Computer Science & Information Systems

Deep Neural Networks - Lab sheet 1

Perceptron for Logic Gates

Prepared by Seetha Parameswaran

1 Objective

The objective is to

- Understand the perceptron as the simplest artificial neuron.
- Implement perceptron learning algorithm from scratch.
- Apply perceptron to learn logic gates (AND, OR, NOR).
- Understand limitations: XOR gate cannot be learned by single perceptron.

2 Steps to be Performed

- **Tool:** Python3
- **Libraries required:** numpy, matplotlib, pandas, sklearn
- **Input:** Truth tables for logic gates (AND, OR, XOR)
- **Model:** Perceptron (Single Neuron)
- **Implementation:** L1-Perceptron.ipynb

2.1 Steps

- Import required Python libraries.
- Define truth tables for logic gates as training data.
- Understand the problem: Binary classification with binary inputs.
- Create a Perceptron class with step activation function.
- Initialize weights randomly and bias to zero.
- Implement perceptron learning rule for weight updates.
- Train the perceptron on each logic gate separately.
- Predict outputs for all input combinations.
- Visualize decision boundaries for each logic gate.
- Compute accuracy for each gate.

- Demonstrate XOR problem: single perceptron cannot learn XOR.
- Visualize why XOR is not linearly separable.

2.2 Mathematical Formulation

Model (Linear Combination):

$$z = \mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + b \tag{1}$$

Step Activation Function:

$$a = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \qquad \text{OR} \qquad a = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases} \tag{2}$$

Perceptron Learning Rule:
For each misclassified example $\mathbf{x}^{(i)}$:

$$\mathbf{w} := \mathbf{w} + \eta(y^{(i)} - a^{(i)})\mathbf{x}^{(i)} \tag{3}$$

$$b := b + \eta(y^{(i)} - a^{(i)}) \tag{4}$$

where:

- η is the learning rate
- $y^{(i)}$ is the true label
- $a^{(i)}$ is the predicted output

Update only when misclassified:

- If $y^{(i)} = 1$ and $a^{(i)} = 0$: increase weights
- If $y^{(i)} = 0$ and $a^{(i)} = 1$: decrease weights
- If correctly classified: no update

3 Results

- Perceptron successfully learned AND gate (100% accuracy).
- Perceptron successfully learned OR gate (100% accuracy).
- Perceptron failed to learn XOR gate .
- Decision boundaries visualized as straight lines for linearly separable gates.
- XOR problem demonstrated: no single straight line can separate XOR outputs.
- Convergence typically achieved within 10-20 epochs for learnable gates.
- Step activation function produces binary outputs {0, 1} or {-1, +1}.

4 Observation

- Perceptron learns linear decision boundaries to separate classes.
- Step activation function makes perceptron suitable for binary classification.
- Perceptron learning rule updates weights only for misclassified examples.
- AND, OR gates are linearly separable, hence learnable.
- XOR gate is not linearly separable, fundamental limitation of single perceptron.
- XOR requires non-linear decision boundary (cannot be drawn with one straight line).
- Multiple perceptrons (multi-layer network) needed to solve XOR problem.
- Perceptron convergence theorem: guaranteed to converge if data is linearly separable.
- Learning rate affects convergence speed but not final solution for separable data.

4.1 Logic Gate Truth Tables

AND Gate (Linearly Separable):

x_1	x_2	Output
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate (Linearly Separable):

x_1	x_2	Output
0	0	0
0	1	1
1	0	1
1	1	1

XOR Gate (NOT Linearly Separable):

x_1	x_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

4.2 Key Insight: Linear Separability

A dataset is **linearly separable** if there exists a hyperplane (line in 2D, plane in 3D) that perfectly separates the two classes. Single perceptron can only learn linearly separable functions.

Solution to XOR: Use multi-layer perceptron (MLP) with at least one hidden layer to create non-linear decision boundaries.