Experiment No. 8

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.

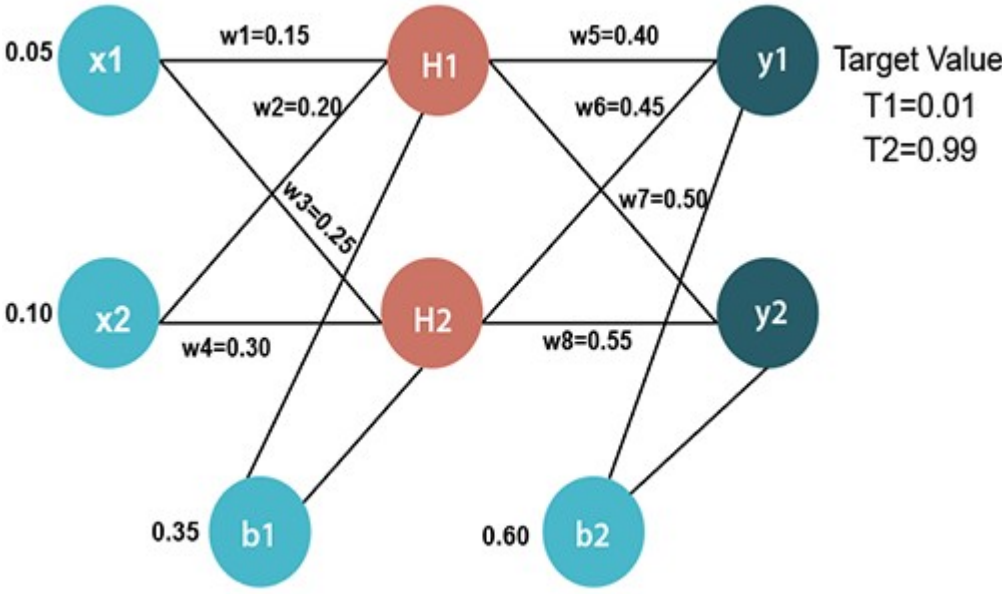
Theory:

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

X1=0.05
X2=0.10

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

$H1=x1\times w_1+x2\times w_2+b1$
$H1=0.05\times 0.15+0.10\times 0.20+0.35$

H1=0.3775

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

$$H1_{final}=\frac{1}{1+\frac{1}{e^{H1}}}$$

$$H1_{final}=\frac{1}{1+\frac{1}{e^{0.3775}}}$$

$$\mathbf{H1_{final}=0.593269992}$$

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

$H2=x1\times w_3+x2\times w_4+b1$
$H2=0.05\times 0.25+0.10\times 0.30+0.35$

H2=0.3925

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

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

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

$$\mathbf{H2_{final}=0.596884378}$$

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

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

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

$$\mathbf{y1_{final} = 0.75136507}$$

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

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

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

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

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

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

So, the total error is

$$= \frac{1}{2}(t1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2$$

$$= \frac{1}{2}(0.01 - 0.75136507)^2 + \frac{1}{2}(0.99 - 0.772928465)^2$$

$$= 0.274811084 + 0.0235600257$$

$$\mathbf{E_{total} = 0.29837111}$$

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.

$$Error_w = \frac{\partial E_{total}}{\partial w}$$

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

$$Error_{w5} = \frac{\partial E_{total}}{\partial w5} \ldots\ldots\ldots (1)$$

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

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

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

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

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

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

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

$$= -(0.01 - 0.75136507)$$

$$\mathbf{\frac{\partial E_{total}}{\partial y1_{final}} = 0.74136507} \ldots\ldots\ldots (4)$$

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

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

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

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

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

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

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

$$= \frac{1 - y1_{final}}{y1_{final}} \times (y1_{final})^2$$

$$= y1_{final} \times (1 - y1_{final})$$

$$= 0.75136507 \times (1 - 0.75136507)$$

$$\frac{\partial y1_{final}}{\partial y1} = \mathbf{0.186815602} \ldots\ldots\ldots (8)$$

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

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

$$= H1_{final}$$

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

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.

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial y1_{final}} \times \frac{\partial y1_{final}}{\partial y1} \times \frac{\partial y1}{\partial w5}$$

$$= 0.74136507 \times 0.186815602 \times 0.593269992$$

$$\mathbf{Error_{w5}} = \frac{\partial \mathbf{E_{total}}}{\partial \mathbf{w5}} = \mathbf{0.0821670407} \ldots\ldots\ldots (\mathbf{11})$$

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

$$w5_{new} = w5 - \eta \times \frac{\partial E_{total}}{\partial w5} \text{ Here, } \eta = \text{learning rate} = 0.5$$

$$= 0.4 - 0.5 \times 0.0821670407$$

$$\mathbf{w5_{new}} = \mathbf{0.35891648} \ldots\ldots\ldots (\mathbf{12})$$

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

$w5_{new}$=0.35891648
$w6_{new}$=408666186
$w7_{new}$=0.511301270

$w8_{new}$=0.561370121

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.

We will calculate the error at w1 as

$$\text{Error}_{w1} = \frac{\partial E_{total}}{\partial w1}$$

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

From equation (2), it is clear that we cannot partially differentiate it with respect to w1 because there is no any w1. We split equation (1) into multiple terms so that we can easily differentiate it with respect to w1 as

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial H1_{final}} \times \frac{\partial H1\_final}{\partial H1} \times \frac{\partial H1}{\partial w1} \dots\dots\dots\dots (13)$$

Now, we calculate each term one by one to differentiate E_{total} with respect to w1 as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial(\frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2)}{\partial H1} \dots\dots\dots\dots (14)$$

We again split this because there is no any $H1^{final}$ term in E^{toatal} as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial E_1}{\partial H1_{final}} + \frac{\partial E_2}{\partial H1_{final}} \dots\dots\dots (15)$$

$\frac{\partial E_1}{\partial H1_{final}}$ and $\frac{\partial E_2}{\partial H1_{final}}$ will again split because in E1 and E2 there is no H1 term. Splitting is done as

$$\frac{\partial E_1}{\partial H1_{final}} = \frac{\partial E_1}{\partial y1} \times \frac{\partial y1}{\partial H1_{final}} \dots\dots\dots (16)$$

$$\frac{\partial E_2}{\partial H1_{final}} = \frac{\partial E_2}{\partial y2} \times \frac{\partial y2}{\partial H1\_final} \dots\dots\dots (17)$$

We again Split both $\frac{\partial E_1}{\partial y1}$ and $\frac{\partial E_2}{\partial y2}$ because there is no any y1 and y2 term in E1 and E2. We split it as

$$\frac{\partial E_1}{\partial y1} = \frac{\partial E_1}{\partial y1_{fianl}} \times \frac{\partial y1_{final}}{\partial y1} \dots\dots\dots (18)$$

$$\frac{\partial E_2}{\partial y2} = \frac{\partial E_2}{\partial y2_{fianl}} \times \frac{\partial y2_{final}}{\partial y2} \dots\dots\dots (19)$$

Now, we find the value of $\frac{\partial E_1}{\partial y1}$ and $\frac{\partial E_2}{\partial y2}$ by putting values in equation (18) and (19) as

From equation (18)

$$\frac{\partial E_1}{\partial y1} = \frac{\partial E_1}{\partial y1_{fianl}} \times \frac{\partial y1_{final}}{\partial y1}$$

$$= \frac{\partial(\frac{1}{2}(T1 - y1_{final})^2)}{\partial y1_{final}} \times \frac{\partial y1_{final}}{\partial y1}$$

$$= 2 \times \frac{1}{2}(T1 - y1_{final}) \times (-1) \times \frac{\partial y1_{final}}{\partial y1}$$

From equation (8)

$$= 2 \times \frac{1}{2}(0.01 - 0.75136507) \times (-1) \times 0.186815602$$

$$\mathbf{\frac{\partial E_1}{\partial y1} = 0.138498562} \dots\dots\dots (20)$$

From equation (19)

$$\frac{\partial E_2}{\partial y2} = \frac{\partial E_2}{\partial y2_{fianl}} \times \frac{\partial y2_{final}}{\partial y2}$$

$$= \frac{\partial(\frac{1}{2}(T2 - y2_{final})^2)}{\partial y2_{final}} \times \frac{\partial y2_{final}}{\partial y2}$$

$$= 2 \times \frac{1}{2}(T2 - y2_{final}) \times (-1) \times \frac{\partial y2_{final}}{\partial y2} \dots\dots\dots (21)$$

$$y2_{final} = \frac{1}{1 + e^{-y2}} \dots\dots\dots\dots\dots\dots (22)$$

$$\frac{\partial y2_{fianl}}{\partial y2} = \frac{\partial(\frac{1}{1 + e^{-y2}})}{\partial y2}$$

$$= \frac{e^{-y2}}{(1 + e^{-y2})^2}$$

$$= e^{-y2} \times (y2_{final})^2 \dots\dots\dots\dots\dots (23)$$

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

$$e^{-y2} = \frac{1 - y2_{final}}{y2_{final}} \dots\dots\dots\dots\dots (24)$$

Putting the value of e^{-y2} in equation (23)

$$= \frac{1 - y2_{final}}{y2_{final}} \times (y2_{final})^2$$

$$= y2_{final} \times (1 - y2_{final})$$

$$= 0.772928465 \times (1 - 0.772928465)$$

$$\mathbf{\frac{\partial y2_{fianl}}{\partial y2} = 0.175510053 \dots\dots\dots (25)}$$

From equation (21)

$$= 2 \times \frac{1}{2}(0.99 - 0.772928465) \times (-1) \times 0.175510053$$

$$\frac{\partial E_1}{\partial y1} = -0.0380982366126414 \ldots\ldots\ldots (26)$$

Now from equation (16) and (17)

$$\frac{\partial E_1}{\partial H1_{final}} = \frac{\partial E_1}{\partial y1} \times \frac{\partial y1}{\partial H1_{final}}$$

$$= 0.138498562 \times \frac{\partial(H1_{final} \times w_5 + H2_{final} \times w_6 + b2)}{\partial H1_{final}}$$

$$= 0.138498562 \times \frac{\partial(H1_{final} \times w_5 + H2_{final} \times w_6 + b2)}{\partial H1_{final}}$$

$$= 0.138498562 \times w5$$

$$= 0.138498562 \times 0.40$$

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

$$\frac{\partial E_2}{\partial H1_{final}} = \frac{\partial E_2}{\partial y2} \times \frac{\partial y2}{\partial H1\_final}$$

$$= -0.0380982366126414 \times \frac{\partial(H1_{final} \times w_7 + H2_{final} \times w_8 + b2)}{\partial H1_{final}}$$

$$= -0.0380982366126414 \times w7$$

$$= -0.0380982366126414 \times 0.50$$

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

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

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial E_1}{\partial H1_{final}} + \frac{\partial E_2}{\partial H1_{final}}$$

$$= 0.0553994248 + (-0.0190491183063207)$$

$$\mathbf{\frac{\partial E_{total}}{\partial H1_{final}} = 0.0364908241736793 \ldots\ldots\ldots (29)}$$

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

$$\frac{\partial H1_{final}}{\partial H1} = \frac{\partial(\frac{1}{1+e^{-H1}})}{\partial H1}$$

$$= \frac{e^{-H1}}{(1+e^{-H1})^2}$$

$$e^{-H1} \times (H1_{final})^2 \ldots\ldots\ldots (30)$$

$$H1_{final} = \frac{1}{1+e^{-H1}}$$

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

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

$$= \frac{1 - H1_{final}}{H1_{final}} \times (H1_{final})^2$$

$$= H1_{final} \times (1 - H1_{final})$$

$$= 0.593269992 \times (1 - 0.593269992)$$

$$\mathbf{\frac{\partial H1_{final}}{\partial H1} = 0.2413007085923199}$$

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:

$$H1 = H1_{final} \times w5 + H2_{final} \times w6 + b2 \dots\dots\dots\dots\dots\dots (32)$$

$$\frac{\partial y1}{\partial w1} = \frac{\partial(x1 \times w1 + x2 \times w3 + b1 \times 1)}{\partial w1}$$

$$= x1$$

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

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.

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial H1_{final}} \times \frac{\partial H1_{final}}{\partial H1} \times \frac{\partial H1}{\partial w1}$$

$$= 0.0364908241736793 \times 0.2413007085923199 \times 0.05$$

$$\mathbf{Error_{w1} = \frac{\partial E_{total}}{\partial w1} = 0.000438568 \dots\dots\dots (34)}$$

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

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

$$= 0.15 - 0.5 \times 0.000438568$$

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

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

$w1_{new}$=0.149780716
$w2_{new}$=0.19956143
$w3_{new}$=0.24975114

$w4_{new}$=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:

```
import numpy as np

class NeuralNetwork:

def __init__(self, input_size, hidden_size, output_size):
# Initialize weights and biases
self.hidden_weights = np.random.randn(hidden_size, input_size)
self.hidden_bias = np.random.randn(hidden_size, 1)
self.output_weights = np.random.randn(output_size, hidden_size)
self.output_bias = np.random.randn(output_size, 1)

def sigmoid(self, x):
return 1 / (1 + np.exp(-x))

def sigmoid_derivative(self, x):
return x * (1 - x)

def feedforward(self, inputs):
# Hidden layer computation
hidden_output = self.sigmoid(np.dot(self.hidden_weights, inputs) + self.hidden_bias)

# Output layer computation
output = self.sigmoid(np.dot(self.output_weights, hidden_output) + self.output_bias)

return hidden_output, output

def backpropagate(self, inputs, target, hidden_output, output):
# Calculate output layer error
output_error = target - output
output_delta = output_error * self.sigmoid_derivative(output)

# Calculate hidden layer error
hidden_error = np.dot(self.output_weights.T, output_delta)
hidden_delta = hidden_error * self.sigmoid_derivative(hidden_output)

# Update output weights and bias
self.output_weights += np.dot(output_delta, hidden_output.T)
self.output_bias += output_delta

# Update hidden weights and bias
self.hidden_weights += np.dot(hidden_delta, inputs.T)
self.hidden_bias += hidden_delta
```

```
def train(self, inputs, targets, epochs, learning_rate):
for _ in range(epochs):
for i in range(len(inputs)):
input_data = inputs[i].reshape(-1, 1)
target_data = targets[i].reshape(-1, 1)

hidden_output, output = self.feedforward(input_data)
self.backpropagate(input_data, target_data, hidden_output, output)

if _ % 100 == 0:
error = np.mean(np.abs(target_data - output))
print(f'Epoch {_}, Error: {error}')


# Example usage

inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
targets = np.array([[0], [1], [1], [0]])


input_size = 2

hidden_size = 3
output_size = 1

epochs = 1000
learning_rate = 0.1

# Create neural network
nn = NeuralNetwork(input_size, hidden_size, output_size)

# Train the neural network
nn.train(inputs, targets, epochs, learning_rate)
```

Output:

```
Epoch 0, Error: 0.8699039239566799
Epoch 100, Error: 0.4753296638656727
Epoch 200, Error: 0.2100970343727372
Epoch 300, Error: 0.1088470664400816
Epoch 400, Error: 0.07885449864123421
Epoch 500, Error: 0.06404650028578399
Epoch 600, Error: 0.05499250046623407
Epoch 700, Error: 0.04878069761801233
Epoch 800, Error: 0.04420134298889308
Epoch 900, Error: 0.040655730666748956
```

Conclusion:

In conclusion, the Backpropagation algorithm is a fundamental method used for training neural networks by iteratively updating weights and biases to minimize the error between predicted and target outputs. Through forward and backward passes, it calculates error gradients and adjusts parameters accordingly. The provided implementation showcases a simple neural network trained using the Backpropagation algorithm to solve a binary classification problem. By feeding input data through the network and updating weights based on error, the algorithm converges towards accurate predictions over multiple epochs. This iterative process demonstrates the effectiveness of Backpropagation in optimizing neural network performance for various machine learning tasks.