XOR Classification Using a Feedforward Neural Network

Nour Al-Sheikh

May 4, 2025

1. Overview

This project implements a simple feedforward neural network from scratch using Python and NumPy to classify the XOR logic gate outputs. The network is trained using binary cross-entropy loss and optimized through backpropagation and gradient descent.

2. Key Components

Sigmoid Activation

The activation function used in both the hidden and output layers is the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

This function outputs values in the range $[0, 1]$, making it suitable for binary classification tasks.

Neural Network Initialization

The network is structured as follows:

- 2 input neurons
- 2 hidden neurons
- 1 output neuron

Weights are initialized using a standard normal distribution, while biases are initialized to zero.

Data Loading

Input data is read from a tab-separated text file `xor.txt`, where each row contains two binary input values and one binary output value. The data is reshaped to column-major form for matrix computation.

Forward Propagation

Forward propagation is computed in two steps:

$$\begin{aligned} \text{Hidden Layer Output:} \quad & h = \sigma(W_{hidden}X + b_{hidden}) \\ \text{Output Layer Output:} \quad & y = \sigma(W_{output}h + b_{output}) \end{aligned}$$

Loss Calculation

Binary cross-entropy loss is used to measure the difference between the predicted and actual labels:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$$

Backpropagation

Gradients of the loss function are computed with respect to all weights and biases. The chain rule is used to propagate the error backward through the layers.

Parameter Updates

Parameters are updated using gradient descent:

$$\theta \leftarrow \theta - \alpha\nabla_\theta L$$

where $\alpha = 0.3$ is the learning rate.

Training Process

The network is trained for 5000 epochs. The loss is printed every 100 epochs to track convergence.

3. Visualizations

Loss Curve

A plot of the loss over time is saved as `loss_plot.png`. It demonstrates the convergence of the training process.

Neural Network Architecture

The neural network diagram (`network_diagram.png`) includes:

- Input, hidden, and output nodes
- Color-coded weights (blue for positive, red for negative)

- Edge thickness proportional to weight magnitude
- Display of bias values inside hidden and output neurons

4. Output and Predictions

After training, the network accurately predicts the XOR output values. Example predictions:

x	y	Predicted Output
0	0	≈ 0.00
0	1	≈ 1.00
1	0	≈ 1.00
1	1	≈ 0.00

The network successfully models a non-linearly separable function like XOR.