Experiment No. 6
Implementation of McCulloch Pitts Model
Date of Performance:
Date of Submission:
Marks:
Sign:

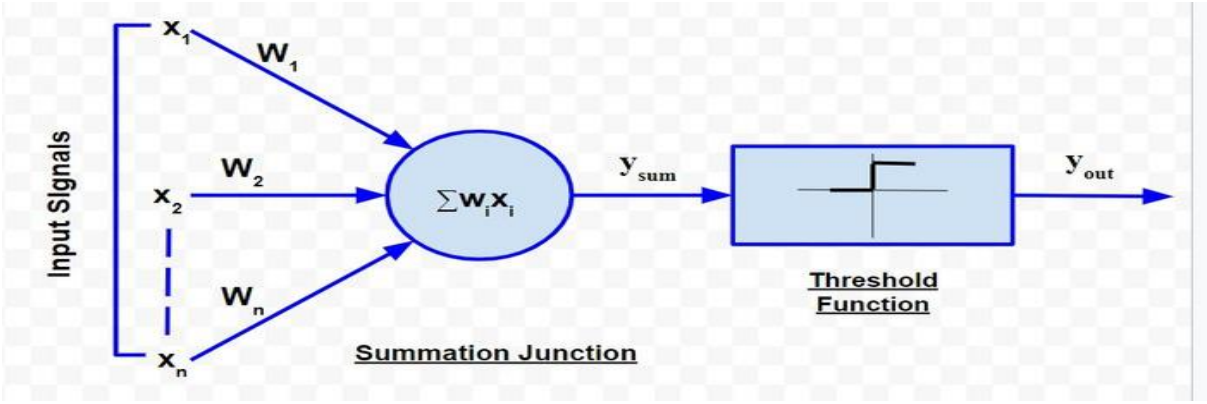
Aim: To implement McCulloch Pitts Model for ANN.

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:

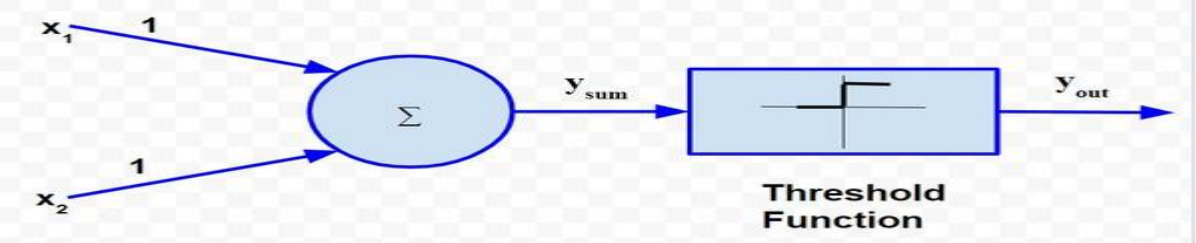
- First scenario: It is not raining, nor it is sunny
- Second scenario: It is not raining, but it is sunny

- 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:

```
import numpy as np
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]
```

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

```
Dot product: 2
```

```
def linear_threshold_gate(dot: int, T: float) -> int:
    if dot >= T:
        return 1
    else:
        return 0
T = 1
activation = linear_threshold_gate(dot, T)
print(f'Activation: {activation}')
```

```
Activation: 1
```

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

```
Activation: 0
```

```
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]
```

dot_products = input_table @ weights

print(f'Dot products: {dot_products}')

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

```
def linear_threshold_gate(dot: int, T: float) -> int:
    if dot >= T:
        return 1
    else:
        return 0
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 serves as a foundational concept in the field of artificial neural networks, specifically in understanding the basic principles of neuron behavior and network connectivity. Proposed by Warren McCulloch and Walter Pitts in 1943, this binary threshold logic model emulates the functioning of a biological neuron, where inputs are weighted and summed, and an output is produced based on a threshold function. Although simplistic compared to modern neural network architectures, the McCulloch-Pitts model laid the groundwork for subsequent developments in neural network theory and computational neuroscience, providing insights into information processing mechanisms in both natural and artificial systems.

2) How it will be used to develop ANN?

   The McCulloch-Pitts model forms the conceptual basis for developing artificial neural networks (ANNs) by simulating the behavior of interconnected neurons. While the original model is binary and simplistic, it inspires the design of more complex ANN architectures. In modern ANNs, neurons are modeled as nodes that receive inputs, apply activation functions to compute outputs, and transmit signals to connected neurons. By extending and refining the principles of the McCulloch-Pitts model, ANNs can exhibit sophisticated behaviors, such as pattern recognition, classification, and regression, making them versatile tools for solving a wide range of machine learning tasks. Through the integration of numerous interconnected neurons and layers, ANNs harness the collective power of distributed computation to learn complex patterns and relationships from data, enabling them to perform tasks with high accuracy and efficiency.