Neural Networks Lab1

Harshini Kavuru
kavuru.7@osu.edu

1 Methodology

The inputs and outputs for the given parity problem are as follows:

Inputs	Outputs
0000	0
0001	1
0010	1
0011	0
0100	1
0101	0
0110	0
0111	1
1000	1
1001	0
1010	0
1011	1
1100	0
1101	1
1110	1
1111	0

Table 1: Parity Problem: 4-bit Inputs and Corresponding Outputs

To solve this 4-bit parity problem we are implementing a neural network with 4 inputs and 4 units in hidden layer and 1 unit in the output layer. All the inputs are connected to all the hidden layer neurons.

In the neural network architecture, the following weight matrices and bias vectors are used:

- **W1**: Represents the weights matrix for the connections between the input layer and the hidden layer. Since the input layer has 4 neurons and the hidden layer has 4 neurons, the dimensions of $W1$ are 4×4.
- **B1**: Represents the bias vector for the 4 hidden neurons. The dimension of $B1$ is 4×1, where each element corresponds to the bias applied to one hidden neuron.
- **W2**: Represents the weights matrix for the connections between the hidden layer and the output layer. The dimensions of $W2$ are 4×1.
- **B2**: Represents the bias for the output layer. The dimension of $B2$ is 1×1.

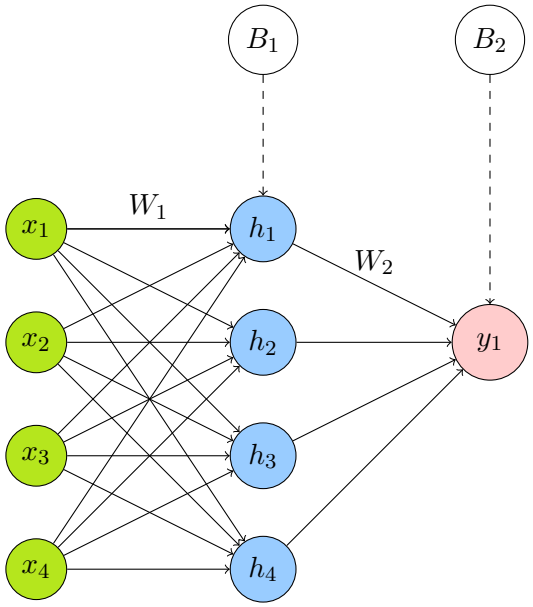


Figure 1: Neural Network with Updated Labels and Spacing

2 Code

The forward pass computes the activations of the hidden and output layers using weight matrices $W1$, $W2$ and bias vectors $B1$, $B2$, applying the sigmoid activation function:

$$\text{hidden\_output} = \sigma(X \cdot W1 + B1)$$

$$\text{final\_output} = \sigma(\text{hidden\_output} \cdot W2 + B2)$$

The backward pass calculates the error and gradients:

$$\text{error} = y - \text{final\_output}$$

$$d_{\text{output}} = \text{error} \times \sigma'(\text{final\_output})$$

$$W2_{\text{gradient}} = \text{hidden\_output}^T \cdot d_{\text{output}}$$

$$B2_{\text{gradient}} = \sum d_{\text{output}}$$

$$\text{hidden\_error} = (d_{\text{output}} \cdot W2^T) \times \sigma'(\text{hidden\_output})$$

$$W1_{\text{gradient}} = X^T \cdot \text{hidden\_error}$$

$$B1_{\text{gradient}} = \sum \text{hidden\_error}$$

The weights and biases are updated using momentum:

$$\Delta W = \text{momentum} \times \Delta W + \text{learning\_rate} \times \text{gradient}$$

$$W = W + \Delta W$$

$$\Delta B = \text{momentum} \times \Delta B + \text{learning\_rate} \times \text{gradient}$$

$$B = B + \Delta B$$

This process continues until the error falls below the threshold of 0.05.

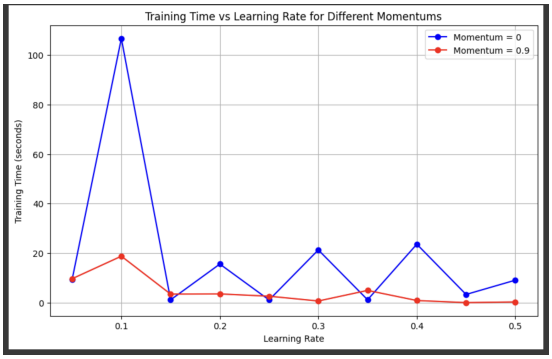
3 Results

Learning Rate	Epochs	Time (s)
0.05	37,925	9.3385
0.1	1,072,717	106.6522
0.15	13,947	1.1905
0.2	164,138	15.6219
0.25	12,893	1.1367
0.3	214,821	21.3228
0.35	13,084	1.2109
0.4	236,695	23.6374
0.45	29,903	3.3551
0.5	100,712	9.1161

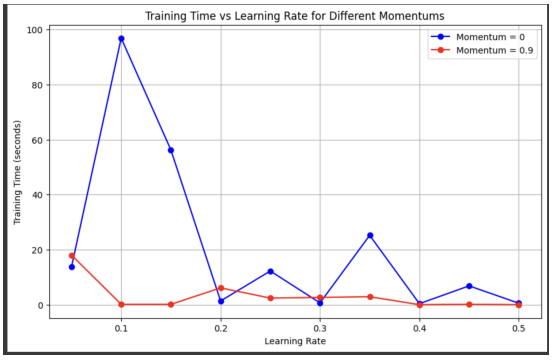
Table 2: Momentum = 0

Learning Rate	Epochs	Time (s)
0.05	104,300	9.6979
0.1	198,692	18.7945
0.15	40,039	3.5030
0.2	41,951	3.5737
0.25	31,080	2.6484
0.3	8,534	0.7185
0.35	45,872	4.9865
0.4	10,856	0.9103
0.45	833	0.0651
0.5	4,178	0.3396

Table 3: Momentum = 0.9

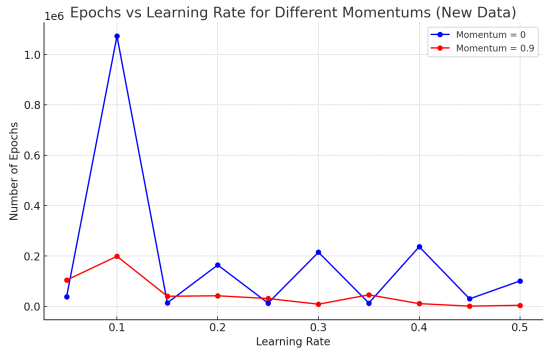


(a) Training Time vs Learning Rate (Graph 1)

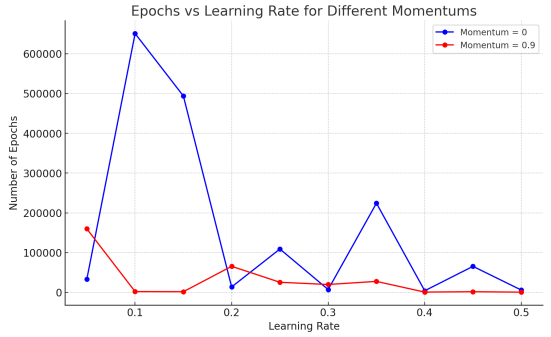


(b) Training Time vs Learning Rate (Graph 2)

Figure 2: Comparison of Training Time vs Learning Rate for Different Momentums for different runs



(a) Number of Epochs vs Learning Rate (Graph 1)



(b) Number of Epochs vs Learning Rate (Graph 2)

Figure 3: Comparison of Number of Epochs vs Learning Rate for Different Momentums for different runs

Figures 2 and 3 represent four graphs from two different runs of the MLP program, which were performed to check for any variation in the results. The inference based on these graphs is as follows:

1. **Momentum = 0.9 consistently improves convergence**, requiring fewer epochs and significantly reducing training time compared to Momentum = 0, especially at lower learning rates (0.05 to 0.2).

2. **Momentum = 0 experiences instability at low learning rates**, with both epochs and training time spiking dramatically, particularly at a learning rate of 0.1, in contrast to the smoother performance of Momentum = 0.9.

3. **At higher learning rates (0.3 and above)**, the difference in performance between Momentum = 0 and Momentum = 0.9 becomes smaller, though Momentum = 0.9 still demonstrates a slight advantage in both epochs and training time.