LAB:6

8/09/25 Implement gradient descent and backpropagation in deep neural network

*AIM:

To implement gradient descent optimization and backpropagation learning algorithm in a ddep neural network for minimizing the error function and improving prediction accuracy.

*objectives

→ To understaind the Forward propagation process is a DNN.
→ To apply the backpropagation algorithm for computing weight updates
→ To implement gradient descent for minimizing the loss function.
→ To test the performance of the trained model on the given data

*pseudocode:

1. Initialize:
   → Define network architecture (layers, neurons per layer).
   → Randomly initialize weights and biases (small values).
   → choose learning rate η.

2. For each epoch (iteration over dataset):
   a. forward propagation:
      → For each layer l from input → output:
         z[l] = w[l] * A[l-1] + b[l]
         A[l] = activation (z[l])
         (Eg: sigmoid, Relu, softmax)

   b. compute loss:
      L = loss (y_true, A[L])

c. Backpropagation:

→ compute derivative of loss wrt output activations:

dA[L] = ∂(L)/∂A[L]

→ For each layer l from output → input:

dZ[l] = dA[l] * activation_derivative(Z[l])
dw[l] = (1/m) * dZ[l] * A[l-1].T
db[l] = (1/m) * sum(dZ[l])
dA[l-1] = w[l].T * dZ[l]

d. Parameter update (Gradient Descent):

→ w[l] = w[l] − α * dw[l]
→ b[l] = b[l] − α * db[l]

3. Repeat until convergence or max epochs

*observation:

→ Backpropagation:

pytorch computed all gradient with loss.

backward()

It understood how gradient are stored in grad and that optimization.

output:

→ fixed LR is Rigid: work easily on, but can stuck if learning rate is high/low for later training.

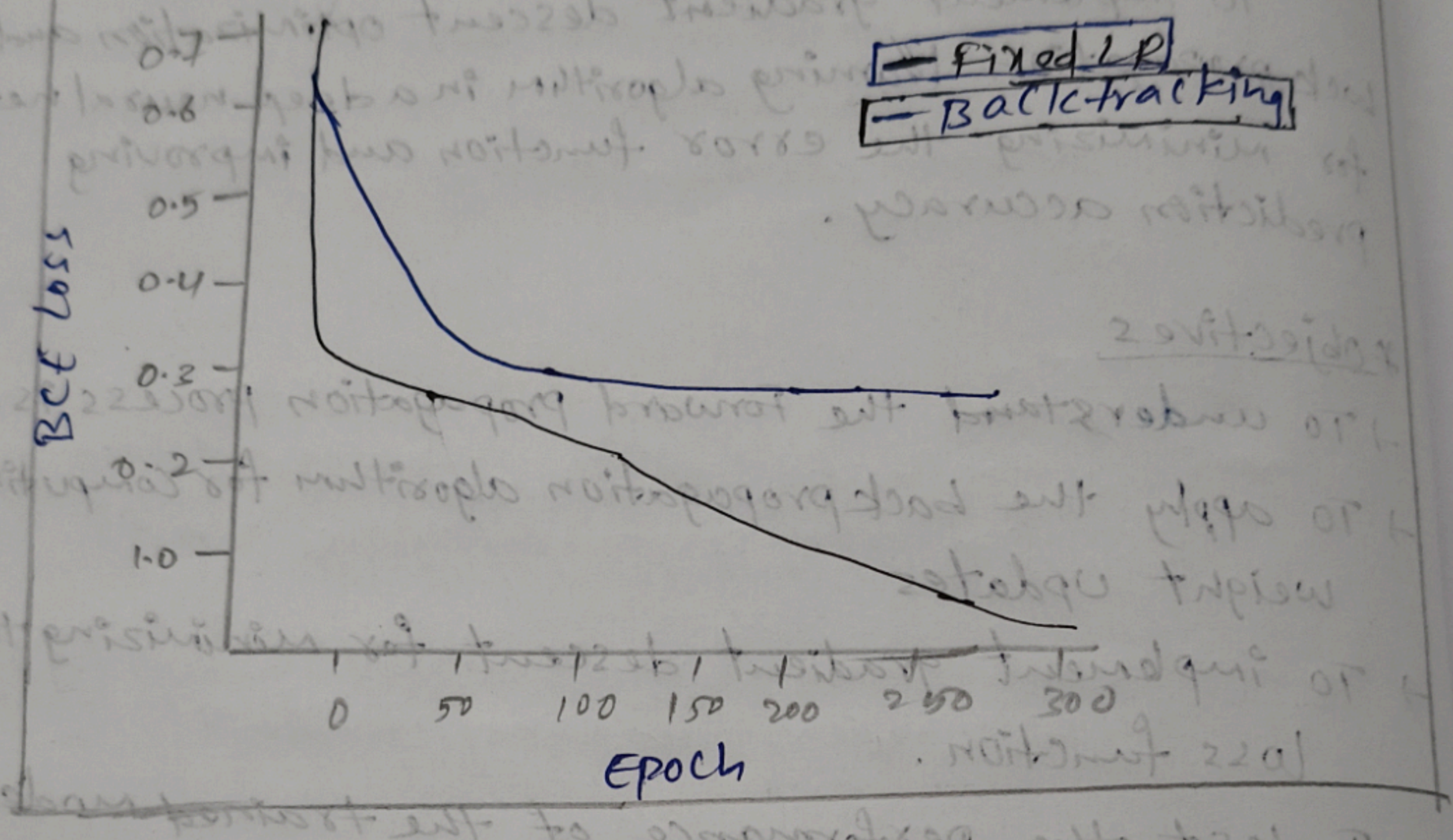
→ Backtracking adapts

→ lower loss = Better optimization, It network is learning a more accurate decision boundary.

*Result:

finally we implemented Backtracking successfully

Loss curves



Final summary:

Fixed LR - Train ACC: 0.8650, Test ACC: 0.8600

Backtrack - Train ACC: 0.9800, Test ACC: 0.9800

Epochs → Loss ↓ Acc ↑

Not Secure http://10.1.38.19/user/ra2311047010037/lab/workspaces/auto-k/tree/DLT/Lab6.ipynb

Settings Help

Go forward one page (Alt+Right Arrow)
Right-click or pull down to show history

Launcher | Lab6.ipynb | Lab5.ipynb

Code | Notebook | Python 3 (ipykernel)

Filter files by name

/ DLT /

Name	Last Modified
exp5.ipynb	56 minutes ago
Lab2.ipynb	last month
Lab3.ipynb	25 days ago
Lab4.ipynb	18 days ago
Lab5.ipynb	1 hour ago
Lab6.ipynb	58 seconds ago

•[7]:

```
import copy
import torch
import torch.nn as nn
import torch.nn.functional as F
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np
import matplotlib.pyplot as plt
torch.manual_seed(42)
np.random.seed(42)
device = torch.device("cpu")
X, y = make_moons(n_samples=500, noise=0.2, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train).astype(np.float32)
X_test = scaler.transform(X_test).astype(np.float32)

X_train_t = torch.from_numpy(X_train).to(device)
X_test_t = torch.from_numpy(X_test).to(device)
Y_train_t = torch.from_numpy(y_train.astype(np.float32)).unsqueeze(1).to(device)
Y_test_t = torch.from_numpy(y_test.astype(np.float32)).unsqueeze(1).to(device)
class SimpleMLP(nn.Module):
    def __init__(self, input_dim=2, hidden=16):
        super().__init__()
        self.fc1 = nn.Linear(input_dim, hidden)
        self.fc2 = nn.Linear(hidden, 1)
```

def forward(self, x):
x = F.relu(self.fc1(x))
x = self.fc2(x)
return x
def accuracy(model, X, y):
model.eval()
with torch.no_grad():
logits = model(X)
preds = (torch.sigmoid(logits) > 0.5).float()
return (preds == y).float().mean().item()
model_fixed = SimpleMLP().to(device)
opt_fixed = torch.optim.SGD(model_fixed.parameters(), lr=0.1)
criterion = nn.BCEWithLogitsLoss()
epochs = 300
losses_fixed = []
for epoch in range(epochs):
model_fixed.train()
opt_fixed.zero_grad()
logits = model_fixed(X_train_t)
loss = criterion(logits, y_train_t)
loss.backward()
opt_fixed.step()
losses_fixed.append(loss.item())
model_bt = SimpleMLP().to(device)
losses_bt = []
alpha_init = 1.0
Launcher
lab6.ipynb
lab5.ipynb
Code
Notebook
Python 3 (ipykern
Filter files by name
/ DLT /
Name
Last Modified
exp5.ipynb
57 minutes ago
lab2.ipynb
last month
lab3.ipynb
25 days ago
lab4.ipynb
18 days ago
lab5.ipynb
1 hour ago
lab6.ipynb
1 minute ago
File
Edit
View
Run
Kernel
Tabs
Settings
Help
Not Secure
http://10.1.36.19/user/1825104/6005/lab/workspaces/auto-K/tree/DLT/Lab6.ipynb

Not Secure http://10.1.38.19/user/ra2311047010037/lab/workspaces/auto-k/tree/DLT/Lab6.ipynb

File Edit View Run Kernel Tabs Settings Help

Launcher | Lab6.ipynb | Lab5.ipynb

Code | Notebook | Python 3 (ip

Filter files by name

/ DLT /

Name	Last Modified
exp5.ipynb	57 minutes ago
Lab2.ipynb	last month
Lab3.ipynb	25 days ago
Lab4.ipynb	18 days ago
Lab5.ipynb	1 hour ago
Lab6.ipynb	1 minute ago

```
alpha_init = 1.0
rho = 0.5
c = 1e-4
tiny_step = 1e-6
for epoch in range(epochs):
    model_bt.train()
    # forward + loss
    logits = model_bt(X_train_t)
    loss = criterion(logits, Y_train_t)
    losses_bt.append(loss.item())
    model_bt.zero_grad()
    loss.backward()
    grads = []
    for p in model_bt.parameters():
        grads.append(p.grad.view(-1))
    grad_vec = torch.cat(grads)
    grad_norm_sq = (grad_vec @ grad_vec).item()
    params_backup = copy.deepcopy(model_bt.state_dict())
 alpha = alpha_init
    accepted = False
while alpha > 1e-10:
        with torch.no_grad():
            i = 0
            for name, p in model_bt.named_parameters():
                numel = p.numel()
                g_slice = grad_vec[i:i+numel].view(p.shape)
                p.copy_(p - alpha * g_slice)
                i += numel
```

Not Secure http://10.1.38.19/user/ra2311047010037/lab/workspaces/auto-k/tree/DLT/Lab6.ipynb

Edit View Run Kernel Tabs Settings Help

Launcher | Lab6.ipynb | Lab5.ipynb

Code

Notebook Python 3

/ DLT /

Name	Last Modified
exp5.ipynb	58 minutes ago
Lab2.ipynb	last month
Lab3.ipynb	25 days ago
Lab4.ipynb	18 days ago
Lab5.ipynb	1 hour ago
Lab6.ipynb	1 minute ago

```
            break
        else:
            # revert to backup and reduce alpha
            model_bt.load_state_dict(params_backup)
            alpha *= rho
if not accepted:
        with torch.no_grad():
            i = 0
            for name, p in model_bt.named_parameters():
                numel = p.numel()
                g_slice = grad_vec[i:i+numel].view(p.shape)
                p.copy_(p - tiny_step * g_slice)
                i += numel
acc_fixed_train = accuracy(model_fixed, X_train_t, Y_train_t)
acc_fixed_test = accuracy(model_fixed, X_test_t, Y_test_t)
acc_bt_train = accuracy(model_bt, X_train_t, Y_train_t)
acc_bt_test = accuracy(model_bt, X_test_t, Y_test_t)

print("Final summary:")
print(f"Fixed LR  - Train Acc: {acc_fixed_train:.4f}, Test Acc: {acc_fixed_test:.4f}")
print(f"Backtrack - Train Acc: {acc_bt_train:.4f}, Test Acc: {acc_bt_test:.4f}")
plt.figure(figsize=(8,4))
plt.plot(losses_fixed, label="Fixed LR")
plt.plot(losses_bt, label="Backtracking")
plt.xlabel("Epoch")
plt.ylabel("BCE Loss")
plt.legend()
plt.grid(True)
plt.title("Loss curves")
```

0 2 Python 3 (ipykernel) | Idle Mem: 197.17 MB Mode: Command Ln 109, Col 81

Not Secure http://10.1.38.19/user/ra2311047010037/lab/workspaces/auto-k/tree/DLT/Lab6.ipynb

Edit View Run Kernel Tabs Settings Help

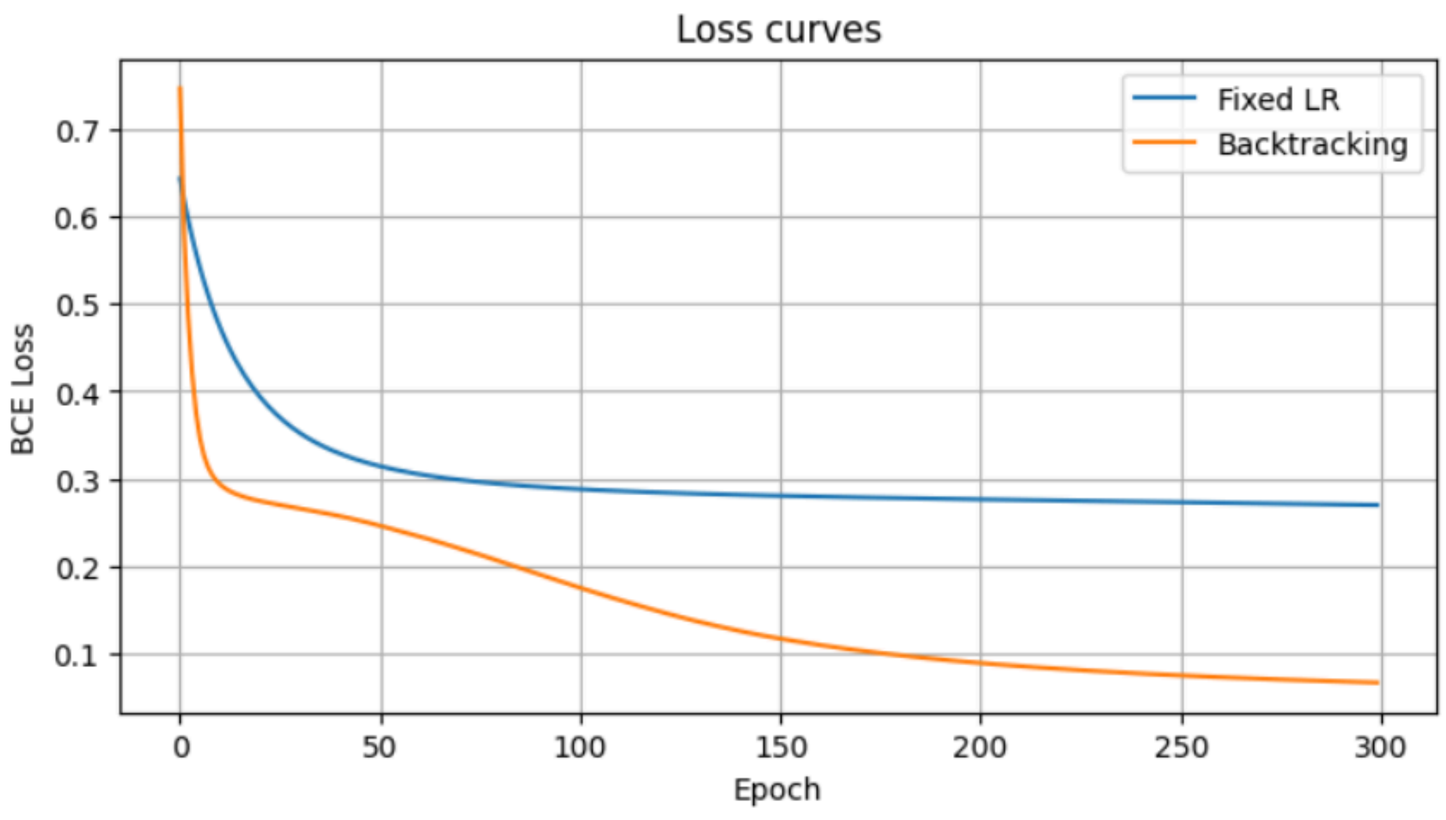
Launcher | Lab6.ipynb | Lab5.ipynb

Code

Notebook | Python 3 (ipykern

```
plt.show()
```

```
Final summary:
Fixed LR  - Train Acc: 0.8650, Test Acc: 0.8600
Backtrack - Train Acc: 0.9800, Test Acc: 0.9800
```



ter files by name

/ DLT /

ne	Last Modified
exp5.ipynb	58 minutes ago
Lab2.ipynb	last month
Lab3.ipynb	25 days ago
Lab4.ipynb	18 days ago
Lab5.ipynb	1 hour ago
Lab6.ipynb	18 seconds ago