Introduction

Artificial Neural Networks are fundamental tools in machine learning. The Perceptron, introduced by Frank Rosenblatt in 1957, is one of the simplest neural networks. Single Layer Perceptrons can only solve linearly separable problems, while Multi-Layer Perceptrons can handle complex, non-linear problems.

In this lab, we implement both Single Layer Perceptron and Multi-Layer Perceptron to solve logic gate problems. The AND gate is linearly separable and can be solved by Single Layer Perceptron, while the XNOR gate is non-linearly separable and requires Multi-Layer Perceptron for successful implementation.

Objectives

The main objectives of this lab are:

1. To understand the difference between linearly separable and non-linearly separable problems
2. To implement Single Layer Perceptron for solving AND gate
3. To implement Multi-Layer Perceptron for solving XNOR gate
4. To demonstrate the limitation of Single Layer Perceptron on non-linear problems
5. To compare the performance of both algorithms on logic gate implementations
6. To gain hands-on experience with neural network programming in Python

AND Gate Data Analysis

Truth Table

The AND gate truth table shows the expected outputs for all possible input combinations in the order used for training:

Table 1: AND Gate Truth Table

Input A	**Input B**	**Output (A AND B)**
1	1	1
1	0	0
0	1	0
0	0	0

Training Progress Overview

The Single Layer Perceptron was trained on AND gate data for 100 epochs. The table below shows the training progress at different intervals:

Table 2: AND Gate Perceptron Training Progress

Epoch	Weights	Bias	Predictions
1	[-0.0168, 0.0212]	-0.1818	[0, 0, 0, 0]
10	[0.1832, 0.1212]	-0.2818	[1, 0, 0, 0]
20	[0.1832, 0.1212]	-0.2818	[1, 0, 0, 0]
30	[0.1832, 0.1212]	-0.2818	[1, 0, 0, 0]
40	[0.1832, 0.1212]	-0.2818	[1, 0, 0, 0]
50	[0.1832, 0.1212]	-0.2818	[1, 0, 0, 0]
60	[0.1832, 0.1212]	-0.2818	[1, 0, 0, 0]
70	[0.1832, 0.1212]	-0.2818	[1, 0, 0, 0]
80	[0.1832, 0.1212]	-0.2818	[1, 0, 0, 0]
90	[0.1832, 0.1212]	-0.2818	[1, 0, 0, 0]
100	[0.1832, 0.1212]	-0.2818	[1, 0, 0, 0]

Final Results

Table 3: AND Gate Final Results

Parameter	Value
Final Weights	[0.1832, 0.1212]
Final Bias	-0.2818
Final Predictions	[1, 0, 0, 0]

Analysis

The training results show that the Single Layer Perceptron successfully learned the AND gate logic. After the first few epochs, the weights and bias stabilized, and the final predictions matched the expected AND gate outputs: [1, 0, 0, 0] for inputs [1,1], [1,0], [0,1], [0,0] respectively. The convergence was achieved quickly, demonstrating the linear separability of the AND gate problem.

Decision Boundary Visualization

The graph below shows the decision boundary learned by the Single Layer Perceptron for the AND gate problem. The decision boundary is a straight line that separates the input space into two regions: one for class 0 (output 0) and one for class 1 (output 1).

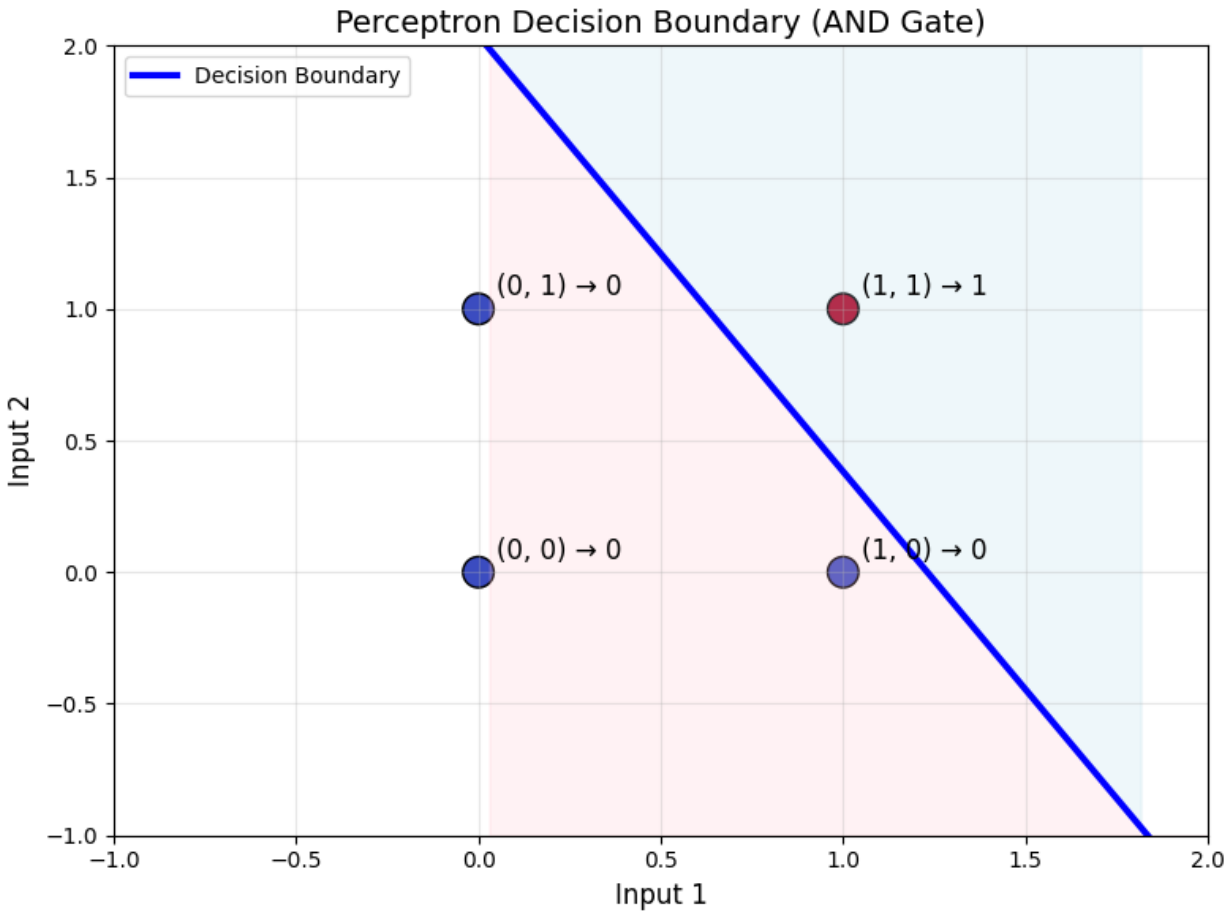

Figure 1: AND Gate Perceptron Decision Boundary

The decision boundary clearly separates the input points, with the point (1,1) belonging to class 1 and all other points ([1,0], [0,1], [0,0]) belonging to class 0. The linear nature of the decision boundary confirms that the AND gate is a linearly separable problem, which is why Single Layer Perceptron can successfully solve it. The weight vector [0.1832, 0.1212] and bias -0.2818 define this separating hyperplane in the 2D input space.

XNOR Gate Data Analysis

Truth Table

The XNOR gate truth table shows the expected outputs for all possible input combinations in the order used for training:

Table 4: XNOR Gate Truth Table

Input A	Input B	Output (A XNOR B)
0	0	1
0	1	0
1	0	0
1	1	1

Training Progress Overview

The Multi-Layer Perceptron was trained on XNOR gate data for 1000 epochs. The table below shows the training progress at different intervals:

Table 5: XNOR Gate MLP Training Progress

Epoch	Layer 1→2 Weights	Layer 2→3 Weights	Accuracy
1	[[-0.24, 0.89, 0.46, 0.20], [-0.68, -0.69, -0.89, 0.73]]	[[0.72], [-0.60], [-0.73], [-0.66]]	50.00%
100	[[-1.15, 2.70, 2.74, 0.84], [-1.78, -1.82, -3.22, 1.29]]	[[1.85], [3.54], [-3.70], [-1.50]]	100.00%
200	[[-1.47, 3.18, 2.93, 1.06], [-1.90, -2.21, -3.54, 1.40]]	[[2.22], [4.53], [-4.60], [-1.79]]	100.00%
300	[[-1.58, 3.36, 3.01, 1.14], [-1.95, -2.35, -3.67, 1.44]]	[[2.38], [4.98], [-5.02], [-1.92]]	100.00%
400	[[-1.65, 3.47, 3.05, 1.19], [-1.99, -2.43, -3.76, 1.47]]	[[2.49], [5.27], [-5.30], [-2.01]]	100.00%
500	[[-1.70, 3.55, 3.08, 1.22], [-2.01, -2.49, -3.82, 1.49]]	[[2.57], [5.49], [-5.52], [-2.07]]	100.00%
600	[[-1.73, 3.61, 3.11, 1.25], [-2.03, -2.54, -3.86, 1.50]]	[[2.63], [5.66], [-5.69], [-2.12]]	100.00%
700	[[-1.76, 3.66, 3.13, 1.27], [-2.04, -2.57, -3.90, 1.51]]	[[2.68], [5.81], [-5.83], [-2.17]]	100.00%
800	[[-1.78, 3.70, 3.14, 1.29], [-2.05, -2.60, -3.93, 1.52]]	[[2.73], [5.93], [-5.95], [-2.20]]	100.00%
900	[[-1.80, 3.73, 3.15, 1.30], [-2.06, -2.62, -3.96, 1.53]]	[[2.76], [6.04], [-6.06], [-2.24]]	100.00%
1000	[[-1.82, 3.76, 3.16, 1.32], [-2.07, -2.64, -3.98, 1.54]]	[[2.80], [6.13], [-6.15], [-2.26]]	100.00%

Final Results

Table 6: XNOR Gate Final Results

Parameter	Value
Total Epochs	1000
Final Accuracy	100.00%
Final Predictions	[1, 0, 0, 1]
Expected Outputs	[1, 0, 0, 1]

Analysis

The training results show that the Multi-Layer Perceptron successfully learned the XNOR gate logic. Unlike the AND gate, the XNOR gate is not linearly separable, which is why a Single Layer Perceptron cannot solve it. The MLP achieved 50% accuracy initially (random guessing level) and then gradually improved to 100% accuracy by epoch 100. The weights continued to evolve throughout the 1000 epochs, indicating ongoing optimization even after achieving perfect classification. The final predictions [1, 0, 0, 1] perfectly match the expected XNOR outputs for inputs [0,0], [0,1], [1,0], [1,1] respectively. This demonstrates the capability of Multi-Layer Perceptrons to solve non-linearly separable problems through the use of hidden layers and non-linear activation functions.

Decision Boundary Visualization

The graph below shows the non-linear decision boundary learned by the Multi-Layer Perceptron for the XNOR gate problem. Unlike the linear decision boundary of the AND gate, the XNOR decision boundary is non-linear, consisting of two separate regions where the output is 1.

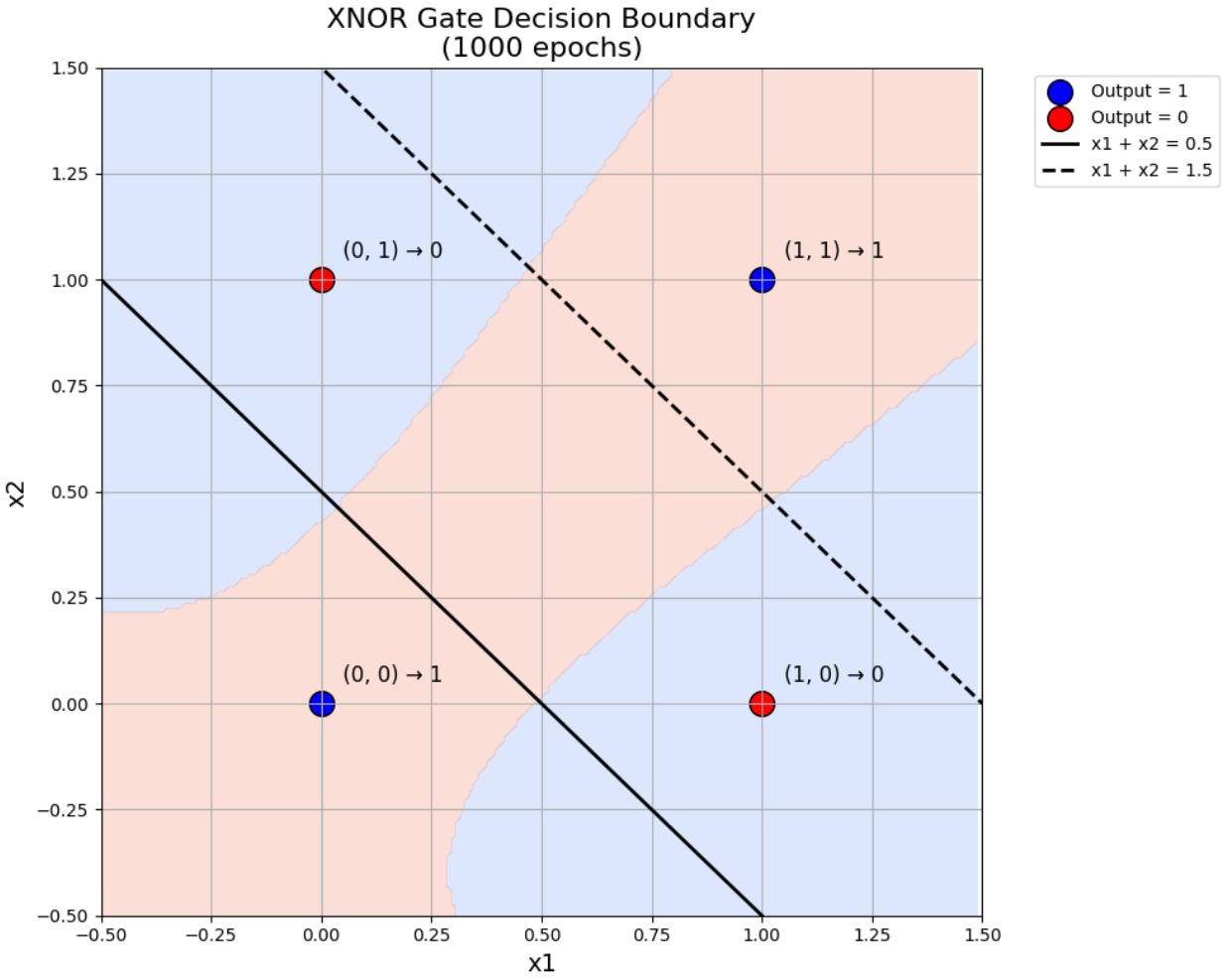

Figure 2: XNOR Gate MLP Decision Boundary

The complex, non-linear nature of the decision boundary confirms that the XNOR gate is a non-linearly separable problem. The MLP successfully creates this complex boundary through its hidden layer representations, demonstrating why multi-layer architectures are necessary for such problems.

MLP Architecture Visualization

The diagram below shows the architecture of the Multi-Layer Perceptron used for solving the XNOR gate problem. The network consists of an input layer with 2 neurons (for the two inputs), a hidden layer with 4 neurons, and an output layer with 1 neuron.

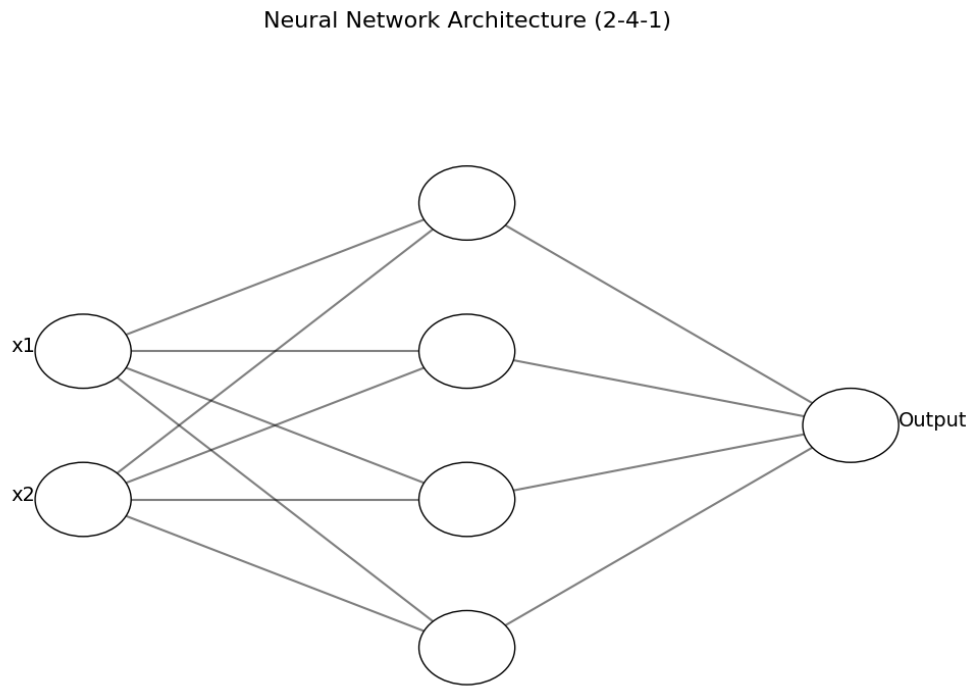

Figure 3: XNOR Gate MLP Network Architecture

The visualization illustrates how the input signals propagate through the network layers, with each connection having its own weight parameter. The hidden layer neurons apply activation functions to transform the weighted inputs, enabling the network to learn non-linear decision boundaries necessary for solving the XNOR problem.

Conclusion

This lab demonstrated the capabilities and limitations of Single Layer Perceptrons (SLPs) and Multi-Layer Perceptrons (MLPs) through AND and XNOR gate implementations. The SLP successfully learned the linearly separable AND gate, while failing on the non-linear XNOR gate. The MLP achieved 100% accuracy on XNOR by forming complex decision boundaries through hidden layers. Key findings include:

- SLPs are effective for linear problems (AND gate) but fail on non-linear cases (XNOR)
- MLPs solve non-linear problems via hidden layers and activation functions
- Decision boundary visualization confirmed the linear/non-linear separation

The results highlight the importance of network architecture selection based on problem complexity. Future work could explore deeper architectures and advanced optimization techniques.