dl5.ipynb | DL2.ipynb | DL6.ipynb

Filter files by name

/ Deep Learning /

Name	Last Modified
DL2.ipynb	8 days ago
dl3.ipynb	25 days ago
dl4.ipynb	17 days ago
dl5.ipynb	8 days ago
DL6.ipynb	8 minutes ago
Untitled.ipy...	7 days ago

Code — Notebook — Python 3 (ipykernel)

[13]:
```python
import torch
import torch.nn.functional as F
import matplotlib.pyplot as plt
```

[4]:
```python
X = torch.tensor([[0, 0],
                  [0, 1],
                  [1, 0],
                  [1, 1]], dtype=torch.float32)
y = torch.tensor([[0], [1], [1], [0]], dtype=torch.float32)
```

[5]:
```python
torch.manual_seed(42)
input_size = 2
hidden1 = 4
hidden2 = 4
output_size = 1
```

[6]:
```python
W1 = torch.randn(input_size, hidden1, requires_grad=True)
b1 = torch.zeros(hidden1, requires_grad=True)
```

[7]:
```python
W2 = torch.randn(hidden1, hidden2, requires_grad=True)
b2 = torch.zeros(hidden2, requires_grad=True)
```

[8]:
```python
W3 = torch.randn(hidden2, output_size, requires_grad=True)
b3 = torch.zeros(output_size, requires_grad=True)
```

[9]:
```python
lr = 0.1  # learning rate
```

Simple 0 5 Python 3 (ipykernel) | Idle Mem: 682.66 MB Mode: Command Ln 5, Col 16 DL6.ipynb 0

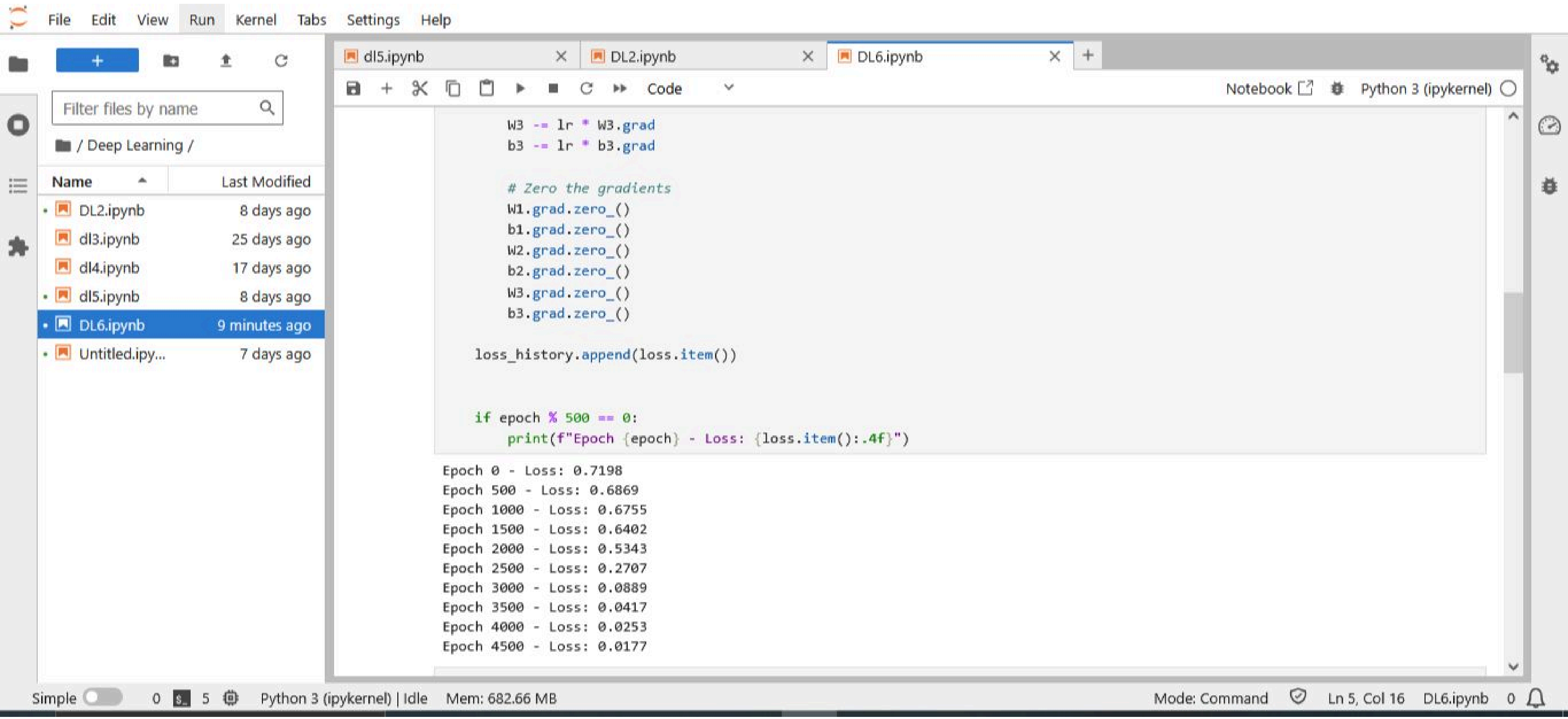
Name	Last Modified
DL2.ipynb	8 days ago
dl3.ipynb	25 days ago
dl4.ipynb	17 days ago
dl5.ipynb	8 days ago
DL6.ipynb	9 minutes ago
Untitled.ipy...	7 days ago

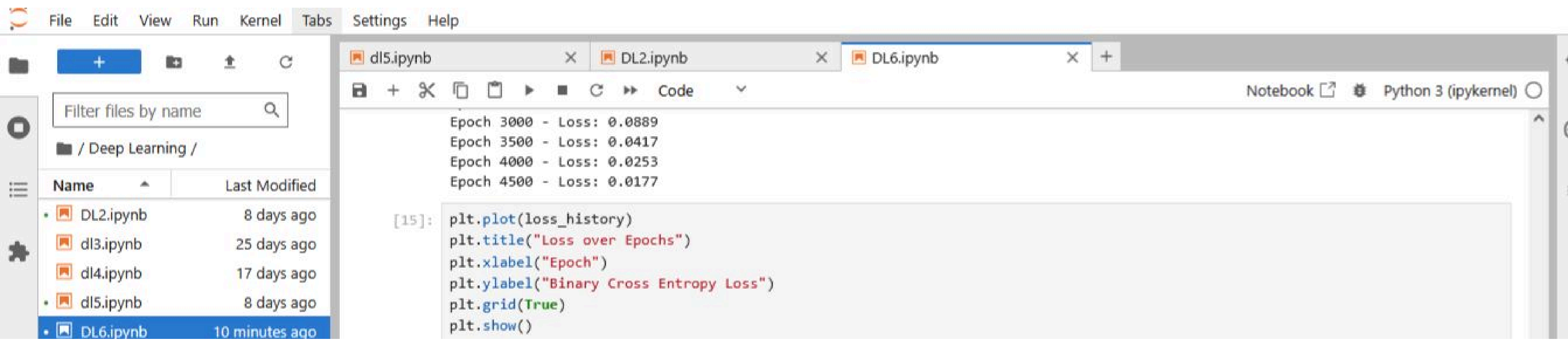
[9]:

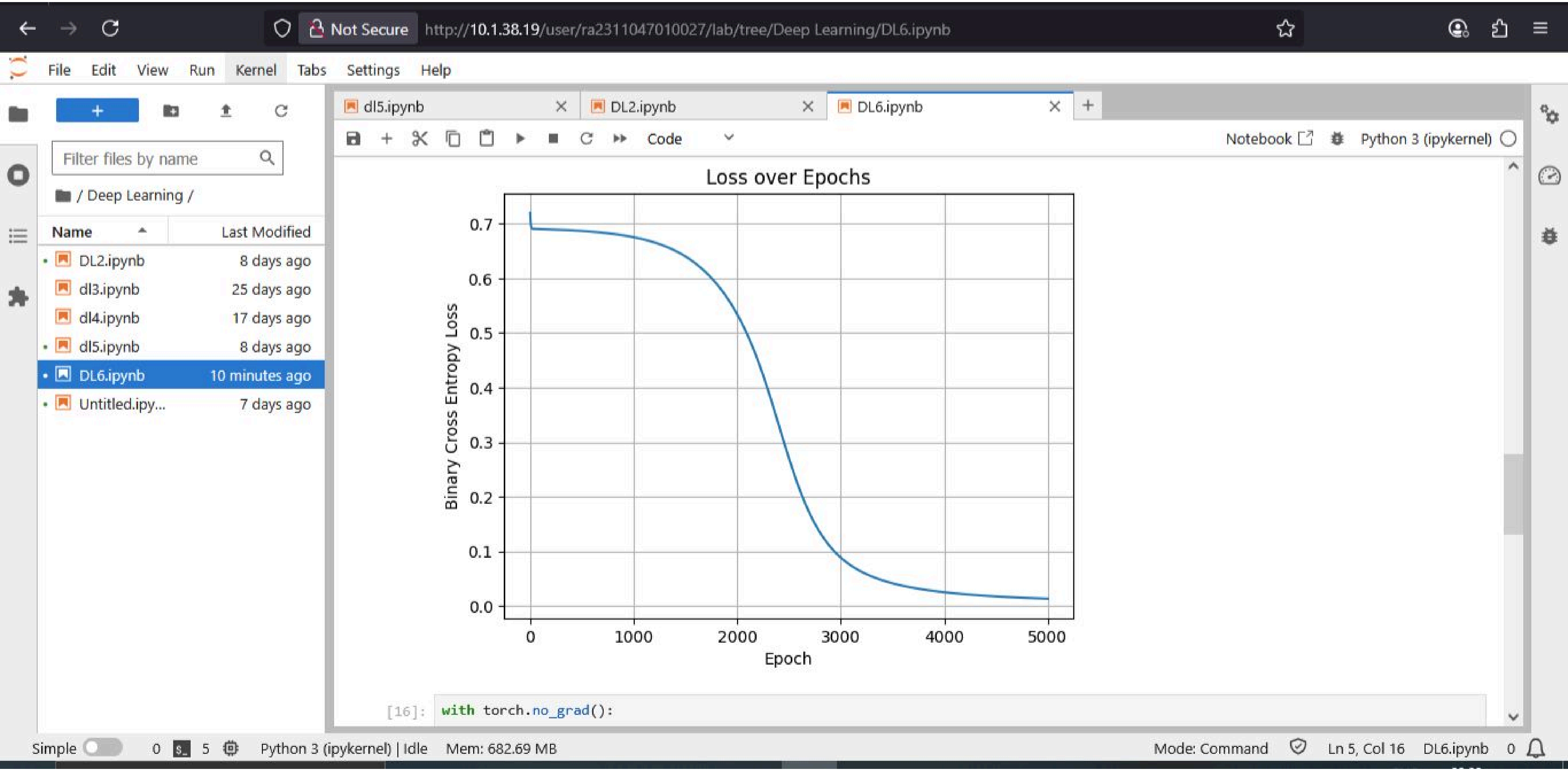
```
lr = 0.1  # Learning rate
epochs = 5000
loss_history = []
```

•[14]:

```
for epoch in range(epochs):
    # Forward pass
    z1 = X @ W1 + b1
    a1 = torch.sigmoid(z1)

    z2 = a1 @ W2 + b2
    a2 = torch.sigmoid(z2)

    z3 = a2 @ W3 + b3
    y_pred = torch.sigmoid(z3)

    loss = F.binary_cross_entropy(y_pred, y)
    loss.backward()
    with torch.no_grad():
        W1 -= lr * W1.grad
        b1 -= lr * b1.grad

        W2 -= lr * W2.grad
        b2 -= lr * b2.grad
```


File Edit View Run Kernel Tabs Settings Help
dl5.ipynb
DL2.ipynb
DL6.ipynb
Code
Notebook
Python 3 (ipykernel)
Filter files by name
/ Deep Learning /
Name
Last Modified
DL2.ipynb 8 days ago
dl3.ipynb 25 days ago
dl4.ipynb 17 days ago
dl5.ipynb 8 days ago
DL6.ipynb 9 minutes ago
Untitled.ipy... 7 days ago
W3 -= lr * W3.grad
b3 -= lr * b3.grad
# Zero the gradients
W1.grad.zero_()
b1.grad.zero_()
W2.grad.zero_()
b2.grad.zero_()
W3.grad.zero_()
b3.grad.zero_()
loss_history.append(loss.item())
if epoch % 500 == 0:
print(f"Epoch {epoch} - Loss: {loss.item():.4f}")
Epoch 0 - Loss: 0.7198
Epoch 500 - Loss: 0.6869
Epoch 1000 - Loss: 0.6755
Epoch 1500 - Loss: 0.6402
Epoch 2000 - Loss: 0.5343
Epoch 2500 - Loss: 0.2707
Epoch 3000 - Loss: 0.0889
Epoch 3500 - Loss: 0.0417
Epoch 4000 - Loss: 0.0253
Epoch 4500 - Loss: 0.0177
Simple 0 5 Python 3 (ipykernel) | Idle Mem: 682.66 MB
Mode: Command Ln 5, Col 16 DL6.ipynb 0


File Edit View Run Kernel Tabs Settings Help
dl5.ipynb
DL2.ipynb
DL6.ipynb
Code
Notebook
Python 3 (ipykernel)
Filter files by name
/ Deep Learning /
Name
Last Modified
DL2.ipynb 8 days ago
dl3.ipynb 25 days ago
dl4.ipynb 17 days ago
dl5.ipynb 8 days ago
DL6.ipynb 10 minutes ago
Epoch 3000 - Loss: 0.0889
Epoch 3500 - Loss: 0.0417
Epoch 4000 - Loss: 0.0253
Epoch 4500 - Loss: 0.0177
[15]:
plt.plot(loss_history)
plt.title("Loss over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Binary Cross Entropy Loss")
plt.grid(True)
plt.show()


Not Secure
http://10.1.38.19/user/ra2311047010027/lab/tree/Deep Learning/DL6.ipynb
File Edit View Run Kernel Tabs Settings Help
dl5.ipynb
DL2.ipynb
DL6.ipynb
Code
Notebook
Python 3 (ipykernel)
Filter files by name
/ Deep Learning /
Name
Last Modified
DL2.ipynb 8 days ago
dl3.ipynb 25 days ago
dl4.ipynb 17 days ago
dl5.ipynb 8 days ago
DL6.ipynb 10 minutes ago
Untitled.ipy... 7 days ago
Loss over Epochs
Binary Cross Entropy Loss
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
1000
2000
3000
4000
5000
Epoch
[16]:
with torch.no_grad():
Simple
0
5
Python 3 (ipykernel) | Idle
Mem: 682.69 MB
Mode: Command
Ln 5, Col 16
DL6.ipynb
0

0 1000 2000 3000 4000 5000
Epoch

[16]:
```python
with torch.no_grad():
    z1 = X @ W1 + b1
    a1 = torch.sigmoid(z1)

    z2 = a1 @ W2 + b2
    a2 = torch.sigmoid(z2)

    z3 = a2 @ W3 + b3
    y_pred = torch.sigmoid(z3)
    predicted = (y_pred > 0.5).float()

print("Predictions:\n", predicted)
print("Ground Truth:\n", y)
```

```
Predictions:
 tensor([[0.],
        [1.],
        [1.],
        [0.]])
Ground Truth:
 tensor([[0.],
        [1.],
        [1.],
        [0.]])
```

6. Implement gradient descent & backpropagation in deep neural network

Aim :

To implement gradient descent and backpropagation algorithms in a simple deep neural network and study their role in training.

Objectives :

1. To understand the working of gradient descent as an optimization method.
2. To implement backpropagation for updating weights in a neural network.
3. To train a simple neural network for a classification task using these algorithms.
4. To observe how loss decreases with iterations

Pseudo Code :

Start

1. Initialize dataset (X, Y) for training (eg: XOR dataset)
2. Initialize neural network Parameters :
   - Input layer, hidden layer, output layer sizes
   - Random weights & biases
   - Learning rate (η)

3. Define forward Propagation:

- $Z1 = W1 * X + b1$
- $A1 = activation(Z1)$ # Sigmoid or ReLU
- $Z2 = W2 * A1 + b2$
- $A2 = activation(Z2)$ # sigmoid / softmax (output)

4. Compute loss using Mean Squared Error or Cross entropy

5. Backpropagation:

- Compute error at output ($dA2 = A2 - y$)
- Calculate gradients for $W2, b2$
- Propagate error to hidden layer
- Calculate gradients for $W1, b1$

6. Update weights and biases using gradient descent:

$$W = W - \eta * dW$$
$$b = b - \eta * db$$

7. Repeat the steps 3-6 for given number of epochs.

8. Observe loss reduction and accuracy improvement

End

Observation:

→ Loss decreases as the number of iterations (epochs) increases

→ Weights and biases adjust to minimize error

→ Backpropagation ensures errors are efficiently distributed layer by layer

→ Learning rate (η) greatly influences Convergence speed.

Result :

Successfully implemented gradient descent and backpropagation in deep neural network.