Pratham Nagar 59 ML_EXP_5

```
import numpy as np
import matplotlib.pyplot as plt


# Parameters
learning_rate = 0.05
epochs = 200


# Input Data (e.g., OR Gate)
X = np.array([
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
])

# Target Output
y = np.array([0, 1, 1, 1])


# Initialize Weights and Bias
weights = np.random.rand(2)
bias = np.random.rand()


# Activation Function
def activation_function(x):
    return 1 if x >= 0 else 0


# Training Loop
for epoch in range(epochs):
    for i in range(len(X)):
        input_vector = X[i]
        target = y[i]

        # Weighted Sum
        weighted_sum = np.dot(weights, input_vector) + bias

        # Output
        output = activation_function(weighted_sum)

        # Hebbian Learning Rule
        delta_w = learning_rate * input_vector * output
        delta_b = learning_rate * output

        # Update Weights and Bias
        weights += delta_w
        bias += delta_b


# Results
print("Trained Weights:", weights)
print("Trained Bias:", bias)
```

```
Trained Weights: [20.57022354 20.42259191]
Trained Bias: 40.355677298982805
```

```
# Testing
for i in range(len(X)):
    input_vector = X[i]
    weighted_sum = np.dot(weights, input_vector) + bias
    output = activation_function(weighted_sum)
    print(f"Input: {input_vector}, Predicted Output: {output}")
```

```
Input: [0 0], Predicted Output: 1
Input: [0 1], Predicted Output: 1
Input: [1 0], Predicted Output: 1
Input: [1 1], Predicted Output: 1
```