import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

# Load Iris dataset fro

iris = load\_iris()

X = iris.data

y = iris.target

# Normalize data

X = (X - np.mean(X, axis=0)) / np.std(X, axis=0)

X = np.c\_[np.ones(X.shape[0]), X] # Add bias term

# One-hot encoding

y\_onehot = np.eye(3)[y]

# Softmax function

def softmax(z):

exp\_z = np.exp(z - np.max(z, axis=1, keepdims=True))

return exp\_z / np.sum(exp\_z, axis=1, keepdims=True)

# Initialize parameters

theta = np.zeros((X.shape[1], 3))

lr = 0.1

n\_iters = 1000

loss\_history = []

# Gradient descent

for i in range(n\_iters):

z = X @ theta

h = softmax(z)

loss = -np.mean(y\_onehot \* np.log(h))

loss\_history.append(loss)

grad = X.T @ (h - y\_onehot) / X.shape[0]

theta -= lr \* grad

# Plot convergence

plt.figure(figsize=(10,6))

plt.plot(loss\_history)

plt.title('Gradient Descent Convergence')

plt.xlabel('Epoch')

plt.ylabel('Loss (Cross-Entropy)')

plt.show()

z = X @ theta

h = softmax(z)

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

accuracy = np.mean(y\_pred == y)

print(f"Final Parameters:\n{theta}")

print(f"Training Accuracy: {accuracy:.2%}")