Devoir de Rattrapage Programmation Parallèle

Etienne Martin SILR2

1) Etat de l'art

Un réseau de neurones est un système (hardware ou software) basé sur la structure des neurones observables dans le vivant.
Le réseau de neurones fait partie de la famille du machine learning, des algorithmes étant capable d'adapter leurs paramètres internes afin d'améliorer leur performance.

Le meilleur moyen de comprendre comment marche un réseau de neurones est d'observer comment se comporte un neurone dans la nature.

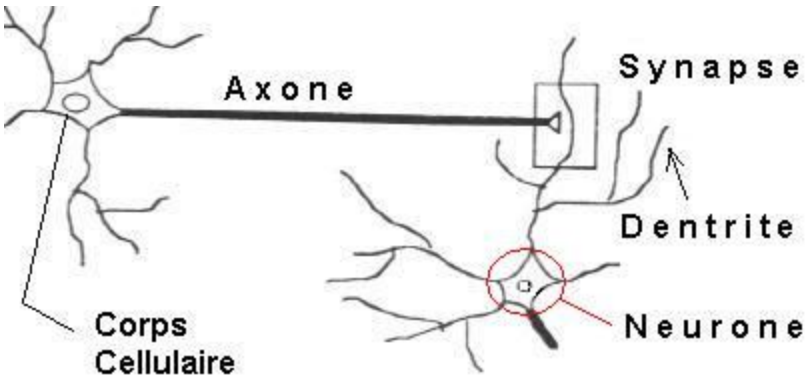

Figure 1 modèle de neurone

Un neurone est l'unité élémentaire d'un cerveau (humain ou animal). Chaque neurone est connecté à des milliers de ses congénères par l'intermédiaire de synapses. Lorsqu'un neurone est stimulé il envoie une légère impulsion électrique à ses voisins ces impulsions peuvent à leur tour activer d'autres neurones ou au contraire en désactiver prorata la nature de la liaison synaptique.

Ainsi dans notre cerveau des centaines de milliers de neurones s'activent et se désactivent mutuellement en fonction de l'infini quantité de données relayée par nos différents sens.

Si nous voulons modéliser de manière simpliste le fonctionnement d'un neurone, nous pouvons l'approcher comme une combinaison des éléments suivants :

- Un certain nombre d'entrées d'informations binaires
- Des coefficients/poids associés à chacune de ces entrées
- Une fonction d'activation aussi appelé fonction de transfert qui prendra en entrée la somme des produits des entrées et de leurs poids et retournera une valeur entre 0 et 1
- Un seuil d'activation, qui nous permettra de savoir en fonction de la valeur retournée si le neurone vient d'être stimulé ou non
- Un nombre non nul de sorties qui prendront l'information d'activation pour la propager à d'autres neurones

Lors d'une résolution de problèmes par un réseau de neurones, les poids à chacune des entrées d'informations sont constamment modifiées de manière excrémentielle afin d'approcher au mieux les solutions recherchées.

Pour ce qui est des réseaux de neurones artificiels deux fonctions de transferts sont très souvent utilisées

La fonction sigmoïde définie par f(x) => 1/(1 + e(-x))

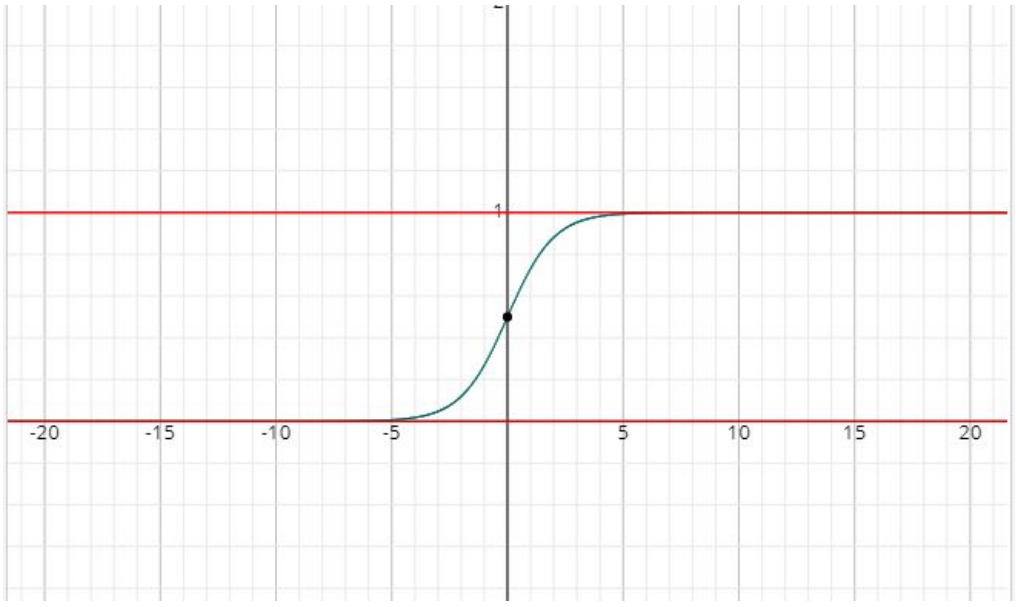

Figure 2 : représentation graphique de la fonction sigmoid

Et la fonction de Heaviside définie tel que f(x) = 0 Pt x >0 et f(x) = 1 Pt x<0

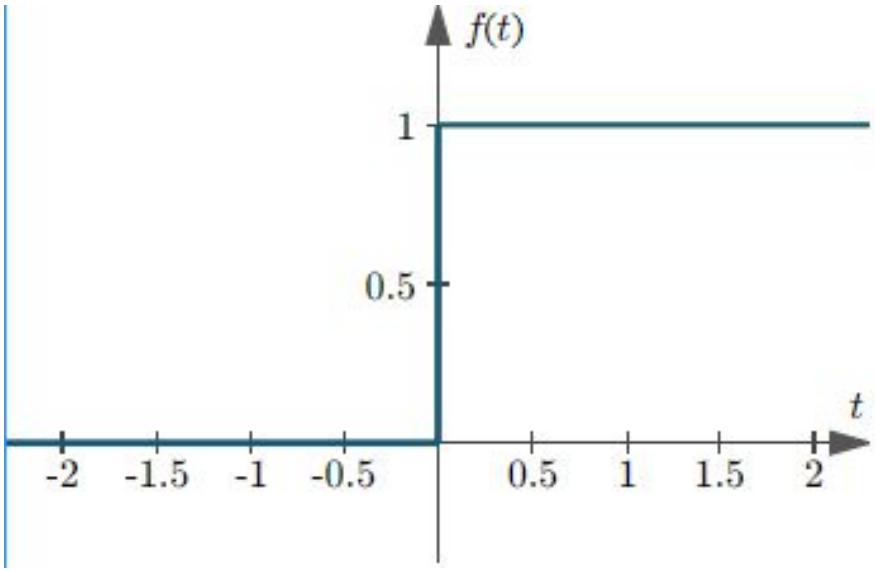

Figure 3 représentation graphique de la fonction de Heaviside

Le deuxième niveau d'abstraction d'un réseau de neurone est la couche de neurone

Il s'agit d'un vecteur de neurones prenant les mêmes entrées et renvoyant autant d'information qu'il y a de neurones dans la couche.

Si plusieurs couches de neurones sont connectées entre elles, les sorties d'informations de la première couche deviendront les entrées de la seconde.

2) Modélisation et Pseudo code

On considère L couches de neurones chacune composée d'une liste (notée Ln) de n neurones avec n variant entre 1 et C (C constante à définir)
Chaque couche peut avoir un nombre de neurones différents

Pour chaque neurone il existe, une liste de nC connections (notée Conn), une fonction de transfert fn(x) (pour notre cas une sigmoïde sera utilisée), un booléen Act représentant l'état d'activation du neurone et un seuil d'activation W.

Chaque connection est composée d'une référence à une sortie d'informations (notée outInfo) de la couche précédente ainsi qu'un poid/coefficient noté ConnW
Il existe autant de connections pour un neurone qu'il y a de neurones dans la couche qui précède la sienne.

Pseudo Code :

```
Somme = 0

Pour Tout l dans L
        Pour Tout neurone n dans l->Ln
                Pour Toutes Connections c dans n->Conn
                        Somme +=  c->outInfo * c->ConnW
                Fin Pour

                Si n->fn(Somme) > W
                        n->act = 1
                Sinon
                        n->act = 0
                Fin SI
                Somme =0
        Fin Pour
Fin Pour
```

3) Implémentation

CF CODE INCLU DANS LE DEPOT

4) Nous nous trouvons face à trois boucles qui peuvent être potentiellement parallélisables :

- La boucle qui itère les couches
- La boucle qui itère les neurones de chaque couche
- La boucle qui itère les connections de chaque neurone

Je ne pense pas que la première le soit vraiment, au final la couche n de neurones a besoin de savoir si les neurones de la couche n-1 ont été activés pour propager l'information à ses propres neurones. Implémenter une parallélisation sur cette boucle me semble contre productif puisque chaque thread devront s'attendre mutuellement.

Pour ce qui est de la seconde boucle tout est bien plus facile.
En effet le résultat de traitement d'un neurone ne dépends pas de celui des autres neurones de la même couche.
Il est donc possible d'implémenter une parallélisation de cette boucle.

La troisième boucle ne présente pas de difficulté d'implémentations
Néanmoins un appel si fréquent à la fonction de création de threads risque de couter en performances.

5) Le code a donc été parallélisé de la manière suivante

Le fork de OpenMp a lieu dans la boucle des neurone d'une couche
L'instruction For de l'api a été utilisé.
Celle-ci répartit la tâche d'une boucle for à tous les threads créés dans la section de code parallélisé
Afin d'améliorer les performances l'instruction schedule a été rajoutée afin de détailler le partage du travail entre les threads

Etant donné que le nombre de neurones peut changer drastiquement entre chaque couche j'ai choisi d'utiliser l'option "dynamic" de schedule sans préciser le nombre de neurone assigné à chaque thread (Chunk).

J'aurai pu aussi utiliser l'option guided, mais après plusieurs tests Dynamic restait meilleur (pour ce contexte d'utilisation)

L'accélération max est de facteur 2.5 (observé sur mon ordinateur avec 15 couches de neurones). Passé la barre des 4 threads aucune observation n'est visible.

NB : à chaque nouvelle couche openmp réassigne n threads puis les fusionne après traitement. S'il y avait un moyen de garder les threads 'vivants' entre chaque itération il serait peut être possible de gagner en performance.

NBB : J'ai essayé d'utiliser l'option collapse d'openMp pour fusionner la boucle des connections et des neurones. Mais étant donné que les boucles doivent être de structures simples et qu'elles ne doivent pas dépendre l'une de l'autre je n'ai pas trouvé de moyens performants de l'implémenter

5) Sources utilisées

Figure 1) + contenu concernant les réseaux de neurones

http://alp.developpez.com/tutoriels/intelligence-artificielle/reseaux-de-neurones/

Figure 2)

https://www.symbolab.com/graphing-calculator

Figure 3)

http://www.intmath.com/laplace-transformation/1a-unit-step-functions-definition.php

Rafraichissement de mémoire sur OpenMp

http://icps.u-strasbg.fr/~bastoul/teaching/openmp/bastoul_cours_openmp.pdf