Pratham Nagar - 64 Experiment 03 ~ Apply Nesterov Accelerated Gradient Algorithm on a feed forward neural network for Iris Flower classification.

```
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Load Wine data
wine = load_wine()
X = wine.data
y = wine.target

# Normalize features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Convert to PyTorch tensors
X = torch.tensor(X, dtype=torch.float32)
y = torch.tensor(y, dtype=torch.long)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

class WineNet(nn.Module):
    def __init__(self):
        super(WineNet, self).__init__()
        self.fc1 = nn.Linear(13, 16)  # 13 features in wine dataset
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(16, 3)   # 3 classes in wine dataset

    def forward(self, x):
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# In the criterion below add Cross Entropy Loss
# In the optimizer add the parameters of the model
# Add learning rate (alpha) as 0.01
# Add momentum as 0.9
# Enable the nesterov function
model = WineNet()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9, nesterov=True)

epochs = 100
for epoch in range(epochs):
    model.train()

    outputs = model(X_train)
    loss = criterion(outputs, y_train)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if (epoch+1) % 10 == 0:
        print(f"Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}")

model.eval()
with torch.no_grad():
    predictions = model(X_test)
    _, predicted_classes = torch.max(predictions, 1)
    acc = accuracy_score(y_test, predicted_classes)
    print(f"Test Accuracy: {acc * 100:.2f}%")
```

```
Epoch [10/100], Loss: 0.9650
Epoch [20/100], Loss: 0.6926
Epoch [30/100], Loss: 0.4419
Epoch [40/100], Loss: 0.2706
Epoch [50/100], Loss: 0.1763
Epoch [60/100], Loss: 0.1267
```

```
Epoch [70/100], Loss: 0.0986
Epoch [80/100], Loss: 0.0811
Epoch [90/100], Loss: 0.0692
Epoch [100/100], Loss: 0.0604
Test Accuracy: 100.00%
```