Chapitre 1

Réseaux de neurones artificiels

1.1 Introduction

Dans ce premier chapitre, nous commençons par introduire les réseaux de neurones artificiels, qui sont la base de l'apprentissage profond, en donnant différentes notions de base et la méthode de leur mise en oeuvre dans le procédé d'apprentissage.

1.2 Neurone biologique

Dans l'antiquité, il n'était pas évident que le cerveau constituait l'organe essentiel de la pensée. Ainsi, pour Aristote « le coeur est au centre des processus sensitifs, le cerveau ayant alors un simple rôle de réfrigération de ce dernier »[1]

Au début du XX^{eme} siècle, Santiago Ramon y Cajal, neuro-scientifique espagnol, met en évidence que les cellules du cerveau sont des entités fonctionnelles autonomes connectées entre elles.

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites.[2]

C'est par les dendrites que l'information est acheminée de l'extérieur vers le noyau du neurone. L'information traitée par le neurone chemine ensuite le long de l'axone pour être transmise aux autres neurones. La jonction entre deux neurones est appelée la synapse.

1.2.1 Composants d'un neurone biologique

Le tableau (1.1) illustre les composants d'un neurone biologique ainsi que leur rôle.

Composant	**Rôle**
Corps cellulaire	intégration des messages nerveux
Dendrites	réception des messages nerveux
Axone	conduction des messages nerveux
Arborisation terminale	transmission des messages nerveux à un autre neurone

Table 1.1: Composants d'un neurone biologique

La figure (1.1) représente un schéma d'un neurone biologique[1].

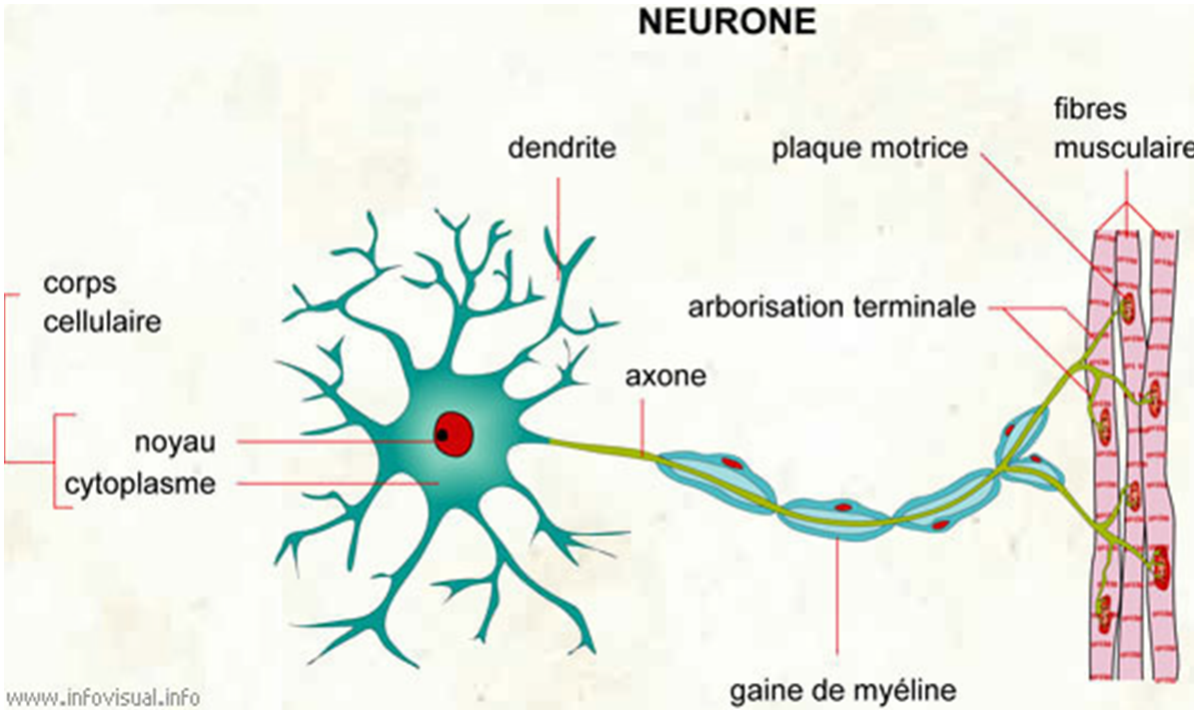

FIGURE 1.1: Schéma du neurone biologique

1.3 NEURONE FORMEL

C'est en 1943 que Mc Culloch (un neurophysicien) et Pitts (logicien) ont proposé les premières notions de neurone formel binaire. Ce concept fut ensuite, mis en réseau avec une couche d'entrée et une sortie par Rosenblatt en 1959, pour simuler le fonctionnement rétinien et tacher de reconnaître des formes. C'est l'origine du perceptron [3].

Cette approche, dite connexionniste, a atteint ses limites technologiques, compte tenu de la puissance de calcul de l'époque.

Un neurone formel est une représentation mathématique et informatique d'un neurone biologique. Le neurone formel possède généralement, plusieurs entrées et une seule sortie, qui correspondent respectivement, aux dendrites et au cône d'émergence du neurone biologique.[4] De manière schématique, un neurone reçoit des signaux entrant via les dendrites et envoie le signal sortant via l'axone. Cela ressemble à une fonction mathématique

1. https://www.infovisual.info/fr/corps-humain/neurone

à plusieurs variables f(x1,x2,x3,xn) avec des coefficients synaptiques (poids), il effectue une sommation pondérée via les poids et déclenche un signal de sortie y tel que :

$$y = w1.x1 + w2.x2 + w3.x3 + ... + wn.xn + b$$

la sortie finale est f(y) tel que f s'appelle la fonction d'activation. L'équation (1.1) illustre l'approximation mathématique du neurone et la figure (1.2) représente un schéma d'un neurone formel.

$$s = \varphi(\sum_{i=0}^{i=p} w_i e_i) \tag{1.1}$$

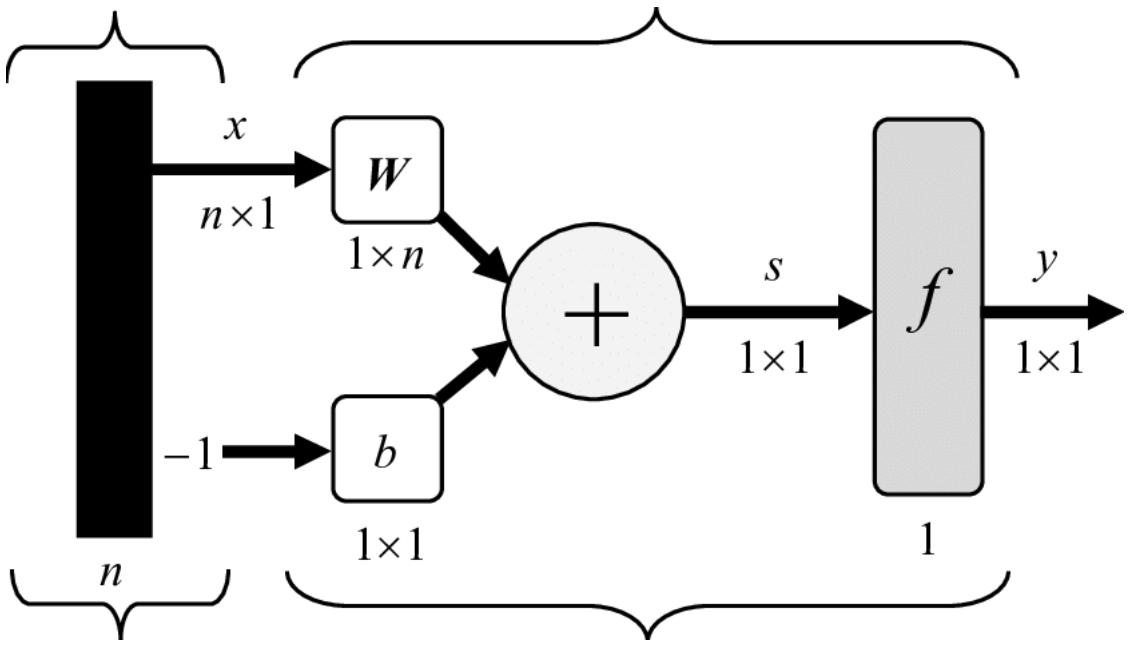

FIGURE 1.2: Schéma du neurone formel [5]

1.3.1 Composants du neurone formel

Dans cette section nous allons présenter les principaux composants d'un neurone formel

- **Entrées :** les entrées peuvent provenir directement du système ou d'autres neurones.
- **Biais :** permet d'ajouter de la flexibilité au réseau, en permettant de varier le seuil de déclenchement du neurone.
- **Poids :** Facteurs multiplicateurs qui affectent l'influence de chaque entrée sur la sortie du neurone.

- **Noyau :** Intègre toutes les entrées et le biais et calcule la sortie du neurone selon une fonction d'activation.
- **Sortie :** Peut être soit, directement une des sorties du système ou distribuée vers d'autres neurones

1.3.2 A quoi sert un perceptron

Le perceptron simple sert comme classifieur linéaire binaire, si le problème est linéairement séparable. Il permet de classifier l'ensemble de données d'entrée en deux différentes classes [6].

Exemple :

$$f(x) = \begin{cases} +1 & si \quad < w, x > +b > 0 \\ -1 & sinon \end{cases} \tag{1.2}$$

ou :

w : le poids synaptique du neurone.

x : l'entrée du neurone.

b : Le biaie du neurone.

1.3.3 Interprétation géométrique

<w,x>+b=0 est l'équation d'un hyperplan qui sépare X en deux demi-espaces correspondants aux deux classes.

Exemple du AND

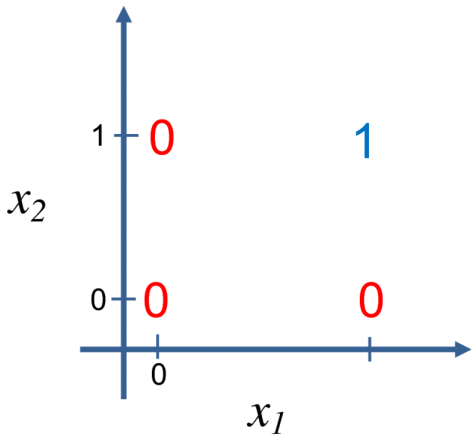

FIGURE 1.3: Schéma de l'opérateur AND [7]

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

TABLE 1.2: Table de vérité AND

D'après la figure (1.3), il est clair qu'on peut séparer l'ensemble en deux classes par un seul hyperplan, donc on peut déduire que l'opérateur "And" est linéairement séparable, alors il peut être discrétisé par un perceptron linéaire à seuil.

Exemple du XOR

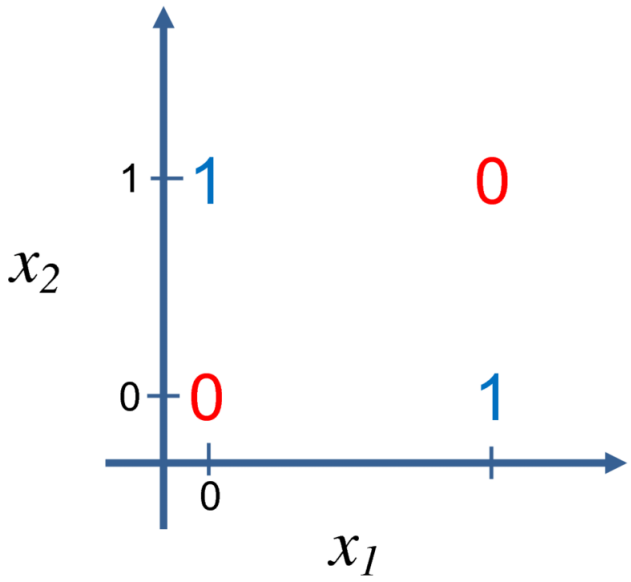

FIGURE 1.4: Schéma de l'opérateur Xor [7]

A	B	A Xor B
0	0	0
0	1	1
1	0	1
1	1	0

TABLE 1.3: Table de vérité Xor

On remarque que le Xor est non linéairement séparable car il est impossible de séparer les deux classes par un tenseur de dimension 1 (une droite). C'est cette limite du perceptron simple qui donne lieu au perceptron multicouches (MLP) qui fera l'objet de la section suivante.

1.4 PERCEPTRON MULTICOUCHE

En anglais « multi layer perceptron MLP» est un ensemble de neurones artificiels, organisés en plusieurs couches successives, au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement. Il s'agit donc d'un réseau de type feedforward [8], chaque couche est constituée d'un nombre variable de neurones, les neurones de la couche de sortie correspondent toujours aux sorties du système. La couche d'entrée reçoit les signaux (ou variables) d'entrée et la couche de sortie fournit les résultats.

Enfin, les neurones des autres couches (couches cachées) n'ont aucun lien avec l'extérieur et sont appelés neurones cachés [9]. La figure (1.5) représente le schéma d'un perceptron multi couches.

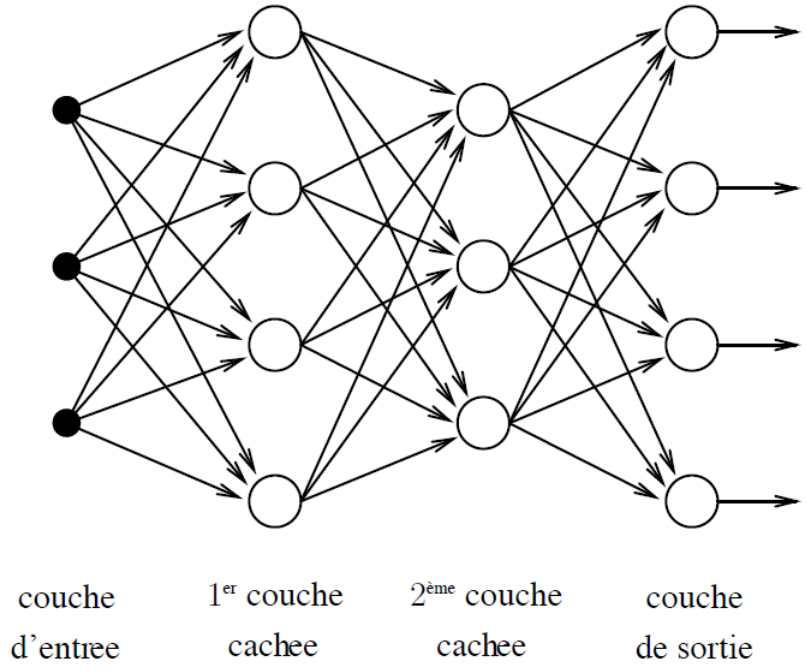

FIGURE 1.5: Schéma MLP [9]

1.5 FONCTION D'ACTIVATION

La fonction d'activation est une composante élémentaire des réseaux de neurones, elle prend en entrée la valeur du produit scalaire et le biais, et elle renvoie la valeur de la sortie. Les différents types de neurones se distinguent par la nature de leur fonction d'activation, les principaux types de ces fonctions sont :

- **Identité** :

$$f(x) = x \tag{1.3}$$

- **Seuil** :

$$f(x) = \begin{cases} 1 & si \quad x > 0 \\ 0 & si \quad x \leq 0 \end{cases} \tag{1.4}$$

- **Sigmoïde** :

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.5}$$

- **Tangente Hyperbolique** :

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{1.6}$$

– **ReLu** :

$$f(x) = \begin{cases} 0 & si \quad x < 0 \\ x & si \quad x \geq 0 \end{cases} \tag{1.7}$$

1.6 La propagation et la rétropropagation du gradient

C'est un algorithme qui permet de stabiliser un réseau à plusieurs couches. Cet algorithme a été trouvé, de façon indépendante, par plusieurs équipes des chercheurs (Rumelhart, Parker et LeCun). La topologie d'un tel réseau est donc formée de plusieurs couches de neurones, sans communication à l'intérieur d'une même couche. Le but est encore de minimiser l'erreur quadratique entre les sorties obtenues et celles souhaitées, qui correspond de nouveau à une descente du gradient.

Le principe est de redistribuer sur toutes les couches, y compris les couches cachées, une partie de l'erreur de manière récursive, en partant de la couche de sortie et en remontant vers la couche d'entrée [10]. La figure (1.6) montre le gradient de l'erreur totale du réseau.

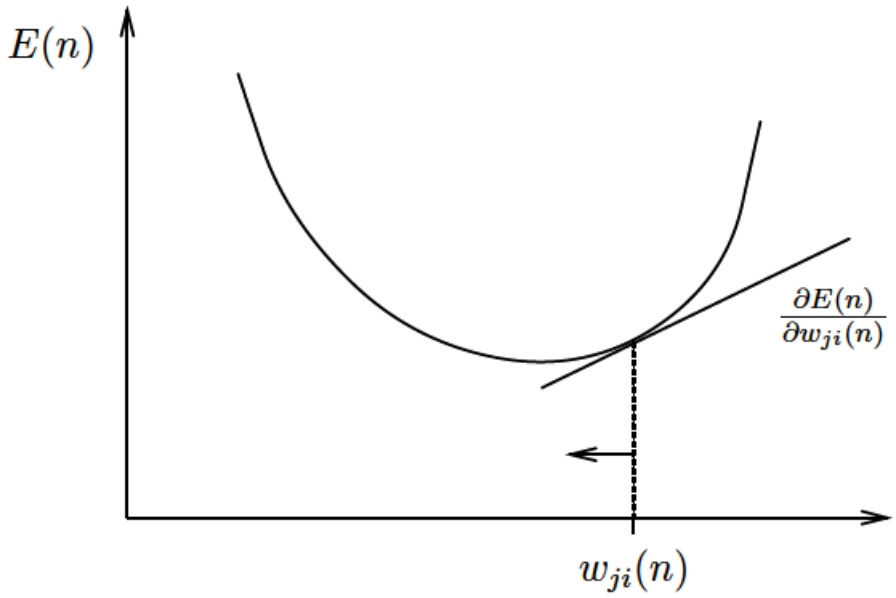

Figure 1.6: Gradient de l'erreur totale du réseau [10]

1.6.1 Formalisation de l'apprentissage

Pour un réseau multi-couches à n sorties, composé de plusieurs couches (couches cachées et couche de sortie), la somme de la j-éme unité est donnée par [10] :

$$S_j = \sum_{j=0}^{i=n} W_{ji} X_i + \theta_j \tag{1.8}$$

$$I_j = f\left(S_j\right) = f\left(\sum_{j=0}^{i=n} W_{ji} X_i + \theta_j\right) \tag{1.9}$$

ou :
W_i : poids de la connexion i^{me} entrée.
e_j : bais du neurone j
F : fonction d'activation de ce neurone j

L'objectif de la rétro-propagation est l'adaptation des paramètres W_{ij}, de telle façon à minimiser la fonction du coût donnée par :

$$E_p = \frac{1}{2} \sum_{k=1}^{T} E = \frac{1}{2} \sum_{k=1}^{T} (\delta)^2 \tag{1.10}$$

$$\delta_k = Y_k - R_k \tag{1.11}$$

Avec :
E_p : somme de l'erreur quadratique moyenne de moindres carrés
R_K : sortie actuelle du réseau
y_K : sortie désirée
T : longueur de l'ensemble d'apprentissage

1.6.2 Ajustement des poids

Après avoir calculer la sortie R_k et l'erreur E correspondantes à l'ensemble des entrées, à partir de l'équation associée, les poids du réseau sont alors ajustés par la méthode du gradient [10] :

$$W_{ji}^L(n+1) = W_{ji}^L(n) + \Delta W_{ji}^L(n) \tag{1.12}$$

$$\Delta W_{ji}^{L}(n) = -\mu \frac{\delta E}{\delta W_{ji}^{L}(n)} \tag{1.13}$$

Avec :
μ : pas d'apprentissage représentant la vitesse de convergence, dont la valeur est généralement choisie expérimentalement ($0 < \mu < 1$)
Si μ est trop petit, la convergence est lente mais la direction de la descente est optimale.
Si μ est trop grand, la convergence est rapide mais la précision est médiocre.

L'application des poids du réseau est faite, tout d'abord, pour la couche de sortie puis pour les couches cachées
on a :

- Pour la couche de sortie :

$$W_{kj}^{R}(n+1) = W_{kj}^{R}(n) + \mu \delta_{k}^{R} I_{j} \tag{1.14}$$

- Pour la couche cachée :
  On note que chaque adaptation des poids dans la couche cachée dépend de l'erreur totale de la couche de sortie, ce qui conduit à la notion de rétro-propagation.
  L'équation d'adaptation des poids dans ce cas est :

$$W_{ji}^{l}(n+1) = W_{ji}^{i}(n) + \mu \delta_{j}^{l} X_{i} \tag{1.15}$$

Toutes ces étapes sont résumées dans le pseudo-algorithme

Algorithm 1 propagation et rétropropagation

1-Initialiser les poids W_{ij} et les biais des neurones à des petites valeurs aléatoires.
2-Présenter le vecteur d'entrée et de sortie désirés.
3-Calculer La somme des entrées des neurones de la couche cachée
3-Calculer Les sorties de neurones de la couche cachée
4-Calculer La somme des entrées des neurones de la couche de sortie
5-Calculer Les sorties des réseaux
6-Calculer l'erreur pour les neurones de la couche de sortie
7-Réinjecter l'erreur de sortie
8-Ajuster Les poids de la couche de sortie
9-Ajuster Les poids de la couche cachée
10-Calculer l'erreur E
if $E - E_p < \varepsilon$ **then**
  FIN
else
  étape 6
end if

1.7 Conclusion

Dans ce chapitre, les principales notions des réseaux de neurones et leur apprentissage ont été présentés. En effet, le neurone n'est rien d'autres qu'une approximation formelle d'un neurone biologique. Le perceptron multi-couches est très efficace pour quelques problèmes. En revanche, il est limité pour d'autres.

Nous avons abordé, également, les principaux types de fonctions d'activation, les plus utilisées. Puis, nous avons expliqué les démarches de l'algorithme de propagation et rétropropagation du gradient et comment il corrige les poids synaptiques, lors de la phase de l'apprentissage du réseau de neurones.

Le chapitre suivant sera consacré à une description de l'apprentissage profond (Deep Learning en Anglais) et les différents types de réseaux de neurones profonds.