Introduction aux Réseaux Neuronal

Sources :

* [Using neural nets to recognize handwritten digits](http://neuralnetworksanddeeplearning.com/chap1.html)
* <https://www.miximum.fr/blog/introduction-au-deep-learning-2/>
* <https://www.miximum.fr/blog/introduction-au-deep-learning-1/>

# 

# 1 - Implémenter Neuron & Layer

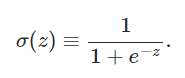
Un neurone est défini par une liste de poids associé à chaque neurone de la couche précédente.

La fonction d’agrégation respecte cette formule :



Pour un neurone J, w le poid associé au neurone j, x la valeur de l’input j, b le biais.

La fonction d’activation respecte cette formule :



Où Z est la valeur de l'agrégation.

Pour le Layer, un layer est un ensemble de neurone. Implémenter le Compute qui calcule la fonction d’activation de chaque neurone du layer.

# 2 - Évolution en matrice

Pour la suite de l’exercice nous allons nous baser sur l’écriture matricielle, le neurone n’existe plus et fait place à une Matrice<double> et un Vector<double>. Utiliser cette notation matricielle et implémenter MatrixAgregation, Matrix Activation et MatrixForward (qui s’occupe de faire traverser l’information au travers des différents layers). La dérivé de la fonction d’activation se définit :

*σ(x) \* (1 - σ(x))*

# 3 - Mécanisme d’apprentissage :

Basé sur cet article, implémenter le Train

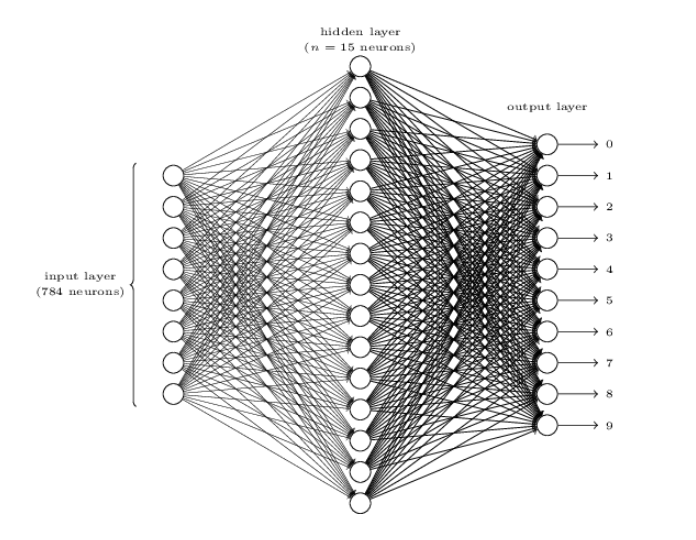
<https://www.miximum.fr/blog/introduction-au-deep-learning-2/>

# 

# Établissement du réseau en couche

Nous avons donc à notre arsenal plusieurs types de neurones possible, il nous reste plus qu’à les assembler.

Notre premier réseaux est définit comme ceci :



# Mécanisme d’apprentissage

Nous arrivons au coeur du problème : l’apprentissage.

Pour commencer nous allons construire nos neurones en prenant des paramètres aléatoires.

Il faudra ensuite soumettre ce réseau à des images et lui annoncer le résultat attendu, l’idée sera d’utiliser ensuite la *rétropropagation du gradient de l’erreur* (un nom bien compliqué)

# Soumission d’un numéro mystère et correction