NAME	**Shreya Shetty**
UID	**2019140059**
CLASS	**TE IT**
BATCH	**B**
SUBJECT	**Soft Computing Lab**

Aim:

To implement (MP-Neuron) Mc-Culloch Pitts Model

Theory:

ANN (Artificial Neural Network)

The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.

Weights

Each neuron is connected to other neurons by means of directed communication links, each with associated weight. The weight represents information being used by the net to solve a problem.

Activation Function

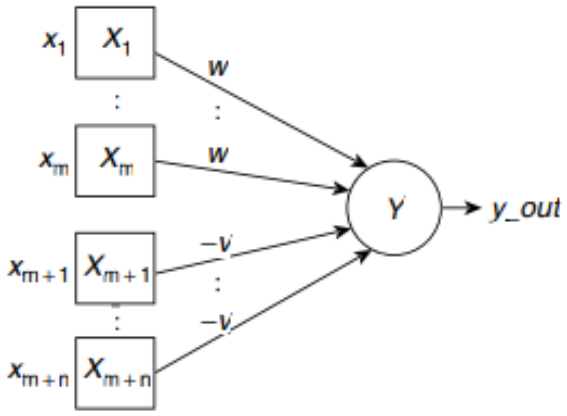
An activation function is a function outputs a small value for small inputs and a larger value if its inputs exceed the threshold. If the inputs are large enough, the activation function "fires", otherwise it does nothing.

Threshold

Threshold is a set value based upon which the final output of the network may be calculated. The threshold value is used in activation function.

McCulloch-Pitts Neuron

- The earliest artificial neural model was proposed by McCulloch and Pitts in 1943. It consists of a number of input units connected to a single output unit. The interconnecting links are unidirectional.
- There are two kinds of inputs, namely, excitatory inputs and inhibitory inputs. In the below Figure, the excitatory inputs are shown as inputs X_1 , …, X_m and the inhibitory inputs are X_{m+1} , …, X_{m+n} .
- The excitatory inputs are connected to the output unit through positively weighted links. Inhibitory inputs have negative weights on their connecting paths to the output unit.



- All excitatory weights have the same positive magnitude w and all inhibitory weights have the same negative magnitude −v.
- The activation y_out = f(y_in) is binary, i.e., either 1 (in case the neuron fires), or 0 (in case the neuron does not fire).
- The activation function is a binary step function. It is 1 if the net input y_in is greater than or equal to a given threshold value q, and 0 otherwise.
- The inhibition is absolute. A single inhibitory input should prevent the neuron from firing irrespective of the number of active excitatory inputs.

The net input y_in to the neuron Y is given by

$$y\_in = \sum_{i=1}^{m} x_i \times w + \sum_{j=m+1}^{n} x_j \times (-v) = w\sum_{i=1}^{m} x_i - v\sum_{i=m+1}^{n} x_j$$

If θ be the threshold value, then the activation of Y, i.e., y_out, is obtained as

$$y\_out = f(y\_in) = \begin{cases} 1, & if\ y\_in \geq \theta, \\ 0, & otherwise. \end{cases}$$

To ensure absolute inhibition, y_in should be less than the threshold even when a single inhibitory input is on while none of the excitatory inputs are off. Assuming that the inputs are binary, i.e., 0 or 1, the cri terion of absolute inhibition requires

$$y\_in = w\sum_{i=1}^{m} x_i - v < \theta$$

$$\therefore w \times m - v < \theta.$$

Tool/Language:

Programming language: Python
Libraries Used: numpy

Implementation:

OR GATE:

Shreya Shetty TE IT 2019140059

Page No.
Date

* Implement OR function using McCulloch-Pitts neuron.

Consider the truth table for OR function -

x_1	x_2	$y = x_1 OR x_2$	$y_{in} = x_1w_1 + x_2w_2$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	2

Let $w_1 = 1$ & $w_2 = 1$

With these assumed weights the net input is calculated as.

$(1,1)$, $y_{in} = x_1w_1 + x_2w_2 = 1\times1 + 1\times1 = 2$

$(1,0)$, $y_{in} = x_1w_1 + x_2w_2 = 1\times1 + 0\times1 = 1$

$(0,1)$, $y_{in} = x_1w_1 + x_2w_2 = 0\times1 + 1\times0 = 1$

$(0,0)$, $y_{in} = x_1w_1 + x_2w_2 = 0\times1 + 0\times1 = 0$

The input is high when atleast one of the inputs is high.

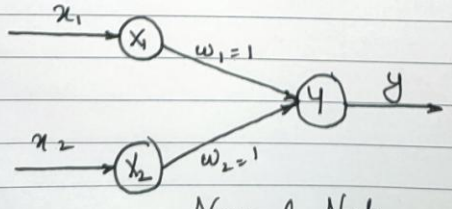
The threshhold value is set to 1.

$\therefore \theta = 1$

The output of neuron Y can be written as -

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

where '1' represents θ y_{in}



Neural Net

NOR GATE:

Shreya Shetty TE IT 2019140059

Page No.
Date

* Implement NOR function using McColloch-Pitts Neuron.

Consider the truth table for NOR function -

x_1	x_2	$y = x_1$ NOR x_2	$y_{in} = x_1w_1 + x_2w_2$
0	0	1	0
0	1	0	-1
1	0	0	-1
1	1	0	-2

Let $w_1 = -1$, $w_2 = -1$

With these assumed weights the net input is calculated as -

(0,0), $y_{in} = x_1w_1 + x_2w_2 = 0 \times (-1) + 0 \times (-1) = 0$

(0,1), $y_{in} = x_1w_1 + x_2w_2 = 0 \times (-1) + 1 \times (-1) = -1$

(1,0), $y_{in} = x_1w_1 + x_2w_2 = 1 \times (-1) + 0 \times (-1) = -1$

(1,1), $y_{in} = x_1w_1 + x_2w_2 = 1 \times (-1) + 1 \times (-1) = -2$

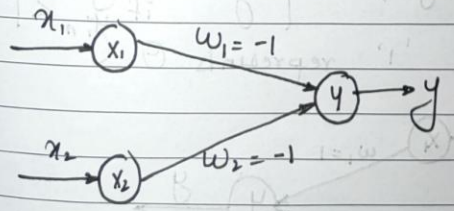
The input is high when both inputs is low.

$\therefore \Theta = 0$ i.e. threshold is set to 0.

The output of neuron Y can be written as -

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

where '0' represents Θ



Neural Net

Code:

```
import numpy as np
inputs = np.array([[0,0], [0,1], [1,0], [1,1]]) # Array of binary input
print('*****************************************************')
print('McCulloch-Pitts Neuron to implement logical OR operation')
print('\nInput Table:\n',inputs)

# OR FUNCTION
weights_or = np.array([1,1])  # Weights Array of OR
print('\nWeights:',weights_or)

# Net input  yin=x1*w1+x2*w2
# Calculating net input for the 4 inputs by taking dot product of Input Array
and weights array
y_in = inputs @ weights_or
print('\nyin: ',y_in)

# Threshold will be 1 for OR operation
t = 1
print('\n----------------------------------------')
print('|  x1\t|  x2\t|  y_in\t|  y = x1 OR x2  |', end =" ")
print('\n----------------------------------------', end =" ")
# Finding output of neuron Y is calculated as y = f(yin)
# y = 1 if y_in >= threshold
# y = 0 if y_in < threshold
y_out=[]
for i in range(0,4):
    if y_in[i] >= t:
        y_out.append(1)
    else:
        y_out.append(0)
    print(f'\n|  {inputs[i][0]}\t|  {inputs[i][1]}\t|  {y_in[i]}\t|\t
 {y_out[i]}  \t |', end =" ")
print('\n----------------------------------------')
print('\n*****************************************************')
```

```
# NOR Function
# array of weights
print('\n*****************************************************')
print('McCulloch-Pitts Neuron to implement logical NOR operation\n')
print('Input Table:\n',inputs)
weights_nor = np.array([-1,-1])  # Weights Array of NOR
print('\nWeights:',weights_nor)

# Calculating net input for the 4 inputs
y_in = inputs @ weights_nor
print('\nyin: ',y_in)
```

```
# Threshold will be 0 for NOR operation
t = 0
print('\n-----------------------------------------')
print('|  x1\t|  x2\t|  y_in\t| y = x1 NOR x2  |', end =" ")
print('\n-----------------------------------------', end =" ")
# Finding output of neuron Y is calculated as y = f(yin)
# y = 1 if y_in >= threshold
# y = 0 if y_in < threshold
y_out=[]
for i in range(0,4):
    if y_in[i] >= t:
        y_out.append(1)
    else:
        y_out.append(0)
    print(f'\n|  {inputs[i][0]}\t|  {inputs[i][1]}\t|  {y_in[i]}\t|\t
 {y_out[i]}  \t |', end =" ")
print('\n-----------------------------------------')
print('\n*********************************************************')
```

Output:

OR Operation:

```
*********************************************************
McCulloch-Pitts Neuron to implement logical OR operation

Input Table:
 [[0 0]
 [0 1]
 [1 0]
 [1 1]]

Weights: [1 1]

yin:  [0 1 1 2]

-----------------------------------------
|  x1   |  x2   |  y_in |  y = x1 OR x2  |
-----------------------------------------
|  0    |  0    |  0    |        0       |
|  0    |  1    |  1    |        1       |
|  1    |  0    |  1    |        1       |
|  1    |  1    |  2    |        1       |
-----------------------------------------

*********************************************************
```

NOR Operation:

```
**********************************************************
McCulloch-Pitts Neuron to implement logical NOR operation

Input Table:
 [[0 0]
 [0 1]
 [1 0]
 [1 1]]

Weights: [-1 -1]

yin:  [ 0 -1 -1 -2]

----------------------------------------
|  x1   |  x2   |  y_in | y = x1 NOR x2  |
----------------------------------------
|  0    |  0    |  0    |         1      |
|  0    |  1    |  -1   |         0      |
|  1    |  0    |  -1   |         0      |
|  1    |  1    |  -2   |         0      |
----------------------------------------

**********************************************************
```

Conclusion:

In this experiment, I learned about various terminologies such as neuron, threshold, weights, and activation function and I implemented various logic gates (OR and NOR) using Mc-Culloch Pitts Model in Python.