XOR Logic Gate Neural Network

By Ayyappan T
Emali: ayyappant27@gmail.com

Introduction:

The XOR (exclusive OR) logic gate is a fundamental component in digital circuit design, capable of producing an output of true (1) only when exactly one of its inputs is true, but not both. However, XOR presents a challenge for simple linear models due to its non-linear behavior. In this context, neural networks offer a powerful solution by learning complex patterns and relationships in data.

This document explores the implementation of a neural network to solve the XOR logic gate problem using PyTorch, a popular deep learning framework. The neural network architecture consists of an input layer with two neurons (representing the two input features of the XOR gate), a hidden layer with two neurons using sigmoid activation, and an output layer with one neuron using sigmoid activation for binary classification.

The training process involves optimizing the model's parameters using stochastic gradient descent (SGD) with a binary cross-entropy loss function. The goal is to train the model to accurately predict the XOR gate's outputs for different input combinations.

Additionally, this document includes a visualization of the decision boundary learned by the neural network, showcasing how the model separates the XOR gate's inputs into distinct classes. Furthermore, a Gradio interface is provided for users to interact with the trained model, input custom values, and observe real-time predictions.

By combining neural network principles, deep learning techniques, and interactive visualization, this project demonstrates the effectiveness of neural networks in solving complex non-linear problems like the XOR logic gate, while also providing a user-friendly interface for experimentation and understanding.

Model Architecture:

The neural network architecture for solving the XOR logic gate problem consists of one hidden layer and one output layer. The input layer has two neurons corresponding to the two input features of the XOR gate. The hidden layer has two neurons with a sigmoid activation function, and the output layer has one neuron with a sigmoid activation function, representing the binary output of the XOR gate. The model architecture can be summarized as follows:

- Input Layer (2 neurons) -> Hidden Layer (2 neurons, sigmoid activation) -> Output Layer (1 neuron, sigmoid activation)

Training Process:

The model is trained using stochastic gradient descent (SGD) optimization with a learning rate of 0.1 and binary cross-entropy loss as the loss function. The training process involves iterating over the dataset for a specified number of epochs (10,000 epochs in this case). In each epoch, the model performs a forward pass to compute the predicted outputs, calculates the loss using the binary cross-entropy formula, performs a backward pass to compute gradients, and updates the weights using the optimizer. The training loop is structured as follows:

1. Initialize the model, loss function, and optimizer.
2. Iterate over the dataset for a specified number of epochs.
3. Perform a forward pass to compute predicted outputs.
4. Compute the loss using binary cross-entropy.
5. Perform a backward pass to compute gradients.
6. Update the weights using the optimizer.
7. Repeat steps 3-6 for each epoch.

Results:

After training the model, we evaluate its performance by comparing the predicted outputs to the actual outputs (ground truth). The accuracy is calculated as the percentage of correct predictions. The results of the trained model on the XOR logic gate dataset are as follows:

- Predicted Outputs:
    - [0, 1, 1, 0] (rounded to the nearest integer)
- Actual Outputs:
    - [0, 1, 1, 0]
- Accuracy:
    - 100%

The model achieves 100% accuracy on the XOR logic gate dataset, correctly predicting the XOR outputs for all input combinations.