import numpy as np

def sigmoid(x):

  return 1 / (1 + np.exp(- x))

# Arquitetura da MPL 4x3x2

N\_input = 3

N\_hidden = 4

N\_output = 2

#Vetor com valores de entrada aleatórios

X         = np.array([0.5, 0.1, -0.2])

target    = np.array([0.3, 0.8])

learnrate = 0.5

#Pesos da Camada Oculta

weigths\_in\_hidden = np.array([[-0.08, 0.08, -0.03, 0.03],

                              [ 0.05, 0.10,  0.07, 0.02],

                              [-0.07, 0.04, -0.01, 0.01]])

#Pesos da Camada de Saída

weigths\_in\_out = np.array([[-0.18, 0.11],

                           [-0.09, 0.05],

                           [-0.04, 0.05],

                           [-0.02, 0.07]])

#============================= Passagem Forward pela rede =============================

#Camada Oculta

#Calcule a combinação linear de entradas e pesos sinápticos

hidden\_layer\_in = np.dot(X, weigths\_in\_hidden)

#Aplicando a função de ativação

hidden\_layer\_out = sigmoid(hidden\_layer\_in)

#Camada de Saída

#Calcule a combinação linear de entradas e pesos sinápticos

output\_layer\_in = np.dot(hidden\_layer\_out, weigths\_in\_out)

#Aplicando a função de Ativação

output\_layer\_out = sigmoid(output\_layer\_in)

print("As saídas da rede são: ", output\_layer\_out)

#============================= Passagem Backward =============================

#Cálculo do Erro

error = target - output\_layer\_out

print('Erro da Rede: ', error)

#Calcule o termo de erro de saída (Gradiente da Camada de Saída)

output\_error\_term = error \* output\_layer\_out \* (1 - output\_layer\_out)

#Calcule a contribuição da camada oculta para o erro

hidden\_error = np.dot(weigths\_in\_out, output\_error\_term)

#print('weights\_hidden\_output: ',weights\_hidden\_output)

#print('output\_error\_term: ',output\_error\_term)

#Calcule o termo de erro da camada oculta (Gradiente da Camada Oculta)

hidden\_error\_term = hidden\_error \* hidden\_layer\_out \* (1 - hidden\_layer\_out)

#Calcule a variação do peso da camada de saída

delta\_w\_h\_o = learnrate \* output\_error\_term \* hidden\_layer\_out[:, None]

print('delta\_w\_h\_o: ', delta\_w\_h\_o)

#Calcule a variação do peso da camada oculta

delta\_w\_i\_h = learnrate \* hidden\_error\_term \* X[:, None]

print('delta\_w\_i\_h: ', delta\_w\_i\_h)

#============================= Atualização dos Pesos =============================

weights\_input\_hidden = learnrate \* delta\_w\_i\_h

print('weights\_input\_hidden: ', weights\_input\_hidden)

weights\_hidden\_output = learnrate \* delta\_w\_h\_o

print('weights\_hidden\_output: ', weights\_hidden\_output)