**ST.FRANCIS INSTITUTE OF TECHNOLOGY**

Mount Poinsur, S.V.P. Road, Borivli (West), Mumbai - 400103

**Computer Engineering Department**

**Academic Year:** 2021-2022 **Class/Branch**: BE CMPN

**Subject:** CSC 703 Artificial Intelligence & Soft Computing **Semester**: VII

# Name:Rebecca Dias Class: BECMPNA2 Rollno:19 PID:182027

**Experiment No. 7**

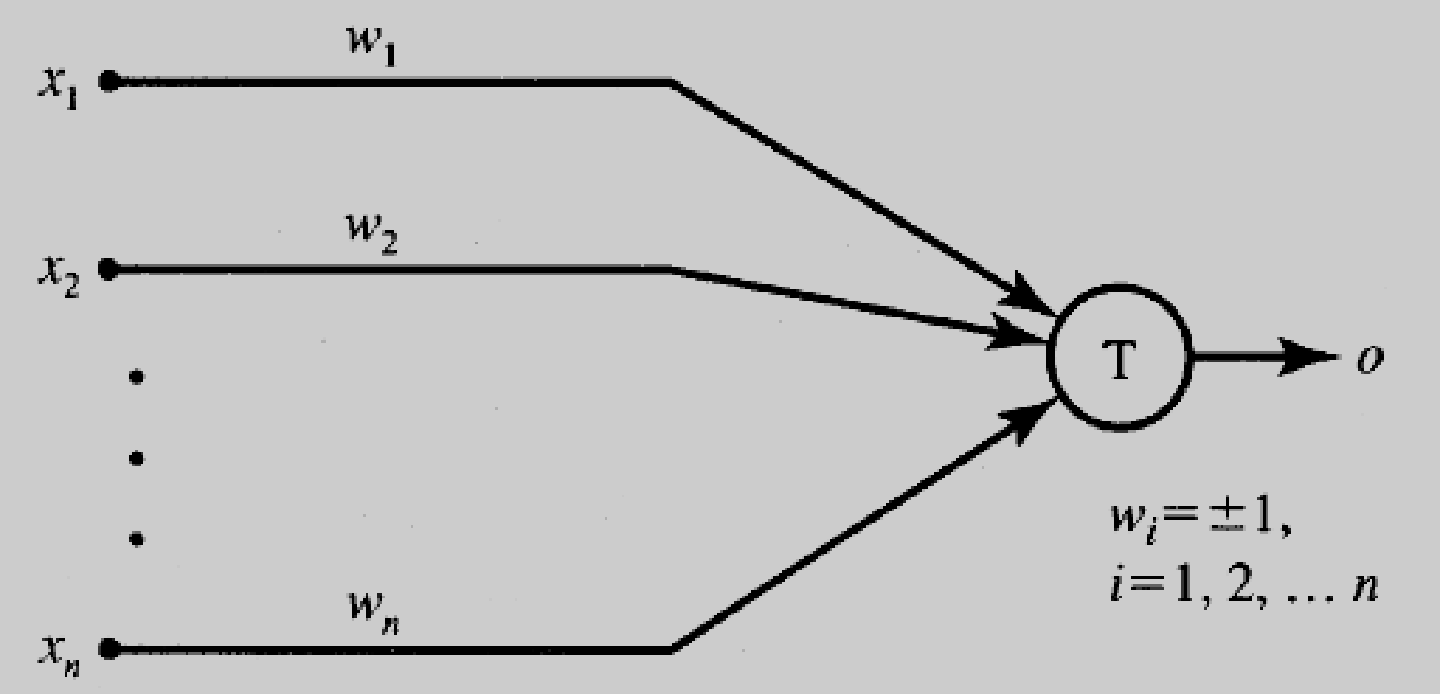
**Mcculloch Pitt’s Model**

**Aim:** To implement Mc Culloc Pitt’s Model

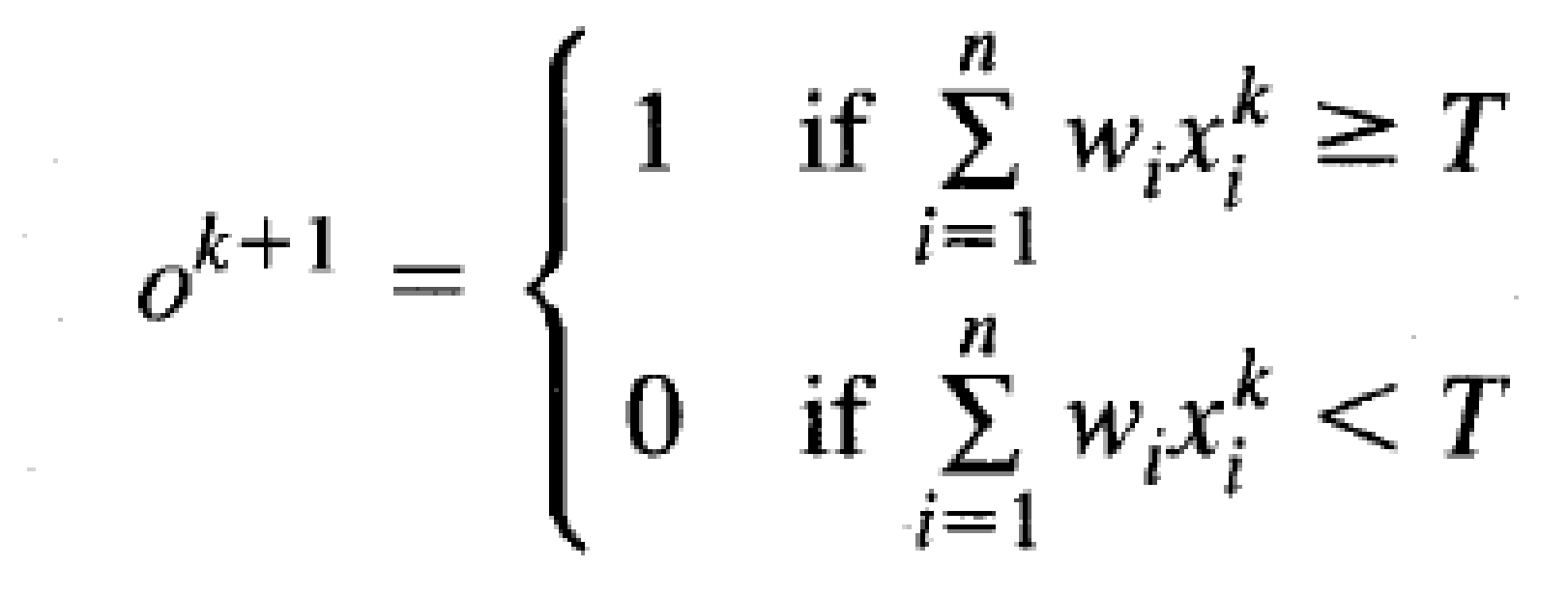
**Theory:**

**McCulloch-Pitts Neuron Model**

The first formal definition of a synthetic neuron model based on the highly simplified considerations of the biological model was formulated by McCulloch and Pitts (1943). The McCulloch-Pitts model of the neuron is as shown below:



The inputs xi, for i = 1, 2, . . . , n, are 0 or 1, depending on the absence or presence of the input impulse at instant k. The neuron's output signal is denoted as o. The firing rule for this model is defined as follows:

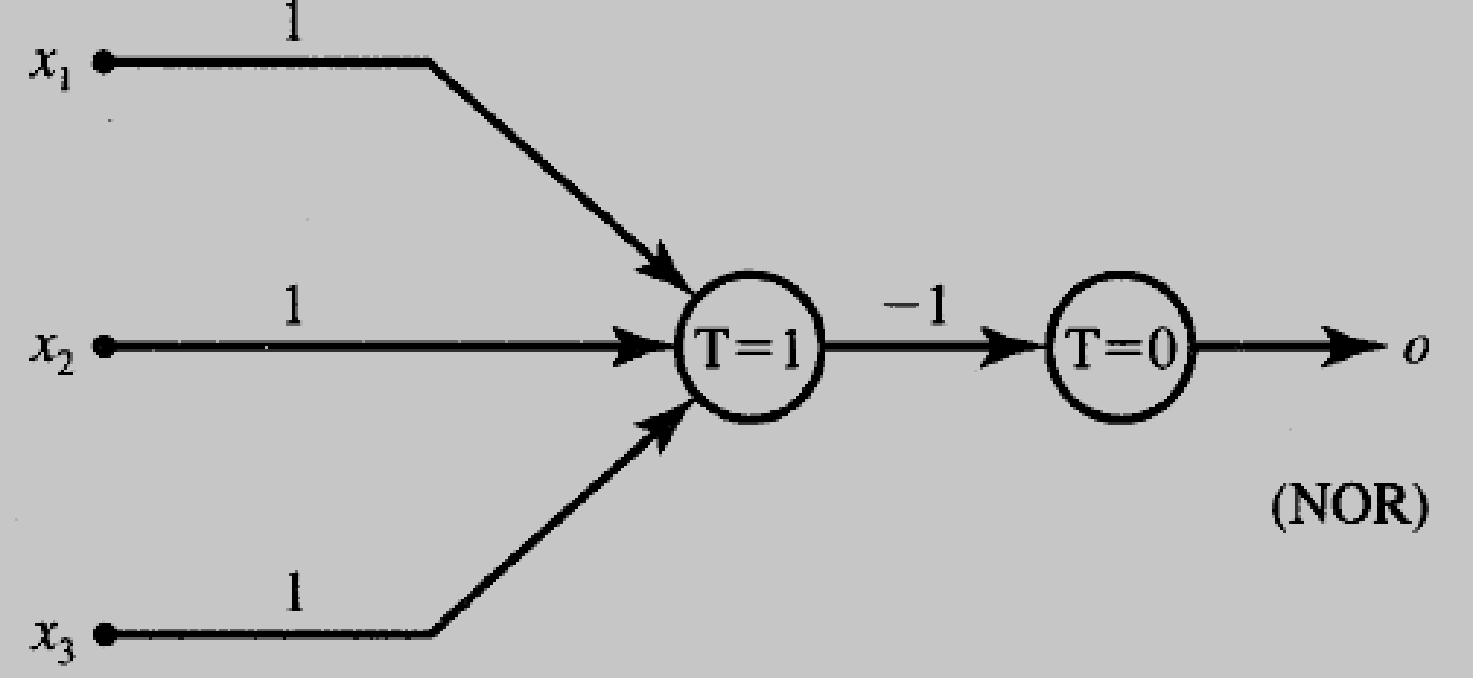


where superscript k = 0, 1, 2, . . . denotes the discrete-time 'instant, and wi is the

multiplicative weight connecting the i'th input with the neuron's membrane. We will assume that a unity delay elapses between the instants k and k + 1. Note that wi = + 1 for excitatory synapses, wi = - 1 for inhibitory synapses for this model, and T is the neuron's threshold value, which needs to be exceeded by the weighted sum of signals for the neuron to fire.

Although this neuron model is very simplistic, it has substantial computing potential. It can perform the basic logic operations NOT, OR, and AND, provided

its weights and thresholds are appropriately selected.



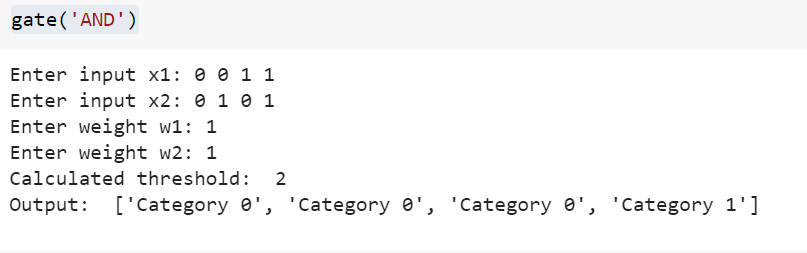
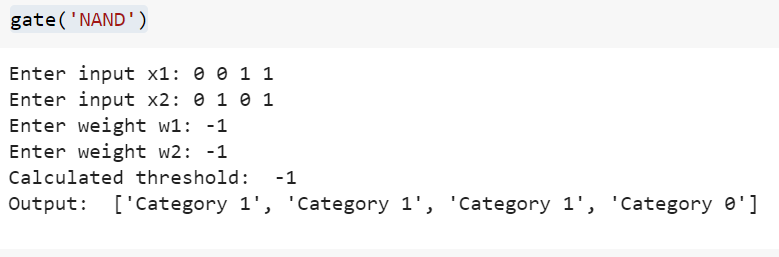
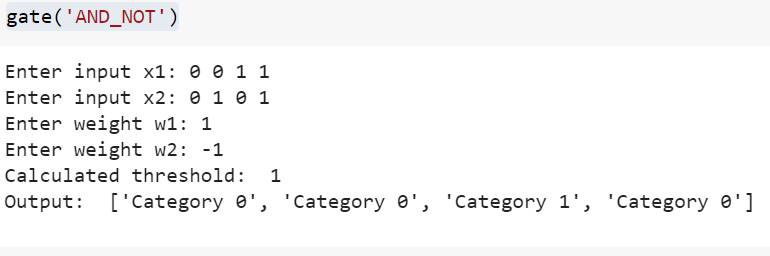
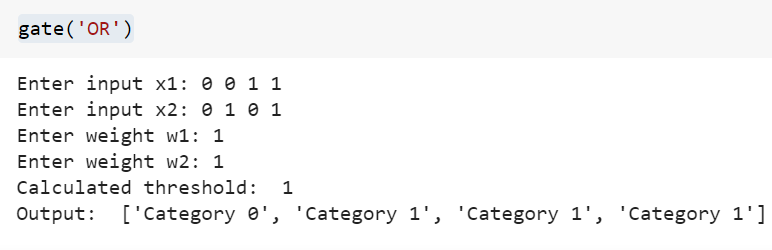
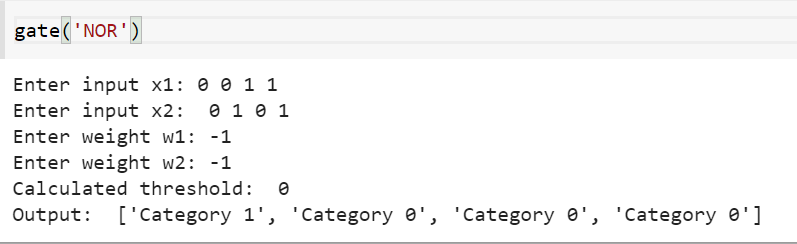
**Experiment :** The experiment is to manually find and design Mc. Culloc Pitts Model for the following logic functions.

1. AND
2. OR
3. AND NOT
4. NAND
5. NOR

Code:

|  |
| --- |
| def gate(name):      x1 = list(map(int,input("Enter input x1: ").split()))      x2 = list(map(int,input("Enter input x2: ").split()))      w1 = list(map(int,input("Enter weight w1: ").split()))      w2 = list(map(int,input("Enter weight w2: ").split()))      yin = []      for i in range(len(x1)):          net = w1[0]\*x1[i] + w2[0]\*x2[i]          yin.append(net)      m = []      if name == 'AND':          for i in range(len(x1)):              if x1[i] == 1 and x2[i] == 1:                  m = yin[i]      elif name == 'OR':          for i in range(len(x1)):              if x1[i] == 1 and x2[i] == 0:                  m=yin[i]      elif name == 'NOR':          for i in range(len(x1)):              if x1[i] == 0 and x2[i] == 0:                  m=yin[i]      elif name == 'NAND':          for i in range(len(x1)):              if x1[i] == 1 and x2[i] == 0:                  m=yin[i]      elif name == 'AND\_NOT':          if w1[0] > 0 and w2[0] < 0:              for i in range(len(x1)):                  if x1[i] == 1 and x2[i] == 0:                      m=yin[i]          elif w1[0] < 0 and w2[0] > 0:              for i in range(len(x1)):                  if x1[i] == 0 and x2[i] == 1:                      m=yin[i]      t = m      print("Calculated threshold: ",t)      out = []      for i in yin:          if i >= t:              out.append("Category 1")          else:              out.append("Category 0")      print("Output: ", out)  gate('AND')  gate('NAND')  gate('AND\_NOT')  gate('OR')  gate('NOR') |

Output:

**Post Experiment**

1. Identify any GATE logic function that cannot be implemented by a single layer linear model.

Ans: A "single-layer" linear model can't implement XOR. XOR is where if one is 1 and other is 0 but not both. The reason is because the classes in XOR are not linearly separable. You cannot draw a straight line to separate the points (0,0),(1,1) from the points (0,1),(1,0). Led to invention of multi-layer networks.

**Conclusion:** Implement the McCulloch Pitt’s Model for various GATE Logic Functions. Understood the difference between Linearly separable and Non separable categories. Output was shown in Python.