In [1]:
```python
import numpy as np

# Sigmoid activation function and its derivative
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return sigmoid(x) * (1 - sigmoid(x))

# Initializing weights and bias
np.random.seed(42)  # for reproducibility
weights = np.random.rand(2)  # weights for two inputs
bias = np.random.rand(1)  # one bias

# Dataset
inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
targets = np.array([0, 1, 1, 1])  # Adjusted to correct OR problem representation

# Learning rate
learning_rate = 0.1  # Increased for faster convergence

# Training loop
for epoch in range(10000):  # Reduced number of epochs
    total_error = 0
    indices = np.arange(len(inputs))
    np.random.shuffle(indices)

    for i in indices:
        input_layer = inputs[i]
        target = targets[i]

        z = np.dot(input_layer, weights) + bias
        output = sigmoid(z)
        error = 0.5 * (target - output) ** 2
        total_error += error

        dE_dy = output - target
        dy_dz = sigmoid_derivative(z)
        dz_dw = input_layer
        dz_db = 1

        gradient_weights = dE_dy * dy_dz * dz_dw
        gradient_bias = dE_dy * dy_dz * dz_db

        weights -= learning_rate * gradient_weights
        bias -= learning_rate * gradient_bias

    if epoch % 1000 == 0:
        print(f"Epoch {epoch}, Average Loss: {total_error / len(inputs)}")

print("Final weights:", weights)
print("Final bias:", bias)

for i in range(len(inputs)):
    z = np.dot(inputs[i], weights) + bias
    output = sigmoid(z)
    print(f"Input: {inputs[i]}, Predicted Output: {output[0]:.4f}, Actual Target: {targets[i]}")
```

```
Epoch 0, Average Loss: [0.06961777]
Epoch 1000, Average Loss: [0.00885499]
Epoch 2000, Average Loss: [0.0040677]
Epoch 3000, Average Loss: [0.00256532]
Epoch 4000, Average Loss: [0.00185353]
Epoch 5000, Average Loss: [0.00144339]
Epoch 6000, Average Loss: [0.00117835]
Epoch 7000, Average Loss: [0.00099367]
Epoch 8000, Average Loss: [0.00085793]
Epoch 9000, Average Loss: [0.00075414]
Final weights: [6.18145689 6.18155127]
Final bias: [-2.84461123]
Input: [0 0], Predicted Output: 0.0550, Actual Target: 0
Input: [0 1], Predicted Output: 0.9657, Actual Target: 1
Input: [1 0], Predicted Output: 0.9657, Actual Target: 1
Input: [1 1], Predicted Output: 0.9999, Actual Target: 1
```

What is neural network? A neural network is a computational model inspired by the way biological neural networks in the human brain process information. It consists of interconnected nodes, or neurons, organized in layers. Here are the key components:

Neurons (Nodes): Basic units that receive input, process it, and pass the output to the next layer. Layers: Input Layer: Receives the initial data. Hidden Layers: Intermediate layers that process inputs from the input layer. Output Layer: Produces the final output. Weights: Parameters that determine the strength of the connection between neurons. Biases: Additional parameters that adjust the output along with the weighted sum of inputs. Activation Functions: Functions applied to the input of a neuron to introduce non-linearity (e.g., sigmoid, ReLU). Neural networks are used for various tasks such as classification, regression, and pattern recognition. They are the foundation of deep learning, where networks with many layers (deep neural networks) are used to model complex patterns in data.