Lab 1

Daniel Mehta

In [2]:

```
import numpy as np
```

AND Implementation

In [4]:

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

y = np.array([0, 0, 0, 1])  # AND outputs
```

In [5]:

```
weights = np.zeros(2)
bias = 0

learning_rate = 0.1
iterations = 1000
```

In [6]:

```
def activation(x):
    return 1 if x >= 0 else 0
```

In [7]:

```
for epoch in range(iterations):
    for i in range(len(X)):

        weighted_sum = np.dot(X[i], weights) + bias
        prediction = activation(weighted_sum)

        error = y[i] - prediction
        weights += learning_rate * error * X[i]
        bias += learning_rate * error

    if epoch % 100 == 0:
        print(f"Epoch {epoch+1}: Weights: {weights}, Bias: {bias}")
```

```
Epoch 1: Weights: [0.1 0.1], Bias: 0.0
Epoch 101: Weights: [0.2 0.1], Bias: -0.20000000000000004
Epoch 201: Weights: [0.2 0.1], Bias: -0.20000000000000004
Epoch 301: Weights: [0.2 0.1], Bias: -0.20000000000000004
Epoch 401: Weights: [0.2 0.1], Bias: -0.20000000000000004
Epoch 501: Weights: [0.2 0.1], Bias: -0.20000000000000004
Epoch 601: Weights: [0.2 0.1], Bias: -0.20000000000000004
Epoch 701: Weights: [0.2 0.1], Bias: -0.20000000000000004
Epoch 801: Weights: [0.2 0.1], Bias: -0.20000000000000004
Epoch 901: Weights: [0.2 0.1], Bias: -0.20000000000000004
```

In [8]:

```
print(f"Final Weights: {weights}")
print(f"Final Bias: {bias}")
```

```
Final Weights: [0.2 0.1]
Final Bias: -0.20000000000000004
```

In [9]:

```
def predict(inputs):
    weighted_sum = np.dot(inputs, weights) + bias
    return activation(weighted_sum)
for i in range(len(X)):
    print(f"Input: {X[i]} => Predicted Output: {predict(X[i])}, Actual Output: {y[i]}")
```

```
Input: [0 0] => Predicted Output: 0, Actual Output: 0
Input: [0 1] => Predicted Output: 0, Actual Output: 0
Input: [1 0] => Predicted Output: 0, Actual Output: 0
Input: [1 1] => Predicted Output: 1, Actual Output: 1
```

OR Implementation

In [11]:

```
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])
y = np.array([0, 1, 1, 1]) # Changed for OR
```

In [12]:

```
weights = np.zeros(2)
bias = 0

learning_rate = 0.1
iterations = 1000
```

In [13]:

```
def activation(x):
    return 1 if x >= 0 else 0
```

In [14]:

```
for epoch in range(iterations):
    for i in range(len(X)):

        weighted_sum = np.dot(X[i], weights) + bias
        prediction = activation(weighted_sum)

        error = y[i] - prediction
        weights += learning_rate * error * X[i]
        bias += learning_rate * error

    if epoch % 100 == 0:
        print(f"Epoch {epoch+1}: Weights: {weights}, Bias: {bias}")
```

```
Epoch 1: Weights: [0.  0.1], Bias: 0.0
Epoch 101: Weights: [0.1 0.1], Bias: -0.1
Epoch 201: Weights: [0.1 0.1], Bias: -0.1
Epoch 301: Weights: [0.1 0.1], Bias: -0.1
Epoch 401: Weights: [0.1 0.1], Bias: -0.1
Epoch 501: Weights: [0.1 0.1], Bias: -0.1
Epoch 601: Weights: [0.1 0.1], Bias: -0.1
Epoch 701: Weights: [0.1 0.1], Bias: -0.1
Epoch 801: Weights: [0.1 0.1], Bias: -0.1
Epoch 901: Weights: [0.1 0.1], Bias: -0.1
```

In [15]:

```
print(f"Final Weights: {weights}")
print(f"Final Bias: {bias}")
```

```
Final Weights: [0.1 0.1]
Final Bias: -0.1
```

In [16]:

```
def predict(inputs):
    weighted_sum = np.dot(inputs, weights) + bias
    return activation(weighted_sum)
for i in range(len(X)):
    print(f"Input: {X[i]} => Predicted Output: {predict(X[i])}, Actual Output: {y[i]}")
```

```
Input: [0 0] => Predicted Output: 0, Actual Output: 0
Input: [0 1] => Predicted Output: 1, Actual Output: 1
Input: [1 0] => Predicted Output: 1, Actual Output: 1
Input: [1 1] => Predicted Output: 1, Actual Output: 1
```

In [ ]: