Step-by-Step MLP for XOR

1 Introduction

A Multi-Layer Perceptron (MLP) with one hidden layer can learn the XOR function using backpropagation and gradient descent. The training process consists of the following steps:

1. Initialize weights and biases randomly.
2. Perform forward propagation to compute predictions.
3. Compute error using a loss function.
4. Backpropagate gradients to update weights using gradient descent.
5. Train the model until convergence.

2 Forward Propagation

The forward propagation step consists of computing weighted sums followed by activation functions:

$$Z_1 = XW_1 + b_1 \tag{1}$$
$$H = \sigma(Z_1) \tag{2}$$
$$Z_2 = HW_2 + b_2 \tag{3}$$
$$\hat{Y} = \sigma(Z_2) \tag{4}$$

where:

- X is the input matrix.
- W_1, W_2 are the weight matrices for the hidden and output layers, respectively.
- b_1, b_2 are the bias vectors for the hidden and output layers.
- $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activation function.

3 Loss Function

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

$$E = \frac{1}{2}(y - \hat{Y})^2 \tag{5}$$

where:

- y is the true label.
- $\hat{Y}$ is the predicted output.

4 Backpropagation (Gradient Calculation)

To minimize the error, we compute the gradients using backpropagation.

4.1 Error at the Output Layer

$$\delta^{(2)} = (\hat{Y} - y) \cdot \sigma'(Z_2) \tag{6}$$

where:

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \tag{7}$$

is the derivative of the sigmoid function.

The gradients w.r.t. the output layer weights and bias are:

$$\frac{\partial E}{\partial W_2} = H^T \delta^{(2)} \tag{8}$$

$$\frac{\partial E}{\partial b_2} = \sum \delta^{(2)} \tag{9}$$

4.2 Error at the Hidden Layer

$$\delta^{(1)} = (\delta^{(2)} W_2^T) \cdot \sigma'(Z_1) \tag{10}$$

The gradients w.r.t. the hidden layer weights and bias are:

$$\frac{\partial E}{\partial W_1} = X^T \delta^{(1)} \tag{11}$$

$$\frac{\partial E}{\partial b_1} = \sum \delta^{(1)} \tag{12}$$

5 Gradient Descent Update Rule

The weights and biases are updated using gradient descent:

$$W^{(new)} = W^{(old)} - \eta \cdot \frac{\partial E}{\partial W} \tag{13}$$

$$b^{(new)} = b^{(old)} - \eta \cdot \frac{\partial E}{\partial b} \tag{14}$$

where η is the learning rate.

6 Conclusion

- The network learns the XOR function after several epochs of training.
- The model updates weights using backpropagation and gradient descent.
- The final predictions should approximate the desired outputs: $[0, 1, 1, 0]$.