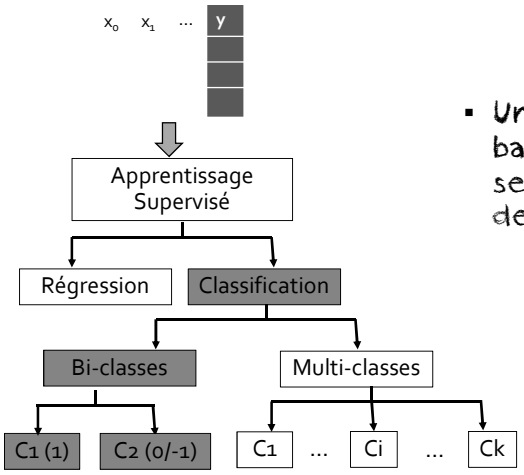
Réseau de neurones

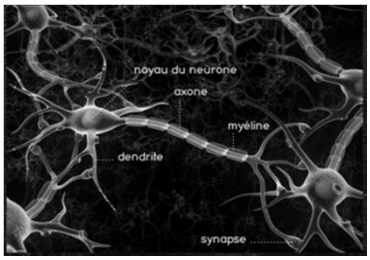
Un neurone

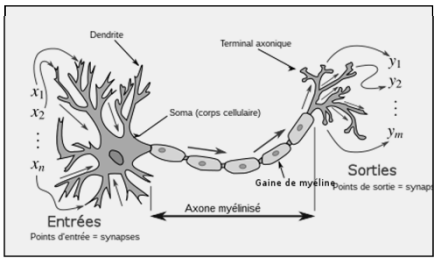
Introduction



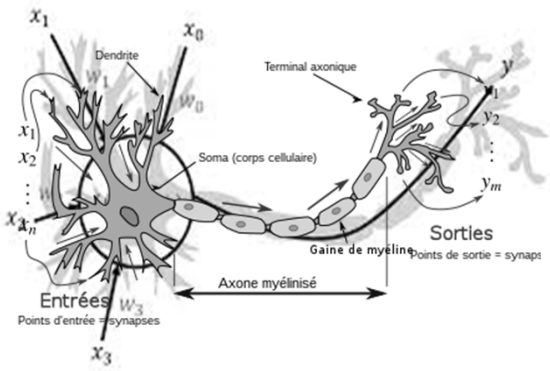
- Une technique de modélisation basée sur les réseaux de neurones sera utilisée dans toute la suite de ce cours

Neurone biologique et neurone artificiel

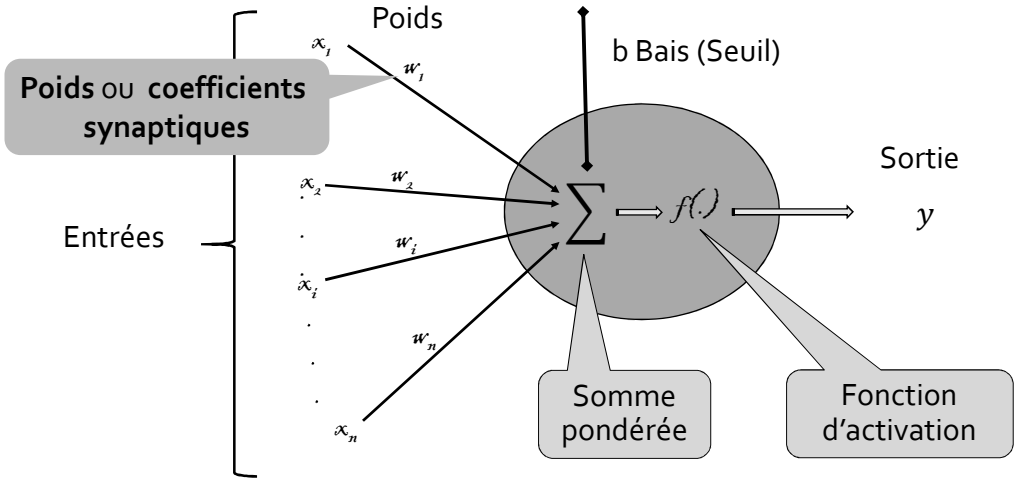




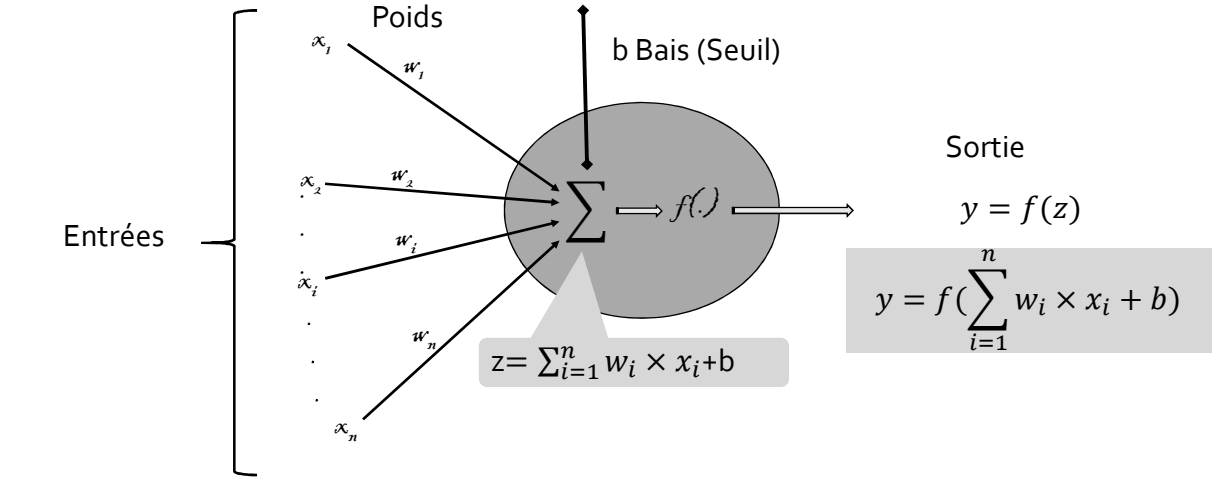
Neurone artificiel



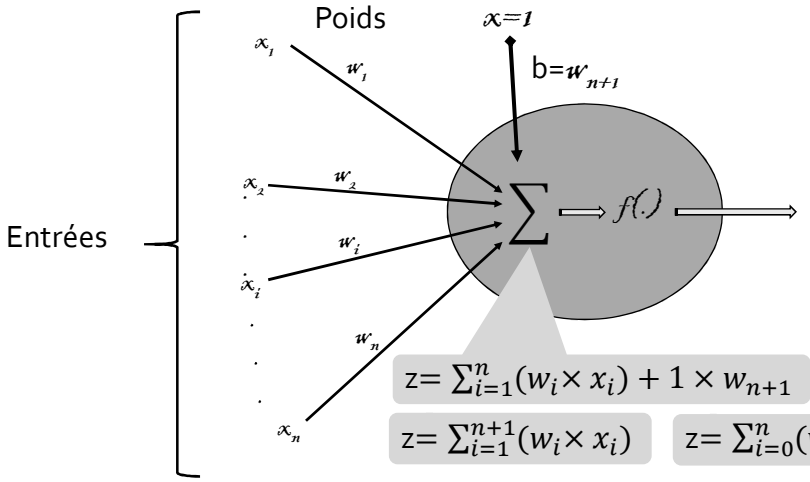
Neurone artificiel



Neurone artificiel



Neurone artificiel



$y = f(z)$

$$y = f(\sum_{i=0}^{n} w_i \times x_i)$$

$z = \sum_{i=1}^{n}(w_i \times x_i) + 1 \times w_{n+1}$

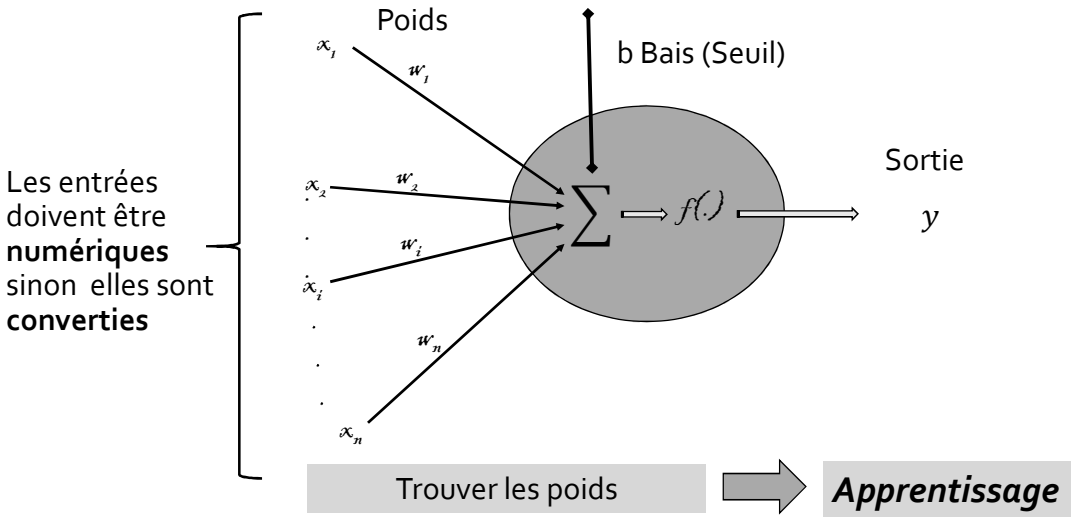
$z = \sum_{i=1}^{n+1}(w_i \times x_i)$

$z = \sum_{i=0}^{n}(w_i \times x_i)$

Fonction d'activation (Fonctions de transfert)

Nom de la fonction	Relation d'entrée/sortie	Icône
seuil	$a = 0$ si $n < 0$ $a = 1$ si $n \geq 0$	
seuil symétrique	$a = -1$ si $n < 0$ $a = 1$ si $n \geq 0$	
linéaire	$a = n$	
linéaire saturée	$a = 0$ si $n < 0$ $a = n$ si $0 \leq n \leq 1$ $a = 1$ si $n > 1$	
linéaire saturée symétrique	$a = -1$ si $n < -1$ $a = n$ si $-1 \leq n \leq 1$ $a = 1$ si $n > 1$	
linéaire positive	$a = 0$ si $n < 0$ $a = n$ si $n \geq 0$	
sigmoïde	$a = \frac{1}{1+\exp^{-n}}$	
tangente hyperbolique	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$	

Apprentissage



Apprentissage

Règles d'apprentissage

Il existe plusieurs Règles d'apprentissage:

- Règle de Hebb (Hebbian rule)
- Règle de Widrow-Hoff (Delta rule)
- Règle de Perceptron (Perceptron rule)
- Règle d'apprentissage Delta (Delta learning rule)

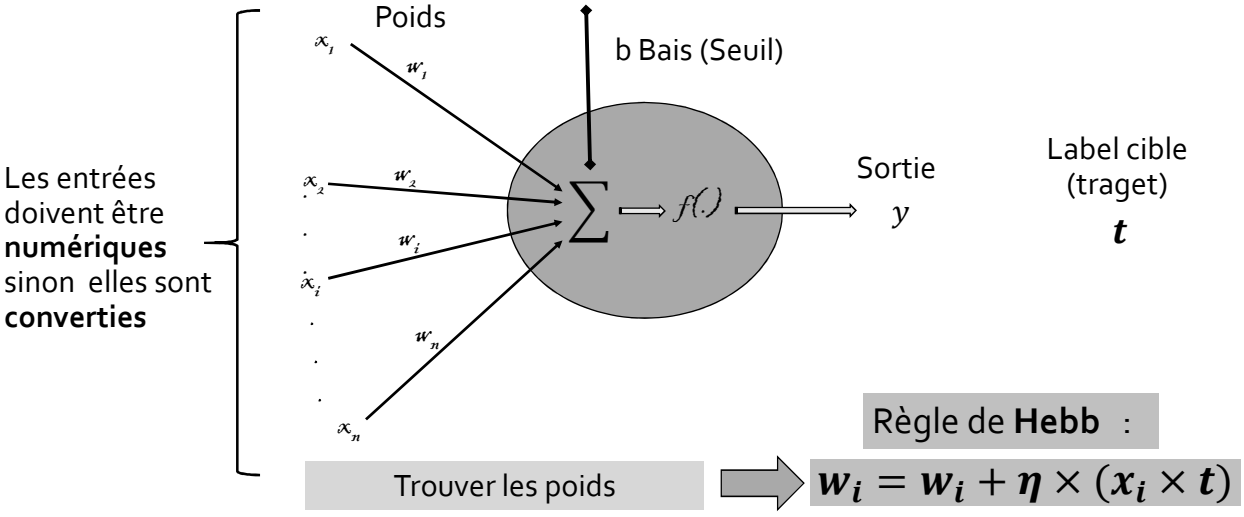
Apprentissage

Règles d'apprentissage

Il existe plusieurs Règles d'apprentissage:

- **Règle de Hebb (Hebbian rule)**
- Règle de Widrow-Hoff (Delta rule)
- Règle de Perceptron (Perceptron rule)
- Règle d'apprentissage Delta (Delta learning rule)

Règle de Hebb (Hebbian rule)



Règle de **Hebb** :

Trouver les poids ⇒ $w_i = w_i + \eta \times (x_i \times t)$

Règle de Hebb (Hebbian rule)

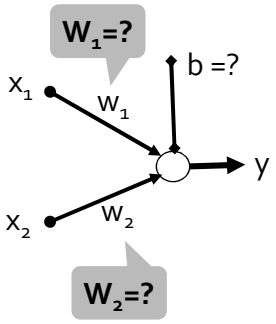
η : pas d'apprentissage (learning rate)
Macro paramètre

Algorithme :

- η est une constante positive.
- Initialisation des poids w_i et du seuil b à des valeurs (petites) choisies au hasard.
- Tant que'il y'a des exemples dans la base d'apprentissage à traiter :
  - Prendre une entrée X = (*x*1, … *x*n) de la base d'apprentissage
  - Calculer la sortie y du réseau pour l 'entrée X
    - $z = \sum(w_i * x_i) + b$
    - $y = f(z)$
  - Si $y \neq t$ (t traget la sortie désirée pour X)
    - **wij = wij + η ∗ (xi ∗ t)** (Modification des poids)
  - Fin Si
- Fin tant que

Règle de Hebb (Hebbian rule)

Exemple



x_1	x_2	t	
1	1	1	(1)
1	-1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)

La base d 'apprentissage (X, t) :

- X est le vecteur associé à l 'entrée (x1, …, xn)
- t la sortie correspondante souhaitée

Règle de **Hebb** :

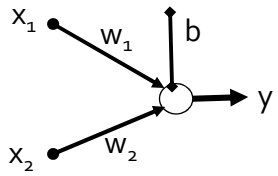
$$w_i = w_i + \eta \times (x_i \times t)$$

Règle de Hebb (Hebbian rule)

Exemple :

- Conditions initiales : η = +1, les poids et le seuil sont nuls.

$W_1=W_2=b=o$



x_1	x_2	t	
1	1	1	(1)
1	-1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)

Base d'apprentissage

Fonction d'activation= Seuil symétrique

$$f(z) = \begin{cases} -1 \ si \ z \leq 0 \\ 1 \ si \ z > 0 \end{cases}$$

Règle de Hebb (Hebbian rule)

x_1	x_2	t	
1	1	1	(1)
1	-1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)

$Z = O*1 + O*1=O$

Algorithme :

- η =1.
- $w1=w2$=b=0.
- Tant que il y'a des exemples dans la base d'apprentissage à traiter :
  - Prendre une entrée X = (x1, … xn) de la base d'apprentissage
  - Calculer la sortie y du réseau pour l 'entrée X
    - $z = \sum(w_i * x_i) + b$
    - $y = f(z)$
  - Si y ≠ t (t traget la sortie désirée pour X)
    - **wij = wij + η * (xi * t)** (Modification des poids)
  - Fin Si
- Fin tant que

Règle de Hebb (Hebbian rule)

x_1	x_2	t	
1	1	1	(1)
1	-1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)

$Z = 0*1 + 0*1 = 0$

$y = f(z) = f(0) = -1$

Fonction d'activation= Seuil symétrique

$$f(z) = \begin{cases} -1 \ si \ z \leq 0 \\ 1 \ si \ z > 0 \end{cases}$$

Algorithme :

- η =1.
- $w1 = w2$=b=0.
- Tant que il y'a des exemples dans la base d'apprentissage à traiter :
  - Prendre une entrée X = ($x1$, … xn) de la base d'apprentissage
  - Calculer la sortie y du réseau pour l 'entrée X
    - $z = \sum(w_i * x_i) + b$
    - $y = f(z)$
  - Si y ≠ t (t traget la sortie désirée pour X)
    - **wij = wij + η * (xi * t)** (Modification des poids)
  - Fin Si
- Fin tant que

Règle de Hebb (Hebbian rule)

x_1	x_2	t	
1	1	1	(1)
1	-1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)

$(y = -1) \neq (t=1)$

$\Rightarrow w1 = ?$

$w2 = ?$

Algorithme :

- η =1.
- $w1 = w2$=b=0.
- Tant que il y'a des exemples dans la base d'apprentissage à traiter :
  - Prendre une entrée X = ($x1$, … xn) de la base d'apprentissage
  - Calculer la sortie y du réseau pour l 'entrée X
    - $z = \sum(w_i * x_i) + b$
    - $y = f(z)$
  - Si y ≠ t (t traget la sortie désirée pour X)
    - **wij = wij + η * (xi * t)** (Modification des poids)
  - Fin Si
- Fin tant que

Règle de Hebb (Hebbian rule)

x_1	x_2	t	
1	1	1	(1)
1	-1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)

Algorithme :

- η =1.
- $w1=w2$=b=0.
- Tant que il y'a des exemples dans la base d'apprentissage à traiter :
  - Prendre une entrée X = (*x*1, … *x*n) de la base d'apprentissage
  - Calculer la sortie y du réseau pour l 'entrée X
    - $z = \sum(w_i * x_i) + b$
    - $y = f(z)$
  - Si y ≠ t (t traget la sortie désirée pour X)
    - **$w_{ij} = w_{ij} + \eta * (x_i * t)$** (Modification des poids)
  - Fin Si
- Fin tant que

$$w1 = w1 + \eta * (x1 * t)$$
$$=0+1*1*1= 1$$
$$w2 = w2 + \eta * (x2 * t)$$
$$=0+1*1*1= 1$$

Règle de Hebb (Hebbian rule)

- Le changement des poids est appliqué avec chaque vecteur des données d'apprentissage
- Lorsque tous les données d'apprentissage ont été parcourus une **epoch** d'apprentissage a été complétée,
- Lorsque tous les sorties du perceptron coïncident avec toutes les valeurs de sorties cibles on dit que le modèle a **convergé** vers une solution,
- Si la base d'apprentissage n'est pas linéairement séparable l'algorithme d'apprentissage du perceptron ne se terminera jamais.

Règle de Hebb (Hebbian rule)

x_1	x_2	t	
1	1	1	(1)
1	-1	1	(2)
-1	1	-1	(3)
-1	-1	-1	(4)

- **Exemple** (1) ⇒ w1 = w2= 1
- **Exemple** (2) : y = f(1.1 + 1.-1)=– 1
  y = – 1 ≠ 1 = t
  ⇒ w1 = w1 + e1 * x = 1 + 1.1= 2
  w2 = w2 + e2 * x = 1 + 1.-1 = 0
- **Exemples** (3) OK →est correctement traité :
  y = f(-2) = – 1 (la sortie est bonne).
- **Exemples** (4) OK
- **Exemples** (1) et (2) OK ⇒ STOP

Règle d'apprentissage du Perceptron (Perceptron learning rule)

Règles d'apprentissage

Il existe plusieurs Règles d'apprentissage:

- Règle de Hebb (Hebbian rule)
- Règle de Widrow-Hoff (Delta rule)
- **Règle de Perceptron (Perceptron rule)**
- Règle d'apprentissage Delta (Delta learning rule)

Règle d'apprentissage du Perceptron (Perceptron learning rule)

Algorithme :

- η est une constante positive.
- Initialisation des poids w_i et du seuil b à des valeurs (petites) choisies au hasard.
- Tant que il y'a des exemples dans la base d'apprentissage à traiter :
  - Prendre une entrée X = ($x1$, … xn) de la base d'apprentissage
  - Calculer la sortie y du réseau pour l 'entrée X
    - $z = \sum(w_i * x_i) + b$
    - $y = f(z)$
  - Si $y \neq t$ (t traget la sortie désirée pour X)
    - $\boldsymbol{w_i^t = w_i^{t-1} + \eta\ (\ t\ -\ y\ ) \times\ x_i}$
  - Fin Si
- Fin tant que

Règle d'apprentissage du Perceptron (Perceptron learning rule)

$$w_i^t = w_i^{t-1} + \Delta w_i^{t-1}$$

$$w_i^t = w_i^{t-1} + \overbrace{\eta\,(\,t\,-\,y\,)\times\,x_i}^{\Delta w_i^{t-1}}$$

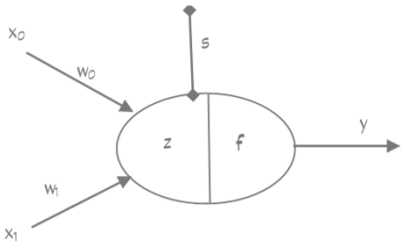
- η : Pas d'apprentissage (Learning rate)
- t : Résultat cible (Traget)
- y : Résultat du modèle
- x_i : Donnée d'apprentissage

Règle d'apprentissage du Perceptron (Perceptron learning rule)

Considérant le perceptron à un neurone de la Figure 1 et les données d'apprentissage du Tableau 1. En utilisant la fonction d'activation seuil et en initialisant

- η=1 et
- (w_0^0, w_1^0, S^0)=(0.282, 0.621, 0.307),

déterminer les valeurs de w0, w1 et S pour les couples (epoch, itération) suivants : (1, 2) ; (2, 3) et (4, 4). Indiquer pour chaque couple si le perceptron converge ou pas.



N°	x_0	x_1	t
1	-1	1	1
2	-1	-1	1
3	0	0	0
4	1	0	0