Terna Engineering College

Department of Artificial Intelligence and Data Science

Program : Sem VI

Course: Machine Learning Lab

Experiment No.09

PART A

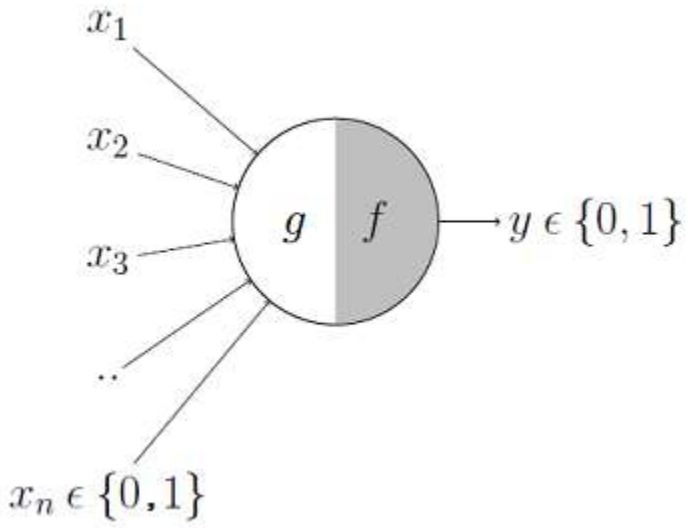
(PART A: TO BE REFERRED BY STUDENTS)

A.1 Aim: To implement McCulloch Pitts modelusing Python.

A.2 Theory:

McCulloch-Pitts Model of Neuron:

The McCulloch-Pitts neural model was the earliest ANN model. The first computational model was proposed by Warren MuCulloch (neuroscientist) and Walter Pitts (logician) in 1943.



It may be divided into 2 parts. The first part, g takes an input (ahem dendrite ahem), performs an aggregation and based on the aggregated value the second part, f makes a decision.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).

e.g.

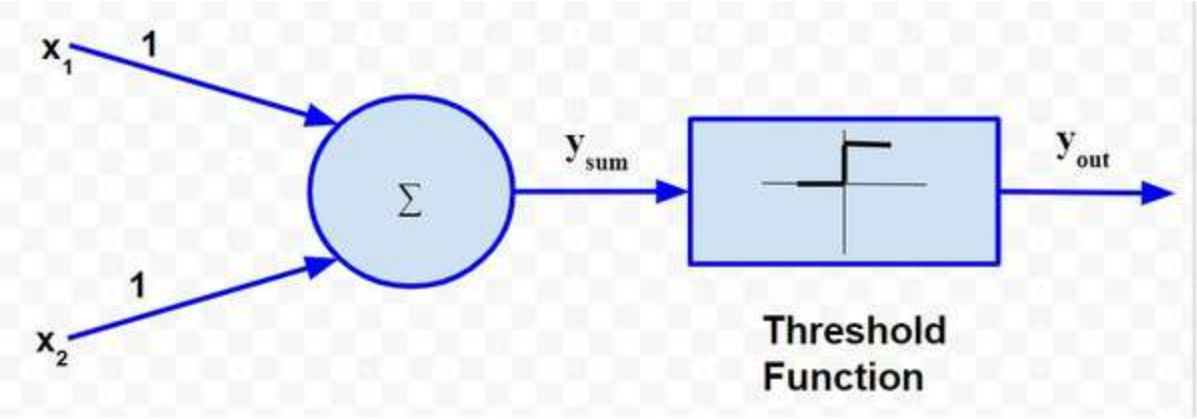
John carries an umbrella if it is sunny or if it is raining. There are four given situations. I need to decide when John will carry the umbrella. The situations are as follows:

- 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:

- X1: Is it raining?
- X2 : Is it sunny?

So, the value of both scenarios can be either 0 or 1. We can use the value of both weights X1 and X2 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 we can say that,

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

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

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 *yout* is 1, John needs to carry an umbrella. Hence, he will need to carry an umbrella in scenarios 2, 3 and 4.

The McCulloch Pitt's model of neuron was mankind's first attempt at mimicking the human brain. And it was a fairly simple one too. It's no surprise it had many limitations-

1. The model failed to capture and compute cases of non-binary inputs. It was limited by its ability to compute every case with 0 and 1 only.
2. The threshold had to be decided beforehand and needed manual computation instead of the model deciding itself.
3. Linearly separable functions couldn't be computed.

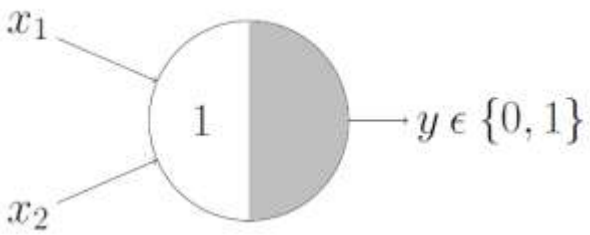
Differences between McCulloch & Pitt's Model and Perceptron Model

1. McCulloh/Pitt's Model accepts only boolean inputs whereas, Perceptron Model can process inputs in various real forms.
2. In McCulloh/Pitt's Model the inputs are not weighted which means that this model is not flexible, however in comparison to this model, Perceptron model accepts weights with respect to the provided inputs which makes it much more flexible.

Similarities between McCulloch & Pitt's Model and Perceptron Model

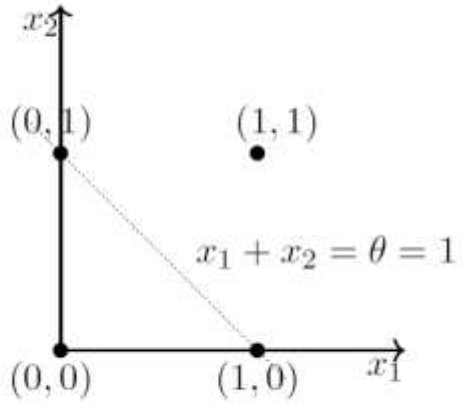
1. Both models can handle linearly separable data.
2. Threshold inputs can be adjusted in both models so that they can fit respective datasets.

OR Function

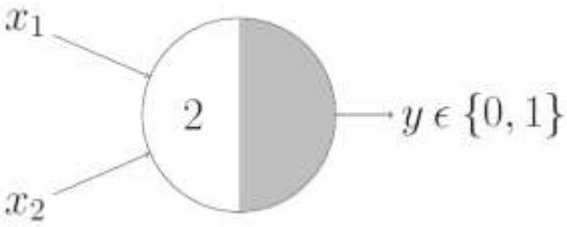


OR function

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$$

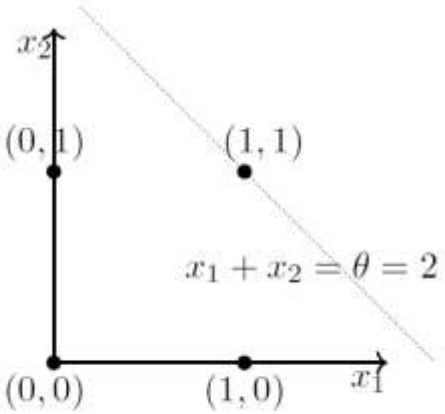


AND Function



AND function

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$$



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. Thesoftcopy mustbeuploaded on the Blackboard or emailed to the concerned lab in charge faculties atthe endofthepractical incasethethere is noBlackboard access available)

Roll.No.A12	Name:Sufiyan Khan
Class:TE–AI&DS	Batch:A1
DateofExperiment:	DateofSubmission: 30-03-24
Grade:	

B.1Input and Output:

```
import numpy as np
import pandas as pd
```

✓ 0.7s

```
class MCPNeuron(object):

    def __init__(self, w = [1,1], t = 1):
        self.w = np.array(w)
        self.t = t

    def decide(self, message):
        x = message
        sum_ = np.inner(self.w,x)
        if sum_ >= self.t:
            return 1
        else:
            return 0

    def TruthTable(self, in_signals, in_labels, out_label):
        table = pd.DataFrame(in_signals, columns = in_labels)
        out_signals = []
        for row in in_signals:
            signal = self.decide(message = row)
            out_signals.append(signal)
        table[out_label] = pd.Series(out_signals)
        return table
```

✓ 0.0s

OR Gate:

```
in_signals = np.array([[0,0], [0,1], [1,0], [1,1]])
in_labels = ['x1', 'x2']
out_label = 'y'
OR = MCPNeuron(w = [1,1], t = 1)
OR_table = OR.TruthTable(in_signals, in_labels = in_labels, out_label = out_label)
print(OR_table)
```

✓ 0.0s

```
   x1  x2  y
0   0   0  0
1   0   1  1
2   1   0  1
3   1   1  1
```

AND Gate:

```
in_signals = np.array([[0,0], [0,1], [1,0], [1,1]])
AND = MCPNeuron(w = [1,1], t = 2)
AND_table = AND.TruthTable(in_signals, in_labels = in_labels, out_label = out_label)
print(AND_table)
```

✓ 0.0s

```
   x1  x2  y
0   0   0  0
1   0   1  0
2   1   0  0
3   1   1  1
```

NOT Gate:

```
NOT_signals = np.array([[0], [1]])
NOT = MCPNeuron(w = [-1], t = 0)
NOT_table = NOT.TruthTable(NOT_signals, in_labels = ['x1'], out_label = 'y')
print(NOT_table)
```

✓ 0.0s

```
   x1  y
0   0  1
1   1  0
```

B.2Conclusion:

Thus we have successfully implemented McCulloch Pitts Model in Python and understood the working of M-P neuron and also tested its functioning by implementing logic gates with it.