Đề kiểm tra lập trình nhập môn phân tích dữ liệu và học sâu

Sinh viên không được phép sử dụng internet

Sinh viên sau khi làm bài xong xuất ra file PDF đồng thời nộp lên Fit-lab và push lên git-hub

Sinh viên làm bắt đầu làm bài từ 15h40 - 18h00

```
# Họ và Tên: Thai Quoc Bao
# MSSV: 207CT09955
```

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader, random_split
import numpy as np
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
```

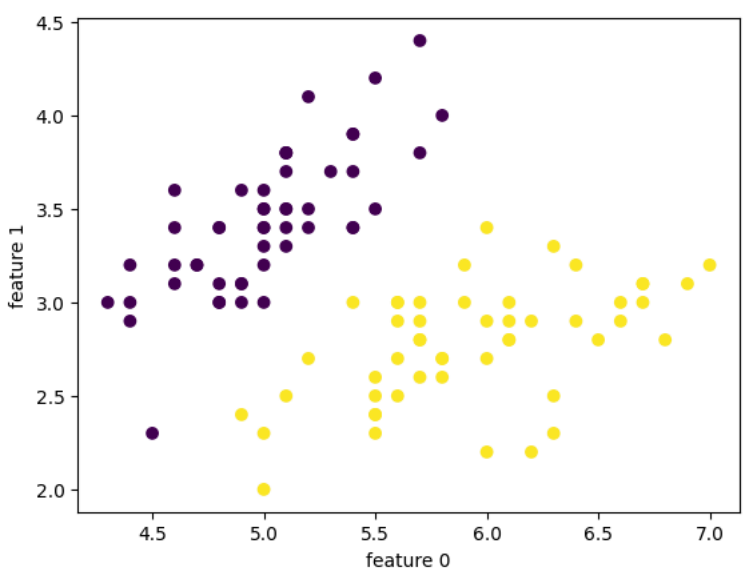
```
#Bước 1: Load data
def load_dataset():
    X, y = load_iris(return_X_y=True)
    X = X[y!=2]
    y = y[y!=2]
    return X,y
#Điền ở đây
X,y = load_dataset()
print( X.shape,y.shape )
```

```
(100, 4) (100,)
```

Kết quả: (100, 4) (100,)

```
#Trực quan hóa dữ liệu data
#Điền code ở đây

plt.scatter(X[:,0], X[:, 1], c=y)
plt.xlabel('feature 0')
plt.ylabel('feature 1')
plt.show()
```



Kết quả

```
# Bước 2: Định nghĩa mô hình hồi quy logistic bằng PyTorch
class LogisticRegressTorch(nn.Module):
    def __init__(self, n_features):
        super(LogisticRegressTorch, self).__init__()
        #Điền ở đây theo comment               # tạo một lớp tuyến tính (nn.Linear) với n_features đầu vào và 1 đầu ra
        self.linear = nn.linear(n_features, 1)

    def forward(self, x):
        return torch.sigmoid(self.linear(x))



# Bước 3: Định nghĩa lớp dữ liệu
class IrisTorch(Dataset):
    def __init__(self, X, y):
        self.X = torch.tensor(X, dtype=torch.float32)
        self.y = torch.tensor(y, dtype=torch.float32).unsqueeze(1)

    def __len__(self):
        return len(self.X) #Điền ở đây theo comment            #trả về số lượng mẫu trong tập dữ liệu (số lượng hàng trong self.X)

    def __getitem__(self, idx):
        return self.X[idx], self.y[idx]  #Điền ở đây theo comment            #trả về một cặp đặc trưng và nhãn tương ứng với chỉ số idx


# Tạo dữ liệu
dataset = IrisTorch(X, y)


# Bước 4: Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra bằng cách chia ngẫu nhiên 70,30.
train_size = int(0.7 * len(dataset))                                                    #70%
test_size =  len(dataset) - train_size                                                   #30%
train_dataset, test_dataset = random_split(dataset, [train_size, test_size])


# Tạo DataLoader
batch_size = 64
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader  =  DataLoader(test_dataset, batch_size=batch_size, shuffle=False)



# Bước 5: Định nghĩa criterion và optimizer
n_features = X.shape[1]

# Định nghĩa mô hình đơn giản với một lớp fully connected
class SimpleNN(nn.Module):
    def __init__(self, input_dim):
        super(SimpleNN, self).__init__()
        self.fc = nn.Linear(input_dim, 1)  # Đầu ra là một giá trị duy nhất (cho bài toán nhị phân)

    def forward(self, x):
        return torch.sigmoid(self.fc(x))  # Sử dụng sigmoid để dự đoán xác suất
n_features = X.shape[1]
model = SimpleNN(n_features)
criterion = nn.BCELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)
```

```
# Huấn luyện mô hình
n_epochs = 200
train_losses = []
test_losses = []
test_accuracies = []

for epoch in range(n_epochs):
    model.train()
    train_loss = 0.0
    for inputs, targets in train_loader:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()
        train_loss += loss.item() * inputs.size(0)

    train_loss /= len(train_loader.dataset)
    train_losses.append(train_loss)

    # Đánh giá trên tập kiểm tra
    model.eval()
    test_loss = 0.0
    correct = 0
    total = 0
    with torch.no_grad():
        for inputs, targets in test_loader:
            outputs = model(inputs)
            loss = criterion(outputs, targets)
            test_loss += loss.item() * inputs.size(0)

            predicted = (outputs >= 0.5).float()
            total += targets.size(0)
            correct += (predicted == targets).sum().item()

    test_loss /= len(test_loader.dataset)
    test_losses.append(test_loss)

    accuracy = correct / total
    test_accuracies.append(accuracy)

    print(f'Epoch {epoch+1}/{n_epochs}, Train Loss: {train_loss:.4f}, Test Loss: {test_loss:.4f}, Test Accuracy: {accuracy:.4f}')
```

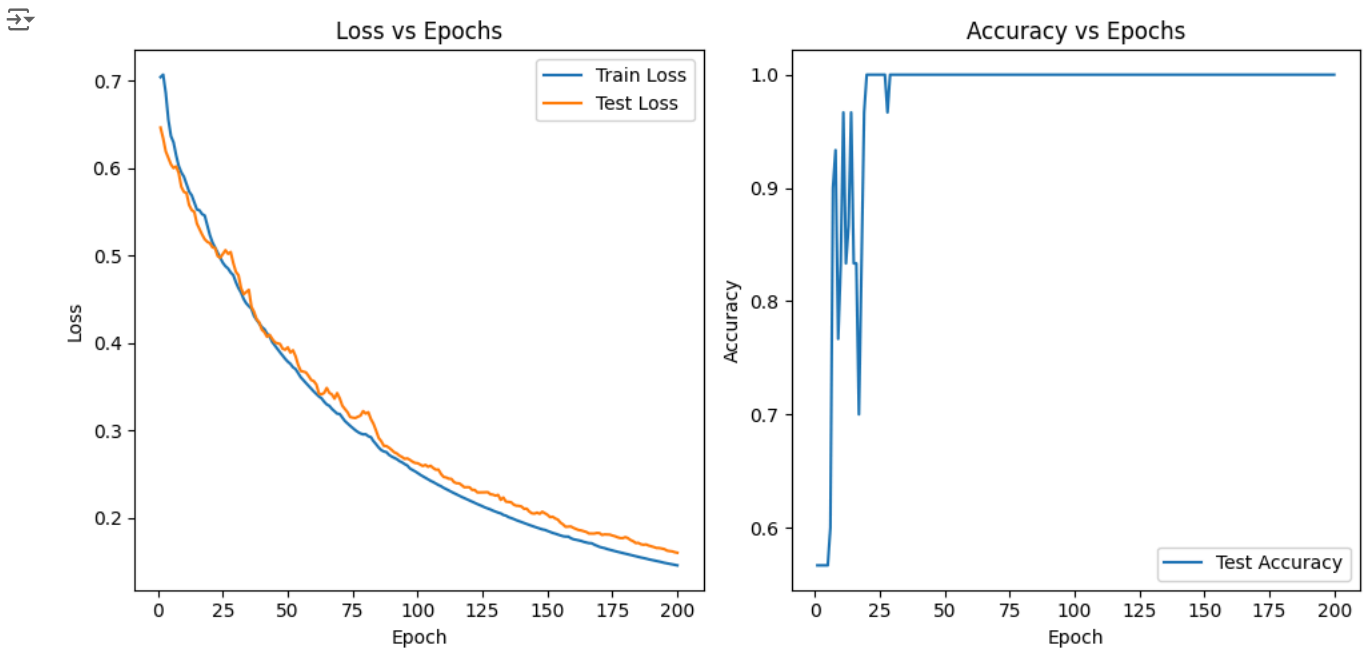
```
Epoch 132/200, Train Loss: 0.2049, Test Loss: 0.2203, Test Accuracy: 1.0000
Epoch 133/200, Train Loss: 0.2031, Test Loss: 0.2230, Test Accuracy: 1.0000
Epoch 134/200, Train Loss: 0.2023, Test Loss: 0.2184, Test Accuracy: 1.0000
Epoch 135/200, Train Loss: 0.2006, Test Loss: 0.2179, Test Accuracy: 1.0000
Epoch 136/200, Train Loss: 0.1995, Test Loss: 0.2177, Test Accuracy: 1.0000
Epoch 137/200, Train Loss: 0.1985, Test Loss: 0.2149, Test Accuracy: 1.0000
Epoch 138/200, Train Loss: 0.1971, Test Loss: 0.2137, Test Accuracy: 1.0000
Epoch 139/200, Train Loss: 0.1960, Test Loss: 0.2134, Test Accuracy: 1.0000
Epoch 140/200, Train Loss: 0.1950, Test Loss: 0.2128, Test Accuracy: 1.0000
Epoch 141/200, Train Loss: 0.1939, Test Loss: 0.2101, Test Accuracy: 1.0000
Epoch 142/200, Train Loss: 0.1927, Test Loss: 0.2103, Test Accuracy: 1.0000
Epoch 143/200, Train Loss: 0.1918, Test Loss: 0.2070, Test Accuracy: 1.0000
Epoch 144/200, Train Loss: 0.1906, Test Loss: 0.2050, Test Accuracy: 1.0000
Epoch 145/200, Train Loss: 0.1896, Test Loss: 0.2046, Test Accuracy: 1.0000
Epoch 146/200, Train Loss: 0.1886, Test Loss: 0.2058, Test Accuracy: 1.0000
Epoch 147/200, Train Loss: 0.1875, Test Loss: 0.2044, Test Accuracy: 1.0000
Epoch 148/200, Train Loss: 0.1866, Test Loss: 0.2068, Test Accuracy: 1.0000
Epoch 149/200, Train Loss: 0.1860, Test Loss: 0.2050, Test Accuracy: 1.0000
Epoch 150/200, Train Loss: 0.1848, Test Loss: 0.2033, Test Accuracy: 1.0000
Epoch 151/200, Train Loss: 0.1837, Test Loss: 0.2006, Test Accuracy: 1.0000
```

```
# Vẽ biểu đồ loss và accuracy
plt.figure(figsize=(10, 5))

# Vẽ biểu đồ loss cho tập huấn luyện và kiểm tra
plt.subplot(1, 2, 1)
plt.plot(range(1, n_epochs + 1), train_losses, label='Train Loss')
plt.plot(range(1, n_epochs + 1), test_losses, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss vs Epochs')
plt.legend()

# Vẽ biểu đồ accuracy cho tập kiểm tra
plt.subplot(1, 2, 2)
plt.plot(range(1, n_epochs + 1), test_accuracies, label='Test Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Accuracy vs Epochs')
plt.legend()

plt.tight_layout()
plt.show()
```



Kết quả: