Assignment 2

Name: : Chukka Chanakya Devendra

Roll Number: : AP21110011577

Submission Date: : 19-08-2024

Question 1

Write a Python program to implement XOR logical gates using M-P neuron.

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

def step_function(x, threshold):
    return 1 if x >= threshold else 0

class MPNeuron:
    def __init__(self, weights, threshold):
        self.weights = np.array(weights)
        self.threshold = threshold

    def predict(self, inputs):
        weighted_sum = np.dot(inputs, self.weights)
        return step_function(weighted_sum, self.threshold)

class XORModel:
    def __init__(self):
        self.and_neuron = MPNeuron(weights=[1, 1], threshold=2)
        self.or_neuron = MPNeuron(weights=[1, 1], threshold=1)
        self.nand_neuron = MPNeuron(weights=[-1, -1], threshold=-1)

    def predict(self, x1, x2):
        or_output = self.or_neuron.predict([x1, x2])
        nand_output = self.nand_neuron.predict([x1, x2])
        xor_output = self.and_neuron.predict([nand_output, or_output])
        return xor_output
xor_model = XORModel()

inputs = [(0, 0), (0, 1), (1, 0), (1, 1)]

print("XOR Gate Output:")
for x1, x2 in inputs:
    print(f"Input: ({x1}, {x2}) -> Output: {xor_model.predict(x1, x2)}")
```

```
XOR Gate Output:
Input: (0, 0) -> Output: 0
Input: (0, 1) -> Output: 1
Input: (1, 0) -> Output: 1
Input: (1, 1) -> Output: 0
```

Question 2

Write a Python program to implement AND, OR logical gates using Hebb Network.

In [ ]:
```python
INPUTS = np.array([[1, 1], [1, -1], [-1, 1], [-1, -1]])
LEARNING_RATE = 0.1

def step_function(sum):
    if sum >= 0:
        return 1
    return -1

def calculate_output(weights, instance, bias):
    sum = instance.dot(weights) + bias
    return step_function(sum)
def hebb(outputs, weights, bias):
    for i in range(4):
        weights[0] = weights[0] + (INPUTS[i][0] * outputs[i])
        weights[1] = weights[1] + (INPUTS[i][1] * outputs[i])
        bias = bias + (1 * outputs[i])

        print("Weight updated: " + str(weights[0]))
        print("Weight updated: " + str(weights[1]))
        print("Bias updated: " + str(bias))
        print("----------------------------------------")

    return weights, bias
and_outputs = np.array([1, -1, -1, -1])
```

```python
or_outputs = np.array([1, 1, 1, -1])
weights = np.array([0.0, 0.0])
bias = 0
returned_weights, returned_bias = hebb(or_outputs, weights, bias)
print('prediction for [1, 1]: ' + str(calculate_output(returned_weights, np.array([[1, 1]]), returned_bias)))
print('prediction for [1, -1]: ' + str(calculate_output(returned_weights, np.array([[1, -1]]), returned_bias)))
print('prediction for [-1, 1]: ' + str(calculate_output(returned_weights, np.array([[-1, 1]]), returned_bias)))
print('prediction for [-1, -1]: ' + str(calculate_output(returned_weights, np.array([[-1, -1]]), returned_bias)))
print("----------------------------------------")
print("----------------------------------------")
returned_weights, returned_bias = hebb(and_outputs, weights, bias)
print('prediction for [1, 1]: ' + str(calculate_output(returned_weights, np.array([[1, 1]]), returned_bias)))
print('prediction for [1, -1]: ' + str(calculate_output(returned_weights, np.array([[1, -1]]), returned_bias)))
print('prediction for [-1, 1]: ' + str(calculate_output(returned_weights, np.array([[-1, 1]]), returned_bias)))
print('prediction for [-1, -1]: ' + str(calculate_output(returned_weights, np.array([[-1, -1]]), returned_bias)))
```

```
Weight updated: 1.0
Weight updated: 1.0
Bias updated: 1
----------------------------------------
Weight updated: 2.0
Weight updated: 0.0
Bias updated: 2
----------------------------------------
Weight updated: 1.0
Weight updated: 1.0
Bias updated: 3
----------------------------------------
Weight updated: 2.0
Weight updated: 2.0
Bias updated: 2
----------------------------------------
prediction for [1, 1]: 1
prediction for [1, -1]: 1
prediction for [-1, 1]: 1
prediction for [-1, -1]: -1
----------------------------------------
----------------------------------------
Weight updated: 3.0
Weight updated: 3.0
Bias updated: 1
----------------------------------------
Weight updated: 2.0
Weight updated: 4.0
Bias updated: 0
----------------------------------------
Weight updated: 3.0
Weight updated: 3.0
Bias updated: -1
----------------------------------------
Weight updated: 4.0
Weight updated: 4.0
Bias updated: -2
----------------------------------------
prediction for [1, 1]: 1
prediction for [1, -1]: -1
prediction for [-1, 1]: -1
prediction for [-1, -1]: -1
```