```
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)
```