with respect to  $w_1$  as

$$\frac{\partial E_{\text{total}}}{\partial H_{1\text{final}}} = \frac{\partial(\frac{1}{2}(T_1 - y_{1\text{final}})^2 + \frac{1}{2}(T_2 - y_{2\text{final}})^2)}{\partial H_1} \dots \dots \dots (14)$$

We again split this because there is no any  $H_1^{\text{final}}$  term in  $E^{\text{total}}$  as

$$\frac{\partial E_{\text{total}}}{\partial H_{1\text{final}}} = \frac{\partial E_1}{\partial H_{1\text{final}}} + \frac{\partial E_2}{\partial H_{1\text{final}}} \dots \dots \dots (15)$$

$\frac{\partial E_1}{\partial H_{1\text{final}}}$  and  $\frac{\partial E_2}{\partial H_{1\text{final}}}$  will again split because in  $E_1$  and  $E_2$  there is no  $H_1$  term. Splitting is done as

$$\frac{\partial E_1}{\partial H_{1\text{final}}} = \frac{\partial E_1}{\partial y_1} \times \frac{\partial y_1}{\partial H_{1\text{final}}} \dots \dots \dots (16)$$

$$\frac{\partial E_2}{\partial H_{1\text{final}}} = \frac{\partial E_2}{\partial y_2} \times \frac{\partial y_2}{\partial H_{1\text{final}}} \dots \dots \dots (17)$$

We again Split both  $\frac{\partial E_1}{\partial y_1}$  and  $\frac{\partial E_2}{\partial y_2}$  because there is no any  $y_1$  and  $y_2$  term in  $E_1$  and  $E_2$ . We split it as





# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

$$\frac{\partial E_1}{\partial y_1} = \frac{\partial E_1}{\partial y_{1\_final}} \times \frac{\partial y_{1\_final}}{\partial y_1} \dots \dots (18)$$

$$\frac{\partial E_2}{\partial y_2} = \frac{\partial E_2}{\partial y_{2\_final}} \times \frac{\partial y_{2\_final}}{\partial y_2} \dots \dots (19)$$

Now, we find the value of  $\frac{\partial E_1}{\partial y_1}$  and  $\frac{\partial E_2}{\partial y_2}$  by putting values in equation (18) and (19) as

From equation (18)

$$\begin{aligned} \frac{\partial E_1}{\partial y_1} &= \frac{\partial E_1}{\partial y_{1\_final}} \times \frac{\partial y_{1\_final}}{\partial y_1} \\ &= \frac{\partial \left( \frac{1}{2} (T_1 - y_{1\_final})^2 \right)}{\partial y_{1\_final}} \times \frac{\partial y_{1\_final}}{\partial y_1} \\ &= 2 \times \frac{1}{2} (T_1 - y_{1\_final}) \times (-1) \times \frac{\partial y_{1\_final}}{\partial y_1} \end{aligned}$$

From equation (8)

$$\begin{aligned} &= 2 \times \frac{1}{2} (0.01 - 0.75136507) \times (-1) \times 0.186815602 \\ \frac{\partial E_1}{\partial y_1} &= 0.138498562 \dots \dots (20) \end{aligned}$$

From equation (19)



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

$$\begin{aligned}\frac{\partial E_2}{\partial y_2} &= \frac{\partial E_2}{\partial y_{2\_final}} \times \frac{\partial y_{2\_final}}{\partial y_2} \\ &= \frac{\partial(\frac{1}{2}(T_2 - y_{2\_final})^2)}{\partial y_{2\_final}} \times \frac{\partial y_{2\_final}}{\partial y_2} \\ &= 2 \times \frac{1}{2}(T_2 - y_{2\_final}) \times (-1) \times \frac{\partial y_{2\_final}}{\partial y_2} \dots \dots \dots (21)\end{aligned}$$

$$y_{2\_final} = \frac{1}{1 + e^{-y_2}} \dots \dots \dots (22)$$

$$\begin{aligned}\frac{\partial y_{2\_final}}{\partial y_2} &= \frac{\partial(\frac{1}{1 + e^{-y_2}})}{\partial y_2} \\ &= \frac{e^{-y_2}}{(1 + e^{-y_2})^2} \\ &= e^{-y_2} \times (y_{2\_final})^2 \dots \dots \dots (23)\end{aligned}$$

$$y_{2\_final} = \frac{1}{1 + e^{-y_2}}$$

$$e^{-y_2} = \frac{1 - y_{2\_final}}{y_{2\_final}} \dots \dots \dots (24)$$

Putting the value of  $e^{-y_2}$  in equation (23)

$$\begin{aligned}&= \frac{1 - y_{2\_final}}{y_{2\_final}} \times (y_{2\_final})^2 \\ &= y_{2\_final} \times (1 - y_{2\_final}) \\ &= 0.772928465 \times (1 - 0.772928465) \\ \frac{\partial y_{2\_final}}{\partial y_2} &= 0.175510053 \dots \dots \dots (25)\end{aligned}$$

From equation (21)

$$\begin{aligned}&= 2 \times \frac{1}{2}(0.99 - 0.772928465) \times (-1) \times 0.175510053 \\ \frac{\partial E_1}{\partial y_1} &= -0.0380982366126414 \dots \dots \dots (26)\end{aligned}$$



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

Now from equation (16) and (17)

$$\begin{aligned}\frac{\partial E_1}{\partial H1_{final}} &= \frac{\partial E_1}{\partial y1} \times \frac{\partial y1}{\partial H1_{final}} \\ &= 0.138498562 \times \frac{\partial(H1_{final} \times w_5 + H2_{final} \times w_6 + b2)}{\partial H1_{final}} \\ &= 0.138498562 \times \frac{\partial(H1_{final} \times w_5 + H2_{final} \times w_6 + b2)}{\partial H1_{final}} \\ &= 0.138498562 \times w_5 \\ &= 0.138498562 \times 0.40\end{aligned}$$

$$\frac{\partial E_1}{\partial H1_{final}} = \mathbf{0.0553994248 \dots \dots \dots (27)}$$

$$\begin{aligned}\frac{\partial E_2}{\partial H1_{final}} &= \frac{\partial E_2}{\partial y2} \times \frac{\partial y2}{\partial H1_{final}} \\ &= -0.0380982366126414 \times \frac{\partial(H1_{final} \times w_7 + H2_{final} \times w_8 + b2)}{\partial H1_{final}} \\ &= -0.0380982366126414 \times w_7 \\ &= -0.0380982366126414 \times 0.50\end{aligned}$$

$$\frac{\partial E_2}{\partial H1_{final}} = \mathbf{-0.0190491183063207 \dots \dots \dots (28)}$$

Put the value of  $\frac{\partial E_1}{\partial H1_{final}}$  and  $\frac{\partial E_2}{\partial H1_{final}}$  in equation (15) as

$$\begin{aligned}\frac{\partial E_{total}}{\partial H1_{final}} &= \frac{\partial E_1}{\partial H1_{final}} + \frac{\partial E_2}{\partial H1_{final}} \\ &= 0.0553994248 + (-0.0190491183063207) \\ \frac{\partial E_{total}}{\partial H1_{final}} &= \mathbf{0.0364908241736793 \dots \dots \dots (29)}\end{aligned}$$

We have  $\frac{\partial E_{total}}{\partial H1_{final}}$ , we need to figure out  $\frac{\partial H1_{final}}{\partial H1}$ ,  $\frac{\partial H1}{\partial w1}$  as



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

$$\begin{aligned}\frac{\partial H1_{final}}{\partial H1} &= \frac{\partial \left( \frac{1}{1 + e^{-H1}} \right)}{\partial H1} \\ &= \frac{e^{-H1}}{(1 + e^{-H1})^2} \\ e^{-H1} \times (H1_{final})^2 \dots \dots \dots (30) \\ H1_{final} &= \frac{1}{1 + e^{-H1}}\end{aligned}$$

$$e^{-H1} = \frac{1 - H1_{final}}{H1_{final}} \dots \dots \dots (31)$$

Putting the value of  $e^{-H1}$  in equation (30)

$$\begin{aligned}&= \frac{1 - H1_{final}}{H1_{final}} \times (H1_{final})^2 \\ &= H1_{final} \times (1 - H1_{final}) \\ &= 0.593269992 \times (1 - 0.593269992) \\ \frac{\partial H1_{final}}{\partial H1} &= \mathbf{0.2413007085923199}\end{aligned}$$

We calculate the partial derivative of the total net input to H1 with respect to  $w1$  the same as we did for the output neuron:

$$\begin{aligned}H1 &= H1_{final} \times w5 + H2_{final} \times w6 + b2 \dots \dots \dots (32) \\ \frac{\partial y1}{\partial w1} &= \frac{\partial (x1 \times w1 + x2 \times w3 + b1 \times 1)}{\partial w1} \\ &= x1\end{aligned}$$

$$\frac{\partial H1}{\partial w1} = \mathbf{0.05 \dots \dots \dots (33)}$$

So, we put the values of  $\frac{\partial E_{total}}{\partial H1_{final}}$ ,  $\frac{\partial H1_{final}}{\partial H1}$ , and  $\frac{\partial H1}{\partial w1}$  in equation (13) to find the final result.

$$\begin{aligned}\frac{\partial E_{total}}{\partial w1} &= \frac{\partial E_{total}}{\partial H1_{final}} \times \frac{\partial H1_{final}}{\partial H1} \times \frac{\partial H1}{\partial w1} \\ &= 0.0364908241736793 \times 0.2413007085923199 \times 0.05 \\ \mathbf{Error_{w1}} &= \frac{\partial E_{total}}{\partial w1} = \mathbf{0.000438568 \dots \dots \dots (34)}\end{aligned}$$

Now, we will calculate the updated weight  $w1_{new}$  with the help of the following formula



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

$$w1_{\text{new}} = w1 - \eta \times \frac{\partial E_{\text{total}}}{\partial w1} \text{ Here } \eta = \text{learning rate} = 0.5$$

$$= 0.15 - 0.5 \times 0.000438568$$

$$w1_{\text{new}} = 0.149780716 \dots \dots (35)$$

In the same way, we calculate  $w2_{\text{new}}$ ,  $w3_{\text{new}}$ , and  $w4$  and this will give us the following values

$$w1_{\text{new}} = 0.149780716$$

$$w2_{\text{new}} = 0.19956143$$

$$w3_{\text{new}} = 0.24975114$$

$$w4_{\text{new}} = 0.29950229$$

We have updated all the weights. We found the error 0.298371109 on the network when we fed forward the 0.05 and 0.1 inputs. In the first round of Backpropagation, the total error is down to 0.291027924. After repeating this process 10,000, the total error is down to 0.0000351085. At this point, the outputs neurons generate 0.159121960 and 0.984065734 i.e., nearby our target value when we feed forward the 0.05 and 0.1.

### Implementation:

```
1  import numpy as np
2  import pandas as pd
3  from sklearn.datasets import load_iris
4  from sklearn.model_selection import train_test_split
5  import matplotlib.pyplot as plt
6
7  # Loading dataset
8  data = load_iris()
9  X = data.data
10 y = data.target
11
12 # Split dataset into training and test sets
13 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=20, random_state=4)
14
15 # Hyperparameters
16 learning_rate = 0.1
17 iterations = 5000
18 N = y_train.size
19 input_size = 4
20 hidden_size = 2
21 output_size = 3
22
23 np.random.seed(10)
24 W1 = np.random.normal(scale=0.5, size=(input_size, hidden_size))
25 W2 = np.random.normal(scale=0.5, size=(hidden_size, output_size))
26
27 # Helper functions
28
29 def sigmoid(x):
30     return 1 / (1 + np.exp(-x))
31
```



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

```
32 def mean_squared_error(y_pred, y_true):
33     # One-hot encode y_true (i.e., convert [0, 1, 2] into [[1, 0, 0], [0, 1, 0], [0, 0, 1]])
34     y_true_one_hot = np.eye(output_size)[y_true]
35
36     # Reshape y_true_one_hot to match y_pred shape
37     y_true_resaped = y_true_one_hot.reshape(y_pred.shape)
38
39     # Compute the mean squared error between y_pred and y_true_resaped
40     error = ((y_pred - y_true_resaped)**2).sum() / (2*y_pred.size)
41
42     return error
43
44 def accuracy(y_pred, y_true):
45     acc = y_pred.argmax(axis=1) == y_true.argmax(axis=1)
46     return acc.mean()
47
48 results = pd.DataFrame(columns=["mse", "accuracy"])
49
50 # Training loop
51
52 for itr in range(iterations):
53     # Feedforward propagation
54     Z1 = np.dot(X_train, W1)
55     A1 = sigmoid(Z1)
56     Z2 = np.dot(A1, W2)
57     A2 = sigmoid(Z2)
58
59     # Calculate error
60     mse = mean_squared_error(A2, y_train)
61     acc = accuracy(np.eye(output_size)[y_train], A2)
62     new_row = pd.DataFrame({"mse": [mse], "accuracy": [acc]})
63
64     new_row = pd.DataFrame({"mse": [mse], "accuracy": [acc]})
65     results = pd.concat([results, new_row], ignore_index=True)
66
67     # Backpropagation
68     E1 = A2 - np.eye(output_size)[y_train]
69     dW1 = E1 * A2 * (1 - A2)
70     E2 = np.dot(dW1, W2.T)
71     dW2 = E2 * A1 * (1 - A1)
72
73     # Update weights
74     W2_update = np.dot(A1.T, dW1) / N
75     W1_update = np.dot(X_train.T, dW2) / N
76     W2 = W2 - learning_rate * W2_update
77     W1 = W1 - learning_rate * W1_update
78
79 # Visualizing the results
80
81 results.mse.plot(title="Mean Squared Error")
82 plt.show()
83
84 results.accuracy.plot(title="Accuracy")
85 plt.show()
86
87 # Test the model
88
89 Z1 = np.dot(X_test, W1)
90 A1 = sigmoid(Z1)
91 Z2 = np.dot(A1, W2)
92 A2 = sigmoid(Z2)
93 test_acc = accuracy(np.eye(output_size)[y_test], A2)
94 print("Test accuracy: {}".format(test_acc))
```



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

### **Conclusion:**

The Error Backpropagation Perceptron Training Algorithm implements a powerful technique for training neural networks with several layers. By propagating errors backward through the network, it adjusts weights to reduce the discrepancy between expected and actual output. This recurrent approach allows the network to learn complicated relationships in the data and generalize to previously unseen examples. Despite its effectiveness, backpropagation incurs processing costs and may encounter problems such as vanishing or exploding gradients. However, due to its adaptability and extensive application, it has become a cornerstone of current deep learning, revolutionizing domains such as computer vision, natural language processing, and reinforcement learning.