DEEP LEARNING LAB ASSIGNMENT 1

Name: Manjiri Prashant Netankar

Roll no: 42

PRN: 202201040206

Batch: T4

---

1. Objective:

Implement a simple feedforward neural network from scratch in Python without using any in-built deep learning libraries. This implementation will focus on basic components like forward pass, backward propagation (backpropagation), and training using gradient descent

2. Methodology:

Neural Network Architecture:

• **Input Layer:** 2 neurons (x1=0.35x_1 = 0.35x1=0.35 and x2=0.9x_2 = 0.9x2=0.9).

• **Hidden Layer:** 2 neurons (H3H_3H3 and H4H_4H4) with the Sigmoid activation function.

• **Output Layer:** 1 neuron (O5O_5O5) with the Sigmoid activation function for binary classification.

• **Weights and Biases:**

- Weights from input to hidden layer (W13=0.1W_{13} = 0.1W13=0.1, W14=0.4W_{14} = 0.4W14=0.4, W23=0.8W_{23} = 0.8W23=0.8, W24=0.6W_{24} = 0.6W24=0.6).
- Bias for hidden layer (B3=0.0B_{3} = 0.0B3=0.0, B4=0.0B_{4} = 0.0B4=0.0).
- Weights from hidden to output layer (W35=0.3W_{35} = 0.3W35=0.3, W45=0.9W_{45} = 0.9W45=0.9).
- Bias for output layer (B5=0.0B_{5} = 0.0B5=0.0).

Forward Pass:

1. Hidden Layer Computation:

$$Z_{\text{hidden}} = X \cdot W_{\text{input\_hidden}} + B_{\text{hidden}}$$

$$A_{\text{hidden}} = \frac{1}{1 + e^{-Z_{\text{hidden}}}}$$

2. Output Layer Computation:

$$Z_{\text{output}} = A_{\text{hidden}} \cdot W_{\text{hidden\_output}} + B_{\text{output}}$$

$$A_{\text{output}} = \frac{1}{1 + e^{-Z_{\text{output}}}}$$

Backpropagation:

1. **Output Layer Error:**

$$\text{Error}_{\text{output}} = Y - A_{\text{output}}$$

$$\Delta_{\text{output}} = \text{Error}_{\text{output}} \cdot A_{\text{output}} \cdot (1 - A_{\text{output}})$$

2. **Hidden Layer Error:**

$$\text{Error}_{\text{hidden}} = \Delta_{\text{output}} \cdot W_{\text{hidden\_output}}^{T}$$

$$\Delta_{\text{hidden}} = \text{Error}_{\text{hidden}} \cdot A_{\text{hidden}} \cdot (1 - A_{\text{hidden}})$$

3. **Weight and Bias Updates:**

$$W_{\text{hidden\_output}} += A_{\text{hidden}}^{T} \cdot \Delta_{\text{output}} \cdot \text{Learning Rate}$$

$$B_{\text{output}} += \sum \Delta_{\text{output}} \cdot \text{Learning Rate}$$

$$W_{\text{input\_hidden}} += X^{T} \cdot \Delta_{\text{hidden}} \cdot \text{Learning Rate}$$

$$B_{\text{hidden}} += \sum \Delta_{\text{hidden}} \cdot \text{Learning Rate}$$

❖ **Loss Function:**

- **Mean Squared Error (MSE):**

$$\text{Loss} = \frac{1}{n} \sum (Y - A_{\text{output}})^2$$

❖ **Optimization:**

- The neural network uses Gradient Descent with a learning rate of 0.1 to minimize the loss by updating the weights and biases.

❖ **Declaration:**

I, Manjiri Netankar, confirm that the work submitted in this assignment is my own and has been completed by following the academic integrity guidelines. The code is uploaded on my GitHub repository account, and the repository link is provided below:

GitHub Repository Link :

https://github.com/ManjiriNetankar1505/assignment_1_deep_learning

---

Colab Link :

https://colab.research.google.com/drive/1cnDBPzeyigjgrDL-X4sbTa562WkexrsS?usp=sharing