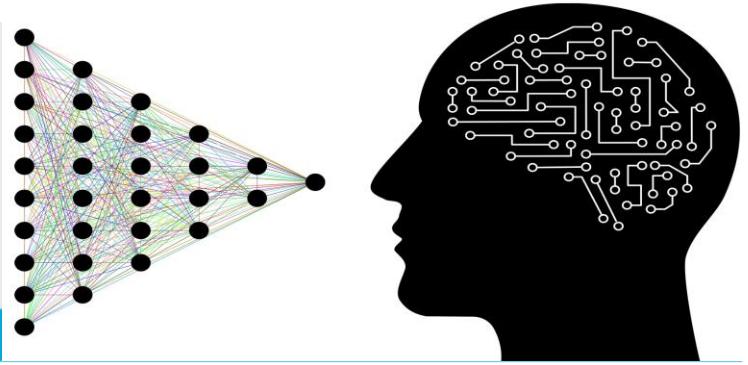
*Université Sidi Mohammed Ben Abdellah*
*Faculté des Sciences Dhar El Mahraz – Fès*

# Intelligence artificielle

## Réseaux de Neurones Artificiels (2)

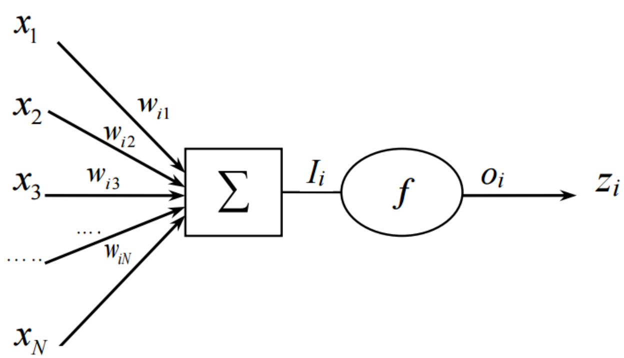


**Pr. EL BOURAKADI Dounia**
**Email: dounia.elbourakadi@usmba.ac.ma**

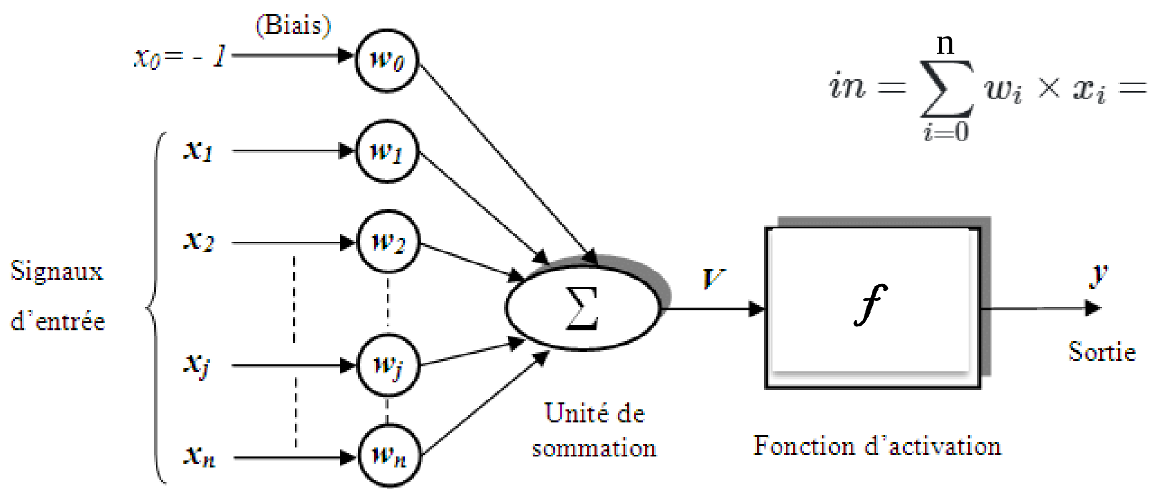
**Année universitaire 2023/2024**

# Neurone artificiel

La sortie d'un neurone artificiel est égale: $O_{\mathrm{i}} = f(I_{\mathrm{i}}) = f(\sum x_j w_{\mathrm{j}})$, avec $1 \leq \ \mathrm{j} \leq \mathrm{N}$




Il y a toutefois un poids supplémentaire appelé "coefficient du biais/seuil" $\boldsymbol{w_0}$; supposé lié à une information $\boldsymbol{x_0}$ = -1 (ou $\boldsymbol{x_0}$ = 1)

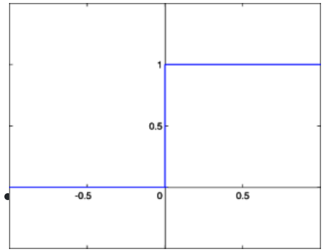


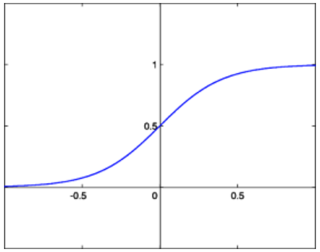

$$in = \sum_{i=0}^{n} w_i \times x_i = (\sum_{i=1}^{n} w_i \times x_i) - w_0$$

# Neurone artificiel

- Seuil des fonctions:

Le seuil de la fonction de Heaviside est en *x = 0* et vaut 1.




Le seuil de la fonction sigmoïde est en *x=0* et vaut 1/2.

- Seuil du neurone artificiel:

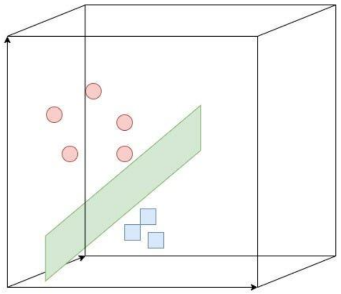
$$in = 0 \Leftrightarrow \sum_{i=0}^{n} w_i \times x_i = 0 \Leftrightarrow (\sum_{i=1}^{n} w_i \times x_i) - w_0 = 0 \Leftrightarrow \sum_{i=1}^{n} w_i \times x_i = w_0$$

- On atteint donc le seuil de la fonction d'activation lorsque la somme pondérée des informations d'entrée vaut le **coefficient de biais.**

De plus: $in \geq 0 \Rightarrow g(in) \geq seuil$

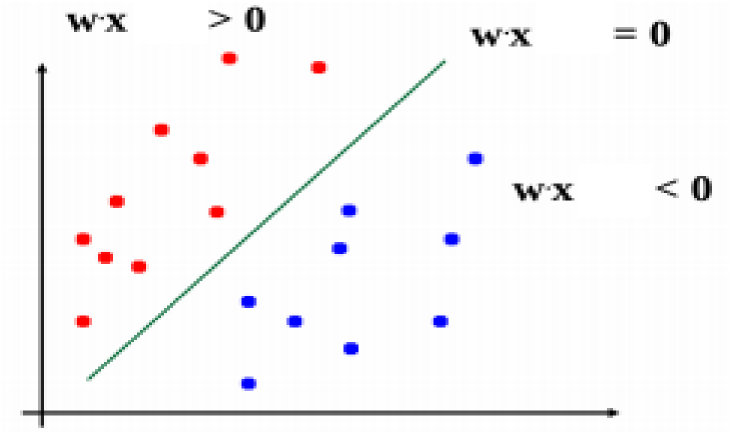
# Neurone artificiel

- Cette équation $in \geq 0 \Rightarrow g(in) \geq seuil$ définit un hyperplan qui sépare l'espace des informations d'entrées en deux parties.



- Les coordonnées seront alors ($x_1$, ..., $x_n$).

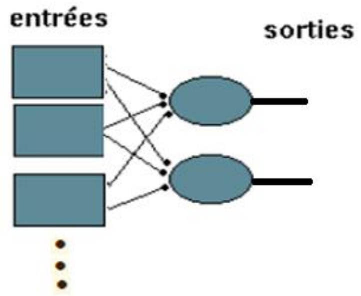
- En dimension 2, un hyperplan est par conséquent une droite.



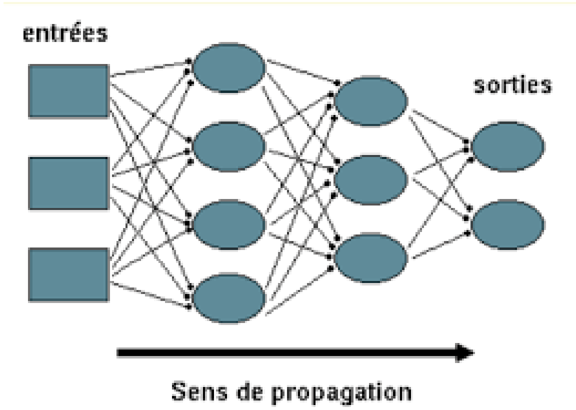
# Types des réseaux de neurones

Il existe deux grandes catégories des RN:

- Les réseaux monocouches: **Perceptron monocouche**




- Les réseaux multicouches: **Perceptron multicouche**; Multi-Layer Perceptron (MLP)

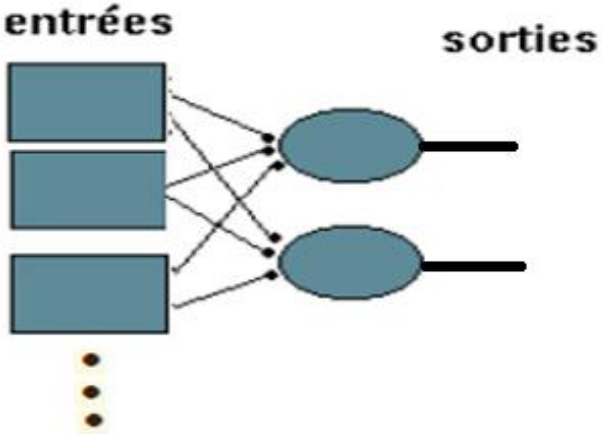


# Perceptron monocouche

Le perceptron est le RN le plus simple qui permet une classification binaire.

Il est décrit de la manière suivante:

- Il possède ***n*** informations en entrée et ***p*** neurones en sortie;
- Les neurones sont généralement alignés verticalement.
- Toutes les entrées sont directement connectées aux sorties.



# Perceptron monocouche

Pour la suite, on notera :

- **X** = $(x_i)_{1 <= i <= n}$ les **n** informations d'entrée ;
- $\boldsymbol{w_{i,j}}$ pour 1 <= i <= n et 1 <= j <= p, le poids reliant l'information $\boldsymbol{x_i}$ et le neurone ***j*** puis $\boldsymbol{y_j}$ l'**activation (sortie)** du *j*-ème neurone ;
- $\boldsymbol{w_{0,j}}$ le **coefficient de biais**, également appelé **seuil**, du *j*-ème neurone ;
- $\boldsymbol{in_j}$ la donnée d'entrée (somme pondérée) du *j*-ème neurone.

Pour définir notre perceptron:

- On commence par initialisation aléatoire des poids.
- Ensuite, on ajuste les poids selon un algorithme d'apprentissage jusqu'à la convergence du processus.

- Deux types de perceptron:
  1. **le perceptron à seuil**
  2. **le perceptron basé sur la descente du gradient**

# Perceptron monocouche à seuil

**Algorithme 1: Perceptron à seuil:**

**1**/ Initialisation des poids (y compris le bias) à des valeurs (petites) choisies au hasard.

**2**/ Présentation d'une entrée $X = (x_0, x_1, ..., x_n)$ de la base d'apprentissage.

**3**/ Calcul de la sortie obtenue ***y*** pour cette entrée : $x = (w_i \,.\, x_i)$

$y = f(x) =$ signe $(x)$ ( si $x \geq 0$ alors $y = +1$ sinon alors $y = 0$ )

**4**/ Si la sortie ***y*** du perceptron est différente de la sortie désirée $\boldsymbol{y_d}$ pour cet exemple d'entrée → alors modification des poids:

$\boldsymbol{w_i(t+1) = w_i(t) + \mu.\mathrm{Err}.\, x_i}$ où l'erreur : $\mathbf{Err} = \boldsymbol{y_d} - \boldsymbol{y}$.

et $\boldsymbol{\mu}$ est le pas de modification choisi entre 0 et 1.

**5**/ On répète de 2) à 4) jusqu'à la convergence du processus. On retient alors les derniers poids .

# Perceptron monocouche à seuil

## Convergence ?

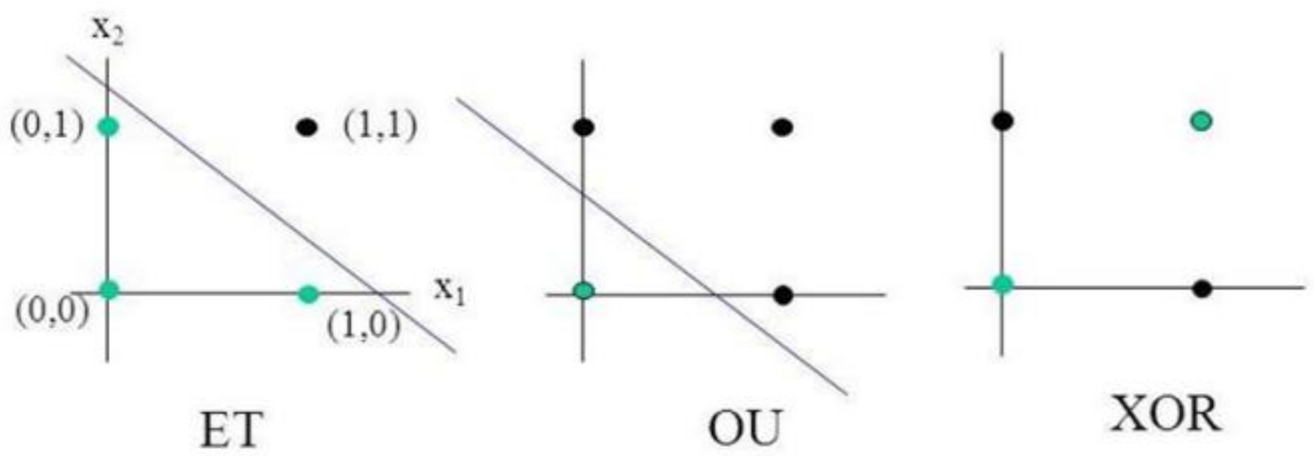
On peut terminer l’algorithme si le processus converge:

1. On fixe un nombre maximum d’itérations.
2. On fixe une erreur minimale à atteindre.
3. Plus aucune correction effectuée en passant toutes les données.
4. L’erreur globale ne diminue plus.
5. Les poids sont stables.

# Perceptron monocouche à seuil

On peut construire des perceptrons capables de réaliser les fonctions logiques : ET / OU. Cependant:

- Le perceptron est incapable de distinguer les patrons non linéairement séparables.
- Exemple: le OU Exclusif (XOR): Données non linéairement séparables !



# Perceptron monocouche à seuil

## Exercice 1: Perceptron qui calcule le ET logique

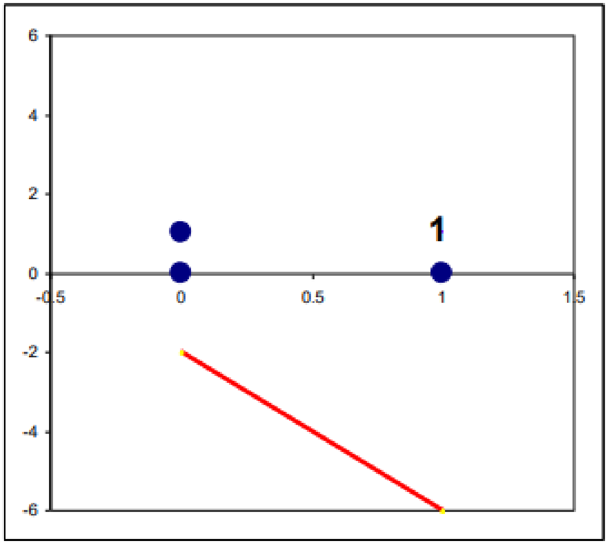
| X1 | X2 | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Données

Initialisation aléatoire des poids : $w_0 = 0.1$; $w_1 = 0.2$; $w_2 = 0.05$

On suppose $x_0 = 1$ et $\mu = 0,1$

$0.1 + 0.2\, x_1 + 0.05\, x_2 = 0 \Leftrightarrow x_2 = -4.0\, x_1 - 2.0$



# Perceptron monocouche à seuil

Observation à traiter

$$\begin{cases} x_0 = 1 \\ x_1 = 0 \\ x_2 = 0 \\ y = 0 \end{cases}$$

Appliquer le modèle

$$0.1\times1+0.2\times0+0.05\times0=0.1$$
$$\Rightarrow \hat{y}=1$$

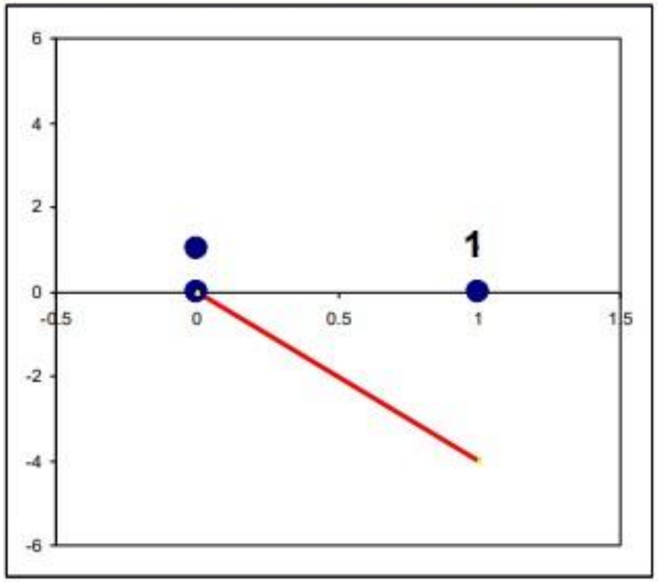
*Màj* des poids

$$\begin{cases} \Delta a_0 = 0.1\times(-1)\times1 = \boxed{-0.1} \\ \Delta a_1 = 0.1\times(-1)\times\boxed{0} = \boxed{0} \\ \Delta a_2 = 0.1\times(-1)\times\boxed{0} = \boxed{0} \end{cases}$$

Valeur observée de Y et prédiction ne matchent pas, une correction des coefficients sera effectuée.

Nouvelle frontière :

$$0.0+0.2x_1+0.05x_2=0 \Leftrightarrow x_2=-4.0x_1+0.0$$




Signal nul ($x_1$ = 0, $x_2$ = 0), seule la constante $a_0$ est corrigée.

| X1 | X2 | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Données

# Perceptron monocouche à seuil

Observation à traiter

$$\begin{cases} x_0 = 1 \\ x_1 = 1 \\ x_2 = 0 \\ y = 0 \end{cases}$$

Appliquer le modèle

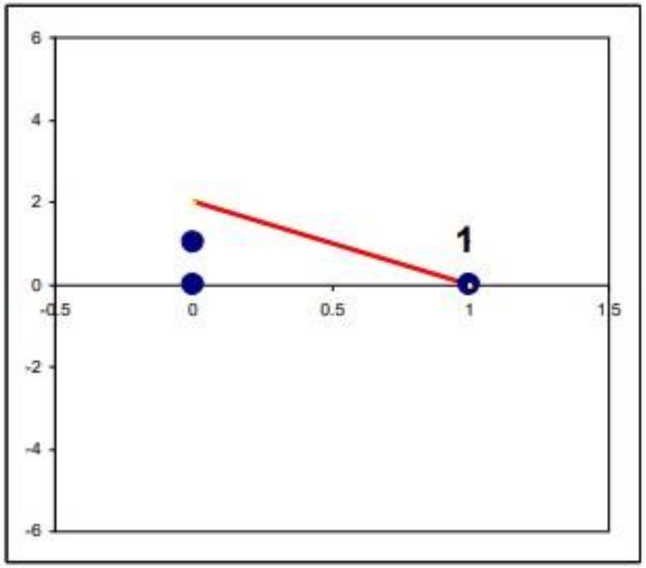
$$0.0 \times 1 + 0.2 \times 1 + 0.05 \times 0 = 0.2$$
$$\Rightarrow \hat{y} = 1$$

Màj des poids

$$\begin{cases} \Delta a_0 = 0.1 \times (-1) \times 1 = -0.1 \\ \Delta a_1 = 0.1 \times (-1) \times \boxed{1} = -0.1 \\ \Delta a_2 = 0.1 \times (-1) \times \boxed{0} = 0 \end{cases}$$

Nouvelle frontière :

$$-0.1 + 0.1 x_1 + 0.05 x_2 = 0 \Leftrightarrow x_2 = -2.0 x_1 + 2.0$$




| X1 | X2 | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Données

# Perceptron monocouche à seuil

Observation à traiter

$$\begin{cases} x_0 = 1 \\ x_1 = 0 \\ x_2 = 1 \\ y = 0 \end{cases}$$

Appliquer le modèle

$$-0.1\times 1 + 0.1\times 0 + 0.05\times 1 = -0.05$$
$$\Rightarrow \hat{y} = 0$$

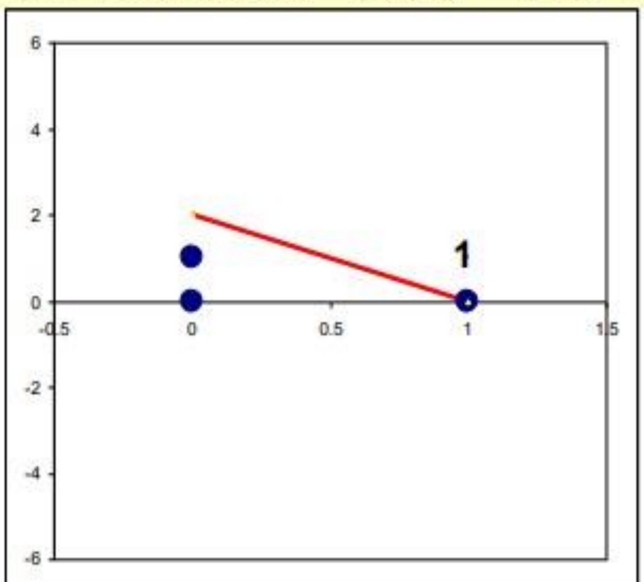
Màj des poids

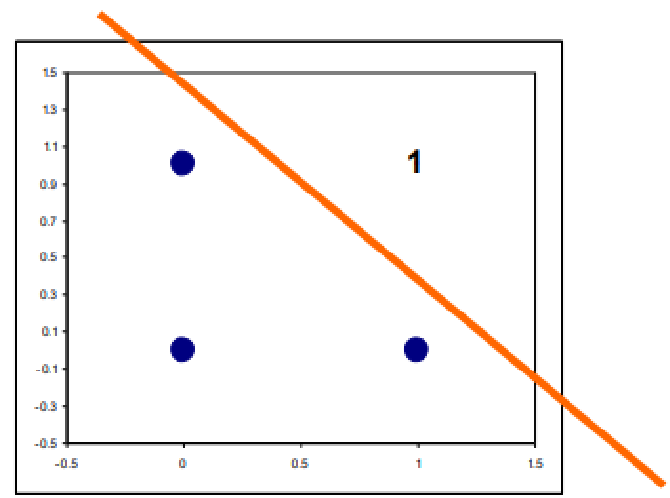
$$\begin{cases} \Delta a_0 = 0.1\times(0)\times 1 = 0 \\ \Delta a_1 = 0.1\times(0)\times 0 = 0 \\ \Delta a_2 = 0.1\times(0)\times 1 = 0 \end{cases}$$

Pas de correction ici ? Pourquoi ? Voir aussi la position du point par rapport à la frontière dans le plan !

Frontière inchangée :

$$-0.1 + 0.1x_1 + 0.05x_2 = 0 \Leftrightarrow x_2 = -2.0x_1 + 2.0$$




Représentation dans le plan

# Perceptron monocouche à seuil

**Exercice 2: Perceptron qui calcule le OU logique**

Appliquer l'algorithme du perceptron à seuil sur les données suivantes:

| $X_1$ | $X_2$ | $Y_d$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

On suppose:

- l'initialisation des poids: $w_0=0; w_1 = 1; w_2 = -1$;
- le taux d'apprentissage *1;*
- $y = f(x) = 1$ si $x > 0$ ; 0 sinon ;
- $X_0=1$

# Perceptron monocouche à seuil

**Exercice 2:**

| étape | $w_0$ | $w_1$ | $w_2$ | Entrée | $\Sigma_0^2\ w_i\ x_i$ | o | c | $w_0$ | $w_1$ | $w_2$ |
|---|---|---|---|---|---|---|---|---|---|---|
| init | | | | | | | | 0 | 1 | -1 |
| 1 | 0 | 1 | -1 | 100 | 0 | 0 | 0 | 0+0x1 | 1+0x0 | -1+0x0 |
| 2 | 0 | 1 | -1 | 101 | -1 | 0 | 1 | 0+1x1 | 1+1x 0 | -1+1x1 |
| 3 | 1 | 1 | 0 | 110 | 2 | 1 | 1 | 1 | 1 | 0 |
| 4 | 1 | 1 | 0 | 111 | 2 | 1 | 1 | 1 | 1 | 0 |
| 5 | 1 | 1 | 0 | 100 | 1 | 1 | 0 | 1+(-1)x1 | 1+(-1)x0 | 0+(-1)x0 |
| 6 | 0 | 1 | 0 | 101 | 0 | 0 | 1 | 0+1x1 | 1+1x0 | 0+1x1 |
| 7 | 1 | 1 | 1 | 110 | 2 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 111 | 3 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 1 | 1 | 100 | 1 | 1 | 0 | 1+(-1)x1 | 1+(-1)x0 | 1 +(-1)x0 |
| 10 | 0 | 1 | 1 | 101 | 1 | 1 | 1 | 0 | 1 | 1 |

Aucune entrée ne modifie le perceptron à partir de cette étape. Vous pouvez aisément vérifier que ce perceptron calcule le OU logique sur les entrées $x_1$ et $x_2$.

# Perceptron monocouche à seuil

**Exercice 3:** Apprentissage d'un ensemble linéairement séparable

| $X_1$ | $X_2$ | $Y_d$ |
|---|---|---|
| 0 | 2 | 1 |
| 2 | 0 | 0 |
| 1 | 1 | 1 |
| 3 | 0,5 | 0 |
| 1 | 2,5 | 1 |

On suppose:

- l'initialisation des poids: $w_0=0; w_1 = 0; w_2 = 0$; et taux d'apprentissage *1;*
- $y = f(x) = 1$ si $x > 0$ ; 0 sinon ;
- $X_0=1$

Quelle est la fonction obtenue par ce perceptron? Dessinez la droite frontière et montrer qu'elle sépare correctement les deux classes définies par cette fonction.

# Perceptron monocouche à seuil

**Exercice 3:** Apprentissage d'un ensemble linéairement séparable

| étape | $w_0$ | $w_1$ | $w_2$ | Entrée | $\Sigma_0^2\, w_i\, x_i$ | o | c | $w_0$ | $w_1$ | $w_2$ |
|---|---|---|---|---|---|---|---|---|---|---|
| init | | | | | | | | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | (1,0,2) | 0 | 0 | 1 | 1 | 0 | 2 |
| 2 | 1 | 0 | 2 | (1,2,0) | 1 | 1 | 0 | 0 | -2 | 2 |
| 3 | 0 | -2 | 2 | (1,1,1) | 0 | 0 | 1 | 1 | -1 | 3 |
| 4 | 1 | -1 | 3 | (1,3,0.5) | -0.5 | 0 | 0 | 1 | -1 | 3 |
| 5 | 1 | -1 | 3 | (1,1,2.5) | 7.5 | 1 | 1 | 1 | -1 | 3 |

Aucune entrée ne modifie le perceptron à partir de cette étape car ce perceptron classifie correctement tous les exemples de la base d’apprentissage. Le perceptron de sortie associe la classe 1 aux couples ($X_1$, $X_2$) tels que $X_2 > X_1/3 - 1/3$

# Perceptron monocouche à seuil

**Exercice 4:**

Appliquer l'algorithme du perceptron à seuil sur les données d'apprentissage suivantes:

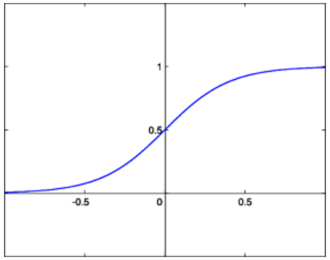
| $X_1$ | $X_2$ | $Y_d$ |
|---|---|---|
| 2 | 0 | 1 |
| 0 | 3 | 0 |
| 3 | 0 | 0 |
| 1 | 1 | 1 |

On suppose l'initialisation des poids: $w_0=0,5$; $w_1 = w_2 = 0$; et le taux d'apprentissage *0,1.*

# Perceptron monocouche à sigmoïde

C'est le perceptron qui utilise la fonction sigmoïde comme fonction d'activation.

Cette fonction possède plusieurs propriétés qui la rendent intéressante comme fonction d'activation:




- Elle n'est pas polynomiale;
- Elle est monotone, continue et dérivable partout;
- Elle renvoie des valeurs dans l'intervalle [0 , 1];
- On peut aussi calculer sa dérivée en un point de façon très efficace à partir de sa valeur en ce point. Ceci réduit le temps calcul nécessaire à l'apprentissage d'un réseau de neurones: $g'(x) = g(x)(1 - g(x))$ .

# Perceptron monocouche à sigmoïde

- L'idée qui sous-tend le perceptron à sigmoïde consiste à ajuster les poids du réseau pour **minimiser la mesure de l'erreur** sur l'ensemble d'apprentissage.

- Pour un seul exemple d'apprentissage avec un vecteur entrée ***x*** et une sortie vraie ***y***, la mesure « classique » de l'erreur est l'**erreur quadratique** qui s'écrit comme suit:

$$E = \frac{1}{2} Err^2 = \frac{1}{2}\left(y - g_W(\mathrm{x})\right)^2,$$

- On peut utiliser **la descente de gradient** pour réduire l'erreur quadratique en calculant la dérivée partielle de ***E*** par rapport à chaque **poids**. On a :

$$\frac{\partial E}{\partial W_j} = Err \times \frac{\partial Err}{\partial W_j}$$

**Remarque**: **si** f = $u^n$ **donc** $f' = (n\, u^{n-1}) \,.\, u' = n \,.\, u^{n-1} \,.\, u'$

# Perceptron monocouche à sigmoïde

- **Démonstration:**

$$\frac{\partial E}{\partial W_j} = Err \times \frac{\partial Err}{\partial W_j}$$

$$= Err \times \frac{\partial}{\partial W_j}\left(y - g_W(x)\right)$$

$$= Err \times \frac{\partial}{\partial W_j}\left(y - g\left(\sum_{j=0}^{n} W_j x_j\right)\right)$$

$$= -Err \times g'\left(\sum_{j=0}^{n} W_j x_j\right) \times x_j$$

$$= -Err \times g'(in) \times x_j$$

**Remarque**: **Si** f(x)=g(ax) **donc** *f ′(x)=a .g′(ax)*

- Lorsqu'on veut réduire ***E***, on actualise les poids comme suit :

$$W_j = W_j - \alpha \frac{\partial E}{\partial W_j}$$

$$W_j = W_j + \alpha \times Err \times g'(in) \times x_j$$

# Perceptron monocouche à sigmoïde

**Algorithme 2: Perceptron à sigmoide basé sur la descente du gradient:**

**1/** Initialisation des poids (y compris le bias) à des valeurs (petites) choisies au hasard.

**2/** Présentation d'une entrée $X = (x_0, x_1, \ldots, x_n)$ de la base d'apprentissage.

**3/** Calcul de la sortie obtenue $y$ pour cette entrée : $x = (w_i \cdot x_i)$

$$y = g(x) = \frac{1}{1+e^{-x}}$$

**4/** Calculer l'erreur par: **Err=** $y_d - y = y_d - g(x)$

Modifier les poids par: $w_i(t+1) = w_i(t) + \alpha.\mathrm{Err}.\, g'(x).\, x_i$

et $\alpha$ est le pas de modification choisi entre 0 et 1.

**5/** On répète jusqu'à la convergence du processus. On retient alors les derniers poids .