LAB Manual

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**Experiment No.04**

**A.1 Aim:**

Implementation of AND gate using perceptron.

**A.2 Prerequisite:**

Basic idea of perceptron

**A.3 Outcome:**

**After successful completion of this experiment students will be able to**

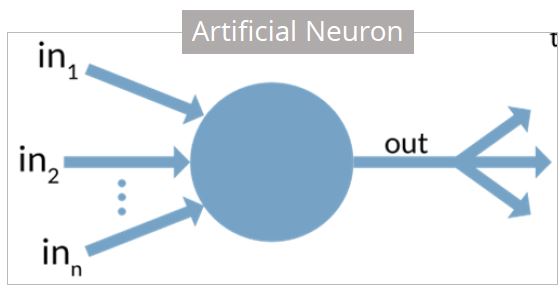
**Design logical gates using perceptron**

**A.4 Theory:**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), the **perceptron** is an algorithm for [supervised learning](https://en.wikipedia.org/wiki/Supervised_classification) of [binary classifiers](https://en.wikipedia.org/wiki/Binary_classification). A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class.[[1]](https://en.wikipedia.org/wiki/Perceptron#cite_note-largemargin-1) It is a type of [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier), i.e. a classification algorithm that makes its predictions based on a [linear predictor function](https://en.wikipedia.org/wiki/Linear_predictor_function) combining a set of weights with the [feature vector](https://en.wikipedia.org/wiki/Feature_vector).

Artificial Neuron

An artificial neuron is a mathematical function based on a model of biological neurons, where each neuron takes inputs, weighs them separately, sums them up and passes this sum through a nonlinear function to produce output.



In the next section, let us compare the biological neuron with the artificial neuron.

Biological Neuron vs. Artificial Neuron

The biological neuron is analogous to artificial neurons in the following terms:

|  |  |
| --- | --- |
| Biological Neuron | Artificial Neuron |
| Cell Nucleus (Soma) | Node |
| Dendrites | Input |
| Synapse | Weights or interconnections |
| Axon | Output |

Artificial Neuron at a Glance

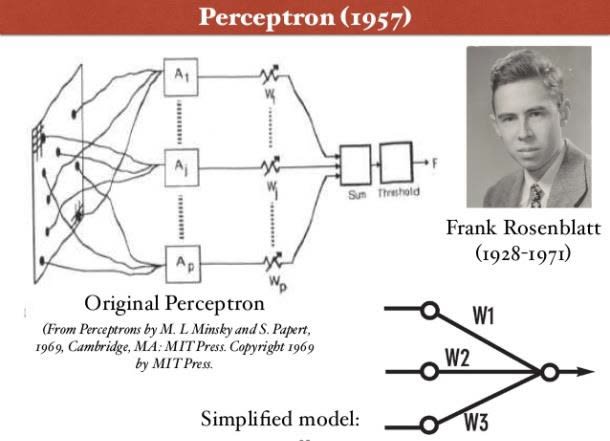
The artificial neuron has the following characteristics:

* A neuron is a mathematical function modeled on the working of biological neurons
* It is an elementary unit in an artificial neural network
* One or more inputs are separately weighted
* Inputs are summed and passed through a nonlinear function to produce output
* Every neuron holds an internal state called activation signal
* Each connection link carries information about the input signal
* Every neuron is connected to another neuron via connection link

In the next section, let us talk about perceptron.

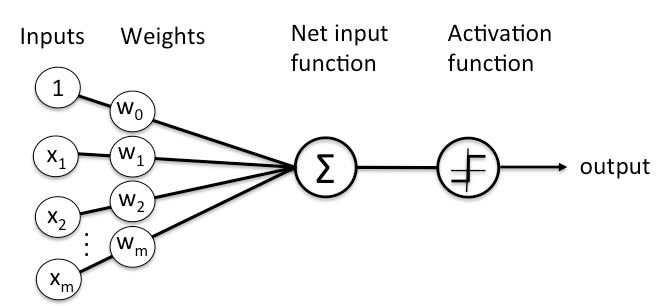
Perceptron

A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.



Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron.

A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.



USE MATLAB toolkit and design a program for implementing AND gate.

Write your program here:

**PART B**

*(PART B: TO BE COMPLETED BY STUDENTS)*

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No.: C026 | Name: Anirbaan Ghatak |
| Class: B. Tech Integrated | Batch: EB1 |
| Date of Experiment: 6/9/23 | Date of Submission: 6/9/23 |
| Date of Grading: | |

## Software Code written by student:

import numpy as np

def perceptron\_and(A, B):

    w = np.array([0.5, 0.5])

    b = -0.7

    x = np.array([A, B])

    result = np.dot(w, x) + b

    if result > 0:

        return 1

    else:

        return 0

print(perceptron\_and(0,0))

print(perceptron\_and(0,1))

print(perceptron\_and(1,0))

print(perceptron\_and(1,1))



* 1. **Observations and learning:**

In machine learning, the perceptron is an algorithm used for binary classification. It helps determine whether an input, represented as a numeric vector, belongs to a particular class or not. This algorithm operates as a linear classifier, meaning it makes predictions by combining a set of weights with the input's features using a linear predictor function.

## Conclusion:

Successfully implemented AND gate in Perceptron model in python.

## Question of Curiosity:

## Implement ANDNOT Function using Perceptron.

## Ans:

import numpy as np

def perceptron\_and(A, B):

    w = np.array([0.5, 0.5])

    b = -0.7

    if B == 0:

        B = 1

    else:

        B = 0

    x = np.array([A,B])

    result = np.dot(w, x) + b

    if result > 0:

      return 1

    else:

      return 0

print(perceptron\_and(0,0))

print(perceptron\_and(0,1))

print(perceptron\_and(1,0))

print(perceptron\_and(1,1))



## Justify the following:

## A perceptron is guaranteed to perfectly learn a given linearly separable function within a finite number of training steps.

**Ans:**

A perceptron, a type of artificial neural network designed for supervised learning, excels in learning linearly separable functions—those functions neatly divided by a hyperplane. Its algorithm adjusts input feature weights based on the prediction error compared to the actual output. Importantly, the perceptron algorithm assures convergence when applied to linearly separable data. This convergence is rooted in the fact that the algorithm continually refines the weights in the direction of the true weight vector, which efficiently separates the data. As a result, a perceptron is guaranteed to accurately learn a given linearly separable function within a finite number of training iterations.

* **A single perceptron can realize the XOR functions.**

**Ans:**

The XOR function poses a challenge for single perceptrons due to its lack of linear separability. However, this complexity can be overcome by expressing XOR as a composite of AND, OR, and NOT functions. Enter the multi-layer perceptron (MLP), with its structured architecture comprising an input layer, one or more hidden layers, and an output layer. These hidden layers grant the MLP the ability to grasp non-linear decision boundaries, making it proficient at learning the XOR function. In essence, while a single perceptron falls short in tackling XOR, an MLP's layered configuration allows it to master this function through the combined efforts of multiple perceptrons.