Abstract

This report presents the design and manual implementation of a Perceptron model to solve a binary classification problem using the Perceptron Learning Law. The Perceptron is one of the earliest and simplest artificial neural network models used for supervised learning. In this study, a logical AND gate dataset is used as a sample binary classification task. The training process is performed manually through iterative weight updates using the Perceptron learning rule. The report includes theoretical background, methodology, iteration tables, working mechanism, implementation steps, results, advantages, limitations, and real-world applications. The results show that the Perceptron successfully classifies linearly separable data and converges after a finite number of iterations.

The Perceptron is one of the earliest and most fundamental models in artificial neural networks, designed to perform supervised binary classification. Introduced by Frank Rosenblatt in 1958, the Perceptron represents a significant milestone in the development of machine learning. This report focuses on building a Perceptron model manually (on paper or using a spreadsheet) to solve a binary classification problem and applying the Perceptron Learning Law to train the model iteratively.

The study begins with the selection of a simple linearly separable dataset, such as a logical AND gate, where the inputs are mapped to binary outputs (0 or 1). The Perceptron model consists of input variables, corresponding weights, a bias term, and a step activation function. The weighted sum of inputs and bias determines the output class. During training, the model compares the predicted output with the actual target value and calculates the error. The Perceptron Learning Law is then applied to update the weights and bias based on the error, input values, and a predefined learning rate. This process continues iteratively until the model correctly classifies all training examples or reaches convergence.

The manual training process is documented through iteration tables that clearly show how weights and bias change over multiple epochs. These tables help in understanding how the decision boundary gradually adjusts to separate the two classes. The results demonstrate that for linearly separable data, the Perceptron successfully converges after a finite number of iterations, validating the Perceptron Convergence Theorem. The model ultimately learns a linear decision boundary that correctly classifies all training samples.

Although the Perceptron is limited to solving only linearly separable binary classification problems and cannot handle more complex tasks such as XOR classification, it remains an essential foundational concept in neural network theory. The simplicity of its structure and learning rule makes it highly suitable for educational purposes and introductory machine learning studies. Furthermore, the core principles of the Perceptron—weighted inputs, bias adjustment, and error-driven learning—form the basis for advanced neural network architectures such as multilayer perceptrons and deep learning models.

1. Introduction

Binary classification is one of the most fundamental tasks in machine learning and artificial intelligence. In this type of problem, a system must classify input data into one of two possible categories, such as yes/no, true/false, spam/not spam, or 0/1. Many real-world applications—such as email filtering, medical diagnosis, and credit approval—depend on accurate binary decision-making systems. One of the earliest and simplest models developed to address such problems is the Perceptron.

The Perceptron model was introduced in 1958 by Frank Rosenblatt as a computational model inspired by the functioning of biological neurons. It was designed to simulate how a human brain might process information and learn from experience. The Perceptron is considered a single-layer artificial neural network and serves as the foundation for many modern neural network architectures used in deep learning today.

At its core, the Perceptron performs linear classification. It takes multiple input values, multiplies each input by a corresponding weight, adds a bias term, and passes the result through an activation function—typically a step function. The output is binary, meaning the model decides whether the input belongs to one class or the other. What makes the Perceptron powerful is its ability to learn the appropriate weights automatically through a training process known as the Perceptron Learning Law.

The Perceptron Learning Law is an error-correction learning mechanism. During training, the model compares its predicted output with the actual target value. If the prediction is incorrect, the weights and bias are adjusted proportionally to the error and input values. This iterative process continues until the model correctly classifies all training examples or reaches a stable state. For datasets that are linearly separable, the Perceptron is guaranteed to converge after a finite number of iterations.

Although the Perceptron has limitations—particularly its inability to solve non-linearly separable problems—it remains an essential educational tool. It introduces key machine learning concepts such as supervised learning, linear decision boundaries, weight adjustment, bias terms, activation functions, and convergence. Understanding the Perceptron provides a strong conceptual foundation for studying more advanced neural networks, including multilayer perceptrons and deep learning models.

The aim of this experiment is to design and manually implement a Perceptron model (using paper or a spreadsheet) to solve a binary classification problem and to apply the Perceptron Learning Law for training the model. The objective is to understand how input features, weights, and bias interact to produce a binary output, and how iterative weight updates based on error correction enable the model to learn a linearly separable decision boundary. Through this process, the experiment seeks to demonstrate the working mechanism, convergence behavior, and practical implementation of the Perceptron algorithm in supervised learning.

2. Objectives

- To understand the structure of a Perceptron model
- To apply the Perceptron Learning Law manually
- To train the model using iteration tables
- To analyze convergence behavior.
- To evaluate performance on a binary classification task

3.Overview

This experiment focuses on building and training a Perceptron model to solve a binary classification problem using the Perceptron Learning Law. The Perceptron is a simple supervised learning algorithm used to classify data into two categories (0 and 1). It consists of input variables, corresponding weights, a bias term, and a step activation function that produces a binary output.

In this task, the model is implemented manually on paper or in a spreadsheet to clearly understand how the algorithm works internally. A simple linearly separable dataset (such as a logical AND gate problem) is used for training. The Perceptron computes the weighted sum of inputs, applies an activation function, compares the predicted output with the target value, and updates the weights using the error-correction learning rule.

Through iterative training (epochs), the weights and bias are adjusted until the model correctly classifies all training examples. This process demonstrates how the Perceptron gradually learns a linear decision boundary that separates the two classes.

Overall, this experiment provides a practical understanding of supervised learning, weight adjustment, convergence, and linear classification, forming the foundation for more advanced neural network models.

4. Theoretical Background

The Perceptron is a single-layer neural network consisting of:

- Input layer
- Weights
- Bias
- Activation function (Step function)

Mathematical Model

Net input:

$$z = w_1 x_1 + w_2 x_2 + b$$

Output using step activation:

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Perceptron Learning Rule

Weights are updated as:

$$w_i(new) = w_i(old) + \eta(t - y)x_i$$
$$b(new) = b(old) + \eta(t - y)$$

Where:

- η= learning rate
- t= target output
- y= predicted output

5. Methodology

Problem Selection: AND Gate

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

Learning rate (η) = 1
Initial weights: w1 = 0, w2 = 0
Initial bias: b = 0

6. Manual Training (Iteration Table)

Iteration 1

x1	x2	t	z	y	Error(t-y)	w1	w2	b
0	0	0	0	1	-1	0	0	-1
0	1	0	-1	0	0	0	0	-1
1	0	0	-1	0	0	0	0	-1
1	1	1	-1	0	1	1	1	0

Iteration 2

x1	x2	t	z	y	Error	w1	w2	b
0	0	0	0	1	-1	1	1	-1
0	1	0	0	1	-1	1	0	-2
1	0	0	-1	0	0	1	0	-2
1	1	1	-1	0	1	2	1	-1

Iteration 3

x1	x2	t	z	y	Error	w1	w2	b
0	0	0	-1	0	0	2	1	-1
0	1	0	0	1	-1	2	0	-2
1	0	0	0	1	-1	1	0	-3
1	1	1	-2	0	1	2	1	-2

Training continues until no errors occur.

Final Weights After Convergence

w1 = 1
w2 = 1
b = -1.5 (or equivalent separating boundary)

The model successfully classifies all AND gate inputs.

7. Decision Boundary

After training the Perceptron on the AND gate dataset, the final converged weights were:

- $w_1 = 1$
- $w_2 = 1$
- $b = -1.5$

1. Decision Function

The Perceptron decision function is:

$$z = w_1 x_1 + w_2 x_2 + b$$

Substituting the final values:

$$z = x_1 + x_2 - 1.5$$

The decision boundary occurs where:

$$z = 0$$

So,

$$x_1 + x_2 - 1.5 = 0$$

2. Equation of Decision Boundary

$$x_2 = 1.5 - x_1$$

This is a **straight line**, which confirms that the Perceptron creates a linear decision boundary.

3.Interpretation

The decision boundary $x_1 + x_2 = 1.5$separates the input space into two regions:

- Region below the line → Class 0
- Region above the line → Class 1

This straight-line separation confirms that the AND gate problem is **linearly separable**, which is why the Perceptron successfully converged.

8. Working Process

Working steps:

1. Inputs are multiplied with weights
2. Weighted sum is computed
3. Bias is added
4. Step activation function produces output
5. Error is calculated
6. Weights are updated using learning rule
7. Process repeats until convergence

The Perceptron finds a linear decision boundary separating the two classes.

9. Implementation (Spreadsheet Approach)

Steps:

1. Create columns: x1, x2, t, w1, w2, b
2. Compute:
   z = w1*x1 + w2*x2 + b
3. Apply step function using IF formula
4. Compute error: t - y
5. Update weights using formula
6. Repeat for multiple epochs

The spreadsheet clearly shows weight adjustments at each iteration.

10. Results and Discussion

- The Perceptron converged after multiple iterations.
- The model successfully classified the AND gate.

- Convergence occurs only because AND is linearly separable.
- Weight updates gradually adjust the decision boundary.

The results of this experiment show that the Perceptron model successfully learned to classify the binary dataset after several training iterations. Starting with initial weights and bias set to zero, the model initially produced incorrect predictions. However, by applying the Perceptron Learning Law, the weights and bias were updated step by step based on the calculated error. With each iteration (epoch), the model gradually reduced misclassifications and adjusted the decision boundary. After a finite number of updates, the Perceptron converged to a set of final weights that correctly classified all input combinations of the AND gate. This confirms that the dataset was linearly separable and that the Perceptron Convergence Theorem holds true for such problems.

From the discussion perspective, the experiment clearly demonstrates how supervised learning works through error correction. The weight update mechanism plays a crucial role in shifting the linear decision boundary until it properly separates the two classes. The convergence behavior depends on the learning rate and the nature of the data. If the data is linearly separable, the Perceptron guarantees convergence; otherwise, it will fail to find a solution. Therefore, the experiment not only validates the effectiveness of the Perceptron for simple binary classification tasks but also highlights its limitations when handling more complex, non-linear problems. Overall, the results provide practical insight into how a simple neural network learns from data and forms the foundation for more advanced machine learning models.

11. Advantages

- **Simple and easy to implement**
  The Perceptron is one of the simplest neural network models. Its structure (inputs, weights, bias, activation function) makes it easy to implement manually or in a spreadsheet.

- **Fast convergence for linearly separable data**
  If the dataset is linearly separable (like AND or OR problems), the Perceptron is guaranteed to converge in a finite number of iterations.

- **Low computational cost**
  The algorithm uses simple arithmetic operations (multiplication, addition, comparison), making it computationally efficient.

- **Good for small binary classification problems**
  The Perceptron introduced the concept of weight adjustment using learning rules. It forms the basis for advanced models like multilayer neural networks and deep learning systems.

- **Works Well for Basic Binary Classification**
  It performs effectively when the problem involves two clearly separable classes.

12.Limitations

- **Cannot Solve Non-Linearly Separable Problems**
  The biggest limitation is that it cannot solve problems like XOR because such data cannot be separated by a single straight line.

- **Only Handles Binary Classification**
  The basic Perceptron works only for two classes (0 and 1).

- **Hard Threshold Output**
  It uses a step activation function, which gives only 0 or 1 output. It does not provide probability values.

- **Sensitive to Learning Rate**
  A very large learning rate may cause instability, while a very small one may slow down convergence.

- **Limited Representation Power**
  Since it has only one layer, it cannot learn complex patterns or hierarchical features.

13. Applications

Although simple, the Perceptron has practical uses in basic classification tasks:

- **Spam Email Detection**
  Classifying emails as spam (1) or not spam (0).

- **Sentiment Analysis (Basic Level)**
  Classifying text as positive or negative.

- **Image recognition (basic level)**

  Identifying whether a simple image contains a specific object or not.

- **Pattern recognition**

  Deciding whether to approve or reject a loan application based on financial inputs.

- **Credit approval prediction**

  Recognizing simple patterns in datasets such as logical gate problems (AND, OR).

14 .conclusion

The construction and manual implementation of the Perceptron model for a binary classification problem clearly demonstrate the fundamental principles of supervised learning and neural network training. Through this experiment, we successfully designed a single-layer Perceptron and applied the Perceptron Learning Law to classify a linearly separable dataset, such as the AND gate problem. By initializing the weights and bias and updating them iteratively based on prediction errors, the model gradually learned the correct decision boundary.

One of the key outcomes of this experiment is understanding how learning occurs through error correction. The Perceptron does not initially "know" the correct classification rule. Instead, it adjusts its weights step by step using the difference between the target output and the predicted output. This iterative adjustment continues until all training examples are classified correctly or the model converges. Observing the weight changes in each iteration table helps in visualizing how the decision boundary shifts toward the correct position.

The results confirm that the Perceptron successfully converges when the dataset is linearly separable. This validates the theoretical concept that a single-layer Perceptron can solve problems where a straight-line boundary can separate the two classes. The final weights obtained after training represent the mathematical equation of that decision boundary.

However, this experiment also highlights the limitations of the Perceptron. It cannot solve non-linearly separable problems such as the XOR function. This limitation emphasizes the need for more advanced models, such as multilayer neural networks, to handle complex real-world problems. Despite this drawback, the Perceptron remains an important foundational model in machine learning.

Overall, this experiment provides a clear understanding of how a machine can learn from data using simple mathematical rules. It explains the concepts of weighted input summation, activation functions, error calculation, and iterative weight updates. The Perceptron model, though simple, forms the basis of modern neural networks and deep learning systems. Understanding its working mechanism builds a strong foundation for studying advanced artificial intelligence techniques and practical machine learning