Name: akash
Adarkar Roll
no 68

DL Experiment 03

Back Propagation in Deep Learning

Code:

```
import numpy as np


class NeuralNetwork:

    def __init__(self, input_size, hidden_size,
        output_size):self.input_size = input_size
        self.hidden_size =
        hidden_size
        self.output_size =
        output_size


        # Initialize weights and biases for the hidden layer and output
        layerself.W1 = np.random.randn(hidden_size, input_size)
        self.b1 = np.zeros((hidden_size, 1))

        self.W2 = np.random.randn(output_size, hidden_size)
        self.b2 = np.zeros((output_size, 1))

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))


    def
        sigmoid_derivative(sel
        f, x):return x * (1 -
        x)

    def forward(self, X):
```

```
    self.z1 = np.dot(self.W1, X) + self.b1
    self.a1 = self.sigmoid(self.z1)
    self.z2 = np.dot(self.W2, self.a1) +
    self.b2self.a2 = self.sigmoid(self.z2)
    return self.a2


def backward(self, X, y,
    learning_rate):m = X.shape[1]


     # Compute the
     gradientsdZ2 =
     self.a2 - y
     dW2 = (1 / m) * np.dot(dZ2, self.a1.T)
     db2 = (1 / m) * np.sum(dZ2, axis=1, keepdims=True)

     dZ1 = np.dot(self.W2.T, dZ2) *
     self.sigmoid_derivative(self.a1)dW1 = (1 / m) *
     np.dot(dZ1, X.T)
     db1 = (1 / m) * np.sum(dZ1, axis=1, keepdims=True)


     # Update weights and biases using gradients and
     learning rateself.W2 -= learning_rate * dW2
     self.b2              -=
     learning_rate   *   db2
     self.W1              -=
     learning_rate   *   dW1
     self.b1              -=
     learning_rate * db1


 def train(self, X, y, epochs,
     learning_rate):for epoch in
     range(epochs):
         # Forward pass
         predictions = self.forward(X)
```

```
            # Compute the mean squared
            error loss loss =
            np.mean((predictions - y) ** 2)

            # Backward pass to update weights and biases
            self.backward(X, y, learning_rate)

            if epoch % 100 == 0:
                print(f"Epoch {epoch}, Loss: {loss:.4f}")

    def predict(self,
        X): return
        self.forward(X
        )

#
Example
usage:
input_si
ze = 2
hidden_size = 4
output_size = 1
learning_rate = 0.1
epochs = 10000

# Generate some sample data
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]]).T
y = np.array([[0, 1, 1, 0]])

# Create the neural network
nn = NeuralNetwork(input_size, hidden_size, output_size)

# Train the neural network
nn.train(X, y, epochs,
```

```
learning_rate)


# Make predictions
predictions =
nn.predict(X)
print("Predictions:", predictions)
```

Output:

```
Epoch 0, Loss: 0.3420
Epoch 100, Loss: 0.2618
Epoch 200, Loss: 0.2558
Epoch 300, Loss: 0.2512
Epoch 400, Loss: 0.2475
Epoch 500, Loss: 0.2443
Epoch 600, Loss: 0.2411
Epoch 700, Loss: 0.2377
Epoch 800, Loss: 0.2337
Epoch 900, Loss: 0.2288
Epoch 1000, Loss: 0.2228
Epoch 1100, Loss: 0.2156
Epoch 1200, Loss: 0.2067
Epoch 1300, Loss: 0.1963
Epoch 1400, Loss: 0.1839
Epoch 1500, Loss: 0.1698
Epoch 1600, Loss: 0.1538
Epoch 1700, Loss: 0.1362
Epoch 1800, Loss: 0.1177
Epoch 1900, Loss: 0.0991
Epoch 2000, Loss: 0.0816
Epoch 2100, Loss: 0.0659
Epoch 2200, Loss: 0.0526
Epoch 2300, Loss: 0.0418
Epoch 2400, Loss: 0.0333

Epoch 2500, Loss: 0.0267
Epoch 2600, Loss: 0.0216
Epoch 2700, Loss: 0.0176
Epoch 2800, Loss: 0.0145
Epoch 2900, Loss: 0.0121
Epoch 3000, Loss: 0.0102
Epoch 3100, Loss: 0.0086
Epoch 3200, Loss: 0.0074
Epoch 3300, Loss: 0.0064
Epoch 3400, Loss: 0.0056
Epoch 3500, Loss: 0.0049
```

Epoch 3600, Loss: 0.0043
Epoch 3700, Loss: 0.0038
Epoch 3800, Loss: 0.0034
Epoch 3900, Loss: 0.0031
Epoch 4000, Loss: 0.0028
Epoch 4100, Loss: 0.0025
Epoch 4200, Loss: 0.0023
Epoch 4300, Loss: 0.0021
Epoch 4400, Loss: 0.0019
Epoch 4500, Loss: 0.0018
Epoch 4600, Loss: 0.0016
Epoch 4700, Loss: 0.0015
Epoch 4800, Loss: 0.0014
Epoch 4900, Loss: 0.0013
Epoch 5000, Loss: 0.0012
Epoch 5100, Loss: 0.0011
Epoch 5200, Loss: 0.0010
Epoch 5300, Loss: 0.0010
Epoch 5400, Loss: 0.0009
Epoch 5500, Loss: 0.0009
Epoch 5600, Loss: 0.0008
Epoch 5700, Loss: 0.0008
Epoch 5800, Loss: 0.0007
Epoch 5900, Loss: 0.0007
Epoch 6000, Loss: 0.0006
Epoch 6100, Loss: 0.0006
Epoch 6200, Loss: 0.0006
Epoch 6300, Loss: 0.0006
Epoch 6400, Loss: 0.0005
Epoch 6500, Loss: 0.0005
Epoch 6600, Loss: 0.0005
Epoch 6700, Loss: 0.0005
Epoch 6800, Loss: 0.0004
Epoch 6900, Loss: 0.0004
Epoch 7000, Loss: 0.0004
Epoch 7100, Loss: 0.0004

Epoch 7200, Loss: 0.0004
Epoch 7300, Loss: 0.0004
Epoch 7400, Loss: 0.0003
Epoch 7500, Loss: 0.0003
Epoch 7600, Loss: 0.0003
Epoch 7700, Loss: 0.0003
Epoch 7800, Loss: 0.0003
Epoch 7900, Loss: 0.0003
Epoch 8000, Loss: 0.0003
Epoch 8100, Loss: 0.0003
Epoch 8200, Loss: 0.0003

```
Epoch 8300, Loss: 0.0002
Epoch 8400, Loss: 0.0002
Epoch 8500, Loss: 0.0002
Epoch 8600, Loss: 0.0002
Epoch 8700, Loss: 0.0002
Epoch 8800, Loss: 0.0002
Epoch 8900, Loss: 0.0002
Epoch 9000, Loss: 0.0002
Epoch 9100, Loss: 0.0002
Epoch 9200, Loss: 0.0002
Epoch 9300, Loss: 0.0002
Epoch 9400, Loss: 0.0002
Epoch 9500, Loss: 0.0002
Epoch 9600, Loss: 0.0002
Epoch 9700, Loss: 0.0002
Epoch 9800, Loss: 0.0002
Epoch 9900, Loss: 0.0002
Predictions: [[0.00331625 0.99047302 0.98564877 0.01688339]]
```