Practice Lab Neural Network 1

Name: Om Vitole

PRN Number: 202302090016

Batch: DL [MDM] (T4)

Date of Submission: 15-01-2025

Neural Network Implementation from Scratch

Objective

This project implements a simple feedforward neural network from scratch in Python without using any in-built deep learning libraries. It covers basic components such as:

- Forward Pass
- Backpropagation
- Training using Gradient Descent

Problem Definition

The neural network is trained to solve a **binary classification problem** based on the **AND logic gate**.

Dataset:

Input (X)	Output (Y)
[0, 0]	[0]
[0, 1]	[0]
[1, 0]	[0]
[1, 1]	[1]

Neural Network Architecture

1. **Input Layer**: 2 neurons representing the binary inputs of the AND operation.

2. **Hidden Layer**: 3 neurons with Sigmoid activation function.
3. **Output Layer**: 1 neuron with Sigmoid activation function.

Task:

The task is to train a simple feedforward neural network to predict the output of the **AND operation**. The model should learn the correct classification for each combination of inputs.

- Input: X=[0,0],[0,1],[1,0],[1,1]X = [0, 0], [0, 1], [1, 0], [1, 1]X=[0,0],[0,1],[1,0],[1,1]
- Expected Output: Y=[0],[0],[0],[1]Y = [0], [0], [0], [1]Y=[0],[0],[0],[1]

The trained neural network should produce outputs close to the expected values, with high accuracy for the given binary classification problem.

Methodology

Neural Network Architecture

1. **Sigmoid Activation Function**:

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

2. **Sigmoid Derivative**:

```
def sigmoid_derivative(x):
    return x * (1 - x)
```

3. **Neural Network Class**: A class is created to define the forward pass, backpropagation, and training process. The network uses randomly initialized weights and biases, updated iteratively.
4. **Training**: The network is trained for 10,000 epochs using a learning rate of 0.05.

Forward Pass:

1. Input data is passed through the input layer.
2. The weighted sum is calculated and passed through the hidden layer with the activation function applied.
3. The hidden layer output is further processed through the output layer to generate the final prediction.

Backpropagation:

1. Errors are computed as the difference between predicted and actual output values.
2. Gradients are calculated and used to update weights and biases using Gradient Descent.

Loss Function:

- **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and actual values.

Optimization :

- **Gradient Descent**:
  Gradient Descent is used to minimize the loss function by iteratively updating the weights and biases in the direction of the negative gradient. This helps the model improve its predictions over time by reducing the error between predicted and actual outputs.

- **Learning Rate**:
  The learning rate is a key parameter that determines the size of each step during the weight and bias update process.
  - A small learning rate results in slower but more stable convergence.
  - A large learning rate allows faster progress but may risk overshooting the optimal solution.

By repeating these steps over many iterations, the network becomes more accurate, learning the patterns in the data to minimize the loss.

Declaration

I, **Om Vitole**, confirm that the work submitted in this assignment is my own and has been completed following academic integrity guidelines. The code is uploaded to my GitHub repository, and the repository link is provided below: **GitHub Repository Link**:

https://github.com/omvitole7/Neural_network_from_scratch

Signature:

Om Dnyaneshwar Vitole