

CODE:-

```
import numpy as np
```

```
def bipolar(x):
```

```
    return np.where(x == 0, -1, 1)
```

```
def predict(weights, X):
```

```
    s = np.dot(X, weights)
```

```
    return np.where(s >= 0, 1, -1)
```

```
def train_hebb_verbose(X, y, lr=0.1, epochs=5):
```

```
    w = np.zeros(X.shape[1])
```

```
    for epoch in range(1, epochs+1):
```

```
        print(f"\nEpoch {epoch}")
```

```
        for xi, yi in zip(X, y):
```

```
            w += lr * yi * xi
```

```
            print(f"Input: {xi}, Target: {yi}, Updated Weights: {w}")
```

```
        preds = predict(w, X)
```

```
        preds01 = np.where(preds == -1, 0, 1)
```

```
        print(f"Predictions after epoch {epoch}: {preds01.tolist()}")
```

```
    return w
```

```
X = np.array([
```

```
    [1, 0, 0],
```

```
    [1, 0, 1],
```

```
    [1, 1, 0],
```

```
    [1, 1, 1]
```

```
])
```

```
choice = input("Enter gate (AND/OR): ").strip().upper()
```

```
if choice == "AND":
```

```
    targets = np.array([0, 0, 0, 1])
```

```
elif choice == "OR":
```

```
    targets = np.array([0, 1, 1, 1])
```

```
else:
```

```
    print("Invalid choice! Please enter AND or OR.")
```

```
    exit()
```

```
y = bipolar(targets)
```

```
final_w = train_hebb_verbose(X, y, lr=0.2, epochs=5)
```

```
print("\nFinal Weights:", final_w)
```

OUTPUT:-

Enter gate (AND/OR): AND

Epoch 1

Input: [1 0 0], Target: -1, Updated Weights: [-0.2 0. 0.]
Input: [1 0 1], Target: -1, Updated Weights: [-0.4 0. -0.2]
Input: [1 1 0], Target: -1, Updated Weights: [-0.6 -0.2 -0.2]
Input: [1 1 1], Target: 1, Updated Weights: [-0.4 0. 0.]
Predictions after epoch 1: [0, 0, 0, 0]

Epoch 2

Input: [1 0 0], Target: -1, Updated Weights: [-0.6 0. 0.]
Input: [1 0 1], Target: -1, Updated Weights: [-0.8 0. -0.2]
Input: [1 1 0], Target: -1, Updated Weights: [-1. -0.2 -0.2]
Input: [1 1 1], Target: 1, Updated Weights: [-0.8 0. 0.]
Predictions after epoch 2: [0, 0, 0, 0]

Epoch 3

Input: [1 0 0], Target: -1, Updated Weights: [-1. 0. 0.]
Input: [1 0 1], Target: -1, Updated Weights: [-1.2 0. -0.2]
Input: [1 1 0], Target: -1, Updated Weights: [-1.4 -0.2 -0.2]
Input: [1 1 1], Target: 1, Updated Weights: [-1.2 0. 0.]
Predictions after epoch 3: [0, 0, 0, 0]

Epoch 4

Input: [1 0 0], Target: -1, Updated Weights: [-1.4 0. 0.]
Input: [1 0 1], Target: -1, Updated Weights: [-1.6 0. -0.2]
Input: [1 1 0], Target: -1, Updated Weights: [-1.8 -0.2 -0.2]
Input: [1 1 1], Target: 1, Updated Weights: [-1.6 0. 0.]
Predictions after epoch 4: [0, 0, 0, 0]

Epoch 5

Input: [1 0 0], Target: -1, Updated Weights: [-1.8 0. 0.]
Input: [1 0 1], Target: -1, Updated Weights: [-2. 0. -0.2]
Input: [1 1 0], Target: -1, Updated Weights: [-2.2 -0.2 -0.2]
Input: [1 1 1], Target: 1, Updated Weights: [-2. 0. 0.]
Predictions after epoch 5: [0, 0, 0, 0]

Final Weights: [-2. 0. 0.]