Lab Report 4: Backpropagation Step-by-Step

Course Code: COMP-341L
Course Name: Artificial Neural Networks Lab
Lab Number: 4
Lab Title: Backpropagation (Manual Analysis)
Date: February 10, 2026

Name: Ali Hamza
Roll Number: B23F0063AI106
Section: B.S AI - Red

Academic Integrity

This report and the conclusion are written in my own words. I have not copied from colleagues or submitted anyone else's work. Plagiarism, copy-pasting, or submitting a colleague's report results in zero marks for all involved (as per course policy).

1 Objective

To manually trace one iteration of backpropagation on a small neural network: forward pass, error calculation, output and hidden deltas, and weight updates. To interpret the results and reflect on backpropagation as blame assignment.

2 Network (from Lab 04.pdf)

- **Inputs:** $x_1 = 0.35$, $x_2 = 0.7$
- **Weights:** $w_{1,1} = 0.2$, $w_{2,1} = 0.2$, $w_{1,2} = 0.3$, $w_{2,2} = 0.3$, $w_{1,3} = 0.3$, $w_{2,3} = 0.9$
- **Target:** $y_{\text{target}} = 0.5$
- **Learning rate:** $\eta = 1$
- **Activation:** Sigmoid
- **Notation:** Hidden outputs y_3 (h1), y_4 (h2); output y_5 (o1)

3 Task 1: Forward Pass Analysis (Understanding the Prediction)

Steps

1. Weighted sum at each hidden neuron

$$a_1 = w_{1,1}x_1 + w_{2,1}x_2 = 0.2 \times 0.35 + 0.2 \times 0.7 = 0.21$$

$$a_2 = w_{1,2}x_1 + w_{2,2}x_2 = 0.3 \times 0.35 + 0.3 \times 0.7 = 0.315$$

2. Sigmoid at hidden layer

$$y_3 = \sigma(a_1) = \frac{1}{1 + e^{-a_1}} \approx 0.552$$

$$y_4 = \sigma(a_2) \approx 0.578$$

3. Weighted sum and output at output neuron

$$a_3 = w_{1,3}y_3 + w_{2,3}y_4 = 0.3 \times 0.552 + 0.9 \times 0.578 \approx 0.686$$

$$y_5 = \sigma(a_3) \approx 0.665$$

4. Comparison

Target $= 0.5$, predicted $y_5 \approx 0.665 \rightarrow$ prediction is **incorrect**.

Code description (Task 1)

Relevant snippets from `backprop_manual_calc.py`:

```
def sigmoid(x):
    return 1 / (1 + math.exp(-x))
```

```
a1 = w11 * x1 + w21 * x2
a2 = w12 * x1 + w22 * x2
y3 = sigmoid(a1)
y4 = sigmoid(a2)
```

```
a3 = w13 * y3 + w23 * y4
y5 = sigmoid(a3)
```

The script prints each step and compares `y5` to the target so we can see the prediction is wrong.

Conceptual: Why is the network's prediction considered incorrect?

The prediction is incorrect because $y_5 \neq y_{\text{target}}$. The network output (0.665) is higher than the target (0.5), so the model overestimated the risk score for this input.

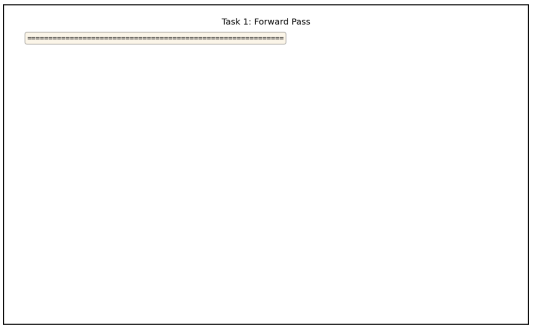

Figure 1: Task 1 – Forward pass

4 Task 2: Error Calculation (Identifying the Mistake)

$$\text{Error} = y_{\text{target}} - y_5 = 0.5 - 0.665 = -0.165$$

Code description (Task 2)

```
error = y_target - y5
```

The script prints this value and explains that a negative error means the prediction is too high.

Conceptual: What does the sign of the error tell you about the prediction?

A **negative** error means the predicted value y_5 is **greater** than the target. So the network predicted too high; we need to reduce the output toward 0.5. The magnitude (0.165) indicates how far the prediction is from the target.

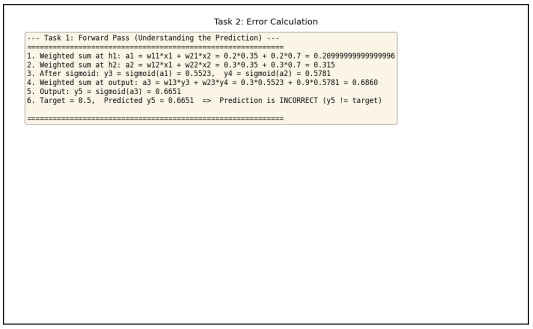

Figure 2: Task 2 – Error calculation

5 Task 3: Output Neuron Responsibility (δ_5)

$$\delta_5 = y_5(1 - y_5)(y_{\text{target}} - y_5)$$

$$\delta_5 = 0.665 \times 0.335 \times (-0.165) \approx -0.037$$

Code description (Task 3)

```
delta5 = y5 * (1 - y5) * (y_target - y5)
```

Conceptual: Why do we multiply the error with the derivative of the sigmoid function?

We multiply the error by the derivative $\sigma'(a) = y(1 - y)$ because of the **chain rule**. We need the gradient of the loss with respect to the weighted sum of the neuron. The derivative shows sensitivity; neurons near saturation learn slowly.

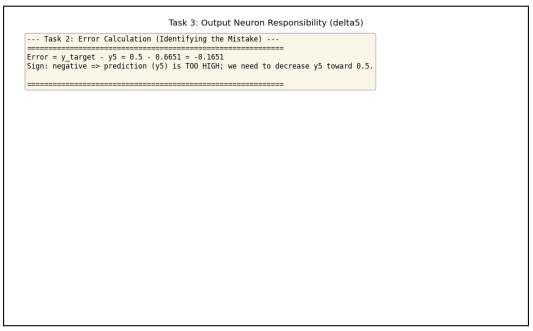

Figure 3: Task 3 – Output delta

6 Task 4: Hidden Neuron Responsibility (δ_3 and δ_4)

$$\delta_3 \approx -0.0027, \quad \delta_4 \approx -0.0081$$

```
delta3 = y3 * (1 - y3) * (w13 * delta5)
delta4 = y4 * (1 - y4) * (w23 * delta5)
```

Hidden neurons do not use the target directly. Responsibility is propagated backward using output weights.

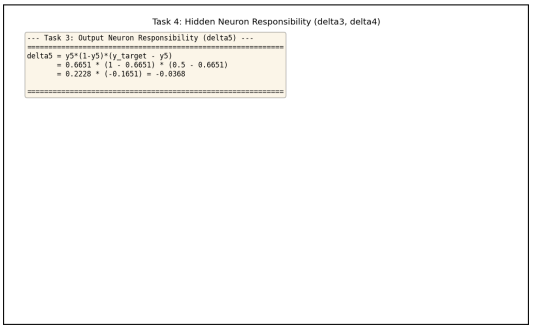

Figure 4: Task 4 – Hidden deltas

7 Task 5: Weight Updates (Learning from Mistakes)

$$\Delta w = \eta \times \delta \times \text{input}$$

```
dw13 = eta * delta5 * y3
dw23 = eta * delta5 * y4
w13_new = w13 + dw13
w23_new = w23 + dw23
```

```
dw11 = eta * delta3 * x1
dw21 = eta * delta3 * x2
dw12 = eta * delta4 * x1
dw22 = eta * delta4 * x2
```

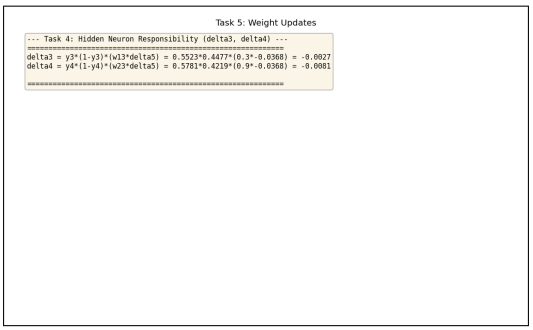

Figure 5: Task 5 – Weight updates

8 Task 6: Interpretation & Reflection (Critical Thinking)

Backpropagation assigns blame backward using the chain rule. Learning rate controls convergence stability. The backward pass flows opposite to the forward activation.

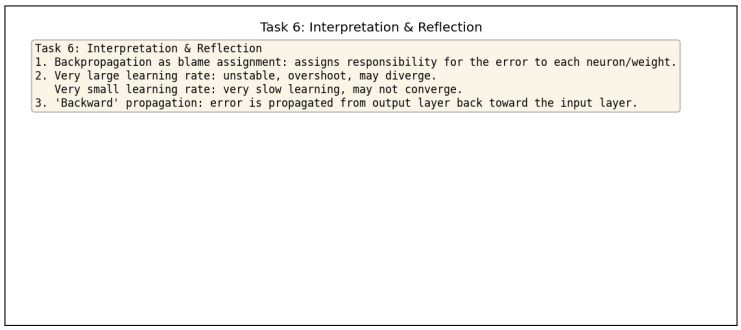

Figure 6: Task 6 – Reflection

9 Results (Screenshots)

Screenshots were captured from the terminal output of `backprop_manual_calc.py` (one section per task). Run the script using:

```
../../venv/bin/python3 backprop_manual_calc.py
```

10 Conclusion

In this lab I worked through one full step of backpropagation by hand on a small network. I saw how the forward pass gives a prediction, how we measure the mistake with the error, and how responsibility is assigned backward using weights and derivatives. This confirmed that backpropagation is simply the chain rule applied step by step.

11 References

Lab 04 Manual (Lab 04.pdf), LAB TASKS, pages 16–18.