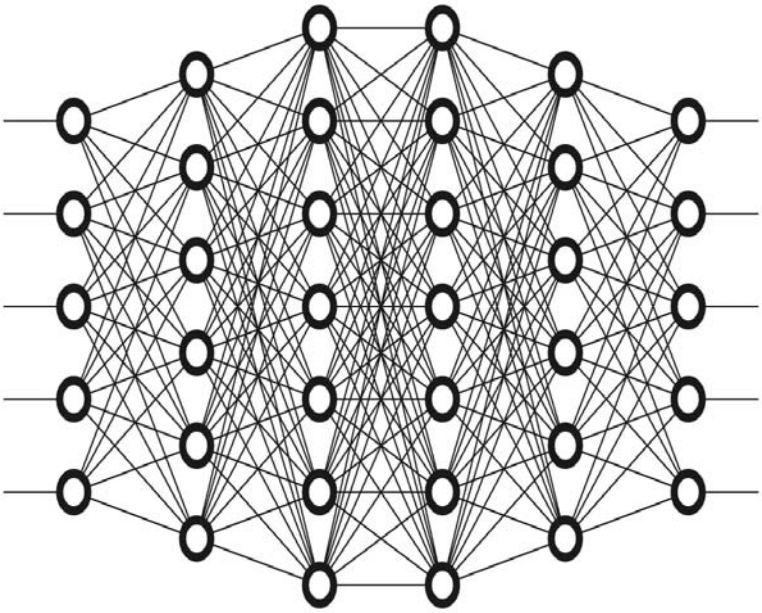




Les réseaux de neurones

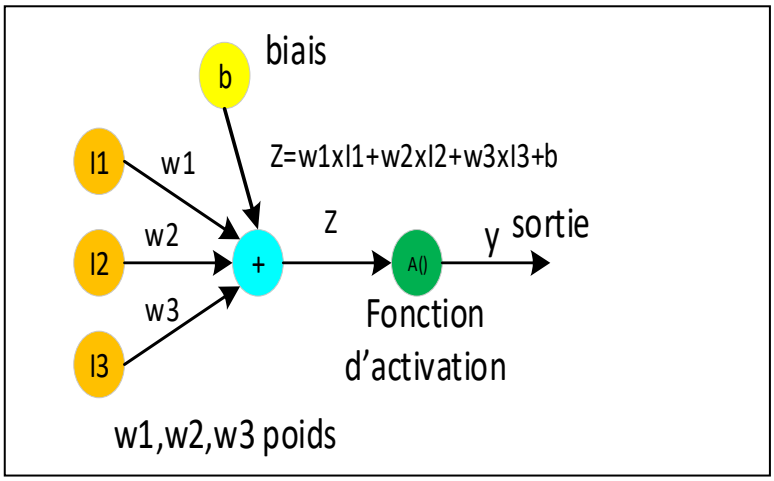


Les réseaux de neurones

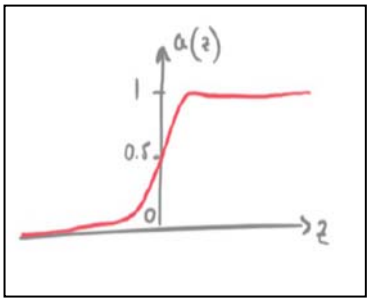
Un réseau de neurones ou plus exactement un réseau de neurones artificiels est un concept inspiré du fonctionnement du cerveau et des neurones biologiques. La structure à perceptron multi-couche ou **MultiLayer Perceptron** (MLP en anglais) a permis l'avènement de ces réseaux de neurones artificiels. L'intelligence artificielle (IA) et l'apprentissage automatique (Machine Learning) ont beaucoup progressé ces dernières années. Grâce au cloud cette technologie est devenue facilement accessible à tout le monde.

Structure d'un neurone

Un neurone est constitué des éléments suivants :



La fonction d'activation détermine l'activité du neurone



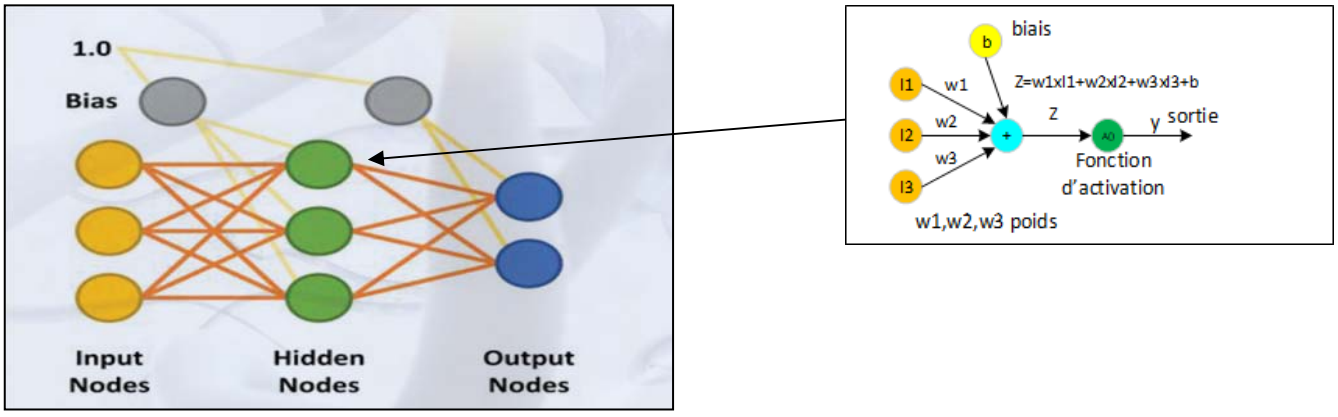
$$y = a(z) = \frac{1}{1+e^{-z}} \; fct\ sigmoide$$

Le rôle du **biais** est d'améliorer la capacité du réseau à résoudre le problème et de solliciter **la fonction d'activation** vers la gauche ou la droite.

Les connexions entre les neurones de différentes couches ont une valeur qui leur est associée : **Le poids** (w1,w2,w3).

Le perceptron multicouche

Ce sont ces valeurs qui déterminent, après calcul, l'activité d'un neurone et impactent les neurones de la couche suivante.

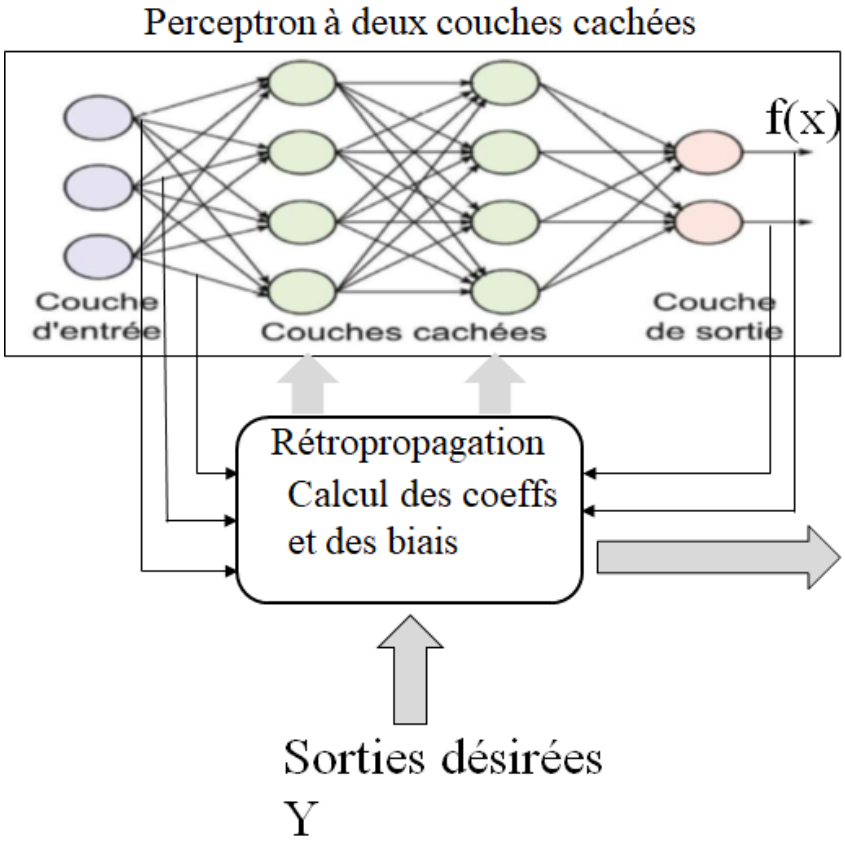


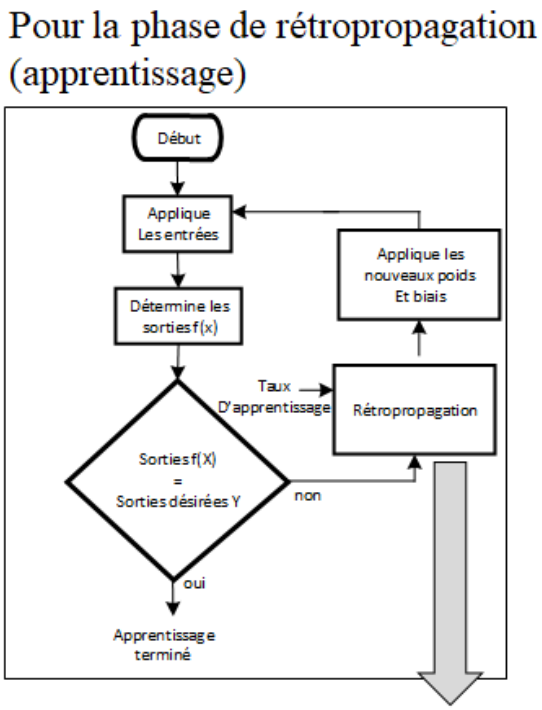
Perceptron multicouche
à une seule couche cachée

Un réseau de neurones ou plus exactement, un perceptron multicouche est composé de neurones, organisés en couches interconnectées.

L'information présentée via les neurones de la couche d'entrée circule au travers du réseau et aboutit, sur la couche de sortie en passant par les couches de neurones cachées.

La phase d'entrainement ou d'apprentissage supervisé d'un réseau de neurones consiste à ajuster la valeur de chacun de ses poids et biais, de manière à obtenir le résultat souhaité en fonction de l'entrée donnée.





Le processus consiste ici à déterminer la contribution de chaque poids et biais à l'erreur totale. L'erreur est calculée sous la forme d'une erreur quadratique moyenne ou MSE en anglais (pour Mean Squared Error).

$$J(a,b) = \frac{1}{2m}\sum_{i=1}^{m} erreur^i$$

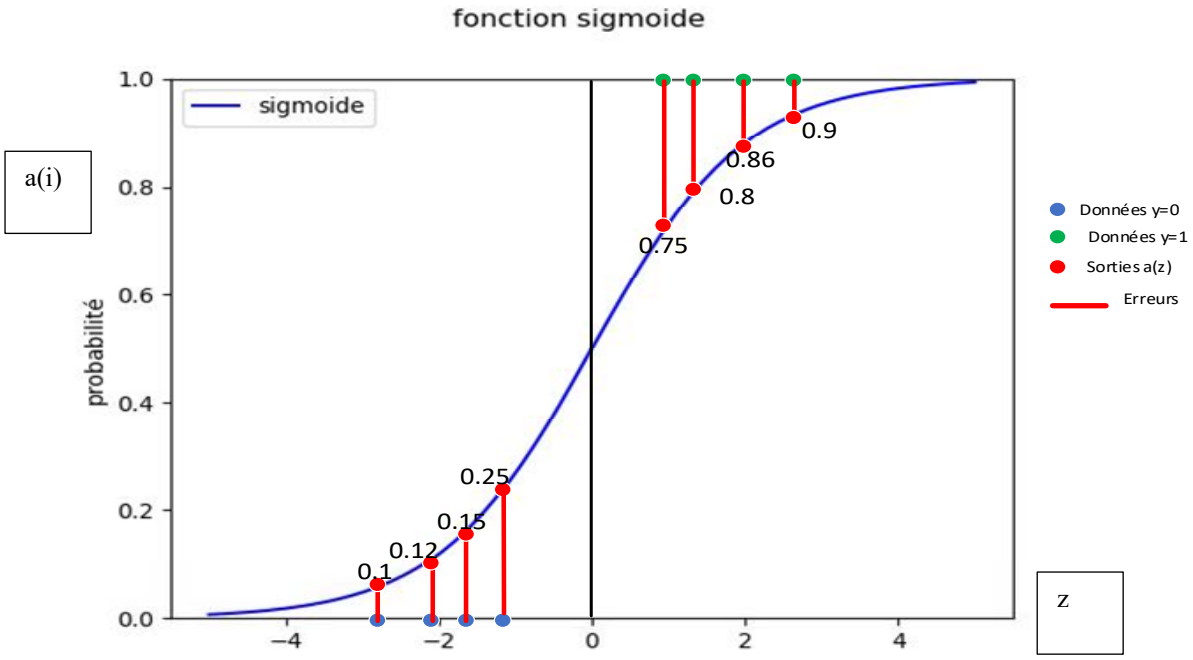
$$J(a,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f\left(x^{(i)}\right) - y^{(i)}\right)^2$$

i : indice des différentes données
m : nombre de données

La phase de rétropropagation permet grâce à l'algorithme de la descente du gradient de calculer de manière judicieuse les coefficient W et les biais b. l'algorithme de la descente du gradient se base sur la fonction cout

La fonction cout

Une fonction cout (loss fonction) est une fonction qui permet de quantifier les erreurs effectuées par un modèle.



La fonction Log Loss

$$L = \frac{-1}{m} \times \sum_{i=1}^{m} y_i \times \log(a_i) + (1 - y_i) \times \log(1 - a_i)$$

m : nombre de données
y_i: $donnée\ prédite\ n°i$
a_i: $sortie\ n°i$

Le modèle pour un neurone simple

z(I1,I2)=w1.I1+w2.I2+b

$$a(z) = \frac{1}{1 + e^{-z}}$$

La fonction cout

Démonstration de la formule du cout

Une façon d'évaluer la performance de notre modèle est de calculer sa vraisemblance. La vraisemblance indique la plausibilité du modèle vis-à-vis de vraies données.
Pour calculer la vraisemblance (Likelihood) de notre modèle on va calculer le produit de toutes les probabilités de nos sorties a(z)
En utilisant la loi de Bernoulli

$$L = \prod_{i=1}^{m} a_i^{yi} \times (1 - a_i)^{1-yi}$$

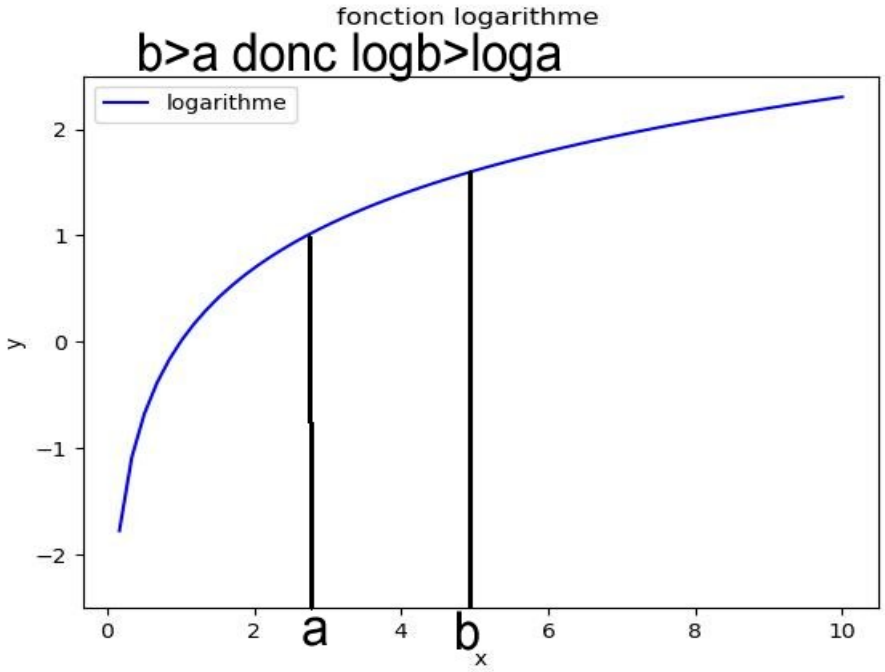
De tous les produits de nos données allant de 1 à m de nos probabilités retournés pour ces données.
Y_i peut prendre 2 valeurs 0 ou 1 i variant de 1 à 8 (8 valeur possibles dans ce cas pour $e^{-(w1.I1+w2.I2+b)}$)
Dans le cas précédent on obtient :

$$L = 0.1^0 \times (1-0.1)^{1-0} \times (1-0.12) \times (1-0.15) \times (1-0.25) \times 0.75^1 \times (1 - 0.75)^{1-1} \times 0.8 \times 0.86 \times 0.9$$
$$L = 0.9 \times 0.88 \times 0.85 \times 0.75 \times 0.75 \times 0.8 \times 0.85 \times 0.9 = 0.23$$

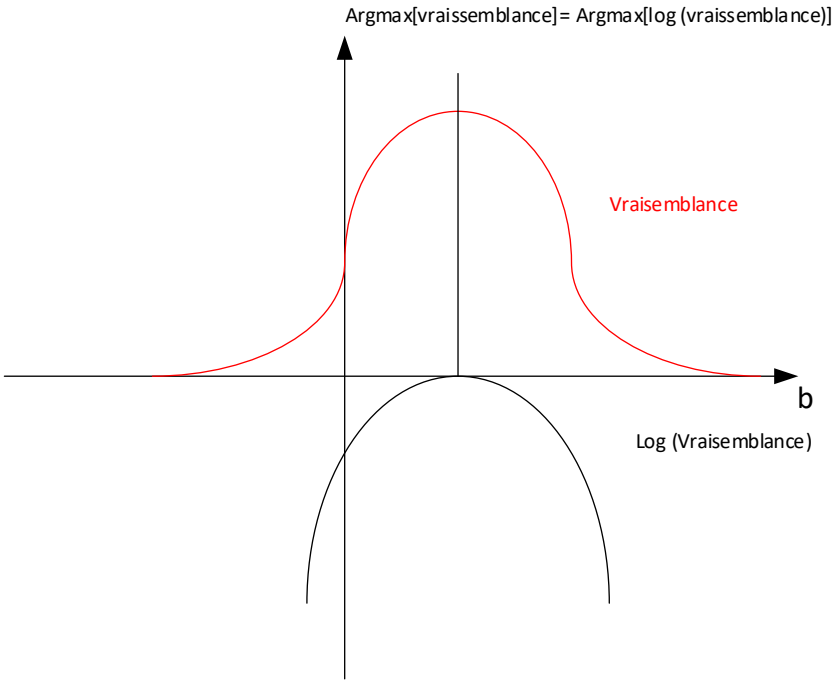
Le problème c'est que plus m est grand plus la vraisemblance tend vers 0 d'où le log:

$$L = \prod_{i=1}^{m} \log\left(a_i^{yi} \times (1-a_i)^{1-yi}\right)$$
$$L = \log(0.9) + \log(0.88) + \cdots ....$$

Comme la fonction logarithme est une fonction monotone croissante elle conserve l'ordre de nos termes :



Pour trouver b optimale par exemple on déterminera le maximum de la vraisemblance ou son log



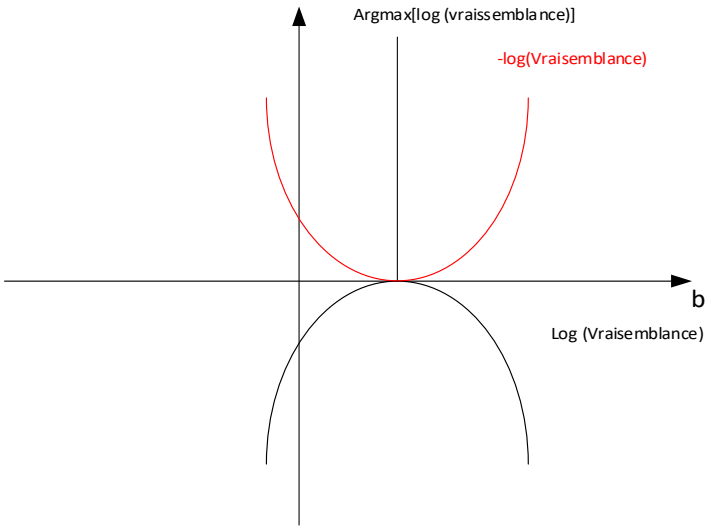
Par la propriété des logs on obtient :

$$L = \prod_{i=1}^{m} yi \times \log(a_i) + (1 - yi) \times \log(1 - a_i)$$

Comme on a un résultat négatif sur m échantillons on va normaliser avec :

$$L = \frac{-1}{m} \times \sum_{i=1}^{m} y_i \times \log(a_i) + (1 - y_i) \times \log(1 - a_i)$$

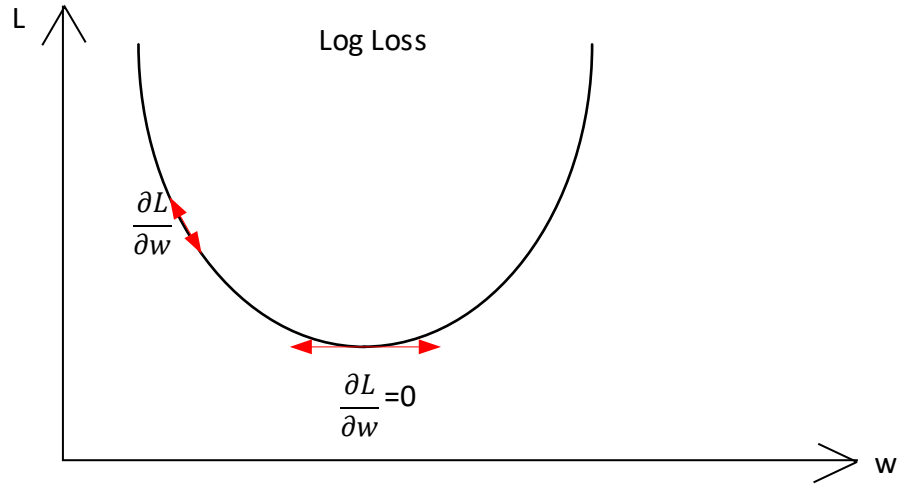
Maximiser f(x) revient à minimiser -f(x) d'où le -1 devant l'expression



On va se servir de la fonction cout pour minimiser les erreurs de notre modèle pour cela on va utiliser l'algorithme de la descente du gradient.

La descente de gradient

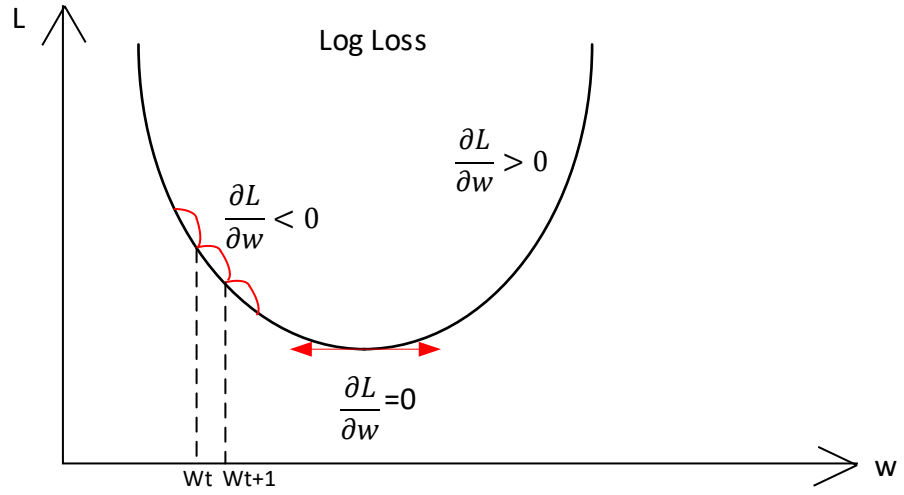
Cette algorithme consiste à ajuster les paramètres w et b de notre neurone de façon à minimiser les erreurs du modèle ou à minimiser la fonction cout (Log Loss)
Il faut déterminer comment cette fonction cout varie suivant les différents paramètres. On calcule ainsi le gradient ou la dérivée de notre fonction pour obtenir le minimum ou la pente s'annule



$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$

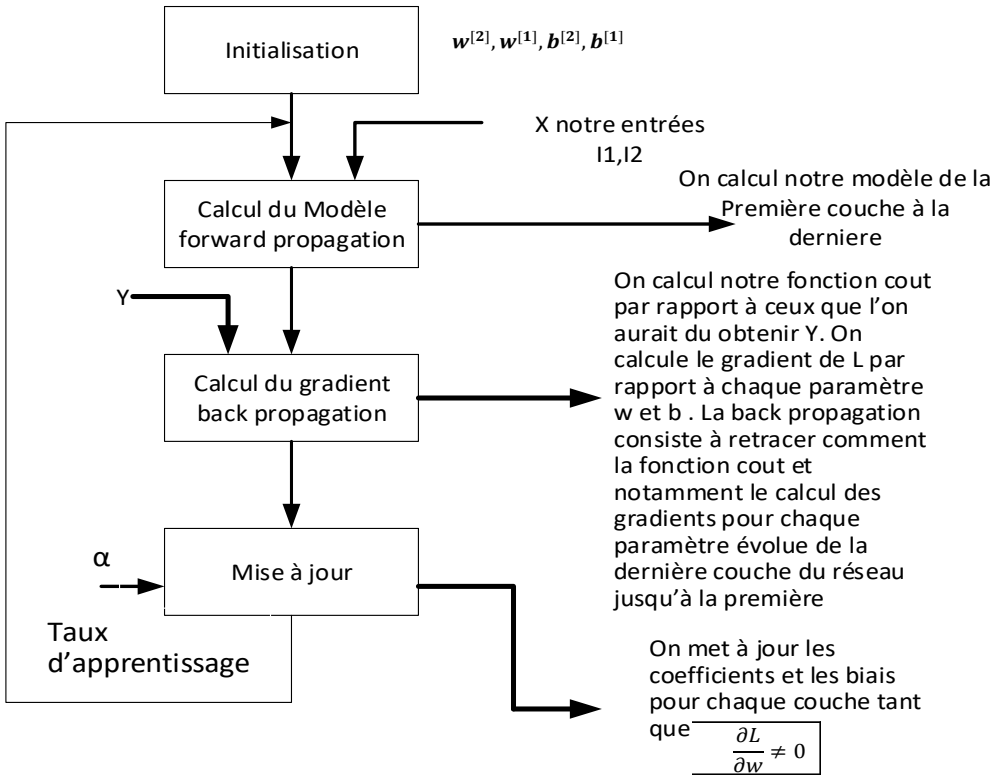
w_{t+1}: $paramètre$ à $l'intant\ t+1$
w_t: $paramètre$ à $l'intant\ t$
α: $pas\ d'apprentissage$
$\frac{\partial L}{\partial w_t}$ $gradient$ à $l'instant\ t$

En répétant cette formule en boucle on va pouvoir trouver le minimum pour notre fonction :

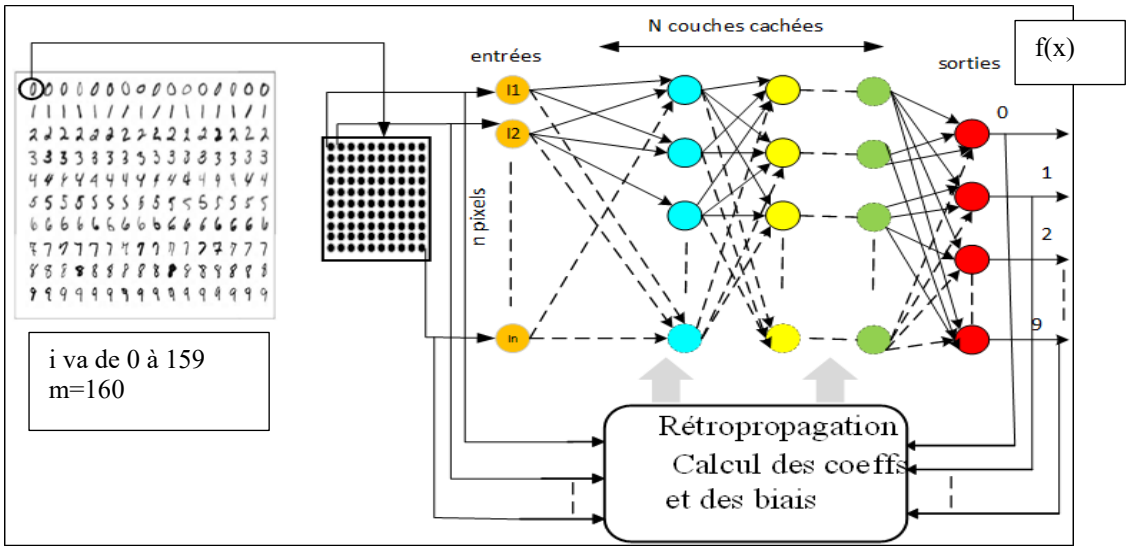


La fonction Log Loss est une fonction convexe qui n'admet qu'un minimum. Au minimum de la fonction cout on aura les paramètres w et b optimums

Les étapes de l'entrainement du réseau de neurones



Exemple d'application: Reconnaissance de chiffre



Ces modèles de perceptron (MLP) ont évolué vers d'autres modèles comme **les réseaux de neurones convolutifs** (reconnaissance d'image) ou **les Large Language Model (LLM)** qui sont des réseaux de neurones entraînés sur de larges volumes de données extraits du Web.

Les Generative Pre-trained Transformer (GPT) renvoient à un type de grand modèle de langage. Eux aussi s'appuient sur une architecture Transformer, mais ont été préentraînés sur un large corpus de texte en vue de

générer des séquences de texte cohérentes en fonction d'instructions fournies en entrée. Plus précisément, le rôle d'un modèle GPT est de prédire le prochain mot d'une phrase en fonction d'un contexte régi par les instructions et par les données utilisées lors de son pré entraînement.