Department of Electrical and Computer Engineering
Pak-Austria Fachhochschule: Institute of Applied Sciences & Technology

Subject: Artificial Neural Network (ANN)

Assignment 1

Foundational Models – Perceptron vs. Adaline and the XOR Problem

Class: BSAI F23 Red

Due Date: 21-02-2026 @ 11:59 (Upload via MS Teams)

Weightage: 2.5%

Program Learning Outcomes (PLOs)

- **PLO 1:** Foundational Knowledge Application in Neural Networks
- Demonstrate the ability to apply foundational neural network concepts, including the McCulloch-Pitts neuron model, Perceptron learning rule, and Adaline delta rule, to solve simple binary classification problems.
- **PLO 2:** Architectural Analysis and Limitation Identification
- Critically analyze the architectures of single-layer networks (Perceptron and Adaline) and identify their fundamental limitation in solving non-linearly separable problems, such as the XOR function.

Student Learning Outcomes (SLOs)

- **SLO 1:** Implementation of Classical Learning Rules
  Implement the Perceptron learning algorithm and the Adaline (LMS/delta) learning rule from scratch using Python and NumPy to classify linearly separable data.
- **SLO 2:** Comparative Evaluation and Critical Analysis

Compare and contrast the learning rules, convergence behavior, and outputs of the Perceptron and Adaline models. Critically analyze why these single-layer networks fail for the XOR problem and propose a multi-layer architecture as a solution..

Intellectual Skills

On successful completion of this assignment, you will be able to:

- Differentiate between the error-driven learning of the Perceptron and the gradient-descent-based LMS rule of Adaline.
- Understand the concept of linear separability and its implications for the representational power of single-layer neural networks.
- Critically analyze the historical significance of the XOR problem in neural network research.
- Relate the choice of activation function (step vs. linear) to the learning algorithm and model output.

Transferable Skills

- **Analytical & Critical Thinking:** Moving beyond simply running code to interpreting *why* the Perceptron learning rule converges for AND/NAND/OR but not for XOR.
- **Problem Solving:** Addressing the challenge of non-linearly separable problems by designing a simple multi-layer network.
- **Communication:** Explaining technical concepts like decision boundaries, weight updates, and the limitations of single-layer models in a clear and concise report.

About Assessment

This is an individual assignment. You will act as a machine learning historian, tasked with implementing and analyzing two of the earliest neural network models: the Perceptron and the Adaline. Your goal is to demonstrate their functionality on simple linearly separable problems (like AND and OR) and then experimentally prove their shared limitation by attempting to solve the non-linearly separable XOR problem. You will implement both models from scratch. For the XOR problem, you will then propose a multi-layer network architecture that can solve it, foreshadowing the need for more complex models like the Multi-Layer Perceptron (MLP):

Your Task

1. Dataset Creation and Familiarization

- Analytical & Critical Thinking: Moving beyond simply running code to interpreting why the Perceptron learning rule converges for AND/NAND/OR but not for XOR.
- Problem Solving: Addressing the challenge of non-linearly separable problems by designing a simple multi-layer network.
- Communication: Explaining technical concepts like decision boundaries, weight updates, and the limitations of single-layer models in a clear and concise report.

2. Model Implementation (from scratch)

You will implement two classes or sets of functions in Python using only NumPy.

- **Perceptron**
  - Implement the Perceptron learning rule as described in your slides: weight updates occur only when the actual output (`y`) differs from the target (`t`), using the formula `Δw_i = α * x_i * t`.
  - Use a **binary step activation function** (e.g., output = 1 if net input >= 0 else -1).
  - Train the model on the **AND** and **OR** datasets until convergence.
- **Adaline (Adaptive Linear Neuron)**
  - Implement the Adaline learning rule (Widrow-Hoff or delta rule) as described: `Δw_i = α * (t - y_in) * x_i`. Note that the error is calculated using the net input (`y_in`) *before* the activation function.
  - Use a **linear activation function** during training (i.e., the output is the net input, `y_in`). For final classification, you will apply a threshold (e.g., predict class 1 if `y_in >= 0` else -1).

- Train the model on the **AND** and **OR** datasets. Track the Mean Squared Error (MSE) over epochs to observe convergence.

3. The XOR Problem – Demonstrating the Limitation

- **Attempt to Train:** Take your trained (or re-initialized) Perceptron and Adaline models and attempt to train them on the **XOR** dataset.
- **Analyze the Results:**
  - Document what happens. Does the Perceptron learning rule converge? Why or why not? Show the inequality contradictions as explained in your slides.
  - For Adaline, plot the MSE over epochs. Does it converge to a near-zero error? If not, explain what the final decision boundary looks like and why it cannot separate the XOR data. Show the equation of the final decision boundary line.

4. Proposing a Solution – Multi-Layer Network

- **Conceptual Design:** Based on the solution described in your lecture slides (e.g., in Week 2 -- Perceptrop, Structure, Adaline.ppt), design a multi-layer network (with a hidden layer) that *can* solve the XOR problem.
- **Network Architecture:** Clearly draw and describe the architecture. You should have:
  - An input layer with 2 neurons (for x1, x2) and a bias.
  - A hidden layer with 2 neurons. Specify their activation function (e.g., step).
  - An output layer with 1 neuron. Specify its activation function.
- **Manual Weight Setting (No Learning):** Explain the role of each neuron in the hidden layer. For example, you can show that the first hidden neuron acts as an AND gate for one pattern, and the second for another. Then, explain how the output neuron combines these to produce the final XOR output. Provide the specific weight values that would make this network work.

5. Report and Code Submission

- Write a report (approx. 400 words) that includes:
  - Your analysis of linear separability with plots for AND, OR, and XOR.
  - A comparison of the learning rules and convergence behavior of Perceptron and Adaline for the AND/OR problems.
  - A detailed explanation and proof of why both models fail on the XOR problem.
  - Your proposed multi-layer network solution for XOR, including a diagram and the manually set weights.

  - A discussion on why hidden layers with non-linear activation functions are necessary.
- Append your well-commented Python code (.ipynb).

Referencing & Formatting

- **Referencing:** IEEE style – cite academic papers, official documentation, and authentic websites only.
- **Submission:** One Word/PDF report **plus** the Python file/notebook.
- **Formatting:** Arial font size 11, 1.5 line spacing.
- **Anonymity: Roll number** on the front page only. Do **not** write your name.

Contact: abid.ali@paf-iast.edu.pk

Marking Rubric: Assignment 2

Total Marks: 100

Criterion (Weight)	Exemplary (80-100%)	Satisfactory (70-79%)	Developing (60-69%)	Unsatisfactory (<60%)	PLO/SLO Map
1. Dataset & EDA (15%)	Datasets for AND, OR, XOR are correctly created in bipolar format. Plots are clear, well-labeled, and perfectly illustrate linear (in)separability.	Datasets are correct. Plots are included but may lack clear labels or perfect visual clarity.	Datasets have minor errors (e.g., using 0/1 instead of -1/1). Plots are missing or incorrect.	Datasets are incorrect or missing.	PLO 2, SLO 2

2. Perceptron Implementation (20%)	Perceptron is correctly implemented from scratch and converges on AND/OR. Code is clean, efficient, and well-commented. The step activation function is correctly used.	Perceptron is correctly implemented and converges. Code works but may be poorly structured or commented.	Perceptron implementation has minor bugs or does not reliably converge on AND/OR.	Perceptron is not implemented or code does not run.	PLO 1, SLO 1
3. Adaline Implementation (20%)	Adaline is correctly implemented from scratch with the delta rule. MSE is tracked and plotted, showing clear convergence for AND/OR. The distinction between training	Adaline is correctly implemented and converges. MSE plot is present.	Adaline implementation has minor bugs, or the delta rule is incorrectly applied.	Adaline is not implemented or code does not run.	PLO 1, SLO 1

	(linear) and classification (threshold) is clear.				
4. XOR Analysis (25%)	Provides a clear, mathematically sound explanation (including inequalities) for Perceptron's failure on XOR. Shows Adaline's MSE plot and explains why the linear boundary fails. The proposed multi-layer solution is correct, with a clear diagram and justified weights.	Explains the failure of both models qualitatively. Proposes a multi-layer solution but may lack precise weights or a complete diagram.	Fails to fully explain the failure of one model. Proposed solution is vague or incorrect.	No clear explanation of XOR failure. No solution proposed.	PLO 2, SLO 2
5. Report Quality (20%)	Excellent structure, clear diagrams, professional	Well-structured, minor writing errors,	Poor structure, several errors, diagrams	Disorganized, missing references to the lecture material,	Transferable Skills

	writing, no errors. Code is exceptionally well-documented. Provides deep insights into the historical significance of these models.	good diagrams, code works.	missing or unclear.	code is unreadable.	