

EX-N0:06 IMPLEMENT GRADIENT DESCENT & BACK
09.09.25 PROPAGATION IN DEEP NEURAL NETWORK

AIM

TO implement gradient descent, and backpropagation algorithm in a deep neural network and study their role in learning.

OBJECTIVE

TO understand the working of gradient optimization.

TO implement backpropagation for updating neural network weights.

TO observe the effect of iterations (epochs) on loss reduction.

PSEUDOCODE

Initialize weights and bias randomly for each epoch

a. forward pass

compute weighted sum

$$(z = (w^T x + b))$$

Apply activation function

$$(A = f(z))$$

b. compute loss

L = difference bw predicted & actual

c. backward pass

→ compute gradients of loss w.r.t weights & biases

→ update weights: $w = w - \eta * \frac{dL}{dw}$

YORK & MONTGOMERY IN THE LATE 19TH CENTURY
SCHOOL OF ECONOMIC HISTORY IN THE UNIVERSITY OF TORONTO

sample

MIA

Epoch	Training Loss	Accuracy	Remark
1	0.95	65.0	High error, random limit
9	0.48	82.3	Loss decreasing
10	0.25	90.1	fast convergence
20	0.18	95.5	model stabilized

Epoch	Loss
0	0.0134
500	0.0107
1000	0.0088
1500	0.0075
2000	0.0065
2500	0.0057
3000	0.0051
3500	0.0046
4000	0.0042
4500	0.0038

\Rightarrow update biases : $b = b - \mu * \frac{dL}{db}$

3. Repeat until converges

formula used

$$z = w \cdot x + b$$

$$L = \frac{1}{n} \sum (y - \hat{y})^2$$

$$w = w - \mu \frac{\frac{dL}{dw}}{=}$$

OBSERVATION

1. Initially, the model started with random weights leading to high loss and low accuracy.
2. with each epoch, gradient descent gradually reduced the loss, showing effect of iterative weight update.
3. Backpropagation efficiently adjusted weights layer by layer, improving model accuracy.
4. The choice of learning rate and number of epochs strongly influenced convergence speed and final accuracy.

~~Result~~

Implemented gradient descent + Backpropagation in DNN.

L6.ipynb (6) - JupyterLab Activation function explaine LINEARITY AND NN LINEAR Network Login Inbox (3,460) - dl7457@smr

Not secure 10.1.38.19/user/ra2311047010011/lab/tree/DEEP%20LEARNING/L6.ipynb

File Edit View Run Kernel Tabs Settings Help

+ C Filter files by name

/ DEEP LEARNING /

Name	Last Modified
data	25 days ago
Breast_cancer_da...	last month
Exp2.ipynb	last month
installpytorch.ip...	20 minutes ago
L_3.ipynb	25 days ago
L1.1.ipynb	last month
L1.ipynb	last month
L2.ipynb	25 days ago
L3.ipynb	25 days ago
L4.2.ipynb	20 minutes ago
L4.ipynb	20 minutes ago
L5.ipynb	20 minutes ago
L6.ipynb	20 minutes ago
Untitled.ipynb	last month

L4.ipynb installpytorch.ipynb L4.2.ipynb L5.ipynb L6.ipynb Notebook Python 3 (ipykernel)

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

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

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

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

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

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

[7]: lr = 0.1
epochs = 5000
loss_history = []

[11]: for epoch in range(epochs):
        Y1 = X @ W1 + b1
```

Simple 0 5 Python 3 (ipykernel) | Idle Mem: 1.07 GB Mode: Command Ln 7, Col 1 L6.ipynb 0 ENG 09:14 INTL 09-09-2025

DL6.ipynb (4) - JupyterLab x www.google.com x +

Not secure 10.1.38.19/user/ra2311047010027/lab/tree/Deep%20Learning/DL6.ipynb

File Edit View Run Kernel Tabs Settings Help

dl5.ipynb x DL2.ipynb x DL6.ipynb ● +

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 18 days ago
- dl5.ipynb 8 days ago
- DL6.ipynb 17 minutes ago

• [9]:

```
lr = 0.1
epochs = 5000
loss_history = []
```

• [14]:

```
for epoch in range(epochs):
    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
        W3 -= lr * W3.grad
        b3 -= lr * b3.grad
    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:
```

Simple 0 s 4 Python 3 (ipykernel) | Idle Mem: 639.08 MB Mode: Command 0 Ln 22, Col 24 DL6.ipynb 0 ENG 09:16 INTL 09-09-2025

The screenshot shows a Jupyter Notebook environment with three tabs open: 'dl5.ipynb', 'DL2.ipynb', and 'DL6.ipynb'. The 'DL6.ipynb' tab is active and contains Python code for training a neural network. The code defines hyperparameters (lr, epochs, loss_history), initializes weights and biases (W1, b1, W2, b2, W3, b3), and performs forward passes through three layers (z1, a1, z2, a2, z3, y_pred). It calculates the binary cross-entropy loss, performs backpropagation (loss.backward()), and updates the weights and biases using gradient descent. Gradients are zeroed out after each epoch. The current cell (cell 14) is highlighted with an orange vertical bar on the left. The status bar at the bottom indicates the mode is 'Command', memory usage is 639.08 MB, and the file is 'DL6.ipynb'.

Name	Last Modified
■ data	25 days ago
✚ Breast_cancer_da...	last month
✚ Exp2.ipynb	last month
• ✎ installPyTorch.ip...	22 minutes ago
✚ L_3.ipynb	25 days ago
✚ L1.1.ipynb	last month
✚ L1.ipynb	last month
✚ L2.ipynb	25 days ago
✚ L3.ipynb	25 days ago
• ✎ L4.2.ipynb	22 minutes ago
• ✎ L4.ipynb	22 minutes ago
• ✎ L5.ipynb	22 minutes ago
• ✎ L6.ipynb	22 minutes ago
✚ Untitled.ipynb	last month

```

b1 -= lr * b1.grad
W2 -= lr * W2.grad
b2 -= lr * b2.grad
W3 -= lr * W3.grad
b3 -= lr * b3.grad

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: 0.0134
 Epoch 500 - Loss: 0.0107
 Epoch 1000 - Loss: 0.0088
 Epoch 1500 - Loss: 0.0075
 Epoch 2000 - Loss: 0.0065
 Epoch 2500 - Loss: 0.0057
 Epoch 3000 - Loss: 0.0051
 Epoch 3500 - Loss: 0.0046
 Epoch 4000 - Loss: 0.0042
 Epoch 4500 - Loss: 0.0038

```
[9]: plt.plot(loss_history)
plt.title("Loss over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Binary Cross Entropy Loss")
plt.grid(True)
plt.show()
```

L6.ipynb (6) - JupyterLab Activation function explaine LINEARITY AND NN LINEAR Network Login Inbox (3,460) - dl7457@smr

Not secure 10.1.38.19/user/ra2311047010011/lab/tree/DEEP%20LEARNING/L6.ipynb

File Edit View Run Kernel Tabs Settings Help

+ C Filter files by name

/ DEEP LEARNING /

Name	Last Modified
data	25 days ago
Breast_cancer_da...	last month
Exp2.ipynb	last month
installpytorch.ip...	23 minutes ago
L_3.ipynb	25 days ago
L1.1.ipynb	last month
L1.ipynb	last month
L2.ipynb	25 days ago
L3.ipynb	25 days ago
L4.2.ipynb	23 minutes ago
L4.ipynb	23 minutes ago
L5.ipynb	23 minutes ago
L6.ipynb	23 minutes ago
Untitled.ipynb	last month

L4.ipynb installpytorch.ipynb L4.2.ipynb L5.ipynb L6.ipynb Notebook Python 3 (ipykernel)

Loss over Epochs

The graph plots 'Binary Cross Entropy Loss' on the y-axis (ranging from 0.0 to 0.7) against 'Epoch' on the x-axis (ranging from 0 to 5000). The curve starts at a loss of about 0.7 at epoch 0 and decreases monotonically, reaching a loss of approximately 0.05 at epoch 5000.

Epoch	Binary Cross Entropy Loss
0	0.70
1000	0.68
2000	0.55
3000	0.10
4000	0.05
5000	0.05

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

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

Simple 0 5 Python 3 (ipykernel) | Idle Mem: 1.07 GB Mode: Command Ln 7, Col 1 L6.ipynb 0 ENG 09:17 INTL 09-09-2025

L6.ipynb (6) - JupyterLab Activation function explained LINEARITY AND NN LINEAR Network Login Inbox (3,460) - dl7457@smi

Not secure 10.1.38.19/user/ra2311047010011/lab/tree/DEEP%20LEARNING/L6.ipynb Verify it's you

File Edit View Run Kernel Tabs Settings Help

+ C Filter files by name

/ DEEP LEARNING /

Name	Last Modified
data	25 days ago
Breast_cancer_da...	last month
Exp2.ipynb	last month
installpytorch.ip...	23 minutes ago
L_3.ipynb	25 days ago
L1.1.ipynb	last month
L1.ipynb	last month
L2.ipynb	25 days ago
L3.ipynb	25 days ago
L4.2.ipynb	23 minutes ago
L4.ipynb	23 minutes ago
L5.ipynb	23 minutes ago
L6.ipynb	23 minutes ago
Untitled.ipynb	last month

L4.ipynb installpytorch.ipynb L4.2.ipynb L5.ipynb L6.ipynb Notebook Python 3 (ipykernel)

Epoch

```
[10]: 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.]])
```

[]:

Simple 0 5 Python 3 (ipykernel) | Idle Mem: 1.07 GB Mode: Command 09:17 09-09-2025

0 5 ENG INTL 09:17 09-09-2025