Name:- Medhansh K PRN:- 122A8035 B3/TE:- AIDS

EXPERIMENT NO 8

**AIM:** Implement HEBBIAN network for AND function

**Theory:** The **Hebb Learning Rule** updates weights based on the correlation between inputs and outputs. For an **AND function**, weights are initialized to zero and updated as **w(new) = w(old) + x \* y** for each training example.

Code: **import** numpy **as** np

In [5]:

1

X = np.array([[1, 1], [1, **-**1], [**-**1, 1], [**-**1, **-**1]])

2

y = np.array([1, **-**1, **-**1, **-**1])

In [6]:

1

weights = np.zeros(X.shape[1]) 2

bias = 0

In [7]:

1

**for** i **in** range(len(X)): 2

weights += X[i] **\*** y[i]

bias += y[i]

3

In [9]:

1

print(f"Trained Weights: {weights}") 2

print(f"Trained Bias: {bias}")

Trained Weights: [ 2. 2.]

Trained Bias: -2

In [10]:

1

**def** test\_hebb\_net(X, weights, bias):

2

**for** x **in** X:

3

output = np.dot(x, weights) **+** bias

4

print(f"Input: {x} -> Output: {1 if output >= 0 else -1}")

In [11]:

1

test\_hebb\_net(X, weights, bias)

Input: [1 1] -> Output: 1

Input: [ 1 -1] -> Output: -1

Input: [-1 1] -> Output: -1

Input: [-1 -1] -> Output: -1

**Conclusion:**

The Hebbian learning rule allows the network to model the AND function by adjusting weights based on the input-output correlation. After training, the network learns the correct weights and can correctly classify the AND function. This simple learning rule forms the basis for more advanced learning algorithms in neural networks.