Experiment No. 1
Implement Mc-Culloch Pitts model for binary logic functions.
Date of Performance: 08/01/24
Date of Submission: 01/02/24

Experiment No. 1

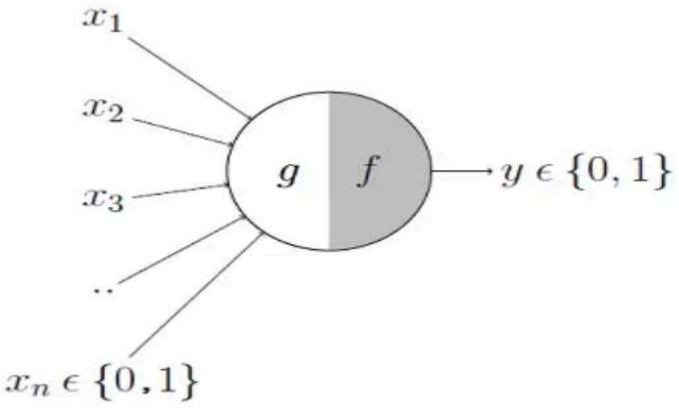
Title: Implement Mc-Culloch Pitts model for binary logic functions.

Aim: To study Mc-Culloch Pitts model

Objective: To implement basic neural network models for simulating logic gates.

Theory:

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



The McCulloch-Pitts neuron takes binary inputs (0 or 1) and produces a binary output based on a threshold criterion. 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.
These inputs can either be *excitatory* or *inhibitory*. Inhibitory inputs are those that have maximum effect on the decision making irrespective of other inputs i.e., if ***x_3*** is 1 (not home) then my output will always be 0 i.e., the neuron will never fire, so ***x_3*** is an inhibitory input. Excitatory inputs are NOT the ones that will make the neuron fire on their own but they might fire it when combined together. Formally, this is what is going on:

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

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

We can see that ***g*(x)** is just doing a sum of the inputs — a simple aggregation. And ***theta*** here is called thresholding parameter. For example, if I always watch the game when the sum turns out to be 2 or more, the ***theta*** is 2 here. This is called the Thresholding Logic.

While the McCulloch-Pitts neuron is a foundational concept, it is quite simplistic compared to modern artificial neural networks, which incorporate more complex architectures and activation functions.

Code:

```
#AND
print("AND operation")
def fire(theta,sum):
    if(sum>=theta):
        return 1
    else:
        return 0
inp = [[1,1,1],[1,1,0],[0,0,1],[0,0,0]]
for i in inp:
    sum=0
    for j in i:
        sum+=j
    print("AND(",i,")=",fire(len(i),sum))


#OR
print("OR operation")
def fire(theta,sum):
    if(sum>=theta):
        return 1
    else:
        return 0
inp = [[1,1,1],[1,1,0],[0,0,1],[0,0,0]]
for i in inp:
    sum=0
    for j in i:
        sum+=j
    print("OR(",i,")=",fire(1,sum))

#OR
print("NOT operation")
def firen(sum,theta):
    if(sum==theta):
        return 1
    else:
        return 0

inpt = [0,1]
for i in inpt:
    print("NOT(",i,")=",firen(0,i))
```

Output:

```
AND operation

AND( [1, 1, 1] )= 1

AND( [1, 1, 0] )= 0

AND( [0, 0, 1] )= 0

AND( [0, 0, 0] )= 0

OR operation

OR( [1, 1, 1] )= 1

OR( [1, 1, 0] )= 1

OR( [0, 0, 1] )= 1

OR( [0, 0, 0] )= 0

NOT operation

NOT( 0 )= 1

NOT( 1 )= 0

>
```

Conclusion: McCulloch-Pitts neuron is simple yet powerful binary input processing and threshold-based decision-making laid the groundwork for modern artificial neural networks. It provides essential insights into neuronal behavior, driving the evolution of more complex architectures and activation functions. While its simplicity may contrast with contemporary complexities, its profound impact is undeniable. By elucidating fundamental neuronal behaviors, it has facilitated the emergence of advanced architectures and activation functions