import numpy as np

class BackPropagationFeedForward:

def \_\_init\_\_(self, input\_size=2, hidden\_size=2, output\_size=1):

np.random.seed(42)

self.W\_input\_hidden = np.random.uniform(size=(input\_size, hidden\_size))

self.b\_hidden = np.random.uniform(size=(1, hidden\_size))

self.W\_hidden\_output = np.random.uniform(size=(hidden\_size, output\_size))

self.b\_output = np.random.uniform(size=(1, output\_size))

self.loss\_history = []

def sigmoid(self, x):

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

def sigmoid\_derivative(self, x):

return x \* (1 - x)

def forward\_propagation(self, X):

self.hidden\_input = np.dot(X, self.W\_input\_hidden) + self.b\_hidden

self.hidden\_output = self.sigmoid(self.hidden\_input)

self.final\_input = np.dot(self.hidden\_output, self.W\_hidden\_output) + self.b\_output

self.final\_output = self.sigmoid(self.final\_input)

return self.final\_output

def backward\_propagation(self, X, y, output, learning\_rate=0.1):

output\_error = y - output

output\_delta = output\_error \* self.sigmoid\_derivative(output)

hidden\_error = output\_delta.dot(self.W\_hidden\_output.T)

hidden\_delta = hidden\_error \* self.sigmoid\_derivative(self.hidden\_output)

self.W\_hidden\_output += self.hidden\_output.T.dot(output\_delta) \* learning\_rate

self.b\_output += np.sum(output\_delta, axis=0, keepdims=True) \* learning\_rate

self.W\_input\_hidden += X.T.dot(hidden\_delta) \* learning\_rate

self.b\_hidden += np.sum(hidden\_delta, axis=0, keepdims=True) \* learning\_rate

def train(self, X, y, epochs=10000, learning\_rate=0.1):

for \_ in range(epochs):

output = self.forward\_propagation(X)

self.backward\_propagation(X, y, output, learning\_rate)

loss = np.mean(np.square(y - output))

self.loss\_history.append(loss)

def predict(self, X):

output = self.forward\_propagation(X)

return np.round(output)

def plot\_loss(self):

plt.plot(self.loss\_history)

plt.title("Loss Curve")

plt.xlabel("Epochs")

plt.ylabel("Loss (MSE)")

plt.grid(True)

plt.show()

# Training data for AND Gate

X = np.array([[0, 0],

[0, 1],

[1, 0],

[1, 1]])

y = np.array([[0],

[0],

[0],

[1]])

# Train the model

model = BackPropagationFeedForward()

model.train(X, y)

# Predictions

print("\nPredictions:")

print(model.predict(X))