(1-1). Train the Perceptron Model to simulate OR operations. There are two input units and one
output unit. Input patterns are (0, 0), (0, 1), (1, 0), and (1, 1). The output will be either 1 or 0. Initial weights are given as 0.3 and 0.7. Learning coefficient is 0.6. Threshold value of output unit
is 3.2

x_1	x_2	y
0	0	o
0	1	1
1	0	1
1	1	1

Where x_1 and x_2 are the inputs and y represent the output

$x_1 = 0$,
$x_2 = 0$
t= 0
wi.xi = 0*0.3 + 0*0.7 = 0 < 3.2
Actual output = 0

$x_1 = 0$,
$x_2 = 1$ t= 1
wi.xi = 0*0.3 + 1*0.7 = 0.7
0.7< 3.2,
output = 0

wi = wi + n(t - o)xi
$w_1 = 0.3 + 0.6(1 - 0)0 = 0.3$
$w_2 = 0.7 + 0.6(1 - 0)1 = 1.3$

i)
$x_1 = 0$, $x_2 = 0$ t = 0
wi.xi = 0*0.3 + 0*1.3 = 0
0< 3.2, output = 0

ii)
$x_1 = 0$, $x_2 = 1$ and Target = 1
wi.xi = 0*0.3 + 1*1.3 = 1.3
Threshold = 3.2< the threshold of 3.2, so the output = 0
wi = wi + n(t - o)xi
w 1 = 0.3 + 0.6(1 - 0)0 = 0.3
w 2 = 1.3 + 0.6(1 - 0)1 = 1.9

i)
$x_1 = 0$, $x_2 = 0$ and Target = 0
wi.xi = 0*0.3 + 0*1.9 = 0

< the threshold of 3.2, so the output = 0

ii)
x_1 = 0, x_2 = 1 and Target = 1
wi.xi = 0*0.3 + 1*1.9 = 1.9
< han the threshold of 3.2, so the output = 0
wi = wi + n(t - o)xi
w 1 = 0.3 + 0.6(1 - 0)0 = 0.3
w 2 = 1.9 + 0.6(1 - 0)1 = 2.5

i)
x_1 = 0, x_2 = 0 and Target = 0
wi.xi = 0*0.3 + 0*2.5 = 0
< threshold of 3.2, so the output = 0
ii)
x_1 = 0, x_2 = 1 and Target = 1
wi.xi = 0*0.3 + 1*2.5 = 2.5
This is not greater than the threshold of 3.2, so the output = 0
wi = wi + n(t - o)xi
w 1 = 0.3 + 0.6(1 - 0)0 = 0.3
w 2 = 2.5 + 0.6(1 - 0)1 = 3.1i)
x_1 = 0, x_2 = 0 and Target = 0
wi.xi = 0*0.3 + 0*3.1 = 0
0 is not greater than the threshold of 3.2, s
output = 0

ii)
x_1 = 0,x_2 = 1 and Target = 1
wi.xi = 0*0.3 + 1*3.1 = 3.1
This is not greater than the threshold of 3.2,
output = 0

wi = wi + n(t - o)xi
w 1 = 0.3 + 0.6(1 - 0)0 = 0.3
w 2 = 3.1 + 0.6(1 - 0)1 = 3.7

i)
x_1 = 0, x_2 = 0 and Target = 0
wi.xi = 0*0.3 + 0*3.7 = 0
0 is not greater than the threshold of 3.2, so the
output = 0

ii)
x_1 = 0, x 2 = 1 and Target = 1
wi.xi = 0*0.3 + 1*3.7 = 3.7
This is greater than the threshold of 3.2, so the output = 1
Actual output = 1
Updated weights, w 1 =0.3, w 2 = 3.7

iii)
x_1 = 1, x_2 = 0 and Target = 1
wi.xi = 1*0.3 + 0*3.7 = 0.3

This is not greater than the threshold of 3.2, so the output = 0
wi = wi + n(t – o)xi
w 1 = 0.3 + 0.6(1 - 0)1 = 0.9
w 2 = 3.7 + 0.6(1 - 0)0 = 3.7i)
x_1 = 0, x_2 = 0 and Target = 0
wi.xi = 0*0.9 + 0*3.7 = 0

This is not greater than the threshold of 3.2,
output = 0

ii)
x_1 = 0, x 2 = 1 and Target = 1
wi.xi = 0*0.9 + 1*3.7 = 3.7
>3.2,

output = 1

iii)
x_1 = 1, x_2 = 0 and Target = 1
wi.xi = 1*0.9 + 0*3.7 = 0.9
< threshold of 3.2,
output = 0
wi = wi + n(t - o)xi
w 1 = 0.9 + 0.6(1 - 0)1 = 1.5
w 2 = 3.7 + 0.6(1 - 0)0 = 3.7

i)
x_1 = 0, x_2 = 0 and Target = 0
wi.xi = 0*1.5 + 0*3.7 = 0
< than the threshold of 3.2,

output = 0

ii)
x_1 = 0, x_2 = 1 and Target = 1
wi.xi = 0*1.5 + 1*3.7 = 3.7
>than the threshold of 3.2,

output = 1

iii)
x_1 = 1, x_2 = 0 and Target = 1
wi.xi = 1*1.5 + 0*3.7 = 1.5
< threshold of 3.2,

output = 0

wi = wi + n(t - o)xi
w 1 = 1.5 + 0.6(1 - 0)1 = 2.1
w 2 = 3.7 + 0.6(1 - 0)0 = 3.7i)
x_1 = 0, x_2 = 0 and Target = 0
wi.xi = 0*2.1 + 0*3.7 = 0

< than the threshold of 3.2,

output = 0
ii)
x_1 = 0, x_2 = 1 and Target = 1
wi.xi = 0*2.1 + 1*3.7 = 3.7
>of 3.2,

output = 1
iii)
x_1 = 1, x_2 = 0 and Target = 1
wi.xi = 1*2.1 + 0*3.7 = 2.1
< than the threshold of 3.2,

output = 0
wi = wi + n(t - o)xi
w 1 = 2.1 + 0.6(1 - 0)1 = 2.7
w 2 = 3.7 + 0.6(1 - 0)0 = 3.7
i)
x 1 = 0, x 2 = 0 and Target = 0
$x_1$xi = 0*2.7 + 0*3.7 = 0
<the threshold of 3.2,

output = 0
ii)
x_1 = 0, x_2 = 1 and Target = 1
wi.xi = 0*2.7 + 1*3.7 = 3.7
>threshold of 3.2, so the output = 1
iii)
x_1 = 1,x_2 = 0 and Target = 1
wi.xi = 1*2.7 + 0*3.7 = 2.7
< the threshold of 3.2,
output = 0
wi = wi + n(t - o)xi
w 1 = 2.7 + 0.6(1 - 0)1 = 3.3
w 2 = 3.7 + 0.6(1 - 0)0 = 3.7i)
x_1 = 0, x_2 = 0 and Target = 0
wi.xi = 0*3.3 + 0*3.7 = 0
< than the threshold of 3.2,

output = 0
ii)
x_1 = 0, x_2 = 1 and Target = 1
wi.xi = 0*3.3 + 1*3.7 = 3.7
> than the threshold of 3.2,

output = 1

iii)
x_1 = 1, x_2 = 0 and Target = 1
wi.xi = 1*3.3 + 0*3.7 = 3.3
> than the threshold of 3.2, so

outout= 1
Actual Output = 1
Updated Weights, w 1 = 3.3, w 2 = 3.7
iv)
x_1 = 1, x 2 = 1 and Target = 1
wi.xi = 1*3.3 + 1*3.7 = 7
> than the threshold of 3.2,
output = 1
Actual Output = 1
Final Weights, w 1 = 3.3, w 2 = 3.7

(1-2). Train the Perceptron Model to simulate NOR (Not OR) operations. There are two input units and one output unit. Input patterns are (0, 0), (0, 1), (1, 0), and (1, 1). The output will be either 1 or 0. Initial weights are given as 0.2 and -0.4. Learning coefficient is 0.5. Threshold value
of output unit is -1.1.

NOR(0, 0) = 1
NOR(0, 1) = 0
NOR(1, 0) = 0
NOR (1, 1) =00wi.xi = 1*-1.3 + 1*-1.4= -2.7
< the threshold of -1.1,

output = 0
Actual Output = 0
Final Weights,
w_1=-1.3, w_1= -1.4

(1-3). Train the Perceptron Model to simulate NAND (Not AND) operations. There are two input
units and one output unit. Input patterns are (0, 0), (0, 1), (1, 0), and (1, 1). The output will be either 1 or 0. Initial weights are given as 0.4 and 0.9. Learning coefficient is 0.8. Threshold value
of output unit is 1.8.
If you do not get the solution with threshold of 1.8, then change the threshold as -1.8 and solve it.
Through your training process, you need to write the followings from the initial status to the final
weights for each epoch.
• actual output
• updated weight

x_1 = 0, x_2 = 0 and Target = 1
wi.xi = 0*0.4 + 0*0.9 = 0
This is not greater than the threshold of 1.8,

output = 0

wi = wi + n(t - o)xi

$w_1 = 0.4 + 0.8(1 - 0)0 = 0.4$
$w_2 = 0.9 + 0.8(1 - 0)0 = 0.9$
We change the threshold to -1.8
$w_1 = 0.4$
$w_2 = 0.9$
Learning coefficient n = 0.8
Threshold = -1.8i)
$x_1 = 0$, $x_2 = 0$ and Target = 1
wi.xi = 0*0.4 + 0*0.9 = 0
1> than the threshold of -1.8,

output = 1
Actual Output = 1
Initial Weights: $w_1 = 0.4$, $w_2 = 0.9$
ii)
$x_1 = 0$, $x_2 = 1$ and Target = 1
wi.xi = 0*0.4 + 1*0.9 = 0.9
0.9>Than the threshold of -1.8,

output = 1
Actual Output = 1
Initial Weights: $w_1 = 0.4$, $w_2 = 0.9$
iii)
$x_1 = 1$, $x_2 = 0$ and Target = 1
wi.xi = 1*0.4 + 0*0.9 = 0.4
0.4 threshold of -1.8,

output = 1
Actual Output = 1
Initial Weights: $w_1 = 0.4$, $w_2 = 0.9$
iv)
$x_1 = 1$, $x_2 = 1$ and Target = 0
wi.xi = 1*0.4 + 1*0.9 = 1.3
> threshold of -1.8,

output = 1
wi = wi + n(t - o)xi
$w_1 = 0.4 + 0.8(0 - 1)1 = -0.4$
$w_2 = 0.9 + 0.8(0 - 1)1 = 0.1$
i)
$x_1 = 0$, $x_2 = 0$ and Target = 1
wi.xi = 0*-0.4 + 0*0.1 = 0
0> -1.8,

output = 1

ii)
$x_1 = 0$, $x_2 = 1$ and Target = 1wi.xi = 0*-0.4 + 1*0.1 = 0.1
0.1 > threshold of -1.8,

output = 1

iii)
$x_1 = 1$, $x_2 = 0$ and Target = 1
wi.xi = 1*-0.4 + 0*0.1 =- 0.4
> the threshold of -1.8,

output = 1
iv)
$x_1 = 1$, $x_2 = 1$ and Target = 0
wi.xi = 1*-0.4 + 1*0.1 = -0.3
> threshold of -1.8,

output = 1
wi = wi + n(t - o)xi
w 1 = -0.4 + 0.8(0 - 1)1= -1.2
w 2 = 0.1 + 0.8(0 - 1)1 = -0.7
i)
$x_1 = 0$, $x_2 = 0$ and Target = 1
wi.xi = 0*-1.2 + 0*-0.7 = 0
> threshold of -1.8,

output = 1

ii)
$x_1 = 0$, $x_2 = 1$ and Target = 1
wi.xi = 0*-1.2 + 1*-0.7 = -0.7
> threshold of -1.8,

output = 1
iii)
$x_1 = 1$,$x_2 = 0$ and Target = 1
wi.xi = 1*-1.2 + 0*-0.7 =- -1.2
> hreshold of -1.8,

output = 1
iv)
$x_1 = 1$, $x_2 = 1$ and Target = 0
wi.xi = 1*-1.2 + 1*-0.7 = -1.9
< the threshold of -1.8,

output = 0
Actual output = 0Final weights, w 1 = -1.2, w 2 = -0.7

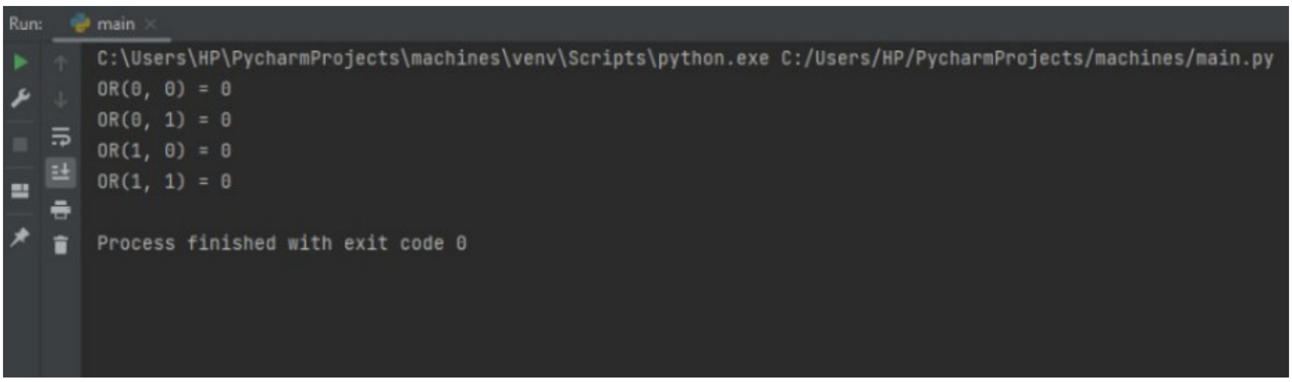
(2-1). Program the Perceptron Model to simulate OR operations. There are two input units and
one output unit. Input patterns are (0, 0), (0, 1), (1, 0), and (1, 1). The output will be either 1 or 0.
Initial weights are given as 0.5 and 0.8. Learning coefficient is 0.4. Threshold value of output unit
is 2.9.

the python code that represent the perceptron algorithm

important to importing nampy library

```
import numpy as np

def StepFunction(threshold):
if threshold >= 3.1:
return 1
else:
return 0
# designing the Perceptron
def designed_Percept(x, weight, bias):
threshold = np.dot(weight, x) + bias
# print("Thereshold: ", threshold)
y = StepFunction(threshold)
return y

# w1 = 0.9, w2 = 0.8, bias = 0.7
def OR_logic(x):
weight = np.array([0.9, 0.8])
bias =0.7
return designed_Perceptron(x, weight, bias)
#  Perceptron Model
input_01 = np.array([0, 1])
input_11 = np.array([1, 1])
input_00 = np.array([0, 0])
input_10 = np.array([1, 0])
print("OR({}, {}) = {}".format(0, 0, OR_logic(input_00)))
print("OR({}, {}) = {}".format(0, 1, OR_logic(input_01)))
print("OR({}, {}) = {}".format(1, 0, OR_logic(input_10)))print("OR({}, {}) = {}".format(1, 1, OR_logic(input_11)))
```

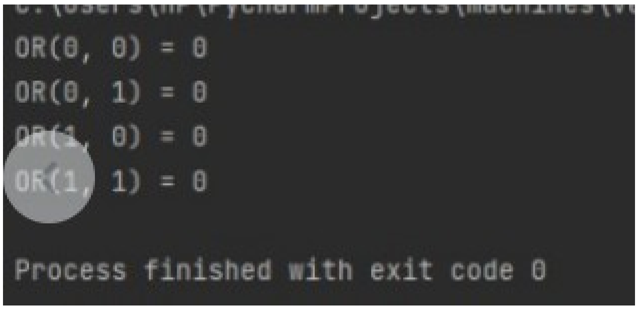
output



(2-2). Program to simulate the following input data and target output relations. The initial weight
for W1 is 0.6, the initial weight for W2 is 0.8, and the initial wight for W3 is 0.9. The learning coefficient is 0.4. The Threshold value is -1.0

```
# importing numpy to do the circulation
import numpy as np

def StepFunction(threshold):
if threshold >= -1.5:
return 1
else:
return 0

def perceptMod(x, weight, b):
threshold = np.dot(weight, x) + b
# print("Thereshold: ", threshold)
y = StepFunction(threshold)
return y
# NOT Logic Function
# weight_NOT = -1, bias_NOT = 0.5
def NOT_Funtion(x):
weight_NOT = -1
bias_NOT = 0.5return perceptMod (x, weight_NOT, bias_NOT)

#  given w1 = 0.9, w2 = 0.8, bias_OR = 0.7
def OR_Function(x):
weight = np.array([0.9, 0.8])
bias_OR = 0.7
return perceptMod(x, weight, bias_OR)
# NOR Logic Function with OR and NOT


def NOR_Functon(x):
OR_out = OR_Function(x)
NOT_output = NOT_Function(OR_out)
return NOT_output
# test
input_01 = np.array([0, 1])
input_11 = np.array([1, 1])
input_00 = np.array([0, 0])
input_10 = np.array([1, 0])
print("NOR({}, {}) = {}".format(0, 0, NOR_Function(input_00)))
print("NOR({}, {}) = {}".format(0, 1, NOR_Function(input_01)))
print("NOR({}, {}) = {}".format(1, 0, NOR_Function(input_10)))
print("NOR({}, {}) = {}".format(1, 1, NOR_Function(input_11)))
```

output



```
OR(0, 0) = 0
OR(0, 1) = 0
OR(1, 0) = 0
OR(1, 1) = 0

Process finished with exit code 0
```