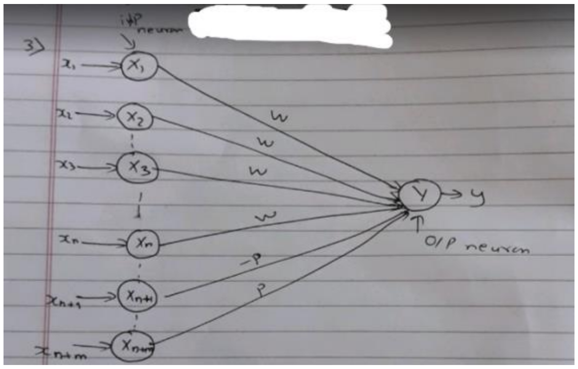
Aim-Generate ANDNOT function using McCulloch-Pitts neural net by python program.

Software for Python: Jupyter Notebook

Theory:

In McCulloch-Pitts neural net, there is a fixed threshold ϴ for each neuron and if the net input to the neuron is greater than threshold then the neuron fires.

It is most widely used in logic functions.



X1,X2,X3,….,Xn,Xn+1,and Xn+m are input neurons and Y is the output neuron.

The input neurons are connected to the output neurons with excitatory weights w (w>0 or positive weights) or inhibitory weights p (p<0 or negative weights).

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

f(yin)=1, yin≥ϴ

=0, yin<ϴ

If inhibitory weights are used, threshold with activation function should satisfy following condition

ϴ= nw-p

n-number of input vectors

w-no of excitatory weights

p-no of inhibitory weights

1)Draw the Truth Table of ANDNOT Function

X1	X2	Y
0	0	0
0	1	0
1	0	1
1	1	0

2)From the truth table, when the first input X1=1 and second input X2=0,the neuron fires so Y=1.

3)Assume both weights w1 and w2 as excitatory, w1=w2=1

4)calculate the net input for 4 inputs using the formula

$yin=x1w1+x2w2$

for $x1=0,x2=0$, $yin1=0\times1+0\times1=0+0=0$

for $x1=0$, $x2=1$, $yin2=0\times1+1\times1=0+1=1$

for $x1=1$, $x2=0$, $yin3=1\times1+0\times1=1+0=1$

for $x1=1$, $x2=1$, $yin4=1\times1+1\times1=1+1=2$

5)If we set $\Theta\geq1$, all neurons with yin1, yin2, yin3, and yin4 will get fired as their values are greater than or equal to 1.

6)It is not possible to fire neurons for inputs x1=1 and x2-0.Hence these weights are not suitable.

7) Assume one weights w1 is excitatory, w1=1 and one weight w2 as inhibitory w2=-1 and bias b=1

8) calculate the net input for 4 inputs using the formula

$yin=x1w1+x2w2$

for $x1=0$, $x2=0$, $yin1=0\times1+0\times-1=0+0=0$

for $x1=0$, $x2=1$, $yin2=0\times1+1\times-1=0-1=-1$

for $x1=1$, $x2=0$, $yin3=1\times1+0\times-1=1+0=1$

for $x1=1$, $x2=1$, $yin4=1\times1+1\times-1=1-1=0$

9) Now it is possible to fire the neuron for input x1=1 and x2=0 only by fixing the threshold of 1.

$\Theta\leq1$

Thus $w1=1$, $w2=-1$, $\Theta\leq1$

Value of Θ can be calculated by equation

$\Theta\leq nw-p$

$\Theta\leq (2\times1)-1$

$\Theta\leq2-1$

$\Theta\leq1$

10)The output neuron Y can be written as

$y=f(yin)=1, yin\geq1$

$\qquad\quad =0, yin<1$

In our case yin3=1

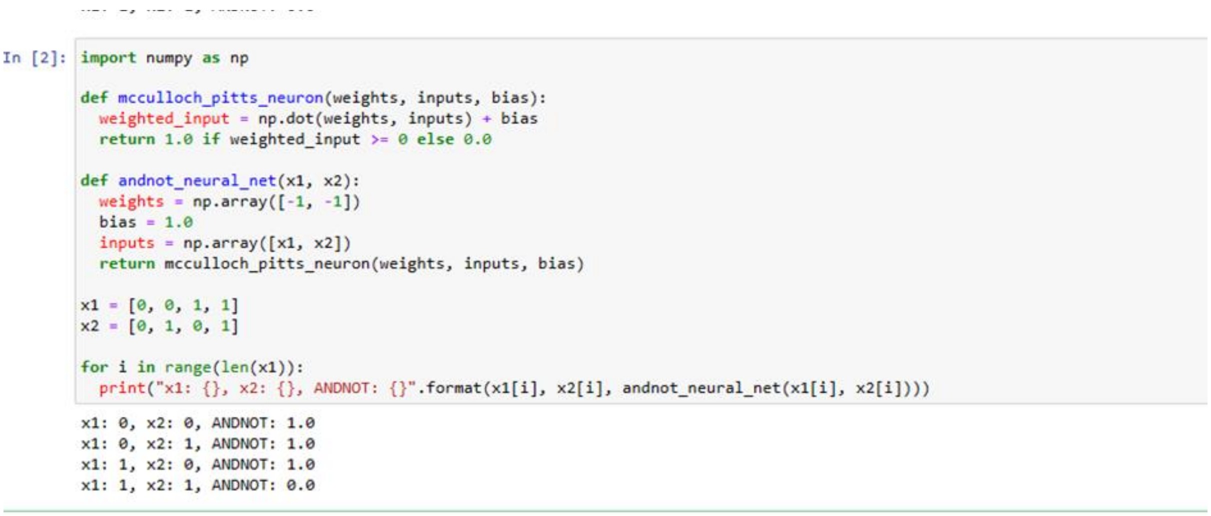
y=f(yin3) =1

y=1

Procedure:

1. First import the library numpy as np.
2. The mcculloch_pitts_neuron function implements a single Mcculloch-Pitts neuron, which computes the dot product of the weights and input arrays and adds bias to obtain the weighted input.
3. If the weighted input is greater than or equal to zero, the function returns 1.0, otherwise it returns 0.0.
4. The andnot_neural _net function implements the ANDNOT logic gate using two McCullochs-Pitts neurons and the appropriate weight and bias.
5. The code defines the inputs x1 and x2, and runs the andnot_neural_net function for each pair of inputs to demonstrate the behavior of the ANDNOT function.

Program Code and Ootput



```
In [2]: import numpy as np

        def mcculloch_pitts_neuron(weights, inputs, bias):
          weighted_input = np.dot(weights, inputs) + bias
          return 1.0 if weighted_input >= 0 else 0.0

        def andnot_neural_net(x1, x2):
          weights = np.array([-1, -1])
          bias = 1.0
          inputs = np.array([x1, x2])
          return mcculloch_pitts_neuron(weights, inputs, bias)

        x1 = [0, 0, 1, 1]
        x2 = [0, 1, 0, 1]

        for i in range(len(x1)):
          print("x1: {}, x2: {}, ANDNOT: {}".format(x1[i], x2[i], andnot_neural_net(x1[i], x2[i])))
```

```
x1: 0, x2: 0, ANDNOT: 1.0
x1: 0, x2: 1, ANDNOT: 1.0
x1: 1, x2: 0, ANDNOT: 1.0
x1: 1, x2: 1, ANDNOT: 0.0
```