

## Backpropagation Mathematical Solution

### Step 1: Forward Propagation

Given a neural network with:

- Inputs:  $x_1 = 0.05$ ,  $x_2 = 0.10$
- Target Outputs:  $y_1 = 0.01$ ,  $y_2 = 0.99$
- Initial Weights:
  - Input to Hidden:  $w_1 = 0.15$ ,  $w_2 = 0.20$ ,  $w_3 = 0.25$ ,  $w_4 = 0.30$
  - Hidden to Output:  $w_5 = 0.40$ ,  $w_6 = 0.45$ ,  $w_7 = 0.50$ ,  $w_8 = 0.55$
- Biases:
  - Hidden Layer:  $b_h = 0.35$
  - Output Layer:  $b_o = 0.60$

Compute hidden layer activation:

$$h_1 = \sigma(x_1 w_1 + x_2 w_2 + b_h)$$

$$h_2 = \sigma(x_1 w_3 + x_2 w_4 + b_h)$$

Where  $\sigma(z) = 1 / (1 + e^{-z})$  (Sigmoid function).

Compute output layer activation:

$$o_1 = \sigma(h_1 w_5 + h_2 w_6 + b_o)$$

$$o_2 = \sigma(h_1 w_7 + h_2 w_8 + b_o)$$

### Step 2: Compute Error

The error is computed using Mean Squared Error (MSE):

$$E = \frac{1}{2} \sum (y_i - o_i)^2$$

### Step 3: Backpropagation (Compute Gradients)

Compute the derivative of error w.r.t. output neuron weights using the chain rule:

$$\partial E / \partial w_5 = \partial E / \partial o_1 * \partial o_1 / \partial \text{net}_1 * \partial \text{net}_1 / \partial w_5$$

Similarly, for  $w_6$ ,  $w_7$ ,  $w_8$ .

Compute the derivative of error w.r.t. hidden neuron weights:

$$\partial E / \partial w_1 = \partial E / \partial h_1 * \partial h_1 / \partial \text{net}_{h1} * \partial \text{net}_{h1} / \partial w_1$$

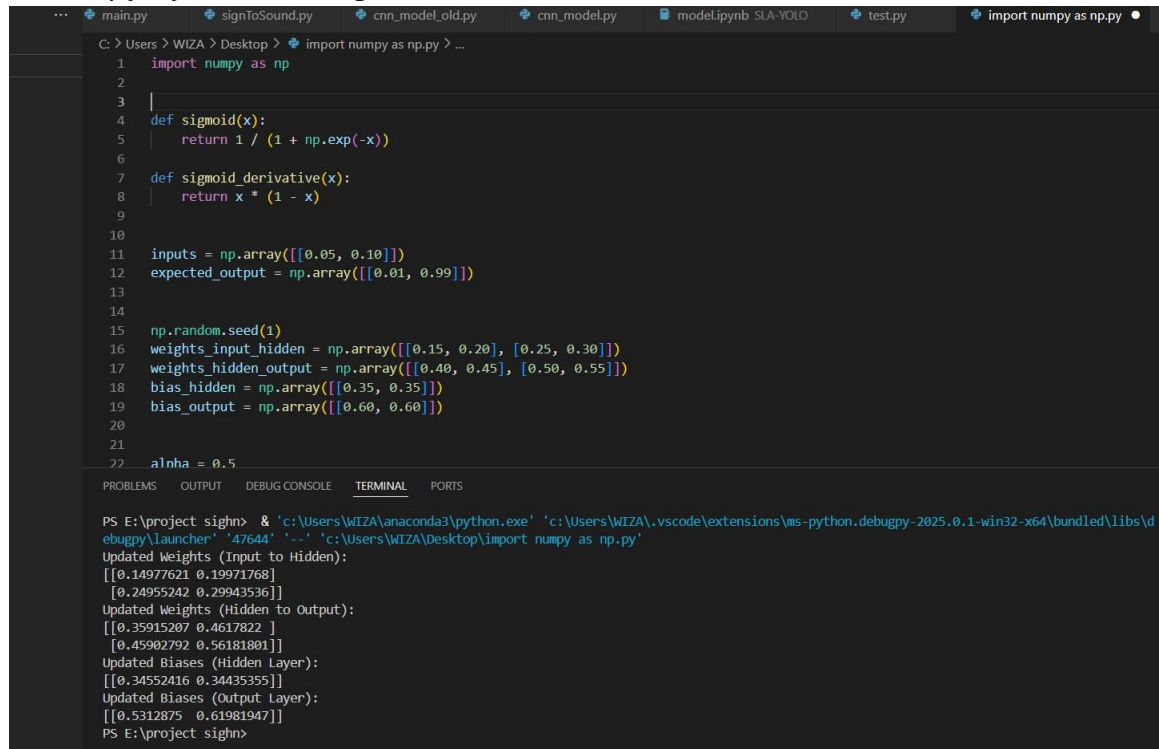
Similarly, for  $w_2$ ,  $w_3$ ,  $w_4$ .

### Step 4: Update Weights

Using Gradient Descent:

$$w_{\text{new}} = w_{\text{old}} - \eta * \partial E / \partial w$$

Where  $\eta$  (eta) is the learning rate.



```
C:\Users\WIZA\Desktop> import numpy as np > ...
1  import numpy as np
2
3  |
4  def sigmoid(x):
5  |     return 1 / (1 + np.exp(-x))
6
7  def sigmoid_derivative(x):
8  |     return x * (1 - x)
9
10
11 inputs = np.array([[0.05, 0.10]])
12 expected_output = np.array([[0.01, 0.99]])
13
14
15 np.random.seed(1)
16 weights_input_hidden = np.array([[0.15, 0.20], [0.25, 0.30]])
17 weights_hidden_output = np.array([[0.40, 0.45], [0.50, 0.55]])
18 bias_hidden = np.array([[0.35, 0.35]])
19 bias_output = np.array([[0.60, 0.60]])
20
21
22 alpha = 0.5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\project sighn> & 'c:\Users\WIZA\anaconda3\python.exe' 'c:\Users\WIZA\.vscode\extensions\ms-python.debugpy-2025.0.1-win32-x64\bundled\libs\debugpy\launcher' '47644' '-' 'c:\Users\WIZA\Desktop\import numpy as np.py'
Updated Weights (Input to Hidden):
[[0.14977621 0.19971768]
 [0.24955242 0.29943536]]
Updated Weights (Hidden to Output):
[[0.35915207 0.4617822 ]
 [0.45902792 0.56181801]]
Updated Biases (Hidden Layer):
[[0.34552416 0.34435355]]
Updated Biases (Output Layer):
[[0.5312875  0.61981947]]
PS E:\project sighn>
```

**import numpy as np**

**def sigmoid(x):**

**return 1 / (1 + np.exp(-x))**

**def sigmoid\_derivative(x):**

**return x \* (1 - x)**

**inputs = np.array([[0.05, 0.10]])**

**expected\_output = np.array([[0.01, 0.99]])**

**np.random.seed(1)**

**weights\_input\_hidden = np.array([[0.15, 0.20], [0.25, 0.30]])**

```
weights_hidden_output = np.array([[0.40, 0.45], [0.50, 0.55]])
```

```
bias_hidden = np.array([[0.35, 0.35]])
```

```
bias_output = np.array([[0.60, 0.60]])
```

```
alpha = 0.5
```

```
hidden_layer_input = np.dot(inputs, weights_input_hidden) + bias_hidden
```

```
hidden_layer_output = sigmoid(hidden_layer_input)
```

```
output_layer_input = np.dot(hidden_layer_output, weights_hidden_output) +  
bias_output
```

```
output_layer_output = sigmoid(output_layer_input)
```

```
delta_output = (expected_output - output_layer_output) *  
sigmoid_derivative(output_layer_output)
```

```
delta_hidden = np.dot(delta_output, weights_hidden_output.T) *  
sigmoid_derivative(hidden_layer_output)
```

```
weights_hidden_output += alpha * np.dot(hidden_layer_output.T, delta_output)
```

```
weights_input_hidden += alpha * np.dot(inputs.T, delta_hidden)
```

```
bias_output += alpha * delta_output
```

```
bias_hidden += alpha * delta_hidden
```

```
print("Updated Weights (Input to Hidden):")
```

```
print(weights_input_hidden)
```

```
print("Updated Weights (Hidden to Output):")
```

```
print(weights_hidden_output)  
print("Updated Biases (Hidden Layer):")  
print(bias_hidden)  
print("Updated Biases (Output Layer):")  
print(bias_output)
```