Step 1: Initialize Inputs

[1]:
```
x1 = 0.5
x2 = 0.8
y_target = 1  # Target output for training
```

Step 2: Initialize Weights

[2]:
```
# Weight initialization
w1 = 0.2
w2 = -0.3
```

Step 3: Calculate Net Input

[3]:
```
# Calculate net input to the neuron
net_input = x1 * w1 + x2 * w2

def sigmoid(x):
    return 1 / (1 + math.exp(-x))  # Sigmoid activation function
```

Step 5: Define Sigmoid Derivative

[5]:
```
def sigmoid_derivative(output):
    return output * (1 - output)  # Derivative of the sigmoid function
```

Step 6: Forward Propagation

[6]:
```
# Calculate the output using the sigmoid function
output = sigmoid(net_input)
```

Step 7: Calculate Error

Step 8: Calculate Weight Updates

[8]:
```
# Calculate weight updates
w1_error = error_term_output * x1
w2_error = error_term_output * x2
```

Step 9: Set Learning Rate

[9]:
```
lr = 0.1  # Learning rate
```

Step 10: Update Weights

Display results

[12]:
```
print("Initial Weights:")
print(f"w1: {w1}")
print(f"w2: {w2}")

print("\nUpdated Weights after Backpropagation:")
print(f"w1_new: {w1_new}")
print(f"w2_new: {w2_new}")
```

```
Initial Weights:
w1: 0.2
w2: -0.3

Updated Weights after Backpropagation:
```

```
w1_new: 0.2066541282958158
w2_new: -0.2893533947266947
```