Name: Ayush Chanchal

Sap: 500097569

Batch: b6

EXPERIMENT: 08

**Describe:**

1. **AND Operation**:
   * The AND operation takes two binary inputs (0 or 1) and produces a 1 as output only if both inputs are 1. Otherwise, it produces a 0.
   * **Neural Network Interpretation**:
     + **Input Layer**: Two nodes representing the two binary inputs (x1, x2).
     + **Output Layer**: One node producing the result.
   * **Weights and Bias**:
     + Assign appropriate weights and bias such that the output node activates (produces a 1) only when both inputs are 1.
2. **NAND Operation**:
   * The NAND operation is the opposite of AND. It produces a 0 as output only if both inputs are 1. Otherwise, it produces a 1.
   * **Neural Network Interpretation**:
     + **Input Layer**: Two nodes representing the two binary inputs (x1, x2).
     + **Output Layer**: One node producing the result.
   * **Weights and Bias**:
     + Assign appropriate weights and bias such that the output node activates (produces a 1) only when both inputs are 0.
3. **OR Operation**:
   * The OR operation takes two binary inputs and produces a 1 as output if at least one of the inputs is 1.
   * **Neural Network Interpretation**:
     + **Input Layer**: Two nodes representing the two binary inputs (x1, x2).
     + **Output Layer**: One node producing the result.
   * **Weights and Bias**:
     + Assign appropriate weights and bias such that the output node activates (produces a 1) when at least one input is 1.
4. **NOR Operation**:
   * The NOR operation is the opposite of OR. It produces a 0 as output if at least one of the inputs is 1.
   * **Neural Network Interpretation**:
     + **Input Layer**: Two nodes representing the two binary inputs (x1, x2).
     + **Output Layer**: One node producing the result.
   * **Weights and Bias**:
     + Assign appropriate weights and bias such that the output node activates (produces a 1) when both inputs are 0.
5. **NOT Operation**:
   * The NOT operation takes a single binary input and produces the opposite output (1 becomes 0 and vice versa).
   * **Neural Network Interpretation**:
     + **Input Layer**: One node representing the binary input (x).
     + **Output Layer**: One node producing the result.
   * **Weights and Bias**:
     + Assign appropriate weights and bias such that the output node activates (produces a 1) when the input is 0.
6. **XOR Operation**:
   * The XOR operation is exclusive OR. It produces a 1 as output if the inputs are different (one 1 and one 0). Otherwise, it produces a 0.
   * **Note**: XOR is not linearly separable and cannot be modelled with a single-layer perceptron. It requires a multi-layer perceptron.
   * **Neural Network Interpretation**:
     + **Input Layer**: Two nodes representing the two binary inputs (x1, x2).
     + **Hidden Layer**: One or more nodes.
     + **Output Layer**: One node producing the result.
   * **Weights and Bias**:
     + Assign appropriate weights and bias in both the hidden and output layers to model the XOR operation.

IMPLEMENTATION:

Source code:

import numpy as np

import matplotlib.pyplot as plt

def function(a):

if a>=0:

return 1

else:

return 0

def NeuralNetwork(x,w,b):

a=np.dot(x,w)+b

y=function(a)

return y

def Not\_Gate(x):

w=-1

b=0.5

return NeuralNetwork(x,w,b)

def And\_Gate(x):

w = np.array([1, 1])

b= -1.5

return NeuralNetwork(x, w, b)

def Or\_Gate(x):

w = np.array([1, 1])

b= -0.5

return NeuralNetwork(x, w, b)

def Nand\_Gate(x):

a=And\_Gate(x)

b=Not\_Gate(a)

return b

def Nor\_Gate(x):

a=Or\_Gate(x)

b=Not\_Gate(a)

return b

x1=0

x2=1

print("Not {}:".format(x1),Not\_Gate(x1))

print("Not {}:".format(x2),Not\_Gate(x2))

x1=np.array([0,0])

x2=np.array([0,1])

x3=np.array([1,0])

x4=np.array([1,1])

print("Or {}:".format(x1),Or\_Gate(x1))

print("Or {}:".format(x2),Or\_Gate(x2))

print("Or {}:".format(x3),Or\_Gate(x3))

print("Or {}:".format(x4),Or\_Gate(x4))

print("And {}:".format(x1),And\_Gate(x1))

print("And {}:".format(x2),And\_Gate(x2))

print("And {}:".format(x3),And\_Gate(x3))

print("And {}:".format(x4),And\_Gate(x4))

print("Nand {}:".format(x1),Nand\_Gate(x1))

print("Nand {}:".format(x2),Nand\_Gate(x2))

print("Nand {}:".format(x3),Nand\_Gate(x3))

print("Nand {}:".format(x4),Nand\_Gate(x4))

print("Nor {}:".format(x1),Nor\_Gate(x1))

print("Nor {}:".format(x2),Nor\_Gate(x2))

print("Nor {}:".format(x3),Nor\_Gate(x3))

print("Nor {}:".format(x4),Nor\_Gate(x4))

Screenshot:

