

Assignment 7: Implementing a Single-Layer Neural Network for Binary Classification

Question 1: Design and Initialize Neural Network

1.1 Network Design and Truth Table Creation

- Creating a truth table and corresponding binary output

Step 1.1 Network Design and Truth Table Creation

The logical function $(x_1 \text{ AND } x_2) \text{ OR } (x_3 \text{ AND } x_4)$

x_1	x_2	x_3	x_4	Output
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

1.2 Manual Forward Propagation Calculation

- I choose example set of inputs from my truth are $x_1=0, x_2=0, x_3=0, x_4=0$ and $x_1=0, x_2=0, x_3=1, x_4=1$ and $x_1=1, x_2=1, x_3=0, x_4=0$ and $x_1=1, x_2=0, x_3=1, x_4=1$
- Using sigmoid activation function and weight=0.5, bias = 1

Step 1.2 Manual Forward Propagation Calculation

$$w = 0.5, b = 1, z = \text{sigmoid}, f(z) = \frac{1}{1+e^{-z}}$$

$$x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0$$

$$z = (0.5 \times 0) + (0.5 \times 0) + (0.5 \times 0) + (0.5 \times 0) + 1$$

$$z = 0 + 0 + 0 + 0 + 1$$

$$z = 1$$

$$f(1) = \frac{1}{1+e^{-1}} = 0.73, y_{\text{pred}} = 0.73$$

$$x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1$$

$$z = (0.5 \times 0) + (0.5 \times 0) + (0.5 \times 1) + (0.5 \times 1) + 1$$

$$z = 0 + 0 + 0.5 + 0.5 + 1$$

$$z = 2$$

$$f(2) = \frac{1}{1+e^{-2}} = 0.88, y_{\text{pred}} = 0.88$$

$$x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$$

$$z = (0.5 \times 1) + (0.5 \times 1) + (0.5 \times 0) + (0.5 \times 0) + 1$$

$$z = 0.5 + 0.5 + 0 + 0 + 1$$

$$z = 2$$

$$f(2) = \frac{1}{1+e^{-2}} = 0.88, y_{\text{pred}} = 0.88$$

$$x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1$$

$$z = (0.5 \times 1) + (0.5 \times 0) + (0.5 \times 1) + (0.5 \times 1) + 1$$

$$z = 0.5 + 0 + 0.5 + 0.5 + 1$$

$$z = 2.5$$

$$f(2.5) = \frac{1}{1+e^{-2.5}} = 0.923, y_{\text{pred}} = 0.923$$

1.3 Manual Backpropagation Calculation

Step 1.3 Manual Backpropagation Calculation

Error Calculation

$$\text{Error} = t - y_{\text{pred}}$$

$$x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0$$

$$w = 0.5$$

$$b = 1$$

$$t = 1$$

$$y_{\text{pred}} = 0.88$$

$$\text{Error} = 1 - 0.88 = 0.12$$

Gradient Computation

$$E(w) = (t - y_{\text{pred}})^2$$

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} (t - y_{\text{pred}})^2 \\ &= -2 \cdot (t - y_{\text{pred}}) \cdot \frac{\partial (t - y_{\text{pred}})}{\partial w_i} \\ &= -2 \cdot (t - y_{\text{pred}}) \cdot \frac{\partial y_{\text{pred}}}{\partial w_i} \end{aligned}$$

$$y_{\text{pred}} = \sigma(z)$$

Using the derivation of the sigmoid function

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z) \cdot (1 - \sigma(z))$$

Thus

$$\frac{\partial y_{\text{pred}}}{\partial w_i} = \frac{\partial \sigma(z)}{\partial z} \cdot \frac{\partial z}{\partial w_i}$$

$$\frac{\partial z}{\partial w_i} = x_i$$

Therefore:

$$\frac{\partial y_{\text{pred}}}{\partial w_i} = \sigma(z) \cdot (1 - \sigma(z)) \cdot x_i$$

Error = 0.12

$$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$$

$$\begin{aligned}\sigma'(z) &= 0.88 \cdot (1 - 0.88) \\ &= 0.1056\end{aligned}$$

$$\frac{\partial y_{\text{pred}}}{\partial w_i} = \sigma'(z) \cdot x_i$$

For each weight:

$$x_1 = 1, \quad \frac{\partial y_{\text{pred}}}{\partial w_1} = 0.1056 \times 1 = 0.1056$$

$$x_2 = 1, \quad \frac{\partial y_{\text{pred}}}{\partial w_2} = 0.1056 \times 1 = 0.1056$$

$$x_3 = 0, \quad \frac{\partial y_{\text{pred}}}{\partial w_3} = 0.1056 \times 0 = 0$$

$$x_4 = 0, \quad \frac{\partial y_{\text{pred}}}{\partial w_4} = 0.1056 \times 0 = 0$$

$$\frac{\partial E}{\partial w_i} = -2 \cdot \text{error} \cdot \frac{\partial y_{\text{pred}}}{\partial w_i}$$

$$\frac{\partial E}{\partial w_1} = -2 \times 0.12 \times 0.1056 = -0.0253$$

$$\frac{\partial E}{\partial w_2} = -2 \times 0.12 \times 0.1056 = -0.0253$$

$$\frac{\partial E}{\partial w_3} = 0$$

$$\frac{\partial E}{\partial w_4} = 0$$

Gradient of the loss function with respect to the bias (b):

$$\frac{\partial E}{\partial b} = -2 \cdot (t - y_{\text{pred}}) \cdot \sigma'(z)$$

$$= -2 \times 0.12 \times 0.1056$$

$$= -0.0253$$

Gradient value = -0.0253

Weight Update

$$w_i = w_i - \eta \cdot \frac{\partial E}{\partial w_i}$$

learning rate, $\eta = 0.001$

$$w_1 = 0.5 - 0.001 \times (-0.0253) = 0.5000253$$

$$w_2 = 0.5 - 0.001 \times (-0.0253) = 0.5000253$$

$$w_3 = 0.5 - 0.001 \times 0 = 0.5$$

$$w_4 = 0.5 - 0.001 \times 0 = 0.5$$

bias update:

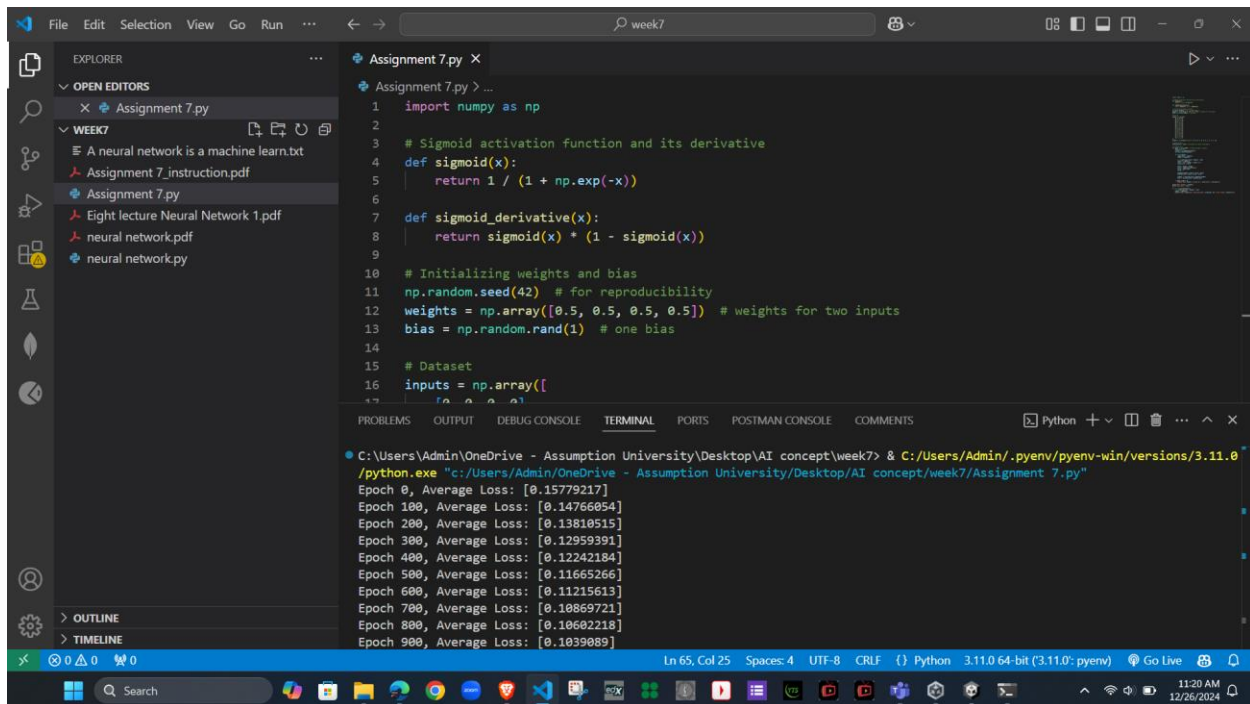
$$b = b - \eta \cdot \frac{\partial E}{\partial b}$$

$$b = 1 - 0.01 \times (-0.0253)$$

$$b = 1.0000253$$

weight update value using learning rate 0.001

1.4 Implementation and Training in Python

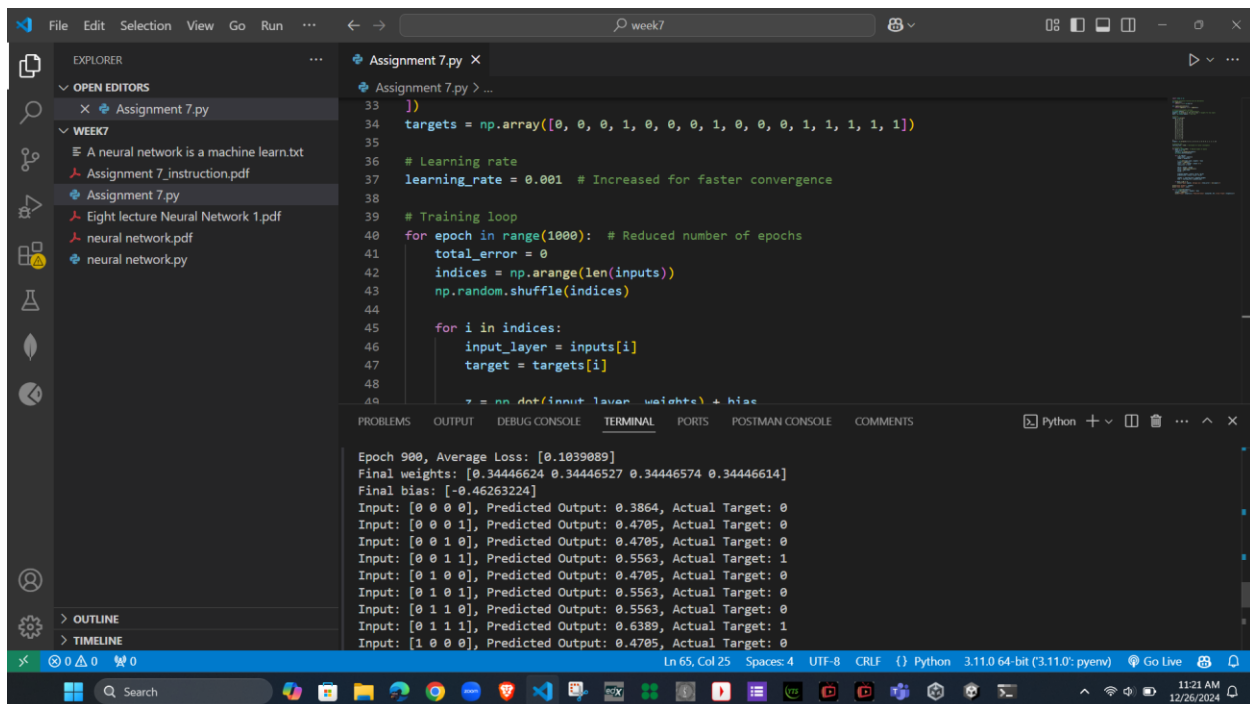


The screenshot shows the VS Code editor with the file 'Assignment 7.py' open. The code defines a sigmoid function and its derivative, initializes weights and bias, and sets up a dataset. The terminal output shows the execution results for epochs 0 to 900.

```
1 import numpy as np
2
3 # Sigmoid activation function and its derivative
4 def sigmoid(x):
5     return 1 / (1 + np.exp(-x))
6
7 def sigmoid_derivative(x):
8     return sigmoid(x) * (1 - sigmoid(x))
9
10 # Initializing weights and bias
11 np.random.seed(42) # for reproducibility
12 weights = np.array([0.5, 0.5, 0.5, 0.5]) # weights for two inputs
13 bias = np.random.rand(1) # one bias
14
15 # Dataset
16 inputs = np.array([
17     [0, 0, 0, 0],
18     [0, 0, 1, 1],
19     [0, 1, 0, 0],
20     [0, 1, 1, 0],
21     [0, 1, 1, 1],
22     [1, 0, 0, 0],
23     [1, 0, 0, 1],
24     [1, 0, 1, 0],
25     [1, 0, 1, 1],
26     [1, 1, 0, 0],
27     [1, 1, 0, 1],
28     [1, 1, 1, 0],
29     [1, 1, 1, 1]
30 ])
```

Terminal Output:

```
C:\Users\Admin\OneDrive - Assumption University\Desktop\AI concept\week7> C:\Users\Admin\pyenv\pyenv-win\versions\3.11.0\python.exe "c:\Users\Admin\OneDrive - Assumption University\Desktop\AI concept\week7\Assignment 7.py"
Epoch 0, Average Loss: [0.15779217]
Epoch 100, Average Loss: [0.14766854]
Epoch 200, Average Loss: [0.13810515]
Epoch 300, Average Loss: [0.12959391]
Epoch 400, Average Loss: [0.12242184]
Epoch 500, Average Loss: [0.11665266]
Epoch 600, Average Loss: [0.11215613]
Epoch 700, Average Loss: [0.10869721]
Epoch 800, Average Loss: [0.10602218]
Epoch 900, Average Loss: [0.1039089]
```



The screenshot shows the VS Code editor with the file 'Assignment 7.py' open. The code continues with the training loop, calculating the total error and printing the results for each epoch. The terminal output shows the final results after 900 epochs.

```
33 ]
34 targets = np.array([0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1])
35
36 # Learning rate
37 learning_rate = 0.001 # Increased for faster convergence
38
39 # Training loop
40 for epoch in range(1000): # Reduced number of epochs
41     total_error = 0
42     indices = np.arange(len(inputs))
43     np.random.shuffle(indices)
44
45     for i in indices:
46         input_layer = inputs[i]
47         target = targets[i]
48
49         z = np.dot(input_layer, weights) + bias
50         predicted_output = sigmoid(z)
51         error = target - predicted_output
52         total_error += error ** 2
53
54     # Print results for each epoch
55     print(f'Epoch {epoch}, Average Loss: {total_error / len(indices)}')
56
57 # Final results
58 final_weights = weights
59 final_bias = bias
60 final_targets = targets
61 final_inputs = inputs
62
63 # Print final results
64 print(f'Final weights: {final_weights}')
65 print(f'Final bias: {final_bias}')
66 print(f'Final targets: {final_targets}')
67 print(f'Final inputs: {final_inputs}')
```

Terminal Output:

```
Epoch 900, Average Loss: [0.1039089]
Final weights: [0.34446624 0.34446527 0.34446574 0.34446614]
Final bias: [-0.46263224]
Input: [0 0 0], Predicted Output: 0.3864, Actual Target: 0
Input: [0 0 0 1], Predicted Output: 0.4705, Actual Target: 0
Input: [0 0 1 0], Predicted Output: 0.4705, Actual Target: 0
Input: [0 0 1 1], Predicted Output: 0.5563, Actual Target: 1
Input: [0 1 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [0 1 0 1], Predicted Output: 0.5563, Actual Target: 0
Input: [0 1 1 0], Predicted Output: 0.5563, Actual Target: 0
Input: [0 1 1 1], Predicted Output: 0.6389, Actual Target: 1
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
```

The screenshot shows the VS Code editor with the file `Assignment 7.py` open. The code defines a neural network with sigmoid and derivative functions, calculates gradients, and updates weights and bias. The terminal displays the output of the program, showing inputs, predicted outputs, and actual targets for 15 different input cases.

```
50     output = sigmoid(z)
51     error = 0.5 * (target - output) ** 2
52     total_error += error
53
54     dE_dy = output - target
55     dy_dz = sigmoid_derivative(z)
56     dz_dw = input_layer
57     dz_db = 1
58
59     gradient_weights = dE_dy * dy_dz * dz_dw
60     gradient_bias = dE_dy * dy_dz * dz_db
61
62     weights -= learning_rate * gradient_weights
63     bias -= learning_rate * gradient_bias
64
65     if epoch % 100 == 0:
```

Terminal Output:

```
Input: [1 0 0 1], Predicted Output: 0.5563, Actual Target: 0
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [1 0 0 1], Predicted Output: 0.5563, Actual Target: 0
Input: [1 0 1 0], Predicted Output: 0.5563, Actual Target: 0
Input: [1 0 1 1], Predicted Output: 0.6389, Actual Target: 1
Input: [1 1 0 0], Predicted Output: 0.5563, Actual Target: 1
```

The screenshot shows the VS Code editor with the file `Assignment 7.py` open. The code continues from the previous screenshot, adding a loop to calculate the final weights and bias, and a loop to calculate the final output for each input. The terminal displays the output of the program, showing inputs, predicted outputs, and actual targets for 15 different input cases.

```
63     bias -= learning_rate * gradient_bias
64
65     if epoch % 100 == 0:
66         print(f"Epoch {epoch}, Average Loss: {total_error / len(inputs)}")
67
68     print("Final weights:", weights)
69     print("Final bias:", bias)
70
71     for i in range(len(inputs)):
72         z = np.dot(inputs[i], weights) + bias
73         output = sigmoid(z)
74         print(f"Input: {inputs[i]}, Predicted Output: {output[0]:.4f}, Actual Target: {targets[i]}")
```

Terminal Output:

```
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [1 0 0 0], Predicted Output: 0.4705, Actual Target: 0
Input: [1 0 0 1], Predicted Output: 0.5563, Actual Target: 0
Input: [1 0 1 0], Predicted Output: 0.5563, Actual Target: 0
Input: [1 0 1 1], Predicted Output: 0.6389, Actual Target: 1
Input: [1 1 0 0], Predicted Output: 0.5563, Actual Target: 1
Input: [1 1 0 1], Predicted Output: 0.6389, Actual Target: 1
Input: [1 1 1 0], Predicted Output: 0.6389, Actual Target: 1
Input: [1 1 1 1], Predicted Output: 0.7141, Actual Target: 1
C:\Users\Admin\OneDrive - Assumption University\Desktop\AI concept\week7>
```

