Name: Prathamesh Rajbhoj

Roll no : 53

Batch : A2

Practical 06

Aim : To implement the perceptron algorithm for AND OR and NOR gate.

```
import numpy as np

def conditionFunction(g):
  if g>= 0:
    return 1
  else:
    return 0

def perceptron(x, w, b):
  g=np.dot(w, x)+b
  y=conditionFunction(g)
  return y

def AndLogic(x):
  w=np.array([1, 1])
  b=-1.5
  return perceptron(x, w, b)

def ORLogic(x):
  w=np.array([1, 1])
  b=-0.5
  return perceptron(x, w, b)

def NOTLogic(x):
    wNOT = -1
    bNOT = 0.5
    return perceptron(x, wNOT, bNOT)

def NORLogic(x):
   output_OR = ORLogic(x)
   output_NOT = NOTLogic(output_OR)
   return output_NOT


test1 = np.array([0, 1])
test2 = np.array([1, 0])
test3 = np.array([0, 0])
test4 = np.array([1, 1])
test5 = np.array([0])
test6 = np.array([1])

print("AND({}, {}) = {}".format(0, 1, AndLogic(test1)))
print("AND({}, {}) = {}".format(1, 0,  AndLogic(test2)))
print("AND({}, {}) = {}".format(0, 0,  AndLogic(test3)))
print("AND({}, {}) = {}".format(1, 1,  AndLogic(test4)))
print("=======================")
print("OR({}, {}) = {}".format(0, 1, ORLogic(test1)))
print("OR({}, {}) = {}".format(1, 0, ORLogic(test2)))
print("OR({}, {}) = {}".format(0, 0, ORLogic(test3)))
print("OR({}, {}) = {}".format(1, 1, ORLogic(test4)))
print("=======================")
print("NOR({}, {}) = {}".format(0, 1, NORLogic(test1)))
print("NOR({}, {}) = {}".format(1, 0, NORLogic(test2)))
print("NOR({}, {}) = {}".format(0, 0, NORLogic(test3)))
print("NOR({}, {}) = {}".format(1, 1, NORLogic(test4)))
print("=======================")
print("NOT({}) = {}".format(0, NOTLogic(test5)))
print("NOT({}) = {}".format(1, NOTLogic(test6)))
```

```
AND(0, 1) = 0
AND(1, 0) = 0
AND(0, 0) = 0
AND(1, 1) = 1
=======================
OR(0, 1) = 1
OR(1, 0) = 1
OR(0, 0) = 0
OR(1, 1) = 1
=======================
NOR(0, 1) = 0
NOR(1, 0) = 0
NOR(0, 0) = 1
NOR(1, 1) = 0
=========================
NOT(0) = 1
NOT(1) = 0
```