Build a Perceptron model to solve a binary classification problem and apply the Perceptron learning law.

Abstract

Artificial Neural Networks (ANN) form the foundation of modern machine learning systems. The Perceptron is one of the earliest and simplest neural network models designed for binary classification tasks. This report presents the design and implementation of a Perceptron model built manually using either paper-based computation or a spreadsheet. The objective is to demonstrate how the Perceptron learning law updates weights iteratively to minimize classification error.

A binary classification problem is constructed using a simple logical dataset (AND gate). The Perceptron model is defined mathematically, initial weights are assigned, and training is performed step-by-step using the Perceptron learning rule. Iterative weight updates are shown clearly to illustrate convergence. The experiment also analyzes learning rate effects, decision boundary formation, and convergence conditions.

The results demonstrate that the Perceptron successfully classifies linearly separable data and converges after a finite number of iterations. However, its limitations in handling non-linearly separable problems are also discussed. This report provides theoretical explanation, mathematical derivation, practical computation steps, tabulated training iterations, and graphical interpretation.

1. Introduction

Artificial Neural Networks are computational models inspired by the biological neuron. Among the earliest models developed was the Perceptron, introduced by Frank Rosenblatt in 1958. The Perceptron is a supervised learning algorithm used for binary classification.

Binary classification refers to problems where the output belongs to one of two classes (0 or 1, -1 or +1). Examples include:

- Spam vs Non-spam email
- Pass vs Fail
- Yes vs No
- True vs False
- AND/OR logical operations

The Perceptron works effectively when the dataset is linearly separable, meaning a straight line (in 2D) or hyperplane (in higher dimensions) can divide the classes.

This report focuses on:

1. Constructing a simple binary dataset.
2. Defining the Perceptron model mathematically.
3. Applying the Perceptron learning law.
4. Demonstrating weight updates step-by-step.
5. Showing convergence and decision boundary formation.
6. Discussing advantages and limitations.

---

2. Objectives

The main objectives of this experiment are:

1. To understand the mathematical structure of the Perceptron.
2. To apply the Perceptron learning rule manually.
3. To train the model on a binary dataset.
4. To observe convergence behavior.
5. To analyze learning rate effects.
6. To interpret the decision boundary.

---

3. Theoretical Background

3.1 Biological Inspiration

A biological neuron consists of:

- Dendrites (input signals)
- Cell body (processing unit)
- Axon (output signal)

The artificial neuron mimics this structure mathematically.

3.2 Mathematical Model of Perceptron

The Perceptron consists of:

Inputs:

x1,x2,...,xnx_1, x_2, ..., x_nx1,x2,...,xn

Weights:

w1,w2,...,wnw_1, w_2, ..., w_nw1,w2,...,wn

Bias:

bbb

Net Input:

z=∑i=1nwixi+bz = \sum_{i=1}^{n} w_i x_i + bz=i=1∑nwixi+b

Activation Function (Step Function):

y={1if z≥00if z<0y = \begin{cases} 1 & \text{if } z \ge 0 \\ 0 & \text{if } z < 0 \end{cases}y={10if z≥0if z<0

3.3 Perceptron Learning Law

The Perceptron updates weights using:

wi(new)=wi(old)+η(t−y)xiw_i(new) = w_i(old) + \eta (t - y) x_iwi(new)=wi(old)+η(t−y)xi b(new)=b(old)+η(t−y)b(new) = b(old) + \eta (t - y)b(new)=b(old)+η(t−y)

Where:

- η\etaη = learning rate
- ttt = target output
- yyy = predicted output

The term (t−y)(t - y)(t−y) is called the error.

---

4. Problem Statement

We consider the AND logical function.

AND Gate Dataset

x1	**x2**	**Target (t)**
0	0	0
0	1	0
1	0	0
1	1	1

This dataset is linearly separable.

---

5. Methodology

5.1 Initialization

Let:

- Learning rate η=1\eta = 1η=1
- Initial weights: w1=0,w2=0w_1 = 0, w_2 = 0w1=0,w2=0
- Bias: b=0b = 0b=0

5.2 Training Iteration 1

Input (0,0), Target = 0

z=0z = 0z=0

Output = 1 (since z ≥ 0)

Error = 0 - 1 = -1

Update:

w1=0w_1 = 0w1=0 w2=0w_2 = 0w2=0 b=−1b = -1b=−1

Input (0,1), Target = 0

z=−1z = -1z=−1

Output = 0

Correct → No update.

Input (1,0), Target = 0

z=−1z = -1z=−1

Output = 0

Correct → No update.

Input (1,1), Target = 1

z=−1z = -1z=−1

Output = 0

Error = 1

Update:

w1=1w_1 = 1w1=1 w2=1w_2 = 1w2=1 b=0b = 0b=0

5.3 Training Iteration 2

Repeat process.

Eventually weights converge to:

w1=1,w2=1,b=−1.5w_1 = 1, w_2 = 1, b = -1.5w1=1,w2=1,b=−1.5

Model correctly classifies all inputs.

---

6. Results

After multiple iterations:

Final Model:

y=step(x1+x2−1.5)y = step(x_1 + x_2 - 1.5)y=step(x1+x2−1.5)

Decision boundary:

x1+x2=1.5x_1 + x_2 = 1.5x1+x2=1.5

All training samples are correctly classified.

7.Decision Boundary Interpretation

The decision boundary is a straight line dividing:

Class 0 → Below the line
Class 1 → Above the line

This confirms linear separability.

---

8. Learning Rate Analysis

If learning rate is:

- Too small → Slow convergence
- Too large → Oscillations possible
- Moderate → Faster stable convergence

In this experiment, $\eta = 1$ works efficiently.

---

9. Limitations

1. Cannot solve non-linearly separable problems (e.g., XOR).
2. Only binary classification.
3. Uses hard threshold (no probability output).

---

10. Advantages

1. Simple implementation.
2. Fast convergence for separable data.
3. Computationally efficient.
4. Foundation for multilayer networks.

11.Applications

1. Spam detection

2. Pattern recognition
3. Simple classification tasks
4. Hardware logic implementation

---

12. Conclusion

This experiment successfully demonstrated the construction and training of a Perceptron model using the Perceptron learning law. By manually computing weight updates, we observed how the model gradually reduces classification errors and converges to a stable solution for linearly separable data.

The Perceptron forms the conceptual foundation for more advanced neural network models such as multilayer perceptrons and deep learning systems. Although limited in capability, it remains important for understanding supervised learning principles.

---

13. References

1. Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain.
2. Haykin, S. (2009). Neural Networks and Learning Machines.
3. Bishop, C. (2006). Pattern Recognition and Machine Learning.
4. Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning.
5. ANN Lecture Notes – Course Material.