Name: Nachiket Erlekar

Roll No: 41434 Batch: Q4

Assignment 4

Title: Implement basic logic gates using McCulloch-Pitts or Hebb net neural networks.

Problem Statement: Implement basic logic gates using McCulloch-Pitts or Hebb net neural networks.

Objective:

1. To understand the concept of an artificial neuron.
2. To implement McCulloch-Pitts network.

Outcome:

Understood and implemented McCulloch-Pitts network.

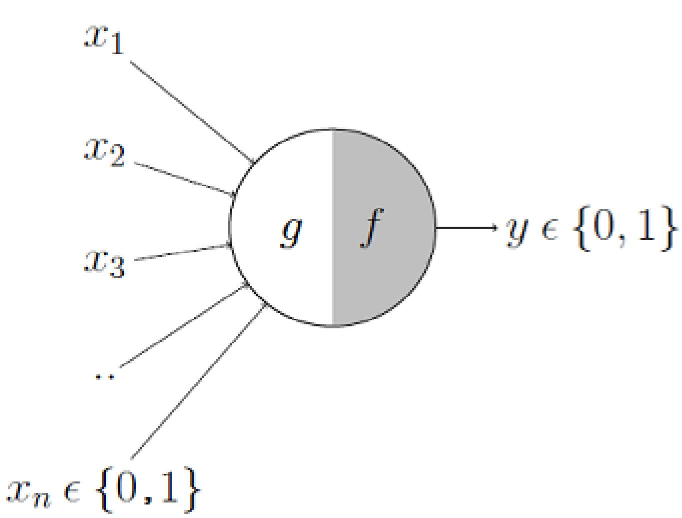
Hardware and Software requirements:

64-bit processor

Python 3, VS Code

Theory:

The first computational model of a neuron was proposed by Warren McCulloch (neuroscientist) and Walter Pitts (logician) in 1943.



The neuron is divided into two parts. The first part, g takes an input and performs an aggregation and based on the aggregated value the second part, f makes a decision.

$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

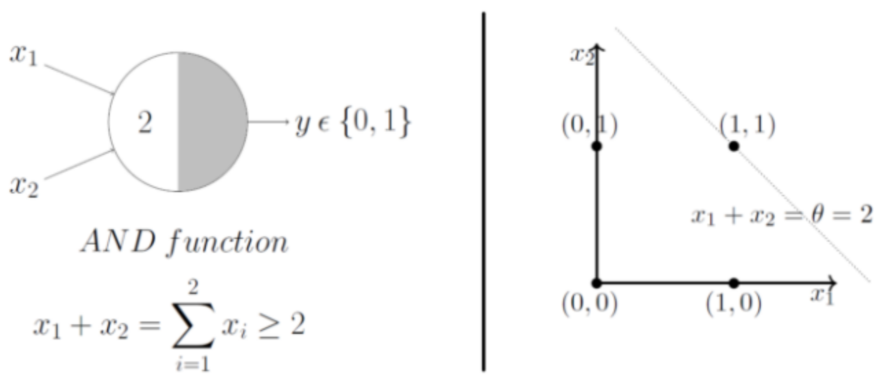
$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

Boolean Functions using M-P neuron:

For Boolean functions structure M-P neurons is simple. Aggregate function calculates sum of all inputs, if sum is more than threshold, the neuron fires.

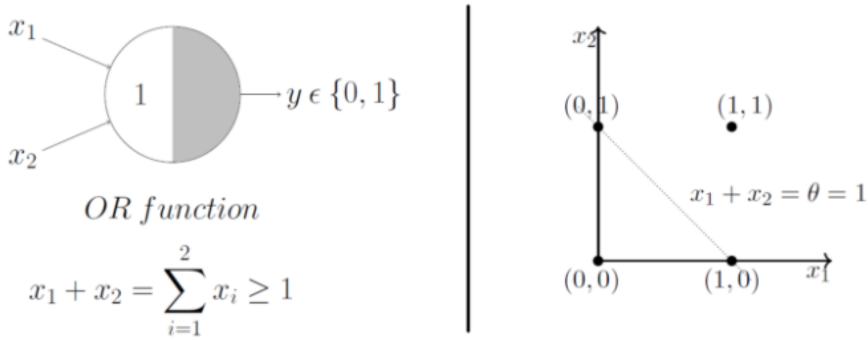
1. AND Function:

An AND function neuron would only fire when all the inputs are ON i.e., g(x) ≥ number of input features.



2. OR Function:

An OR function neuron would only fire when any of the inputs is ON i.e., g(x) ≥ 1.



3. NOT Function:

A NOT function neuron would only fire when input is OFF i.e., g(x) = 0.

4. NOR Function:

A NOR function neuron would only fire when all the inputs are OFF i.e., g(x) = 0.

Test Cases:

Operations	**Input**	**Expected O/P**	**Actual O/P**	**Result**
AND	0,0,0	0	0	Successful
AND	0,0,1	0	0	Successful
AND	1,1,0	0	0	Successful
AND	1,1,1	1	1	Successful
OR	0,0	0	0	Successful
OR	0,1	1	1	Successful
OR	1,0	1	1	Successful
OR	1,1	1	1	Successful
NOT	0	1	1	Successful
NOT	1	0	0	Successful
NOR	0,0	1	1	Successful
NOR	0,1	0	0	Successful
NOR	1,0	0	0	Successful
NOR	1,1	0	0	Successful

Conclusion:

Successfully implemented basic logic gates using McCulloch-Pitts neural network.

Source code and Output

Code

```
import numpy as np
def bitAnd(x):
    w=[2,1]
    y=np.dot(w,x)
    if y > 3 or y==3:
        return 1
    else:
        return 0

def bitOr(x):
    w=[1,1]
    y=np.dot(w,x)
    if y >= 1:
        return 1
    else:
        return 0

def Not(x):
    w=1
    y=w*x
    if y==0:
        return 1
    else:
        return 0

def bitNOR(x):
    w=[1,1]
    y=np.dot(w,x)
    if y > 1:
```

```
            return 0
        else:
            return 1

x1=int(input("Enter x1:"))
x2=int(input("Enter x2:"))
x=np.array([x1,x2])
ResAnd=bitAnd(x)
ResOr=bitOr(x)
xr=int(input('Enter the number whose not needs to be found: '))
ResNOT=Not(xr)
ResNOR=bitNOR(x)

print('----------------------')
print('Result of AND: ',ResAnd)
print('\n')
print('Result of OR: ',ResOr)
print('\n')
print('Result of NOT: ',ResNOT)
print('\n')
print('Result of NOR: ',ResNOT)
print('\n')
```

Output—

```
Enter x1:1

Enter x2:0

Enter the number whose not needs to be found: 1
----------------------
```

Result of AND: 0

Result of OR: 1

Result of NOT: 0

Result of NOR: 0