

Assignment No. 4

Title: : To implement Mc-Culloch pits Model using XOR

Problem Statement:

Implement basic logic gates using Mc-Culloch-Pitts or Hebbnet neural networks

Objectives:

1. To learn neural network.
2. To learn Mc-Culloch-Pitts or Hebbnet neural networks

Software Requirements:

- Ubuntu 16.04

Hardware Requirements:

- Pentium IV system with latest configuration

Theory:

Brain is the basic of human body which corresponds for all the functions. Neurons are responsible for the response of our body. Like the same way, artificial neurons are created which function as similar to that of biological brain. In this paper the response of the artificial neurons are obtained by using different threshold values and activation functions of logic gates. In this paper McCulloch-Pitts model is applied for the purpose of realization of logic gates.

First artificial neurons: The McCulloch-Pitts model

The McCulloch-Pitts model was an extremely simple artificial neuron. The inputs could be either a zero or a one. And the output was a zero or a one. And each input could be either excitatory or inhibitory. Now the whole point was to sum the inputs. If an input is one, and is excitatory in nature, it added one. If it was one, and was inhibitory, it subtracted one from the sum. This is done for all inputs, and a final sum is calculated. Now, if this final sum is less than some value (which you decide, say T), then the output is zero. Otherwise, the output is a one.

Every neuron model consists of a processing element with synaptic input connection and a single input. The "neurons" operated under the following assumptions:-

- i. They are binary devices ($V_i = [0,1]$)
- ii. Each neuron has a fixed threshold, θ values.
- iii. The neuron receives inputs from excitatory synapses, all having identical weights.
- iv. Inhibitory inputs have an absolute veto power over any excitatory inputs.
- v. At each time step the neurons are simultaneously (synchronously) updated by summing the weighted excitatory inputs and setting the output (V_i) to 1 if the sum is greater than or equal to the threshold and if the neuron receives no inhibitory input.

The McCulloch-Pitts Model of Neuron

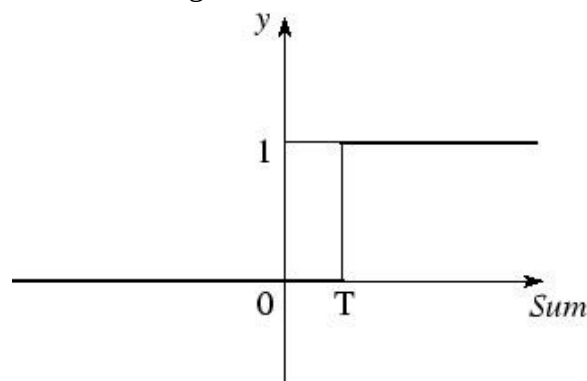
The early model of an artificial neuron is introduced by Warren McCulloch and Walter Pitts in 1943. The McCulloch-Pitts neural model is also known as linear threshold gate. It is a neuron of a set of inputs $I_1, I_2, I_3, \dots, I_m$ and one output y . The linear threshold gate simply classifies the set of inputs into two different classes. Thus the output is y binary. Such a function can be described mathematically using these equations:

$$Sum = \sum_{i=1}^N I_i W_i,$$

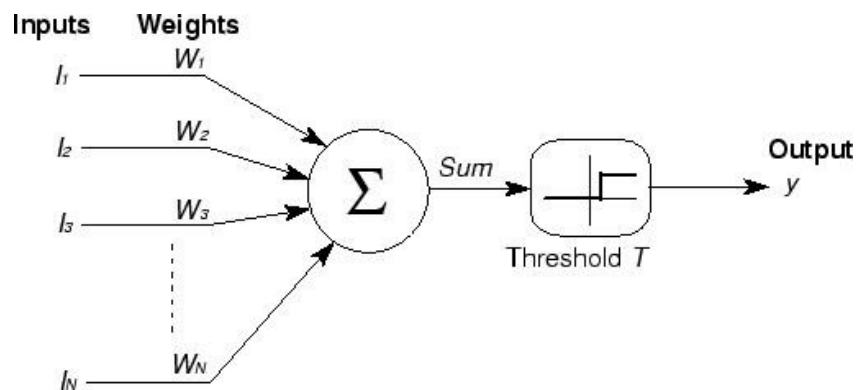
$$y = f(Sum).$$

$W_1, W_2, W_3, W_4, \dots, W_m$ are weight values normalized in the range of either (0, 1) or (-1, 1) and

associated with each input line, Sum is the weighted sum, and T is a threshold constant. The function f is a linear step function at threshold T as shown in figure. The symbolic representation of the linear threshold gate is shown in figure

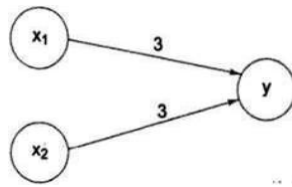


The McCulloch-Pitts model of a neuron is simple yet has substantial computing potential. It also has a precise mathematical definition. However, this model is so simplistic that it only generates a binary output and also the weight and threshold values are fixed. The neural computing algorithm has diverse features for various applications. Thus, we need to obtain the neural model with more flexible computational features.



By using McCulloch-Pitts model, we are going to solve the following logic gates.i. OR Gate ii. NOT Gate iii. AND Gate iv. NAND Gate v. XOR Gate vi. NOR Gate.

Implementation of McCulloch-Pitts model for OR gate



Architecture of AND Gate (Threshold value per unit=3)

The net input is $Y_{in}=3A+3B$. The output is given by

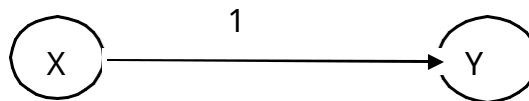
$$Y=f(Y_{in}) = \begin{cases} 1 & \text{if } Y_{in} \geq 3 \\ 0 & \text{if } Y_{in} < 3 \end{cases}$$

Results:

```

MATLAB R2012a
e Edit Debug Parallel Desktop Window Help
[Icons] Current F
Shortcuts How to Add What's New
Command History
Enter weights
Weight w1=3
Weight w2=3
Enter Threshold value
theta=3
Output of Net
0 1 1 1
McCulloch-pitts Net for OR function
Weights of Neuron
3
3
Threshold value
3
  
```

Implementation of McCulloch-Pitts model for NOT gate



Architecture of AND Gate (Threshold value=1)

$$\text{Activation function} = Y=f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} < 1 \\ 0 & \text{if } y_{in} \geq 1 \end{cases}$$

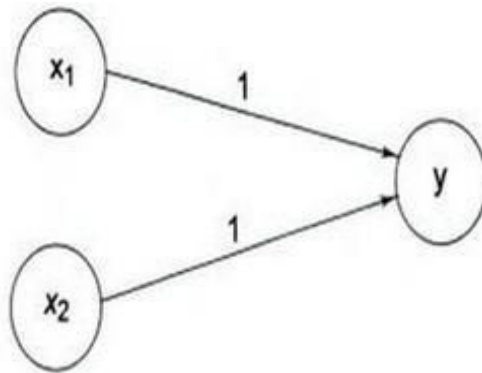
Results:

```
MATLAB R2012a
File Edit Debug Parallel Desktop Window Help
Enter weights
Weight w1=1
Enter Threshold value
theta=1
Output of Net
1 0

McCulloch-pitts Net for NOT function
Weights of Neuron
1

Threshold value
1
```

Implementation of McCulloch-Pitts model for AND gate



Architecture of AND Gate (Threshold value=2)

Net input is $y_{in}=A+B$. Output is given by $Y=f(y_{in})$

$$\text{Activation function} = \begin{cases} 1 & \text{if } y_{in} \geq 2 \\ 0 & \text{if } y_{in} < 2 \end{cases}$$

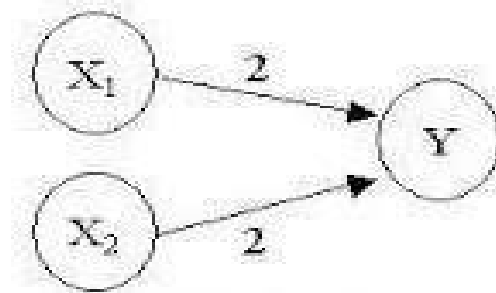
Results:

```
MATLAB R2012a
File Edit Debug Parallel Desktop Window Help
Enter weights
Weight w1=1
Weight w2=1
Enter Threshold value
theta=2
Output of Net
0 0 0 1

McCulloch-pitts Net for AND function
Weights of Neuron
1
1

Threshold value
2
```

Implementation of McCulloch-Pitts model for NAND gate



Architecture of NAND Gate (Threshold value=4)

Net input is $y_{in}=x_1 \cdot x_2$. Output activation function is

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq 4 \\ 0 & \text{if } y_{in} < 4 \end{cases}$$

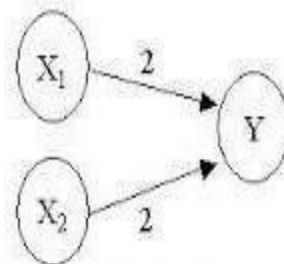
Results:

```

MATLAB R2012a
File Edit Debug Parallel Desktop Window Help
...
Shortcuts How to Add What's New
Command History
Enter weights
Weight w1=2
Weight w2=2
Enter Threshold value
theta=4
Output of Net
    1    1    1    0

McCulloch-pitts Net for NAND function
Weights of Neuron
    2
    2
Threshold value
    4
  
```

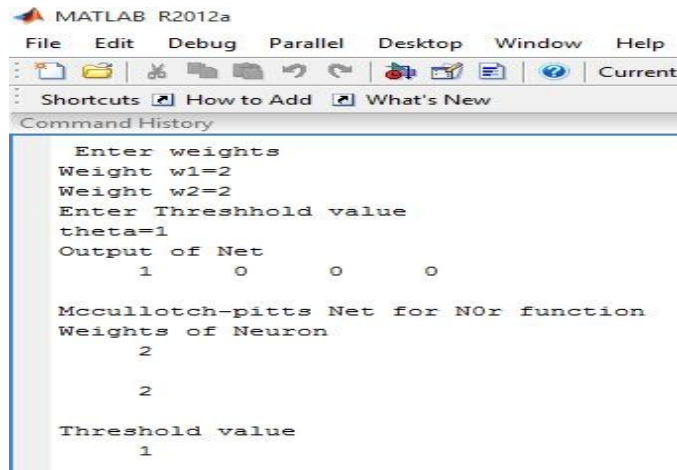
Implementation of McCulloch-Pitts model for NOR gate



Architecture of NOR Gate (Threshold value=1)

Activation function = $\begin{cases} 1 & \text{if } y_{in} \geq 1 \\ 0 & \text{if } y_{in} < 1 \end{cases}$

Results:



```
MATLAB R2012a
File Edit Debug Parallel Desktop Window Help
Shortcuts How to Add What's New
Command History

Enter weights
Weight w1=2
Weight w2=2
Enter Threshold value
theta=1
Output of Net
    1    0    0    0

McCulloch-pitts Net for NOR function
Weights of Neuron
    2
    2
Threshold value
    1
```

Limitations of McCulloch-Pitts model

- i. Weights and thresholds are analytically determined.
- ii. Very difficult to minimize size of a network.

Conclusion

Hence we have studied and implemented basic logic gates using McCulloch-Pitts model.

```
import numpy as np
```

```
def Mcculloch_Pits(x,theta):
    sum = np.sum(x)
    print("Sum :",sum)
    if(sum >= theta):
        y = 1
    else :
        y = 0
    return y

def AND_percep(x):
    theta = 3
    return Mcculloch_Pits(x, theta)

def OR_percep(x):
    theta = 1
    return Mcculloch_Pits(x, theta)
```

```
example1 = np.array([0, 0, 0])
example2 = np.array([0, 0, 1])
example3 = np.array([0, 1, 0])
example4 = np.array([0, 1, 1])
example5 = np.array([1, 0, 0])
example6 = np.array([1, 0, 1])
example7 = np.array([1, 1, 0])
example8 = np.array([1, 1, 1])
```

```
print("Implementation of AND Logic Gate : ")
print("AND({}, {} , {}) = {}".format(0, 0, 0, AND_percep(example1)))
print("AND({}, {} , {}) = {}".format(0, 0, 1, AND_percep(example2)))
print("AND({}, {} , {}) = {}".format(0, 1, 0, AND_percep(example3)))
print("AND({}, {} , {}) = {}".format(0, 1, 1, AND_percep(example4)))
print("AND({}, {} , {}) = {}".format(1, 0, 0, AND_percep(example5)))
print("AND({}, {} , {}) = {}".format(1, 0, 1, AND_percep(example6)))
print("AND({}, {} , {}) = {}".format(1, 1, 0, AND_percep(example7)))
print("AND({}, {} , {}) = {}".format(1, 1, 1, AND_percep(example8)))
```

```
print("Implementation of OR Logic Gate : ")
print("OR({}, {} , {}) = {}".format(0, 0, 0, OR_percep(example1)))
print("OR({}, {} , {}) = {}".format(0, 0, 1, OR_percep(example2)))
print("OR({}, {} , {}) = {}".format(0, 1, 0, OR_percep(example3)))
print("OR({}, {} , {}) = {}".format(0, 1, 1, OR_percep(example4)))
print("OR({}, {} , {}) = {}".format(1, 0, 0, OR_percep(example5)))
print("OR({}, {} , {}) = {}".format(1, 0, 1, OR_percep(example6)))
print("OR({}, {} , {}) = {}".format(1, 1, 0, OR_percep(example7)))
print("OR({}, {} , {}) = {}".format(1, 1, 1, OR_percep(example8)))
```

☞ Implementation of AND Logic Gate :

Sum : 0
AND(0, 0 , 0) = 0
Sum : 1
AND(0, 0 , 1) = 0
Sum : 1
AND(0, 1 , 0) = 0
Sum : 2
AND(0, 1 , 1) = 0
Sum : 1
AND(1, 0 , 0) = 0
Sum : 2
AND(1, 0 , 1) = 0
Sum : 2
AND(1, 1 , 0) = 0
Sum : 3
AND(1, 1 , 1) = 1

Implementation of OR Logic Gate :

Sum : 0
OR(0, 0 , 0) = 0
Sum : 1
OR(0, 0 , 1) = 1
Sum : 1
OR(0, 1 , 0) = 1
Sum : 2
OR(0, 1 , 1) = 1
Sum : 1
OR(1, 0 , 0) = 1
Sum : 2
OR(1, 0 , 1) = 1
Sum : 2
OR(1, 1 , 0) = 1
Sum : 3
OR(1, 1 , 1) = 1