# Réseau de neurones artificiels : XOR

Le but de ce projet est de coder un réseau de neurones capable de retrouver les sorties d’une table XOR.

Table XOR :

|  |  |
| --- | --- |
| Entrées | Sortie |
| 0,0 | 0 |
| 1,0 | 1 |
| 0,1 | 1 |
| 1,1 | 0 |

Une image contenant ligne, diagramme, Police, cercle

Description générée automatiquementOn va créer un réseau de neurones avec 2 entrées qui correspondent aux entrées de la table XOR et une sortie qui correspond au résultat du réseau de neurones (idéalement très similaire à la sortie de la table XOR).

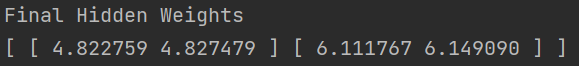
Pour ce projet je choisis de mettre une couche cachée avec 2 neurones. Le nombre de neurones affecte la précision, le temps calcul et les capacités de calcul (ram, processeur…). Ici 2 est suffisant mais vous pouvez jouer avec ce nombre si vous le voulez.

Les neurones sont reliés par des poids qui ont une valeur initial aléatoire. Les poids affectent la fonction d’activation. La fonction d’activation affecte la sortie (forward-propagation).

Une fois que la sortie à une valeur, on va effectuer une ‘back-propagation’ : qui va comparer le résultat de la sortie avec le résultat attendu et chercher à corriger l’erreur. On va donc modifier les poids en prenant en compte l’erreur.

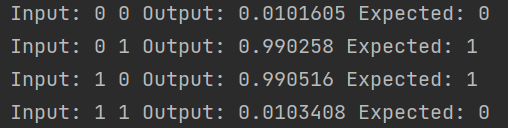
Début de l’entrainement : on effectue la forward et backward propagation pour chacune des 4 pairs entrées possible (0,0 0,1 1,0 1,1) et on répète cette logique des milliers de fois jusqu’à avoir un résultat proche du résultat attendu.

Une fois le réseau entrainé il suffit de récupérer les valeurs des poids.

Par exemple :



Maintenant vous avez une ai fonctionnelle, il suffit de la tester :



Ici on voit une précision à 10-2.

Finalement pour rendre le résultat plus visible j’ai mappé les coordonnées d’une page aux résultats pour tous les input de (0,0) à (1,1). Si la sortie est 0 alors la case est noir si la sortie est 1 la case est blanche (un gradient est utilisé car les résultats ne sont jamais exacte).

