**Assignment No. 4**

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

**Problem Statement:**

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

**Objectives**:

1. To learn neural network.
2. To learn Mc-Culoch-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:-

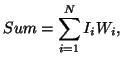
1. They are binary devices (Vi = [0,1])
2. Each neuron has a fixed threshold, theta values.
3. The neuron receives inputs from excitatory synapses, all having identical weights.
4. Inhibitory inputs have an absolute veto power over any excitatory inputs.
5. At each time step the neurons are simultaneously (synchronously) updated by summing the weighted excitatory inputs and setting the output (Vi) 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 *I1, I2, I3****,……I****m* and one output *y*. The linear threshold gate simply classifies the set of

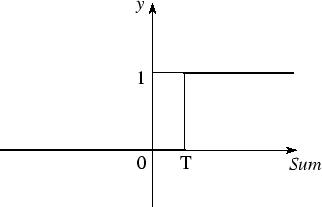
inputs into two different classes. Thus the output is *y* binary. Such a function can be described mathematically using these equations:



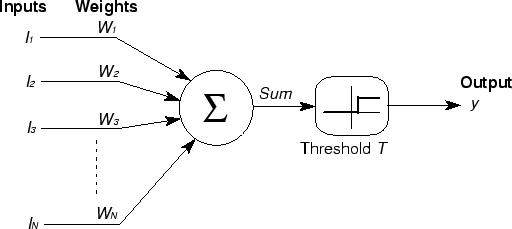


*W1, W2, W3, W4****,……….W****m* are weight values normalized in the range of either (0, 1) or (-1, 1) and

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



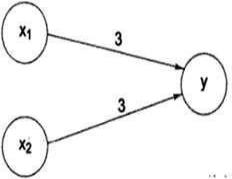
The McCulloch-Pitts model of a neuron is simple yet has substantial computing potential. It also has a precise mathematical definition. However, this model is so simplistic that it only generates a binary output and also the weight and threshold values are fixed. The neural computing algorithm has diverse 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)

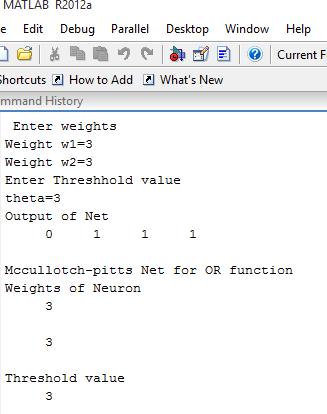


The net input is Yin=3A+3B. The output is given by

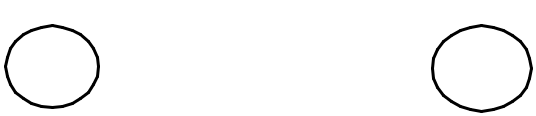
****Y=f (Yin) = 1 if Yin**≥3**

0 if Yin<3

**Results:**

****

* **Implementation of McCulloch-Pitts model for NOT gate**

****

1

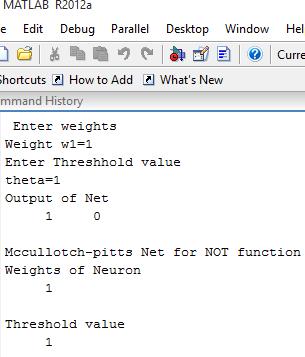
*  Y

Architecture of AND Gate (Threshold value=1)

****

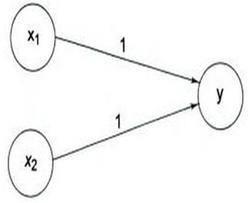
Activation function= Y=f (yin) = 1 if yin<1

0 if yin**≥1**

****

**Results:**

* **Implementation of McCulloch-Pitts model for AND gate**

****

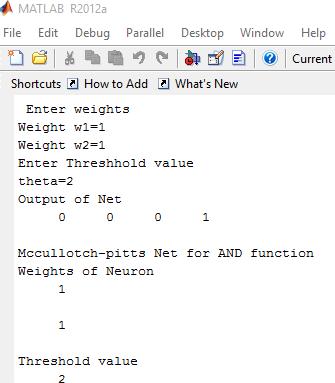
Architecture of AND Gate (Threshold value=2)

Net input is yin=A+B. Output is given by Y=f (yin)

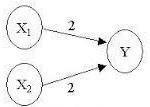


|  |  |  |  |
| --- | --- | --- | --- |
| Activation function= | 1 | if | yin**≥2** |
|  | 0 | if | yin<2 |

**Results:**

****

* **Implementation of McCulloch-Pitts model for NAND gate**

****

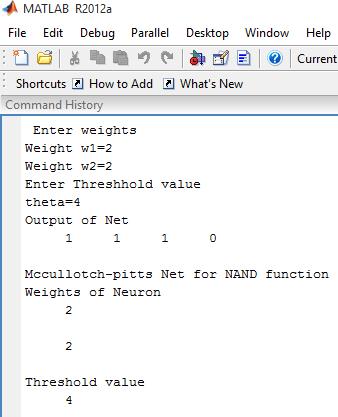
Architecture of NAND Gate (Threshold value=4)

Net input is yin=x1-x2. Output activation function is

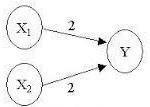
****y= f (yin) = 1 if yin**≥4**

0 if yin <4

**Results:**

****

* **Implementation of McCulloch-Pitts model for NOR gate**

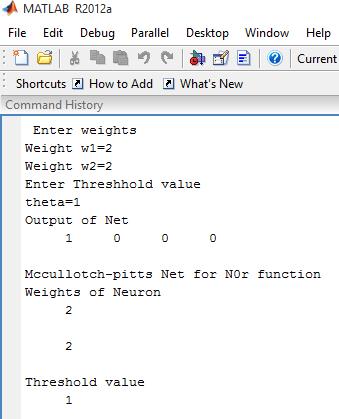
****

Architecture of NOR Gate (Threshold value=1)

****Activation function= **1 if yin ≥** 1

0 if yin <1

**Results:**

****

**Limitations of McCulloch-Pitts model**

1. Weights and thresholds are analytically determined.
2. Very difficult to minimize size of a network.

**Conclusion**

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