

Intelligence Artificielle



Pr. AIT LAHCEN
Printemps 2020

Réseaux de neurones

Introduction

Sous le terme de **réseaux de neurones** artificiels, on regroupe aujourd'hui un certain nombre de modèles dont **l'intention est d'imiter** certaines des fonctions **du cerveau humain** en reproduisant certaines de ses structures de base.

Les réseaux de neurones artificiels sont **composés d'éléments simples** (ou neurones) **connectés** entre eux.

Comme dans la nature, le fonctionnement d'un réseau de neurones est fortement **influencé par les connexions** des éléments entre eux.

Les **valeurs** de ces connexions (ou poids) sont **ajustées** durant une **phase d'entraînement**.

Introduction

Cette phase **dite d'apprentissage** permet aux réseaux de neurones de **réaliser des tâches complexes** dans différents types d'application (**classification, identification, reconnaissance de caractères, de la voix, vision, système de contrôle...**).

Les réseaux de neurones peuvent souvent apporter une **solution simple** à des problèmes encore **trop complexes** ne pouvant être résolus rapidement par les ordinateurs actuels (**puissance de calcul insuffisante**) ou par notre **manque de connaissances**.

La méthode **d'apprentissage** dite **supervisé** est souvent utilisée mais des techniques d'apprentissage **non supervisé** existent.

Introduction

Historique

Les réseaux de neurones ont une **histoire relativement jeune** (environ 60 ans).

Les applications intéressantes des réseaux de neurones n'ont vu le jour qu'il à une vingtaine d'année (**développement de l'informatique**).

En **1943**, **McCulloch et Pitts** étudièrent un ensemble de neurones formels interconnectés, et montrèrent leurs capacités **à calculer certaines fonctions logiques**.

En **1949**, **Hebb**, dans une **perspective psycho-physiologique**, souligne l'**importance du couplage synaptique** dans les **processus d'apprentissage** et présente une règle d'apprentissage.

De nombreux modèles de réseaux de neurones **s'inspirent**, aujourd'hui encore, de cette règle de Hebb.

Introduction

Historique

En 1958, Rosenblatt décrit le **premier modèle opérationnel** de réseaux de neurones mettant en œuvre les idées de Hebb, McCulloch et Pitts : **le Perceptron**.

C'est un réseau de neurones inspiré du système visuel. Il possède deux couches de neurones : une **couche de perception** et une **couche liée à la prise de décision**.

C'est le **premier** système artificiel capable d'apprendre par expérience.

Dans la même période, Le modèle de **Adaline** (ADaptive LInear NEuron) a été présenté par **Widrow et Hoff**. Ce modèle sera par la suite **le modèle de base** des réseaux de neurones multicouches.

Introduction

Historique

En 1969, deux mathématiciens, Minsky et Papert publient une étude montrant les limites du Perceptron.

Cela va avoir une grande incidence sur la recherche dans ce domaine. Elle va fortement diminuer jusqu'en 1972, où Kohonen présente ses travaux sur les mémoires associatives et propose des applications à la reconnaissance de formes.

Le renouveau actuel des réseaux de neurones est dû à des contributions originales, comme celles de Hopfield en 1982, qui en montrant l'analogie des réseaux de neurones avec certains systèmes physiques, a permis de leur appliquer un formalisme riche et bien maîtrisé.

Plus tard, en 1985, des nouveaux modèles mathématiques ont permis de dépasser les limites du Perceptron.

Le neurone biologique

Le neurone biologique

Le neurone

Le neurone est une cellule nerveuse qui est l'élément de base du système nerveux central.

Celui-ci en possèderait environ cent milliards.

Les neurones possèdent de nombreux points communs dans leur organisation générale et leur système biochimique avec les autres cellules.

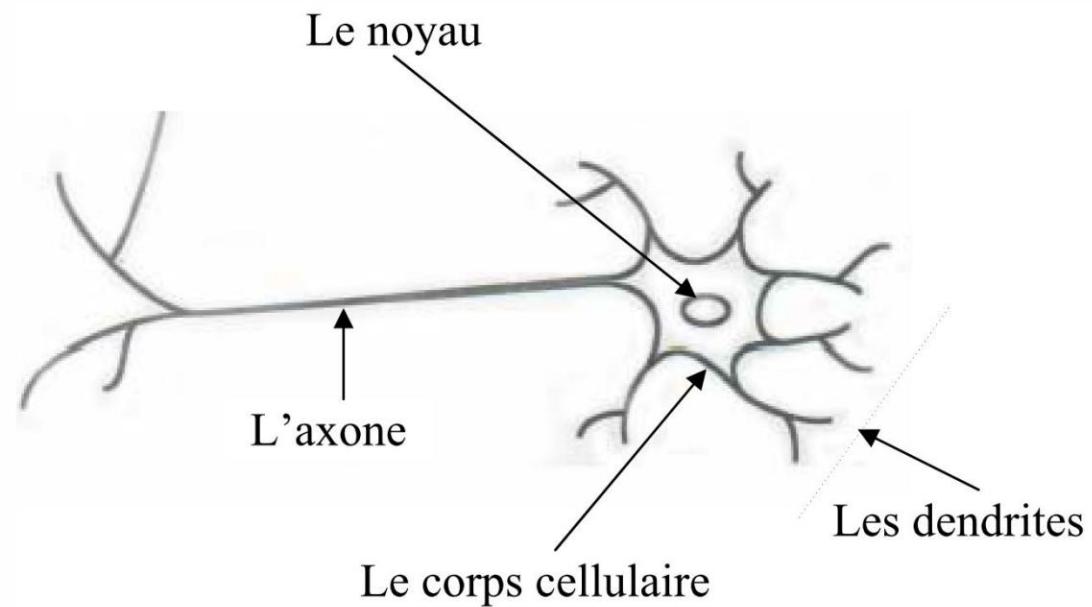
Ils présentent cependant des caractéristiques qui leur sont propres et ils assurent cinq fonctions spécialisées :

Le neurone biologique

Structure d'un neurone

Un neurone est constitué de :

- Le corps cellulaire,
- Le noyau
- Les dendrites,
- L'axone.



Le neurone biologique

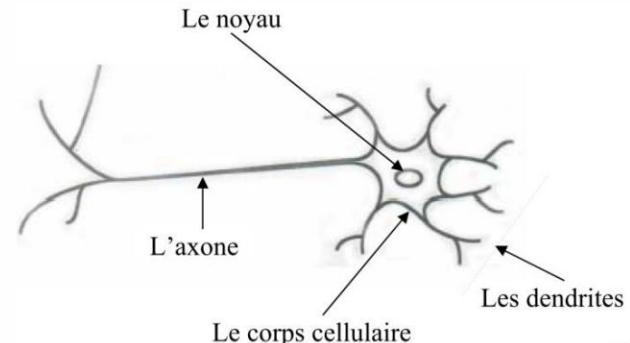
Structure d'un neurone

Le corps cellulaire :

Il contient le noyau du neurone et effectue les transformations biochimiques nécessaires à la synthèse des enzymes et des autres molécules qui assurent la vie du neurone. Sa forme dépend souvent de sa position dans le cerveau. Sa taille est de quelques microns (10^{-6} m) de diamètre.

Les dendrites :

Chaque neurone possède une "chevelure" de dendrites.. Elles sont les **récepteurs principaux** du neurone pour capter les signaux qui lui parviennent.



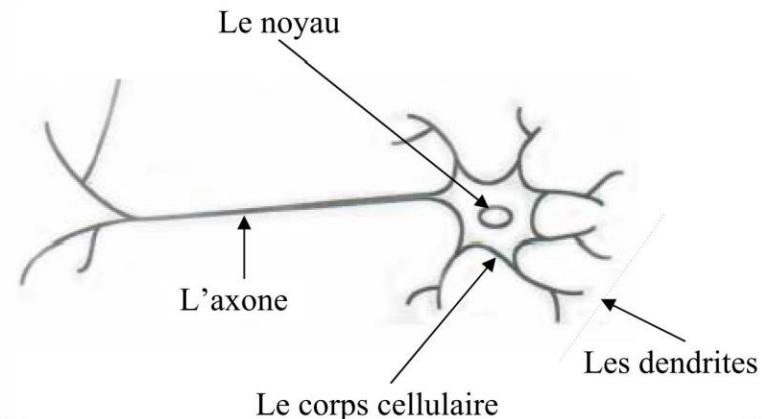
Le neurone biologique

Structure d'un neurone

L'axone :

L'axone, qui est à proprement parler la fibre nerveuse, sert de moyen de transport pour les **signaux émis** par le neurone.

Il se distingue des dendrites par sa forme et par les propriétés de sa membrane externe. En effet, il est généralement plus long que les dendrites, et se ramifie à son extrémité, là où il communique avec d'autres neurones, alors que les ramifications des dendrites se produisent plutôt près du corps cellulaire.

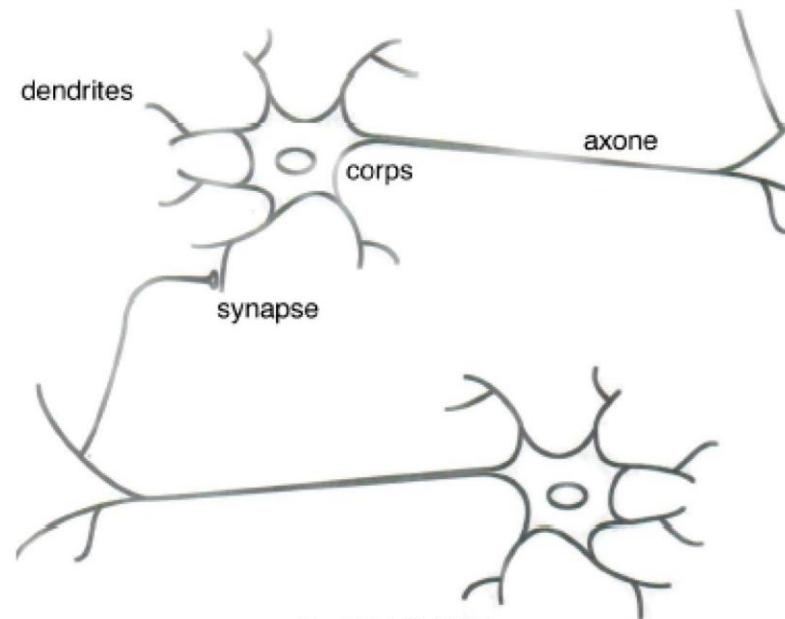


Le neurone biologique

Structure d'un neurone

Synapses :

Pour former le système nerveux, les neurones sont connectés les uns aux autres suivant des répartitions spatiales complexes. Les connexions entre deux neurones se font en des endroits appelés synapses où ils sont séparés par un petit espace synaptique de l'ordre d'un centième de microns.



Le neurone biologique

Fonctionnement des neurones

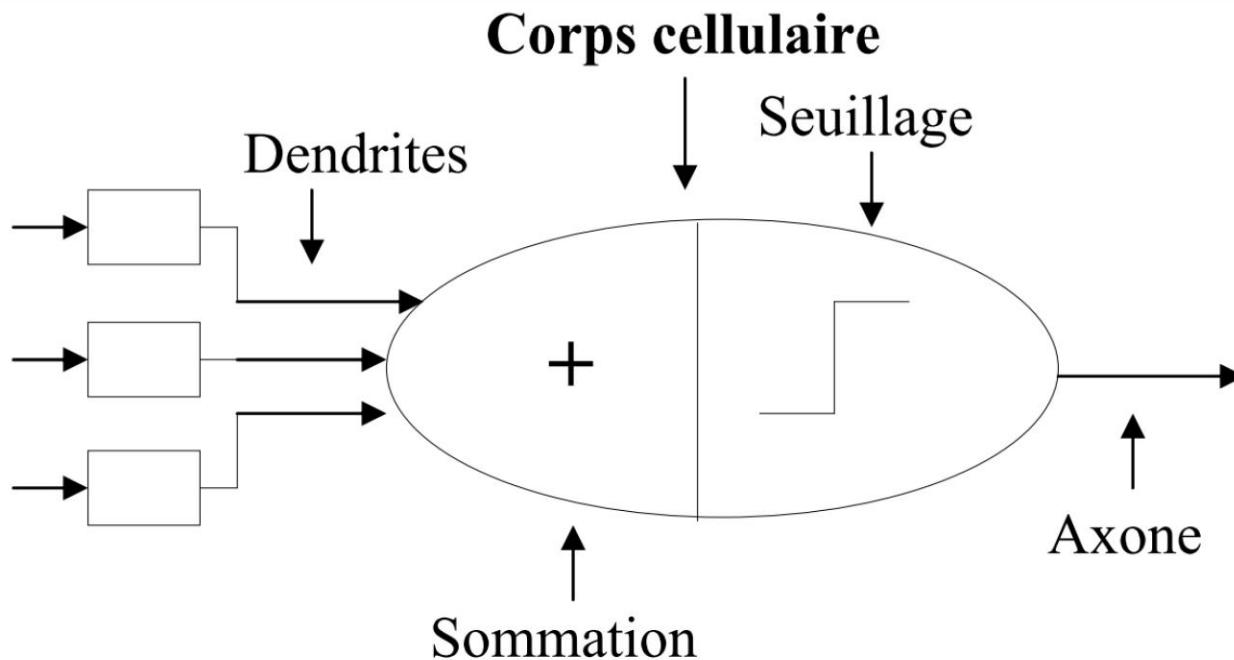
D'une façon simple, on peut dire que le neurone traite les courants électriques qui lui parviennent de ses dendrites, et qu'il transmet le courant électrique résultant de ce traitement aux neurones auxquels il est connecté par l'intermédiaire de son axone.

Le **schéma classique** présenté par les biologistes est celui d'une **cellule effectuant une sommation des influx** nerveux transmis par ses dendrites.

- Si la sommation dépasse un seuil, le neurone répond par un influx nerveux ou potentiel d'action qui se propage le long de son axone.
- Si la sommation est inférieure à un seuil, le neurone reste inactif.

Le neurone biologique

Fonctionnement des neurones



Le neurone biologique

Modélisations

Les débuts : Le modèle de Mac Culloch et Pitts (1943)

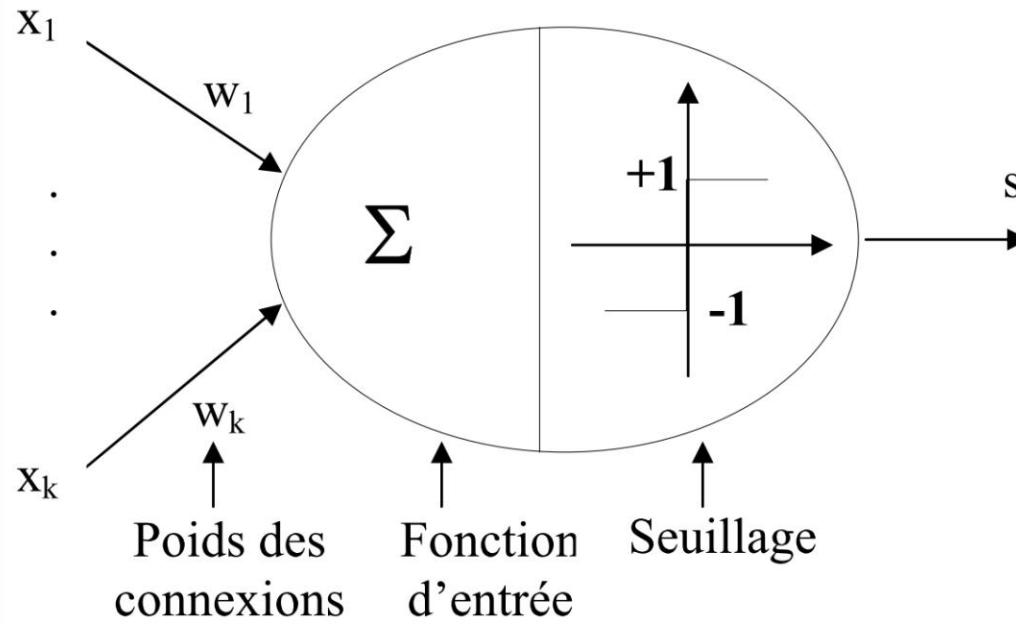
La première modélisation d'un neurone date des années quarante. Elle a été présentée par Mac Culloch et Pitts. S'inspirant de leurs travaux sur le neurone biologique, ils ont proposé le modèle suivant :

- Un neurone formel fait une somme pondérée des potentiels d'action qui lui parviennent (chacun de ces potentiels est une valeur numérique qui représente l'état du neurone qui l'a émis), puis s'active suivant la valeur de cette sommation pondérée.
- Si cette somme dépasse un certain seuil, le neurone est activé et transmet une réponse (sous forme de potentiel d'action) dont la valeur est celle de son activation.
- Si le neurone n'est pas activé, il ne transmet rien.

Le neurone biologique

Modélisations

Les débuts : Le modèle de Mac Culloch et Pitts (1943)



Le neurone biologique

Modélisations - Le neurone sommateur

Le neurone formel sommateur est constitué de deux opérateurs :

- Un opérateur de sommation qui élabore un "potentiel" p égal à la somme pondérée du neurone,
- Un opérateur qui calcule l'état de sortie " s " du neurone en fonction de son potentiel p .

Le neurone biologique

Modélisations - Le neurone sommateur

Les calculs du potentiel p et de la sortie s s'expriment par les relations suivantes :

$$\begin{aligned} p &= \sum_{i=1}^n W_i x_i \\ &= w_1 x_1 + w_2 x_2 + \dots + w_n x_n \end{aligned}$$

et $s = f(p)$

où w_i sont les poids synaptiques, p est la fonction d'entrée totale du neurone et f la fonction d'activation du neurone (ou la fonction de neurone).

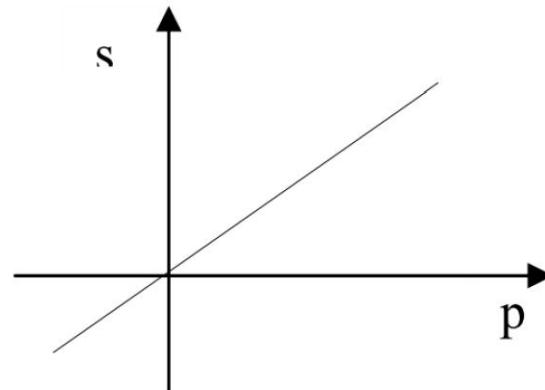
Le neurone biologique

Modélisations - Exemples de fonctions de neurone

La **fonction d'activation** du neurone est l'opérateur qui **définit son comportement**.

Elle **prend différentes formes** selon l'utilisation du réseau, et la nature continue ou discrète, voire binaire de l'état s du neurone. Elle présente en général un seuil. Ainsi, elle peut être :

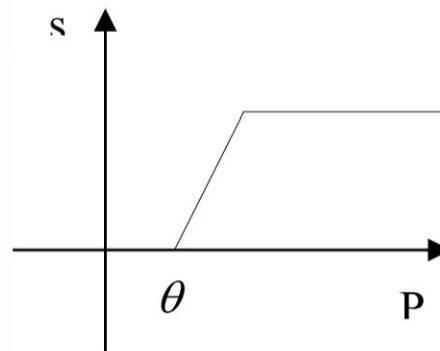
- Une **fonction linéaire** (identité) : $f(p)=p$. Dans ce cas le neurone est linéaire.



Le neurone biologique

Modélisations - Exemples de fonctions de neurone

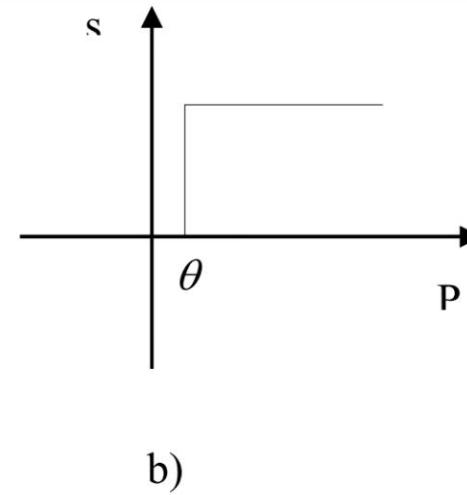
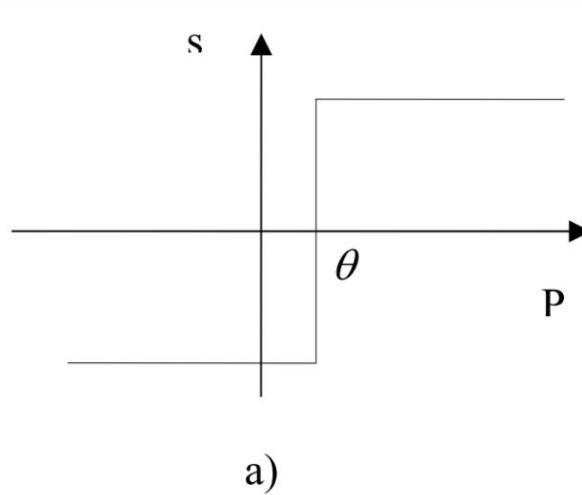
- Une fonction non linéaire (**saturation**), se rapprochant de la caractéristique entrée/sortie des **neurones réels**, décrite par
 - un seuil θ ,
 - une partie linéaire à pente constante A ,
 - et une valeur de saturation : $f(p) = \text{Min}[S_{\max}, \text{Max}(0, A.p)]$



Le neurone biologique

Modélisations - Exemples de fonctions de neurone

- Une fonction "signe" dont les valeurs de la sortie sont -1 et $+1$ ou une fonction "échelon" (seuil) dont les valeurs de la sortie sont 0 ou $+1$ avec éventuellement un seuil θ : $f(p)=\text{Sign}(p - \theta)$ ou $f(p)=U(p - \theta)$. Dans ce cas le neurone est binaire.



a)

b)

La fonction présentent des valeurs négatives. Or les fréquences moyennes ou instantanées de potentiel d'action d'un neurone biologique sont bien sûr des nombres positifs. Il s'agit là d'une liberté que l'on prend par rapport à la biologie.

Le neurone biologique

Modélisations - Exemples de fonctions de neurone

- Une fonction "sigmoïde" dont la forme générale est celle d'une tangente hyperbolique avec des valeurs comprises entre **-1 et +1** ou avec des valeurs comprises entre **0 et +1** :

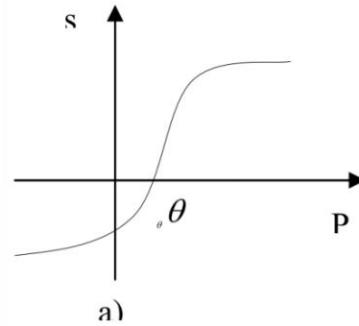
$$f(p) = \frac{e^{(\frac{p-\theta}{T})} - 1}{e^{(\frac{p-\theta}{T})} + 1}$$

ou

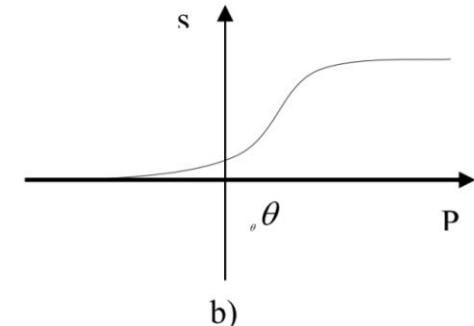
$$f(p) = \frac{e^{(\frac{p-\theta}{T})}}{e^{(\frac{p-\theta}{T})} + 1}$$

$$s \in]-1;+1]$$

$$s \in]0;+1]$$



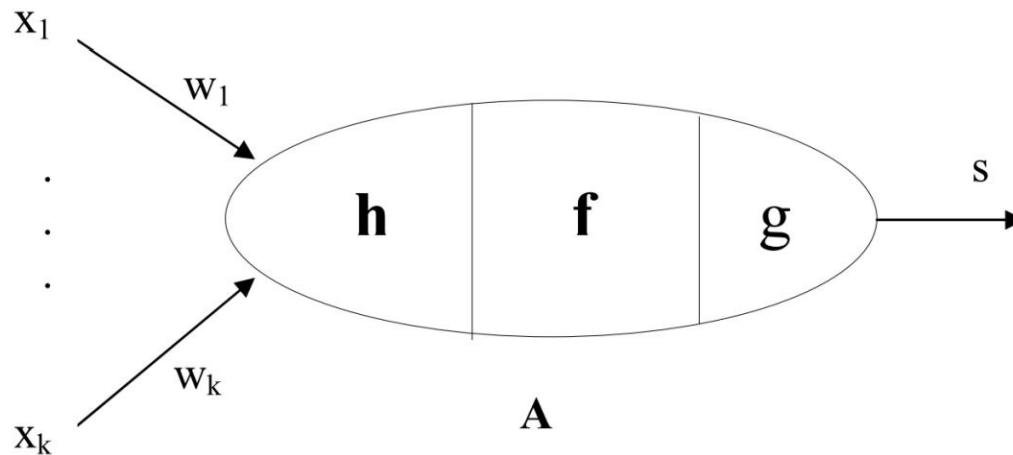
Pr. AIT LAHCEN



T est un paramètre appelé **température**. Quand il tend vers 0, la pente augmente et les fonctions "sigmoïde" tendront alors vers la fonction "signe" et "échelon" définies précédemment.

Le neurone biologique

Modélisations - Modèle général



x_i : entrées du neurone, A : activation du neurone, s : sortie du neurone

w_i : poids (synaptiques), h : fonction d'entrée, f : fonction d'activation (ou de transfert), g : fonction de sortie

$A = f\{h[(x_1, \dots, x_k), (w_1, \dots, w_k)]\}$ et $s = g(A)$ (= A le plus souvent)

La combinaison (h, f, g) définit le type de neurone

Le neurone biologique

Modélisations - Modèle général

D'une façon générale, on peut définir un neurone formel par les éléments suivant :

- La **nature des entrées** (x_1, x_2, \dots, x_n) et **de la sortie** (s) ; elles peuvent être binaires (± 1 ou 0 et 1) ou réelles ;
- La **fonction d'entrée totale** (h) qui définit le **prétraitement effectué** sur les entrées ; elle peut être booléenne, linéaire,...
- La **fonction d'activation** (ou d'état) (f) du neurone qui **définit son état interne** en fonction de son entrée totale ; elle peut être **binaire**, **discrète** ou **continue**.
- La **fonction de sortie** (g) qui **calcule la sortie** du neurone en fonction de son état d'activation.

Le neurone biologique

La structure des connexions

Il existe plusieurs types de connexions possible entre ces neurones. Parmi ces modèles nous étudierons le réseau monocouche et le réseau multicouches.

- Les réseaux à couches sont constitués d'une ou de plusieurs couches de neurones.
- Les neurones qui appartiennent à une même couche ne sont pas connectés entre eux, et chacune des couches recevant des signaux de la couche précédente et transmettant le résultat de ses traitements à la couche suivante (il n'y a pas de connexion "vers l'arrière").
- Les deux couches d'extrêmes correspondent à la couche qui reçoit ses entrées du milieu extérieur d'une part, et à la couche qui fournit le résultat des traitements effectués d'autre part.

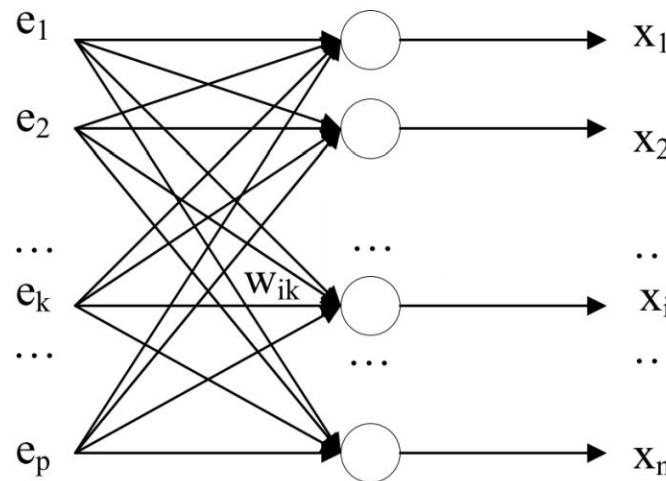
Réseaux monocouches

Réseau monocouche

Structure du réseau

Un réseau de neurone monocouche est constitué d'une couche de neurones (de sortie) connectés à un ensemble d'entrées par l'intermédiaire de connexions modifiables (poids) w_{ik} .

Les neurones ne sont pas interconnectés et reçoivent tous les mêmes informations provenant des entrées.



Réseau monocouche

Structure du réseau

L'évolution du réseau se fait en parallèle.

Tous les neurones calculent, en même temps, la somme des entrées pondérées par des coefficients (poids) et déterminent leur sortie.

Ces coefficients sont déterminés par un apprentissage.

Celui-ci consiste à calculer ces coefficients (poids) w_{ik} à l'aide d'un algorithme approprié, à partir d'un ensemble de forme à apprendre.

Réseau monocouche

La règle d'apprentissage du Perceptron

L'apprentissage **se fait avec superviseur**, c'est à dire que l'on **présente un vecteur à l'entrée du réseau et en même temps on fournit la sortie désirée** :

il s'agit de la classe désirée y_d .

Pour un seul neurone, à chaque pas d'apprentissage, l'erreur **quadratique E à minimiser** est donc :

$$E = (y - y_d)^2, \text{ erreur binaire}$$

Réseau monocouche

La règle d'apprentissage du Perceptron et d'Adaline

Rétro-propagation du gradient

Le **problème de l'apprentissage** dans les perceptrons est de **connaitre la contribution de chaque poids** dans l'erreur globale du réseau.

L'**algorithme de rétro-propagation** de l'erreur permet de faire cela.

- 1.. Propagation de l'entrée jusqu'à la sortie
- 2.. Calcul de l'erreur en sortie
- 3.. Rétro-propagation de l'erreur jusqu'aux entrées

Conditions

Il faut une fonction d'activation dérivable car on a besoin de la dérivé pour rétro-propager l'erreur.

Réseau monocouche

La règle d'apprentissage du Perceptron

Dans le cas du Perceptron les poids sont modifiés selon la règle :

$$\begin{aligned} w_{k+1} &= w_k + \Delta w_k \\ \Delta w_k &= -\alpha(y - y_d) * x_k \end{aligned}$$

où α est un gain positif d'adaptation (contrôle la vitesse d'adaptation).

Un calcul simple permet de justifier ces expressions. En effet, à partir de l'équation de l'erreur $E = (p - p_d)^2$, on a :

$$\frac{\partial E}{\partial w_k} = 2 * (p - p_d) * \frac{\partial p}{\partial w_k}$$

$$p = \sum_{j=1}^n w_j x_j \Rightarrow \frac{\partial p}{\partial w_k} = x_k,$$

$$\frac{\partial E}{\partial w_k} = 2 * (p - p_d) * x_k$$

D'où la proposition de Δw_k

Réseau monocouche

La règle d'apprentissage du Perceptron

1/ Initialisation des poids à des faibles valeurs choisies au hasard.

2/ Présentation d'une entrée $E_p = (e_1, \dots e_n)$ de la base d'apprentissage.

3/ Calcul de la sortie obtenue x pour cette entrée : $a = \sum (w_i \cdot e_i) - \theta$
 $x = \text{signe } (a), (\text{ si } a > 0 \text{ alors } x = +1 \text{ sinon } x = -1)$

4/ Si la sortie x du Perceptron est différente de la sortie désirée d_p pour cet exemple d'entrée E_p , alors modification des poids :

$$w_i(t+1) = w_i(t) + \mu \cdot ((d_p - x) \cdot e_i)$$

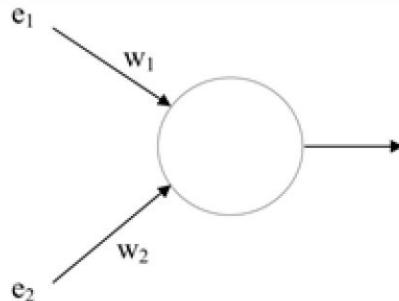
5/ Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement (i.e. modification des poids), retour à l'étape 2.

Réseau monocouche

La règle d'apprentissage du Perceptron

Exemple de fonctionnement de l'algorithme d'apprentissage du Perceptron

Base d'exemples d'apprentissage :



e ₁	e ₂	d	Exemple
1	1	1	(1)
-1	1	-1	(2)
-1	-1	-1	(3)
1	-1	-1	(4)

Conditions initiales : $w_1 = -0.2$, $w_2 = +0.1$, $\theta = 0,2$ ($\mu = +0.1$)

Réseau monocouche

La règle d'apprentissage du Perceptron

Exemple de fonctionnement de l'algorithme d'apprentissage du Perceptron

1/ Conditions initiales : $w_1 = -0.2$, $w_2 = +0.1$, $\theta = 0,2$ ($\mu = +0.1$)

2/ $a(1) = -0.2 + 0.1 - 0.2 = -0.3$

3/ $x(1) = -1$ (la sortie désirée $d(1) = +1$, d'où modification des poids)

$$4/ w_1 = -0.2 + 0.1 * (1 + 1) * (+1) = 0$$

$$w_2 = +0.1 + 0.1 * (1 + 1) * (+1) = +0.3$$

e ₁	e ₂	d	Exemple
1	1	1	(1)
-1	1	-1	(2)
-1	-1	-1	(3)
1	-1	-1	(4)

Réseau monocouche

La règle d'apprentissage du Perceptron

Exemple de fonctionnement de l'algorithme d'apprentissage du Perceptron

$$2/ a(2) = +0.3 - 0.2 = +0.1$$

3/ $x(2) = +1$ (Faux, d'où modification des poids)

$$4/ w_1 = 0 + 0.1 * (-1 - 1) * (-1) = +0.2$$

$$w_2 = +0.3 + 0.1 * (-1 - 1) * (+1) = +0.1$$

e ₁	e ₂	d	Exemple
1	1	1	(1)
-1	1	-1	(2)
-1	-1	-1	(3)
1	-1	-1	(4)

Réseau monocouche

La règle d'apprentissage du Perceptron

Exemple de fonctionnement de l'algorithme d'apprentissage du Perceptron

$$2-3/ a(3) = -0.2 - 0.1 - 0.2 = -0.5 \text{ Ok}$$

$$2-3/ a(4) = +0.2 - 0.1 - 0.2 = -0.1 \text{ Ok}$$

$$2-3/ a(1) = +0.2 + 0.1 - 0.2 = +0.1 \text{ Ok}$$

$$2-3/ a(2) = -0.2 + 0.1 - 0.2 = -0.1 \text{ Ok}$$

e ₁	e ₂	d	Exemple
1	1	1	(1)
-1	1	-1	(2)
-1	-1	-1	(3)
1	-1	-1	(4)

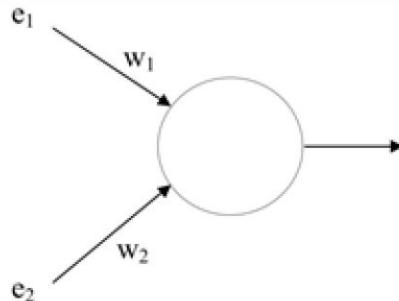
5/ Tous les exemples de la base ont été correctement traités, l'apprentissage est terminé

Réseau monocouche

La règle d'apprentissage du Perceptron

Exemple de fonctionnement de l'algorithme d'apprentissage du Perceptron

Base d'exemples d'apprentissage :



e_1	e_2	d	Exemple
1	1	1	(1)
-1	1	-1	(2)
-1	-1	-1	(3)
1	-1	-1	(4)

Nouvelles valeurs : $w_1 = +0.2$, $w_2 = +0.1$

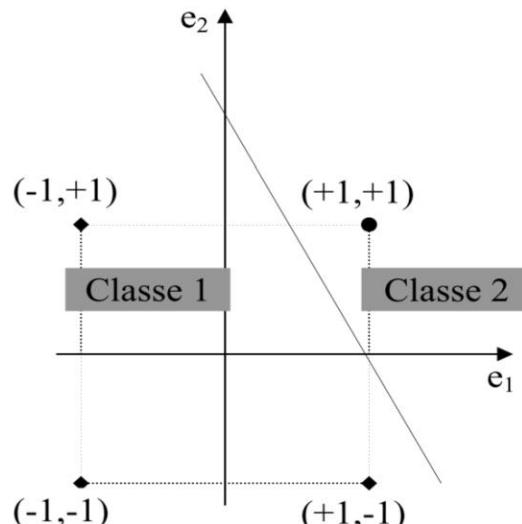
Réseau monocouche

La règle d'apprentissage du Perceptron

Exemple de fonctionnement de l'algorithme d'apprentissage du Perceptron

Le **Perceptron précédent** réalise une **partition** de son espace d'entrée en **2 classes (1 et 2)** selon la valeur de sa sortie (+1 ou -1).

L'équation de la droite séparatrice est : $w_1.e_1 + w_2.e_2 - \theta = 0$. Soit $e_2 = -2.e_1 + 2$



e_1	e_2	d	Exemple
1	1	1	(1)
-1	1	-1	(2)
-1	-1	-1	(3)
1	-1	-1	(4)

Réseau monocouche

Limites des réseaux monocouche : la séparation linéaire

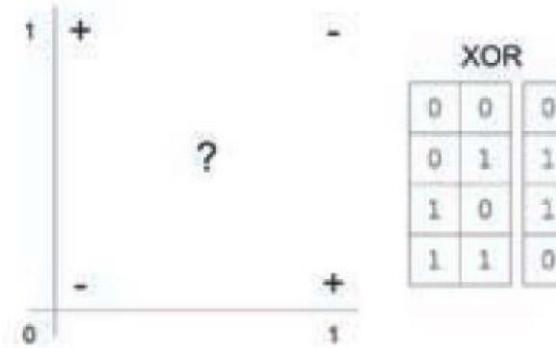
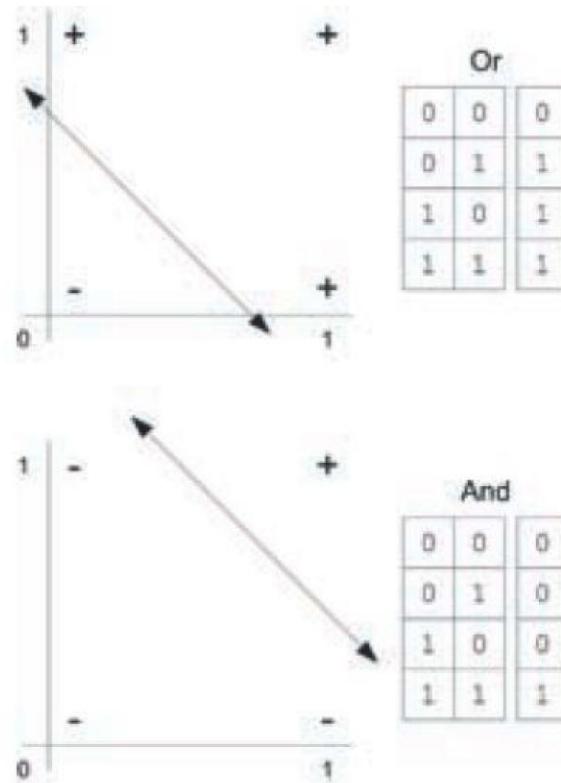
En effet, avec un réseau de neurones monocouche, il n'est possible de [distinguer que des objets linéairement séparables](#).

Pour montrer cela, considérons, la fonction ‘OU EXCLUSIF’ (XOR). Sa table de vérité est la suivante :

e_1	e_2	$s=e_1 \text{ OU EXCLUSIF } e_2$
0	0	0
0	1	1
1	0	1
1	1	0

Réseau monocouche

Limites des réseaux monocouche : la séparation linéaire



Réseau monocouche

La règle d'apprentissage du Perceptron

Exercice : Implémentez en C un perceptron pour apprendre la table du ET booléen ci-dessous. Les conditions initiales sont : $w_1 = 0.3$, $w_2 = -0.1$, $\theta = 0.2$, $\mu = 0.1$

Remarques :

- Il y a 5 étapes
- Vous pouvez utiliser des matrices et des vecteurs pour le stockage des variables.

	entrées		sortie désirée	poids initial		sortie obtenue	poids final	
Etapes	e1	e2	d	w1	w2	x	w1	w2
1	0	0	0	0.3	-0.1			
	0	1	0					
	1	0	0					
	1	1	1					
2	0	0	0					
	0	1	0					
	1	0	0					
	1	1	1					