```
#tarefa 5
#Jóvio L. Giacomolli

import numpy as np

#função sigmoide
def sigmoid(x):
    return 1/(1 + np.exp(-x))

#arquitetura da MPL
n_input = 3
n_hidden = 4
n_output = 2

#vetor dos valores de entrada(aleatoria)
x = np.array([1, 2, 3])
target = 0.6
learnrate = 0.5

#pesos camada oculta

weights_input_hidden = np.array([[0.2, 0.1, -0.08, -0.1],
                                 [0.6, -0.8, 0.05, 0.02],
                                 [0.5, -0.6, -0.01, -0.07]])

#pesos camada de saida
weights_hidden_output = np.array([[0.1, -0.3],
                                  [-0.15, 0.12],
                                  [-0.03, 0.03],
                                  [-0.02, 0.02]])

#camada oculta
#calcule a combinação linear de entradas e pesos sinápticos
hidden_layer_input = np.dot(x, weights_input_hidden)
hidden_layer_output = sigmoid(hidden_layer_input)

#camada de saida

output_layer_in = np.dot(hidden_layer_output, weights_hidden_output)

#aplicar a função de ativação
output = sigmoid(output_layer_in)

print('As saidas da rede são: {}'.format(output))

#backward pass

error = target - output
#calculo do termo do erro
output_error_term = error * output * (1 - output)

hidden_error = np.dot(weights_hidden_output, output_error_term)
```

```
hidden_error_term = hidden_error * hidden_layer_output * (1 - hidden_layer_output)

delta_w_h_o = learnrate * output_error_term * hidden_layer_output[:, None]
print(delta_w_h_o)
print('\n')
delta_w_i_h = learnrate * hidden_error_term * x[:, None]
print(delta_w_i_h)
print('\n')
weights_input_hidden = learnrate * delta_w_i_h
print('Peso da entrada oculta: {}'.format(weights_input_hidden))
print('\n')
weights_hidden_output = learnrate * delta_w_h_o
print('Peso da saída oculta: {}'.format(weights_hidden_output))
```