Projet Apprentissage Automatique

Objectif : Réalisation en python d'un petit réseau de neurones apprenant avec Backprop sur l'exemple XOR

x	0	0	1	1
y	0	1	0	1
z	0	1	1	0

Table 1 : la fonction XOR.

Résultat :

```
import numpy as np
```

Données input et output

```
X = np.array([[0,0], [0,1], [1,0], [1,1]])
Y = np.array([[0], [1],[1],[0]])
```

Parametres

```
epochs = 459000
input_size, hidden_size, output_size = 2, 1, 1
hidden_biais, output_biais = 1, 1
LR = .1 # learning rate
```

Fonction activation (sigmoide et derivee sigmoide)

```
#Sigmoid Function
def sigmoid (x):
    return 1/(1 + np.exp(-x))
#Derivative of Sigmoid Function
def derivatives_sigmoid(x):
    return x * (1 - x)
```

Mise à jour aleatoirement des poids et des connexions

```
w_hidden = np.random.uniform(size=input_size)
w_output = np.random.uniform(size=input_size+hidden_size)
biais_output = np.random.uniform(size=output_size)
biais_hidden = np.random.uniform(size=hidden_size)
```

Iterations

```
for n in range(epochs):
    for (i, x) in enumerate(X) :

        # -------- Forward Propogation
        hidden_layer_input = np.dot(x, w_hidden) + biais_hidden
        hiddenlayer_activation = sigmoid(hidden_layer_input)

        output_layer_input = np.dot(
            np.concatenate(
                (x, hiddenlayer_activation), axis=0), w_output
        ) + biais_output
        outputlayer_activation = sigmoid(output_layer_input)

        # --------- Back propagation
        #error signal
        s_error = Y[i]-outputlayer_activation
        s_error = s_error * derivatives_sigmoid(outputlayer_activation)

        #update output layer weights
        w_output += LR*np.concatenate((x, hiddenlayer_activation), axis=0)*s_error
        biais_output += LR*s_error # update output biais weight

        #hidden error signal
        s_error_hidden = derivatives_sigmoid(hiddenlayer_activation)
        s_error_hidden = s_error_hidden * s_error * w_output[-1]

        w_hidden += x*s_error_hidden*LR #update hidden layer weight
        biais_hidden += LR*s_error_hidden #update hidden biais weight
```

Test

```
for x in X :
    hidden_layer_input = np.dot(x, w_hidden) + biais_hidden
    hiddenlayer_activation = sigmoid(hidden_layer_input)
    output_layer_input = np.dot(
        np.concatenate(
            (x, hiddenlayer_activation), axis=0), w_output
    ) + biais_output
    outputlayer_activation = sigmoid(output_layer_input)
    print(x, "resultat :", outputlayer_activation)
```

```
[0 0] resultat : [ 0.00752672]
[0 1] resultat : [ 0.99171261]
[1 0] resultat : [ 0.99171245]
[1 1] resultat : [ 0.00955539]
```