import numpy as np

class NeuralNetwork:

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

self.W1 = np.random.randn(input\_size, hidden\_size)

self.W2 = np.random.randn(hidden\_size, output\_size)

def forward(self, X):

self.z = np.dot(X, self.W1)

self.a = self.sigmoid(self.z)

self.z2 = np.dot(self.a, self.W2)

self.y\_hat = self.softmax(self.z2)

return self.y\_hat

def backward(self, X, y, lr):

delta3 = self.y\_hat - y

dW2 = np.dot(self.a.T, delta3)

delta2 = np.dot(delta3, self.W2.T) \* self.sigmoid\_prime(self.z)

dW1 = np.dot(X.T, delta2)

self.W1 -= lr \* dW1

self.W2 -= lr \* dW2

def sigmoid(self, z):

return 1 / (1 + np.exp(-z))

def sigmoid\_prime(self, z):

return self.sigmoid(z) \* (1 - self.sigmoid(z))

def softmax(self, x):

"""Compute softmax values for each sets of scores in x."""

e\_x = np.exp(x - np.max(x))

return e\_x / e\_x.sum(axis=1, keepdims=True)