2282437

Zaid Shariff

Lab - 1

Binary Activation

In [1]:

```
import random

# Define the training data (OR gate inputs and corresponding outputs)
training_data = [((0, 0), 0), ((0, 1), 1), ((1, 0), 1), ((1, 1), 1)]

# Initialize weights and bias with random values
weight1 = random.uniform(-1, 1)
weight2 = random.uniform(-1, 1)
bias = random.uniform(-1, 1)

# Learning rate
learning_rate = 0.1

# Training the perceptron
epochs = 10000
for epoch in range(epochs):
    for inputs, target in training_data:
        # Calculate the weighted sum of inputs and bias
        weighted_sum = inputs[0] * weight1 + inputs[1] * weight2 + bias
        # Calculate the perceptron's output
        output = 1 if weighted_sum >= 0 else 0
        # Update weights and bias based on the error
        weight1 += learning_rate * (target - output) * inputs[0]
        weight2 += learning_rate * (target - output) * inputs[1]
        bias += learning_rate * (target - output)
```

```python
# Test the trained perceptron
print("Trained Weights and Bias:")
print("Weight 1:", weight1)
print("Weight 2:", weight2)
print("Bias:", bias)

# Test the perceptron with OR gate inputs
input_pairs = [(0, 0), (0, 1), (1, 0), (1, 1)]

print("Perceptron Output:")
for input_pair in input_pairs:
    weighted_sum = input_pair[0] * weight1 + input_pair[1] * weight2 + bias
    output = 1 if weighted_sum >= 0 else 0
    print(f"Input: {input_pair} -> Output: {output}")
```

```
Trained Weights and Bias:
Weight 1: 0.4179742437011622
Weight 2: 0.8183709821137558
Bias: -0.3216082608782438
Perceptron Output:
Input: (0, 0) -> Output: 0
Input: (0, 1) -> Output: 1
Input: (1, 0) -> Output: 1
Input: (1, 1) -> Output: 1
```

Bipolar Activation

In [2]:

```python
import random

# Define the training data (OR gate inputs and corresponding outputs)
training_data = [((0, 0), -1), ((0, 1), 1), ((1, 0), 1), ((1, 1), 1)]

# Initialize weights and bias with random values
weight1 = random.uniform(-1, 1)
weight2 = random.uniform(-1, 1)
bias = random.uniform(-1, 1)

# Learning rate
learning_rate = 0.1

# Training the perceptron
epochs = 10000
```

```python
for epoch in range(epochs):
    for inputs, target in training_data:
        # Calculate the weighted sum of inputs and bias
        weighted_sum = inputs[0] * weight1 + inputs[1] * weight2 + bias
        # Calculate the perceptron's output
        output = 1 if weighted_sum >= 0 else -1
        # Update weights and bias based on the error
        weight1 += learning_rate * (target - output) * inputs[0]
        weight2 += learning_rate * (target - output) * inputs[1]
        bias += learning_rate * (target - output)

# Test the trained perceptron
print("Trained Weights and Bias:")
print("Weight 1:", weight1)
print("Weight 2:", weight2)
print("Bias:", bias)

# Test the perceptron with OR gate inputs
input_pairs = [(0, 0), (0, 1), (1, 0), (1, 1)]

print("Perceptron Output:")
for input_pair in input_pairs:
    weighted_sum = input_pair[0] * weight1 + input_pair[1] * weight2 + bias
    output = 1 if weighted_sum >= 0 else -1
    print(f"Input: {input_pair} -> Output: {output}")
```

```
Trained Weights and Bias:
Weight 1: 0.24532198690528562
Weight 2: 0.49373943167720724
Bias: -0.12128036692236388
Perceptron Output:
Input: (0, 0) -> Output: -1
Input: (0, 1) -> Output: 1
Input: (1, 0) -> Output: 1
Input: (1, 1) -> Output: 1
```