## Experiment 6: Implement Hebb's Rule for Basic Logical Functions

**Aim:**

To implement Hebb's Learning Rule for simulating basic logical functions such as AND and OR using Python.

**Apparatus Required:**

- Python (Anaconda/Miniconda/Standard Python Installation)
- Jupyter Notebook or any Python IDE
- NumPy library

**Theory:**

Hebb's Rule is one of the earliest learning algorithms proposed by Donald Hebb. The rule can be summarized as:

"Neurons that fire together, wire together."

In artificial neural networks, this rule is used to update weights based on the input and output. The weight update formula in Hebbian learning is:

$$w_i(New) = w_i(Old) + \Delta w_i$$

$$where, \qquad \Delta w_i = \eta * x_i * y$$

Where:

- $w_i$ is the weight vector
- $x_i$ is the input vector
- y is the target output
- $\eta$ is the learning rate

This rule is particularly suited for linearly separable functions like AND and OR.

**Code:**

```
import numpy as np
# Hebbian learning function
def hebbian_learning(X, y, eta=1):
    n_samples, n_features = X.shape
    weights = np.zeros(n_features)   # no bias term, just input weights
   print("=== Training Phase ===")
    for i in range(n_samples):
        x = X[i]
        target = y[i]
        delta_w = eta * x * target
        weights += delta_w
        print(f"Input: {x}, Target: {target}, Δw: {delta_w}, Updated Weights: {weights}")
    return weights
# Prediction function (with threshold)
def predict(x, weights, threshold):
    activation = np.dot(weights, x)
    return 1 if activation >= threshold else 0
# All input combinations for 2-input gate
X = np.array([
    [0,0],
    [0,1],
    [1,0],
    [1,1]
])
# ----------------------
# AND Gate
# ----------------------
print("\n=== AND Gate Training ===")
y_and = np.array([0,0,0,1])
weights_and = hebbian_learning(X, y_and)
print("\n=== AND Gate Testing ===")
for x in X:
    out = predict(x, weights_and, threshold=1)  # threshold = 1
    print(f"Input: {x.tolist()} -> Output: {out}")
# ----------------------
# OR Gate
# ----------------------
print("\n=== OR Gate Training ===")
y_or = np.array([0,1,1,1])
weights_or = hebbian_learning(X, y_or)
print("\n=== OR Gate Testing ===")
for x in X:
    out = predict(x, weights_or, threshold=2)  # threshold = 2 (like in example)
    print(f"Input: {x.tolist()} -> Output: {out}")
```

**Result:**

The weights were successfully learned using Hebb's Rule. The perceptron was able to correctly simulate the logical AND and OR functions.