**Assignment No: 3**

**Prajakta Mane (BE/A/B1821016)**

**Aim**: Implement basic logic gates using Mc-Culoch-Pitts or Hebbnet neural networks.

**Objectives**:

1.The student will be able to obtain the fundamentals and different architecture of neural networks.

2. The student will have a broad knowledge in developing the different algorithms for neural networks.

**Software Requirements**:

Ubuntu 18.04

**Hardware Requirements**:

Pentium IV system with latest configuration

**Outcomes:** The students will be able to,

∙ Describe the relation between real brains and simple artificial neural network models.

∙ Understand the role of neural networks in engineering.

∙ Apply the knowledge of computing and engineering concept to this discipline.

**Theory:**

Neural network was inspired by the design and functioning of human brain and components. **Definition:**

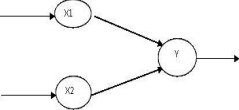
―Information processing model that is inspired by the way biological nervous system (i.e the brain) process information, is called Neural Network.

Neural Network has the ability to learn by examples. It is not designed to perform fix /specific task, rather task which need thinking (e.g. Predictions).

ANN is composed of large number of highly interconnected processing elements(neurons) working in unison to solve problems. It mimic human brain. It is configured for special

application such as pattern recognition and data classification through a learning process. ANN is 85-90% accurate.

**Basic Operation of a Neural Network:**

****

X1 and X2 – input neurons.

Y- output neuron

Weighted interconnection links- W1 and W2.

Net input calculation is :

Yin= x1w1+x2w2

Output is :

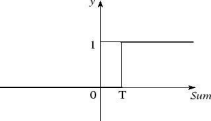
y=f(Yin)

Output= function

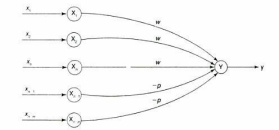
**The McCulloch-Pitts Model of Neuron:**

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…Im and one output y . The linear threshold gate simply classifies the set of inputs into two different classes. Thus the output y is binary. Such a function can be described mathematically using these equations:





W1,W2…Wm 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



A simple M-P neuron is shown in the figure.

It is excitatory with weight (w>0) / inhibitory with weight –p (p<0).

In the Fig., inputs from x1 to xn possess excitatory weighted connection and Xn+1 to xn+m has inhibitory weighted interconnections.

Since the firing of neuron is based on threshold, activation function is defined as



For inhibition to be absolute, the threshold with the activation function should satisfy the following condition:θ >nw –p

Output will fire if it receives ―k‖ or more excitatory inputs but no inhibitory inputs where kw≥θ>(k-1) w

- The M-P neuron has no particular training algorithm.

- An analysis is performed to determine the weights and the threshold. - It is used as a building block where any function or phenomenon is modelled based on a logic function.

-

X1 X2 Y

|  |
| --- |
| 1 |
|  |

0 1

-

Activation function Yin is as follows:

Yin=x1w1+x2w2

As we know,

--

Let Z1= and Z2= 

X1 X2 Z1

1 0 1

-

For Z1,

W11=1 and W12=-1

Θ=1

X1 X2 Z2

0 0

| 0 |
| --- |
| 1 |

0 1

1 0 0

1 1 0

For Z2,

W11=-1 and W12=1

Θ=1

Y=Z1+Z2

Z1 Z2 Y

0 0 0

0 1 1

1 0 1

1 1 0

For Y,

W11=1 and W12=1

Θ=1

**Input:**

| Input1 | Input2 | Output |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Python Code:**

import numpy as np

class Perceptron:

def \_\_init\_\_(self, input\_length, weights=None):

if weights is None:

self.weights = np.ones(input\_length) \* 0.5

else:

self.weights = weights

@staticmethod

def unit\_step\_function(x):

if x > 0.5:

return 1

return 0

def \_\_call\_\_(self, in\_data):

weighted\_input = self.weights \* in\_data

weighted\_sum = weighted\_input.sum()

return Perceptron.unit\_step\_function(weighted\_sum)

p = Perceptron(2, np.array([0.5, 0.5]))

for x in [np.array([0, 0]), np.array([0, 1]),

np.array([1, 0]), np.array([1, 1])]:

y = p(np.array(x))

print(x, y)

**Output:**

[0 0] 0

[0 1] 0

[1 0] 0

[1 1] 1

#Code = 2

import numpy as np

from random import randint

#inputs

X=np.array([[1,1,1],[1,-1,1],[-1,1,1],[-1,-1,1]])

#output

Y\_ad=np.array([[1],[-1],[-1],[-1]])

Y\_o=np.array([[1],[1],[1],[-1]])

print('input is:')

print(X)

print('output for And Gate is: ')

print(Y\_ad)

print('output for Or Gate is: ')

print(Y\_ad)

weights\_ad=np.zeros((3))

weights\_o=np.zeros((3))

print(weights\_ad)

# update weight for and gate /logic

def update\_weight\_ad(X,Y,weights):

for i in range(4):

weights=weights+X[i]\*Y[i]

#print weights

slope =-(weights[0]/weights[1])

c=-(weights[2]/weights[0])

if slope<0 and weights[0]>0:

weights\_main=weights

return weights\_main

def update\_weight\_o(X,Y,weights):

for i in range(4):

weights=weights+X[i]\*Y[i]

#print weights

slope =-(weights[0]/weights[1])

c=-(weights[2]/weights[0])

if slope<0 and weights[0]>0:

weights\_main=weights

return weights\_main

weights\_ad=update\_weight\_ad(X,Y\_ad,weights\_ad)

weights\_o=update\_weight\_o(X,Y\_o,weights\_o)

print('Checking after learning selectg a input')

rand\_int=int(input('Enter the test case no you want to try'))

print('Select a logic you also want to check')

logicgate=input()

print(weights\_ad)

print('selected input is %d '%rand\_int)

print(X[rand\_int])

def check\_learning(X,weights,rand\_int):

Yin=0

for i in range(3):

Yin+=X[rand\_int,i]\*weights[i]

if Yin<0:

Yin=-1

else:

Yin=1

return Yin

if logicgate=='a' or logicgate=='A':

weights\_in=weights\_ad

else:

weights\_in=weights\_o

Yin=check\_learning(X,weights\_in,rand\_int)

print(Yin)

Output:-



**Conclusion:** Mc-Culloch pits Model is implemented for XOR function by using the thresholding activation function. Activation of M-P neurons is binary (i.e) at any time step the neuron may fire or may not fire. Threshold plays major role here.