import numpy as np

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.metrics import log\_loss, accuracy\_score

# Load and preprocess data

X, y = load\_iris(return\_X\_y=True)

X = StandardScaler().fit\_transform(X)

y\_encoded = OneHotEncoder(sparse\_output=False).fit\_transform(y.reshape(-1, 1))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_encoded, test\_size=0.2, random\_state=42)

# Activation functions

def relu(x): return np.maximum(0, x)

def relu\_deriv(x): return (x > 0).astype(float)

def softmax(x): e = np.exp(x - np.max(x, axis=1, keepdims=True)); return e / np.sum(e, axis=1, keepdims=True)

# Initialize parameters

np.random.seed(0)

W1, b1 = np.random.randn(4, 100) \* 0.01, np.zeros((1, 100))

W2, b2 = np.random.randn(100, 3) \* 0.01, np.zeros((1, 3))

epochs, lr = 10000, 0.01

# Training loop

for i in range(epochs):

# Forward pass

Z1 = X\_train @ W1 + b1

A1 = relu(Z1)

Z2 = A1 @ W2 + b2

A2 = softmax(Z2)

# Print progress every 100 epochs

if i % 1000 == 0:

y\_pred\_labels = np.argmax(A2, axis=1)

y\_true\_labels = np.argmax(y\_train, axis=1)

print(f"Epoch {i}, Loss: {log\_loss(y\_train, A2):.4f}, Accuracy: {accuracy\_score(y\_true\_labels, y\_pred\_labels):.4f}")

# Backpropagation

dZ2 = A2 - y\_train

dW2 = A1.T @ dZ2 / len(X\_train)

db2 = np.mean(dZ2, axis=0, keepdims=True)

dZ1 = (dZ2 @ W2.T) \* relu\_deriv(Z1)

dW1 = X\_train.T @ dZ1 / len(X\_train)

db1 = np.mean(dZ1, axis=0, keepdims=True)

# Update weights

W1 -= lr \* dW1

b1 -= lr \* db1

W2 -= lr \* dW2

b2 -= lr \* db2

# Test evaluation

Z1 = X\_test @ W1 + b1

A1 = relu(Z1)

Z2 = A1 @ W2 + b2

A2 = softmax(Z2)

y\_pred\_test = np.argmax(A2, axis=1)

y\_true\_test = np.argmax(y\_test, axis=1)

print(f"\nTest Accuracy: {accuracy\_score(y\_true\_test, y\_pred\_test):.4f}")