Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

ML - Experiment 5

AIM / OBJECTIVE: To implement Backpropagation algorithm.

(1)

Jigar Siddhpura
60004210155
C22

ML – Experiment 5 – Back propagation

Aim: To implement backpropogation algorithm

Theory: 1. A neural network is a group of interconnected I/o units where each connection has a weight associated with it.

2. This model builds upon human nervous system.
3. It helps us conduct image understanding, human learning, computer speech, etc. It is the essence of neural network traing
4. It is the method of fine-tuning weights of NN based on the error rate in prev epochs.
5. Proper tuning of weights decreases the error rate & makes it more generalised.
6. It is a standard method of training ANN. This method helps calculate gradient of a loss function with respect to all the weights in the network
7. This algo computes gradient of the loss function for a single weight by chain rule. Its efficiency computes 1 layer at a time, unlike a native direct computation.
8. It computes the gradient but it does not define how it is used.
9. It generalizes the computation in delta rule.
10. In ANN, we have 3 different types of layers.
    1. Input layer
    2. Hidden layer (can be one or many)
    3. Output layer.

Each layer has its own way of working at its own way of such that we are able to get the desired results & correlate the scenarios to our conditions.

Steps in backpropagation algorithm:

① Create a feed-forward network with n_i inputs, n hidden units & n output ~~in~~ ~~ou~~ inputs.

② Initialize network weights to small no.s.

③ Until the termination condition is met, Do

For each (x, t) in training example, do

Propagate the input forward through the network

1. Input the instance x, to the network & compute the output of every unit.
2. For each network unit k,
$$\delta_k \leftarrow O_k(1-O_k)(t_k - O_k)$$
3. for hidden layers.
$$\delta_h \leftarrow O_h(1-O_h) \sum w_{hk} \delta_k$$
4. Update the network wt, w_{ji}
$$w_{ji} \Leftarrow w_{ji} + \Delta w_{ji}$$
where $\Delta w_{ji} = \eta \delta_j x_{ji}$

⇒ Sum

Iteration 1:

Forward pass

$$z_1 = w_{11} x_1 + w_{21} x_2 + b_1 = 0.6 \times 0 + 1 \times -0.1 + 0.3 = 0.2$$

$$y_3 = \delta(z_1) = \frac{1}{1+e^{-0.2}} = 0.598$$

(2)

$z_2 = w_{12} x_1 + w_{22} x_2 + b_2 = -0.3 \times 0 + 0.9 \times 1 + 0.5 = -0.9$

$y_4 = \delta(z_2) = \frac{1}{1+e^{-0.9}} = 0.711$

$y = w_{31} z_1 + w_{32} z_2 + b_3 = 0.9 \times 0.598 + 0.1 \times 0.711 + -0.2$
$= 0.1103$

$y_5 = \delta(y) = \frac{1}{1+e^{-0.1103}} = 0.527$

$Error = (y_{target} - y_5) = 1 - 0.527$

Backward pass

Output layer

$\delta_5 = y_5(1-y_5)(y_t - y_5) = 0.527(1-0.527)(1-0.52)$
$= 0.117$

Hidden layer

$\delta_3 = y_3(1-y_3) w_{31} \delta_5 = 0.598(1-0.598)(0.9)(0.117)$
$= 0.011$

$\delta_4 = y_4(1-y_4) w_{32} \delta_5 = 0.711(1-0.711)(0.1)(0.117)$
$= 0.002$

$\Delta w_{32} = \eta \delta_5 y_4 = (1)(0.117)(0.711) = 0.083$

$w_{32}(new) = w_{32}(old) + \Delta w_{32} = 0.1 + 0.083 = 0.183$

$\Delta w_{31} = \eta \delta_5 y_3 = (1)(0.117)(0.598) = 0.069$
$w_{31}(new) = w_{31}(old) + \Delta w_{31} = 0.9 + 0.069 = 0.969$

$\Delta w_{11} = \eta \delta_3 x_1 = (1)(0.011)(0) = 0$
$w_{11}(new) = w_{11}(old) + \Delta w_{11} = 0.6 + 0 = 0.6$

$\Delta w_{12} = \eta \delta_2 x_1 = (1)(0.002)(0) = 0$

$w_{12}(new) = w_{12} + \Delta w_{12} = -0.3 + 0 = -0.3$

$\Delta w_{21} = \eta \delta_3 x_2 = (1)(0.011)(1) = 0.011$

$w_{21}(new) = w_{21}(old) + \Delta w_{21} = -0.1 + 0.011 = -0.089$

$\Delta w_{22} = \eta \delta_4 x_2 = (1)(0.002)(1) = 0.002$

$w_{22}(new) = w_{22}(old) + \Delta w_{22} = 0.4 + 0.002 = 0.402$

New update wts:

w_{11}	$= 0.6$	w_{12}	$= 0.402$
w_{12}	$= -0.3$	w_{31}	$= 0.469$
w_{21}	$= -0.089$	w_{32}	$= 0.183$
~~w_{31}~~	~~$= 0.183$~~		

Conclusion: Thus, we have performed backpropagation on the given network & found the error for 500 epochs & plotted the graph for error vs. no. of epochs to understand & observe the error keeps on decreasing, we have performed backpropagation using formulas & algorithm.

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
Import seaborn as sns
# Initialize inputs
x1 = 0
x2 = 1
# weights and biases of output layer
w31 = 0.4
w32 = 0.1
b3 = -0.2
# weights and biases of hidden layer
w11 = 0.6
w21 = -0.1
w12 = -0.3
w22 = 0.4
b1 = 0.3
b2 = 0.5

errors = []
```

```
# Define cost_function_sigmoid activation function
def cost_function_sigmoid(x : int):
    return 1 / (1 + np.exp(-x))
```

```
# Number of epochs
epochs = 800
for epoch in range(epochs):
    # Forward pass
    z1_in = x1 * w11 + x2 * w21 + b1
    z1 = cost_function_sigmoid(z1_in)

    z2_in = x1 * w12 + x2 * w22 + b2
    z2 = cost_function_sigmoid(z2_in)

    y_in = z1 * w31 + z2 * w32 + b3
    y = cost_function_sigmoid(y_in)

    # Calculaing error
    # Assuming a target value = 1
    target = 1
    error_y = (target - y)
    errors.append(error_y)

    # Backward pass
    delta_y = y * (1 - y) * (target - y)
    delta_z1 = z1 * (1 - z1) * w31 * delta_y
    delta_z2 = z2 * (1 - z2) * w32 * delta_y

    # Assuming learning rate (alpha)= 1
    alpha = 1


    # Update weights for output layer
    w31 += alpha * z1 * delta_y
    w32 += alpha * z2 * delta_y
    # Update weights for hidden layer
    w11 += alpha * x1 * delta_z1
    w21 += alpha * x2 * delta_z1
    w12 += alpha * x1 * delta_z2
```

```
    w22 += alpha * x2 * delta_z2

    if (epoch == 0 or epoch == 799):
      print(f"Epoch {epoch + 1}:")
      print("Forward pass:")
      print(f"Hidden layer: = {z1:.4f}, z2 = {z2:.4f}")
      print(f"Error: {error_y:.4f}")
      print("Updated weights after Backward pass:")
      print(f"w31 = {w31:.4f}")
      print(f"w32 = {w32:.4f}")
      print(f"w11 = {w11:.4f}")
      print(f"w21 = {w21:.4f}")
      print(f"w12 = {w12:.4f}")
      print(f"w22 = {w22:.4f}\n")
      print(f"Output: y = {y:.4f}")
```

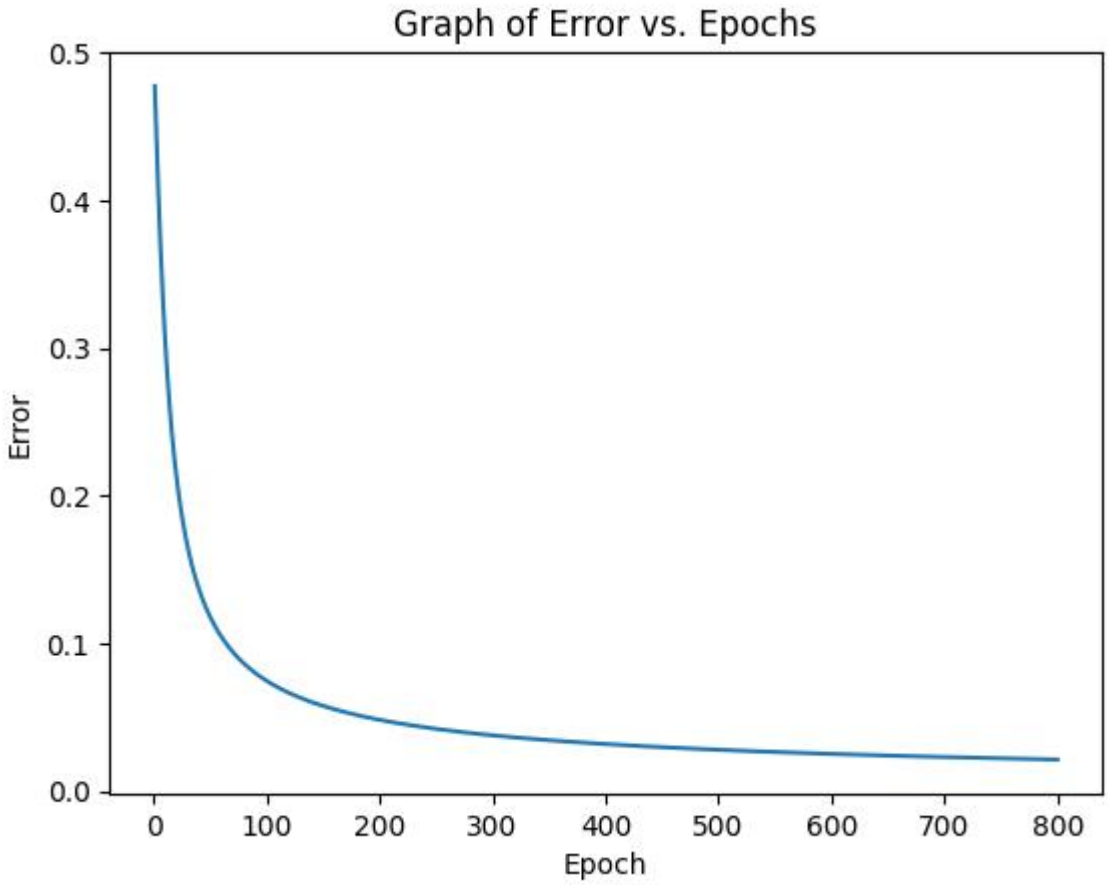
Output for Epoch 1 :

```
Epoch 1:
Forward pass:
Hidden layer: = 0.5498, z2 = 0.7109
Error: 0.4773
Updated weights after Backward pass:
w31 = 0.4655
w32 = 0.1847
w11 = 0.6000
w21 = -0.0882
w12 = -0.3000
w22 = 0.4024

Output: y = 0.5227
```

Output for 800th Epoch :

```
Epoch 800:
Forward pass:
Hidden layer: = 0.7626, z2 = 0.8334
Error: 0.0214
Updated weights after Backward pass:
w31 = 2.4724
w32 = 2.5640
w11 = 0.6000
w21 = 0.8674
w12 = -0.3000
w22 = 1.1099

Output: y = 0.9786
```

```
    sns.lineplot(range(1, epochs + 1), errors)
    plt.xlabel('Epoch')
    plt.ylabel('Error')
    plt.title('Graph of Error vs. Epochs')
    plt.show()
```


Graph of Error vs. Epochs
0.5
0.4
0.3
0.2
0.1
0.0
Error
0
100
200
300
400
500
600
700
800
Epoch