Perceptron From Scratch: A Beginner's Guide

Generated via Python

1. Title & Objective

- Project: Implement a Perceptron Classifier from Scratch.
- Objective: Predict Student Pass/Fail using Study Hours & Attendance.
- Tools: Python, NumPy, Matplotlib.
- Goal: Understand the mathematical foundation of Neural Networks.

2. Problem Statement

- Question: Can we predict if a student passes based on effort?
- Input:
  - Study Hours (0-100)
  - Attendance % (40-100)
- Output:
  - Pass (1)
  - Fail (0)
- Challenge: Find the "line" that separates passing students from failing ones.

3. Real-World Use Cases

- Finance: Loan Approval (Income vs Debt).
- Healthcare: Diagnosis (Blood Pressure vs Age).
- Email: Spam Detection (Keyword Count vs Sender Trust).
- Safety: Fraud Detection (Transaction Amount vs Location).

4. Input Data

- Features (X):
  - x_1: Study Hours
  - x_2: Attendance
- Labels (y):
  - 1: Pass
  - 0: Fail
- Synthetic Data: We generated 100 random examples to train our model.

5. Concepts Used (High Level)

- Perceptron: A single artificial neuron.
- Linear Classifier: Separates data with a straight line.
- Supervised Learning: Learning from labeled examples (Right/Wrong answers).
- Iterative Learning: Improving bit-by-bit over many "Epochs".

6. Concepts Breakdown

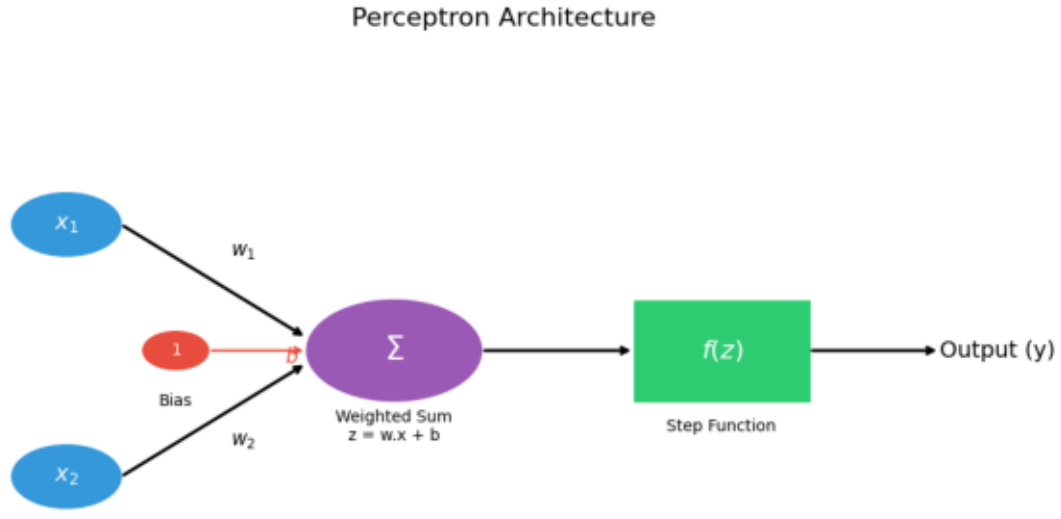
- Weights (W): How important each input is. (e.g., Study > Attendance).
- Bias (b): Shifts the decision boundary (Threshold).
- Step Function: The "Activation". Converts math result into 0 or 1.
    - If Sum > 0 → 1
    - Else → 0

Visual Concept

Inputs --Weights--> Sum

Bias --> Sum

Sum --Step Function--> Output



7. Step-by-Step Solution Flow

1. **Initialize**: Start with random weights (random guess).
2. **Predict**: Guess "Pass" or "Fail" for a student.
3. **Check**: Was the guess right?
4. **Update**: If wrong, nudge weights closer to the correct answer.
5. **Repeat**: Do this for all students, many times.

8. Code Logic Summary

- Linear Sum: $Z = (x_1 \cdot w_1) + (x_2 \cdot w_2) + b$
- Activation: $Prediction = Step(Z)$
- Update Rule:
  - $Error = Target - Prediction$
  - $W_{new} = W_{old} + (LearningRate \times Error \times Input)$

9. Important Functions

- `__init__`: Sets Learning Rate (0.01) and Epochs (100).
- `fit(X, y)`: The training loop. Adjusts weights when mistakes happen.
- `predict(X)`: Uses learned weights to classify new students.