Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2023-24

Class:	TE	**Semester:**	VI
Course Code:	CSL604	**Course Name:**	Machine Learning Lab

Name of Student:	Ojasi Prabhu
Roll No.:	43
Experiment No.:	6
Title of the Experiment:	Implementation of McCulloch Pitts Model
Date of Performance:	
Date of Submission:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Mr Raunak Joshi

Signature :

Date :

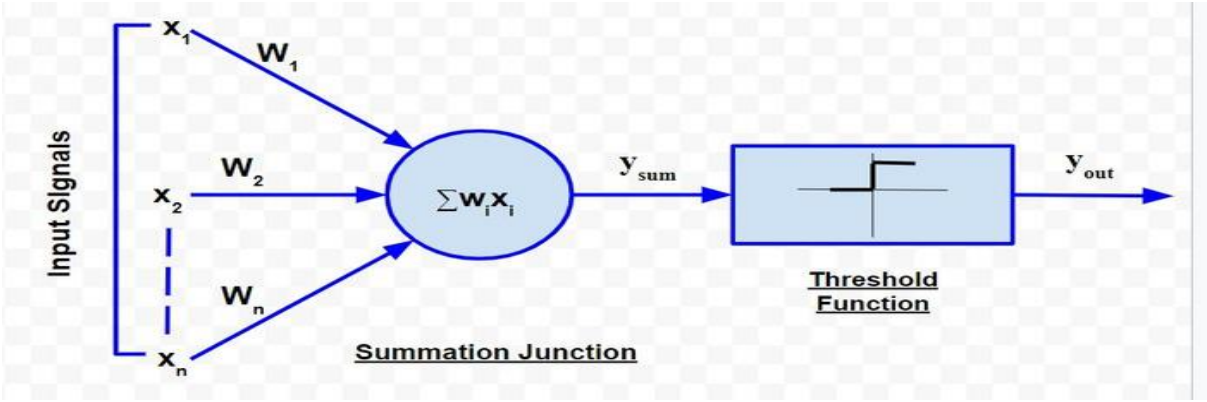
Aim: To implement McCulloch Pitts Model for ANN.

Objective: Able to design a neural network and use activation function as learning rule defined by McCulloch Piits Model.

Theory:

McCulloch-Pitts Model of Neuron

The McCulloch-Pitts neural model, which was the earliest ANN model, has only two types of inputs — **Excitatory and Inhibitory.** The excitatory inputs have weights of positive magnitude and the inhibitory weights have weights of negative magnitude. The inputs of the McCulloch-Pitts neuron could be either 0 or 1. It has a threshold function as an activation function. So, the output signal y_{out} is 1 if the input y_{sum} is greater than or equal to a given threshold value, else 0. The diagrammatic representation of the model is as follows:



Simple McCulloch-Pitts neurons can be used to design logical operations. For that purpose, the connection weights need to be correctly decided along with the threshold function (rather than the threshold value of the activation function).

Example:

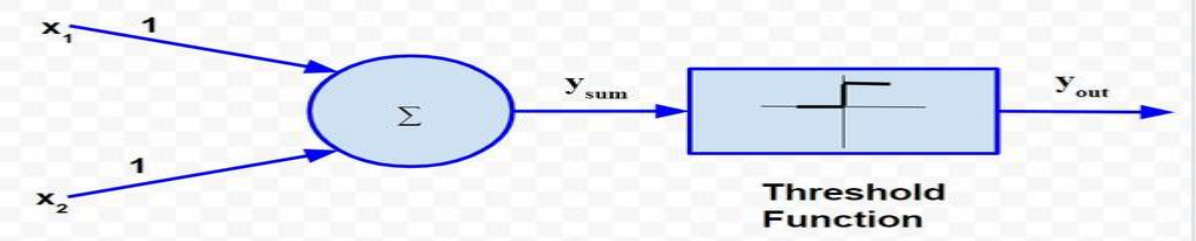
John carries an umbrella if it is sunny or if it is raining. There are four given situations. I need to decide when John will carry the umbrella. The situations are as follows:

- First scenario: It is not raining, nor it is sunny
- Second scenario: It is not raining, but it is sunny
- Third scenario: It is raining, and it is not sunny
- Fourth scenario: It is raining as well as it is sunny

To analyse the situations using the McCulloch-Pitts neural model, I can consider the input signals as follows:

- X_1: Is it raining?
- X_2 : Is it sunny?

So, the value of both scenarios can be either 0 or 1. We can use the value of both weights X_1 and X_2 as 1 and a threshold function as 1. So, the neural network model will look like:



Truth Table for this case will be:

Situation	x1	x2	ysum	yout
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	2	1

So,

$$y_{sum} = \sum_{i=1}^{2} w_i x_i$$

$$y_{out} = f(y_{sum}) = \begin{cases} 1, x \geq 1 \\ 0, x < 1 \end{cases}$$

The truth table built with respect to the problem is depicted above. From the truth table, I can conclude that in the situations where the value of y_{out} is 1, John needs to carry an umbrella. Hence, he will need to carry an umbrella in scenarios 2, 3 and 4.

Implementation:

Step 1: generate a vector of inputs and a vector of weights

```
[1] import numpy as np
```

```
[2] np.random.seed(seed=0)
    I = np.random.choice([0,1], 3)
    W = np.random.choice([-1,1], 3)
    print(f'Input vector:{I}, Weight vector:{W}')
```

```
Input vector:[0 1 1], Weight vector:[-1  1  1]
```

Step 2: compute the dot product between the vector of inputs and weights

```
[3] dot = I @ W
    print(f'Dot product: {dot}')
```

```
Dot product: 2
```

Step 3: define the threshold activation function

```
[4] def linear_threshold_gate(dot: int, T: float) -> int:
        if dot >= T:
            return 1
        else:
            return 0
```

Step 4: compute the output based on the threshold value

```
[5] T = 1
    activation = linear_threshold_gate(dot, T)
    print(f'Activation: {activation}')
```

```
Activation: 1
```

```
[6] T = 3
    activation = linear_threshold_gate(dot, T)
    print(f'Activation: {activation}')
```

```
Activation: 0
```

Application: boolean algebra using the McCulloch-Pitts artificial neuron

```
[7] input_table = np.array([
        [0,0],
        [0,1],
        [1,0],
        [1,1]
    ])

    print(f'input table:\n{input_table}')
```

```
input table:
[[0 0]
 [0 1]
 [1 0]
 [1 1]]
```

```
weights = np.array([1,1])
print(f'weights: {weights}')
```

```
weights: [1 1]
```

Step 2: compute the dot product between the matrix of inputs and weights

```
[9] dot_products = input_table @ weights
    print(f'Dot products: {dot_products}')
```

```
Dot products: [0 1 1 2]
```

Step 3: define the threshold activation function

```
[10] def linear_threshold_gate(dot: int, T: float) -> int:
         if dot >= T:
             return 1
         else:
             return 0
```

Step 4: compute the output based on the threshold value

```
[11] T = 2
     for i in range(0,4):
         activation = linear_threshold_gate(dot_products[i], T)
         print(f'Activation: {activation}')
```

```
Activation: 0
Activation: 0
Activation: 0
Activation: 1
```

Conclusion:

1) What is the use of McCulloch Pitts model?

The McCulloch-Pitts model, a fundamental notion in neural network theory, provides the framework for understanding neural computation. It simplifies neurons into binary threshold units that fire when inputs exceed a particular threshold. This model paves the way for investigating complicated computational tasks, including as pattern recognition and classification, by mimicking the behavior of linked neurons. Though simple, it paved the way for the creation of more advanced neural network structures and learning algorithms.

2) How it will be used to develop ANN?

The McCulloch-Pitts model offers the foundation for creating artificial neural networks (ANNs). By abstracting neurons as binary threshold units and modeling their interconnections with weighted connections, the model serves as the foundation for more complicated neural network topologies. This model motivates the design of activation functions, connection weights, and network topology in ANN development, allowing for the simulation of complicated computational tasks like pattern recognition, classification, and regression. It also provides a theoretical foundation for studying neural networks' behavior and learning capacities.