Neural Network Backpropagation: Complete Mathematical Framework

3-4-3 Architecture with Worked Example

September 2, 2025

Contents

1 Network Architecture

We consider a feedforward neural network with the following structure:

- **Input Layer**: 3 neurons (x_1, x_2, x_3)
- **Hidden Layer**: 4 neurons (h_1, h_2, h_3, h_4)
- **Output Layer**: 3 neurons (y_1, y_2, y_3)

2 Network Parameters

2.1 Input to Hidden Layer

Weight matrix $\mathbf{W}^{(1)} \in \mathbb{R}^{4\times 3}$:

$$\mathbf{W}^{(1)} = \begin{pmatrix} 0.5 & -0.3 & 0.7 \\ -0.2 & 0.8 & 0.1 \\ 0.9 & -0.6 & 0.4 \\ 0.3 & 0.2 & -0.5 \end{pmatrix} \tag{1}$$

Bias vector $\mathbf{b}^{(1)} \in \mathbb{R}^{4\times 1}$:

$$\mathbf{b}^{(1)} = \begin{pmatrix} 0.1 \\ -0.2 \\ 0.3 \\ 0.0 \end{pmatrix} \tag{2}$$

2.2 Hidden to Output Layer

Weight matrix $\mathbf{W}^{(2)} \in \mathbb{R}^{3\times 4}$:

$$\mathbf{W}^{(2)} = \begin{pmatrix} 0.6 & -0.4 & 0.2 & 0.8 \\ -0.1 & 0.7 & -0.3 & 0.5 \\ 0.4 & 0.1 & 0.9 & -0.2 \end{pmatrix} \tag{3}$$

Bias vector $\mathbf{b}^{(2)} \in \mathbb{R}^{3\times 1}$:

$$\mathbf{b}^{(2)} = \begin{pmatrix} 0.2 \\ -0.1 \\ 0.4 \end{pmatrix} \tag{4}$$

3 Forward Pass

3.1 Mathematical Framework

The forward pass computes the network output through two stages:

3.1.1 Input to Hidden Layer

For each hidden neuron $j = 1, 2, 3, 4$:

$$z_j^{(1)} = \sum_{i=1}^{3} W_{ji}^{(1)} x_i + b_j^{(1)} \tag{5}$$

$$h_j = \sigma(z_j^{(1)}) = \frac{1}{1 + e^{-z_j^{(1)}}} \tag{6}$$

3.1.2 Hidden to Output Layer

For each output neuron $k = 1, 2, 3$:

$$z_k^{(2)} = \sum_{j=1}^{4} W_{kj}^{(2)} h_j + b_k^{(2)} \tag{7}$$

$$y_k = \sigma(z_k^{(2)}) = \frac{1}{1 + e^{-z_k^{(2)}}} \tag{8}$$

3.2 Activation Function

We use the sigmoid activation function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{9}$$

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) \tag{10}$$

4 Loss Function

We use the Mean Squared Error (MSE) loss function:

$$L = \frac{1}{2} \sum_{k=1}^{3} (y_k - t_k)^2 \tag{11}$$

where t_k is the target output for neuron k.

The partial derivative of the loss with respect to output y_k is:

$$\frac{\partial L}{\partial y_k} = y_k - t_k \tag{12}$$

5 Backpropagation Algorithm

5.1 Error Terms (Deltas)

5.1.1 Output Layer Delta

For output layer neuron k:

$$\delta_k^{(2)} = \frac{\partial L}{\partial z_k^{(2)}} \tag{13}$$

$$= \frac{\partial L}{\partial y_k} \cdot \frac{\partial y_k}{\partial z_k^{(2)}} \tag{14}$$

$$= (y_k - t_k) \cdot y_k(1 - y_k) \tag{15}$$

5.1.2 Hidden Layer Delta

For hidden layer neuron j:

$$\delta_j^{(1)} = \frac{\partial L}{\partial z_j^{(1)}} \tag{16}$$

$$= \sum_{k=1}^{3} \frac{\partial L}{\partial z_k^{(2)}} \cdot \frac{\partial z_k^{(2)}}{\partial h_j} \cdot \frac{\partial h_j}{\partial z_j^{(1)}} \tag{17}$$

$$= \left(\sum_{k=1}^{3} \delta_k^{(2)} W_{kj}^{(2)} \right) \cdot h_j(1 - h_j) \tag{18}$$

5.2 Gradient Computation

Once we have the deltas, the gradients are:

5.2.1 Output Layer Gradients

$$\frac{\partial L}{\partial W_{kj}^{(2)}} = \delta_k^{(2)} \cdot h_j \tag{19}$$

$$\frac{\partial L}{\partial b_k^{(2)}} = \delta_k^{(2)} \tag{20}$$

5.2.2 Hidden Layer Gradients

$$\frac{\partial L}{\partial W_{ji}^{(1)}} = \delta_j^{(1)} \cdot x_i \tag{21}$$

$$\frac{\partial L}{\partial b_j^{(1)}} = \delta_j^{(1)} \tag{22}$$

6 Worked Example

Let's compute a complete example with:

$$\mathbf{x} = \begin{pmatrix} 1.0 \\ 0.5 \\ -0.3 \end{pmatrix} \tag{23}$$

$$\mathbf{t} = \begin{pmatrix} 0.8 \\ 0.2 \\ 0.6 \end{pmatrix} \tag{24}$$

6.1 Forward Pass Calculations

6.1.1 Hidden Layer Pre-activations

$$z_1^{(1)} = 0.5(1.0) + (-0.3)(0.5) + 0.7(-0.3) + 0.1 = 0.19 \tag{25}$$

$$z_2^{(1)} = (-0.2)(1.0) + 0.8(0.5) + 0.1(-0.3) + (-0.2) = -0.03 \tag{26}$$

$$z_3^{(1)} = 0.9(1.0) + (-0.6)(0.5) + 0.4(-0.3) + 0.3 = 0.78 \tag{27}$$

$$z_4^{(1)} = 0.3(1.0) + 0.2(0.5) + (-0.5)(-0.3) + 0.0 = 0.55 \tag{28}$$

6.1.2 Hidden Layer Activations

$$h_1 = \sigma(0.19) = 0.547 \tag{29}$$

$$h_2 = \sigma(-0.03) = 0.493 \tag{30}$$

$$h_3 = \sigma(0.78) = 0.686 \tag{31}$$

$$h_4 = \sigma(0.55) = 0.634 \tag{32}$$

6.1.3 Output Layer Pre-activations

$$z_1^{(2)} = 0.6(0.547) + (-0.4)(0.493) + 0.2(0.686) + 0.8(0.634) + 0.2 = 0.768 \quad (33)$$
$$z_2^{(2)} = (-0.1)(0.547) + 0.7(0.493) + (-0.3)(0.686) + 0.5(0.634) + (-0.1) = 0.364 \quad (34)$$
$$z_3^{(2)} = 0.4(0.547) + 0.1(0.493) + 0.9(0.686) + (-0.2)(0.634) + 0.4 = 1.090 \quad (35)$$

6.1.4 Output Layer Activations (Final Predictions)

$$y_1 = \sigma(0.768) = 0.683 \quad (36)$$
$$y_2 = \sigma(0.364) = 0.590 \quad (37)$$
$$y_3 = \sigma(1.090) = 0.748 \quad (38)$$

6.1.5 Loss Computation

$$L = \frac{1}{2}[(0.683 - 0.8)^2 + (0.590 - 0.2)^2 + (0.748 - 0.6)^2] \quad (39)$$
$$= \frac{1}{2}[(-0.117)^2 + (0.390)^2 + (0.148)^2] \quad (40)$$
$$= \frac{1}{2}[0.0137 + 0.1521 + 0.0219] \quad (41)$$
$$= 0.094 \quad (42)$$

6.2 Backward Pass - Error Terms

6.2.1 Output Layer Deltas

$$\delta_1^{(2)} = (0.683 - 0.8) \times 0.683 \times (1 - 0.683) = -0.117 \times 0.216 = -0.025 \quad (43)$$
$$\delta_2^{(2)} = (0.590 - 0.2) \times 0.590 \times (1 - 0.590) = 0.390 \times 0.242 = 0.094 \quad (44)$$
$$\delta_3^{(2)} = (0.748 - 0.6) \times 0.748 \times (1 - 0.748) = 0.148 \times 0.189 = 0.028 \quad (45)$$

6.2.2 Hidden Layer Deltas

$$\delta_1^{(1)} = [(-0.025)(0.6) + (0.094)(-0.1) + (0.028)(0.4)] \times 0.547 \times (1 - 0.547) \quad (46)$$
$$= [-0.015 - 0.009 + 0.011] \times 0.247 = -0.013 \times 0.247 = -0.003 \quad (47)$$

$$\delta_2^{(1)} = [(-0.025)(-0.4) + (0.094)(0.7) + (0.028)(0.1)] \times 0.493 \times (1 - 0.493) \quad (48)$$
$$= [0.010 + 0.066 + 0.003] \times 0.250 = 0.079 \times 0.250 = 0.020 \quad (49)$$

$$\delta_3^{(1)} = [(-0.025)(0.2) + (0.094)(-0.3) + (0.028)(0.9)] \times 0.686 \times (1 - 0.686) \quad (50)$$
$$= [-0.005 - 0.028 + 0.025] \times 0.215 = -0.008 \times 0.215 = -0.002 \quad (51)$$

$$\delta_4^{(1)} = [(-0.025)(0.8) + (0.094)(0.5) + (0.028)(-0.2)] \times 0.634 \times (1 - 0.634) \quad (52)$$
$$= [-0.020 + 0.047 - 0.006] \times 0.232 = 0.021 \times 0.232 = 0.005 \quad (53)$$

6.3 Gradient Calculations

6.3.1 Output Layer Weight Gradients

$$\frac{\partial L}{\partial \mathbf{W}^{(2)}} = \begin{pmatrix} -0.025 \times 0.547 & -0.025 \times 0.493 & -0.025 \times 0.686 & -0.025 \times 0.634 \\ 0.094 \times 0.547 & 0.094 \times 0.493 & 0.094 \times 0.686 & 0.094 \times 0.634 \\ 0.028 \times 0.547 & 0.028 \times 0.493 & 0.028 \times 0.686 & 0.028 \times 0.634 \end{pmatrix} \tag{54}$$

$$\frac{\partial L}{\partial \mathbf{W}^{(2)}} = \begin{pmatrix} -0.014 & -0.012 & -0.017 & -0.016 \\ 0.051 & 0.046 & 0.064 & 0.060 \\ 0.015 & 0.014 & 0.019 & 0.018 \end{pmatrix} \tag{55}$$

6.3.2 Output Layer Bias Gradients

$$\frac{\partial L}{\partial \mathbf{b}^{(2)}} = \begin{pmatrix} -0.025 \\ 0.094 \\ 0.028 \end{pmatrix} \tag{56}$$

6.3.3 Hidden Layer Weight Gradients

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \begin{pmatrix} -0.003 \times 1.0 & -0.003 \times 0.5 & -0.003 \times (-0.3) \\ 0.020 \times 1.0 & 0.020 \times 0.5 & 0.020 \times (-0.3) \\ -0.002 \times 1.0 & -0.002 \times 0.5 & -0.002 \times (-0.3) \\ 0.005 \times 1.0 & 0.005 \times 0.5 & 0.005 \times (-0.3) \end{pmatrix} \tag{57}$$

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \begin{pmatrix} -0.003 & -0.002 & 0.001 \\ 0.020 & 0.010 & -0.006 \\ -0.002 & -0.001 & 0.001 \\ 0.005 & 0.003 & -0.002 \end{pmatrix} \tag{58}$$

6.3.4 Hidden Layer Bias Gradients

$$\frac{\partial L}{\partial \mathbf{b}^{(1)}} = \begin{pmatrix} -0.003 \\ 0.020 \\ -0.002 \\ 0.005 \end{pmatrix} \tag{59}$$

7 Parameter Updates

With all gradients computed, we can now update the network parameters using gradient descent:

$$\mathbf{W}^{(2)}_{\text{new}} = \mathbf{W}^{(2)}_{\text{old}} - \alpha \frac{\partial L}{\partial \mathbf{W}^{(2)}} \tag{60}$$

$$\mathbf{b}^{(2)}_{\text{new}} = \mathbf{b}^{(2)}_{\text{old}} - \alpha \frac{\partial L}{\partial \mathbf{b}^{(2)}} \tag{61}$$

$$\mathbf{W}^{(1)}_{\text{new}} = \mathbf{W}^{(1)}_{\text{old}} - \alpha \frac{\partial L}{\partial \mathbf{W}^{(1)}} \tag{62}$$

$$\mathbf{b}^{(1)}_{\text{new}} = \mathbf{b}^{(1)}_{\text{old}} - \alpha \frac{\partial L}{\partial \mathbf{b}^{(1)}} \tag{63}$$

where α is the learning rate (typically a small positive number like 0.01 or 0.001).

8 Summary

This document provides a complete mathematical framework for backpropagation in a 3-4-3 neural network, including:

- Forward pass computations with sigmoid activation
- Mean squared error loss function
- Backward pass with delta calculations
- Complete gradient computations for all parameters
- Worked numerical example with specific input and target values
- Parameter update formulas for gradient descent

The computed gradients can be used to iteratively improve the network's performance through gradient descent optimization.