import numpy as np

import matplotlib.pyplot as plt

# Initialisation des paramètres

x1, x2 = 0.5, 0.3 # Entrées

w1, w2 = 0.2, 0.4 # Poids des entrées

b = 1 # Biais

wb = 0.1 # Poids du biais

eta = 0.1 # Taux d'apprentissage

y\_real = 1 # Sortie réelle

# Fonction d'activation sigmoïde

def sigmoid(x):

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

# Dérivée de la sigmoïde

def sigmoid\_derivative(x):

return x \* (1 - x)

# Fonction de coût quadratique

def cost\_function(y\_pred, y\_real):

return 0.5 \* (y\_real - y\_pred) \*\* 2

# Stockage des valeurs pour affichage

errors = []

weights\_w1 = []

weights\_w2 = []

weights\_b = []

epochs = 100

# Entraînement avec descente du gradient

for epoch in range(epochs):

# Calcul de la sortie du neurone

net = w1 \* x1 + w2 \* x2 + wb \* b

y\_pred = sigmoid(net)

# Calcul de l'erreur

error = cost\_function(y\_pred, y\_real)

# Calcul du gradient

delta = (y\_pred - y\_real) \* sigmoid\_derivative(y\_pred)

# Mise à jour des poids

w1 -= eta \* delta \* x1

w2 -= eta \* delta \* x2

wb -= eta \* delta \* b

# Sauvegarde des valeurs pour affichage

errors.append(error)

weights\_w1.append(w1)

weights\_w2.append(w2)

weights\_b.append(wb)

# Condition d'arrêt si l'erreur est très faible

if error < 0.001:

break

# Affichage des courbes

plt.figure(figsize=(12, 5))

# Courbe de l'erreur

plt.subplot(1, 2, 1)

plt.plot(errors, label="Erreur quadratique")

plt.xlabel("Épochs")

plt.ylabel("Erreur")

plt.title("Évolution de la fonction de coût")

plt.legend()

# Courbe de l'évolution des poids

plt.subplot(1, 2, 2)

plt.plot(weights\_w1, label="w1")

plt.plot(weights\_w2, label="w2")

plt.plot(weights\_b, label="wb (biais)")

plt.xlabel("Épochs")

plt.ylabel("Valeur des poids")

plt.title("Évolution des poids")

plt.legend()

plt.tight\_layout()

plt.show()

# Affichage des poids finaux

print(f"Poids finaux: w1 = {w1:.4f}, w2 = {w2:.4f}, wb = {wb:.4f}")