pratical-6-and-8-back-and-feed

May 9, 2024

[3]:
```python
import numpy as np

# Define the parameters of the network
input_neuron = 2    # Number of input neurons
hidden_neuron = 4   # Number of hidden neurons
output_neuron = 1   # Number of output neurons
learning_rate = 0.1
epochs = 10000

# Define the training data
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
Y = np.array([[0], [1], [1], [0]])

# Initialize the weights with random values
W1 = np.random.randn(input_neuron, hidden_neuron) * 0.01
W2 = np.random.randn(hidden_neuron, output_neuron) * 0.01

# Initialize the biases with random values
b1 = np.random.randn(1, hidden_neuron) * 0.01
b2 = np.random.randn(1, output_neuron) * 0.01

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

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

# Train the network using backpropagation
for i in range(epochs):
    # Forward pass
    hidden_layer_input = np.dot(X, W1) + b1
    hidden_layer_output = sigmoid(hidden_layer_input)
    output_layer_input = np.dot(hidden_layer_output, W2) + b2
    output_layer_output = sigmoid(output_layer_input)

    # Backward pass
```

```
    output_error = Y - output_layer_output
    output_delta = output_error * sigmoid_derivative(output_layer_output)

    hidden_error = output_delta.dot(W2.T)
    hidden_delta = hidden_error * sigmoid_derivative(hidden_layer_output)

    # Update weights and biases
    W2 += np.dot(hidden_layer_output.T, output_delta) * learning_rate
    b2 += np.sum(output_delta, axis=0, keepdims=True) * learning_rate
    W1 += np.dot(X.T, hidden_delta) * learning_rate
    b1 += np.sum(hidden_delta, axis=0, keepdims=True) * learning_rate

# Test the network with some example inputs
x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_test = np.array([[0], [1], [1], [0]])

hidden_layer_input = np.dot(x_test, W1) + b1
hidden_layer_output = sigmoid(hidden_layer_input)
output_layer_input = np.dot(hidden_layer_output, W2) + b2
output_layer_output = sigmoid(output_layer_input)

print("Input:")
print(x_test)
print("Output:")
print(output_layer_output)
print("Expected Output:")
print(y_test)
```

```
Input:
[[0 0]
 [0 1]
 [1 0]
 [1 1]]
Output:
[[0.50000008]
 [0.50000146]
 [0.49999851]
 [0.49999989]]
Expected Output:
[[0]
 [1]
 [1]
 [0]]
```

[ ]: