adani University

Department of Computer Science and Engineering (AI-ML)
Deep Learning: Principles and Practices
(EICDT24404)

Objective

Implement a **single-neuron classifier from scratch** and explore how **activation functions** affect learning and decision boundaries.

Part A — Implement Perceptron (Step Activation) *(Classic ML)*

Steps

1. Setup Python environment (NumPy, Matplotlib)
2. Generate synthetic dataset (linearly separable)
3. Implement:
    a. dot product
    b. bias addition
    c. step activation
4. Train using Perceptron learning rule
5. Plot:
    a. dataset points
    b. decision boundary

Deliverables

- Weight and bias values after training
- Decision boundary plot (final)

Part B — Upgrade to "Neuron with Sigmoid" *(DL direction, but still 1 neuron)*

What changes from Part A?

- Activation changes:
  - Step → Sigmoid
- Output becomes probability-like
- Training uses **gradient descent for a single neuron**

Steps

6. Replace step activation with:
   a. Sigmoid activation
7. Use loss function:
   a. Binary cross entropy (or MSE, your choice)
8. Train using gradient descent weight update (single neuron)
9. Plot:
   a. loss vs epochs
   b. decision boundary

Deliverables

- Loss curve plot
- Decision boundary plot
- Accuracy comparison (Step vs Sigmoid)

Part C — Compare on Linear vs XOR dataset *(Motivation for MLP later)*

Steps

10. Run BOTH models (Step-perceptron and Sigmoid-neuron) on:

- linearly separable dataset
- XOR dataset (expected failure)

11. Document:

- accuracy results
- reason for failure

Deliverables

- Table comparing performance
- Short explanation: why XOR fails