



Université Constantine 2
جامعة قسنطينة 2

Modélisation et simulation des systèmes complexes

Chapitre 5 Les réseaux de neurones artificiels

Pr Salima Ouadfel

Salima.ouadfel@univ-constantine2.dz

Etudiants concernés

Faculté/Institut	Département	Niveau	Spécialité
Nouvelles technologies	IFA	Master1	STIC



Université Constantine 2
جامعة قسنطينة 2

Modélisation et simulation des systèmes complexes

Chapitre 5 Les réseaux de neurones artificiels

Pr Salima Ouadfel

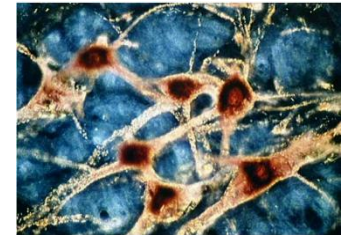
Faculté des NTIC

Salima.ouadfell@univ-constantine2.dz



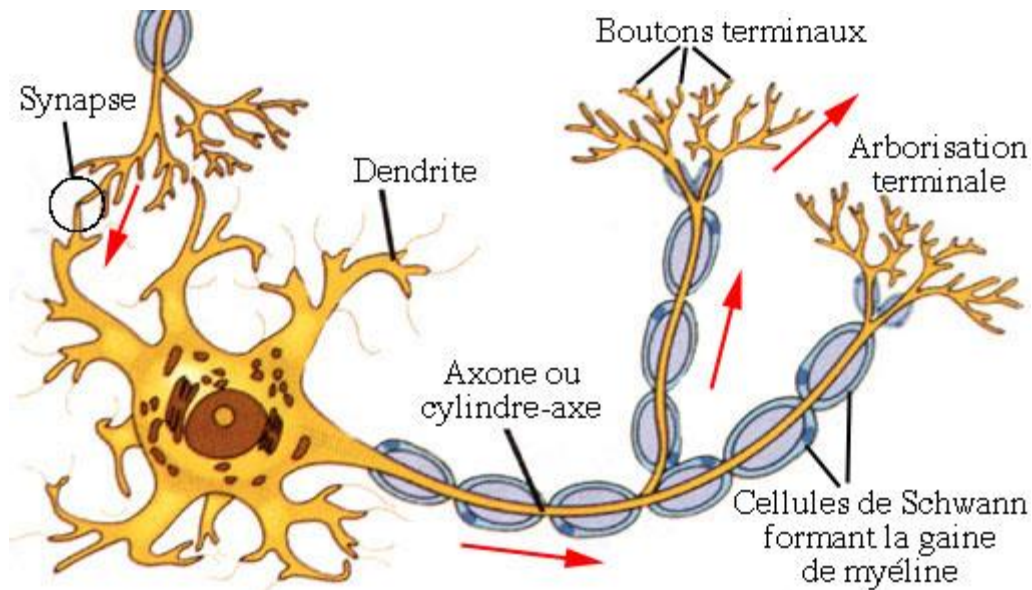
Inspiration: Les réseaux de neurones naturels

- Robuste et tolérant aux fautes
- Adaptable
- S'accommode aux données bruitées
- Massivement parallèle
- Capable d'apprentissage

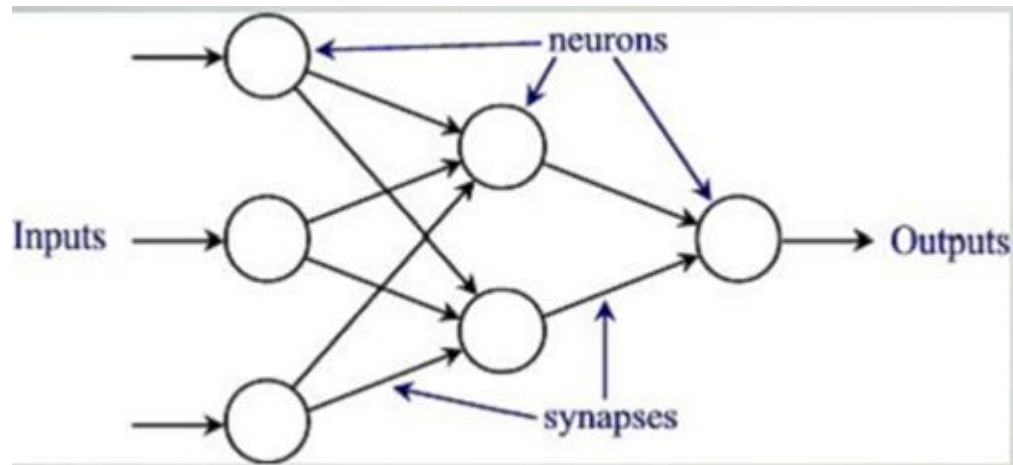


Inspiration: Les réseaux de neurones naturels

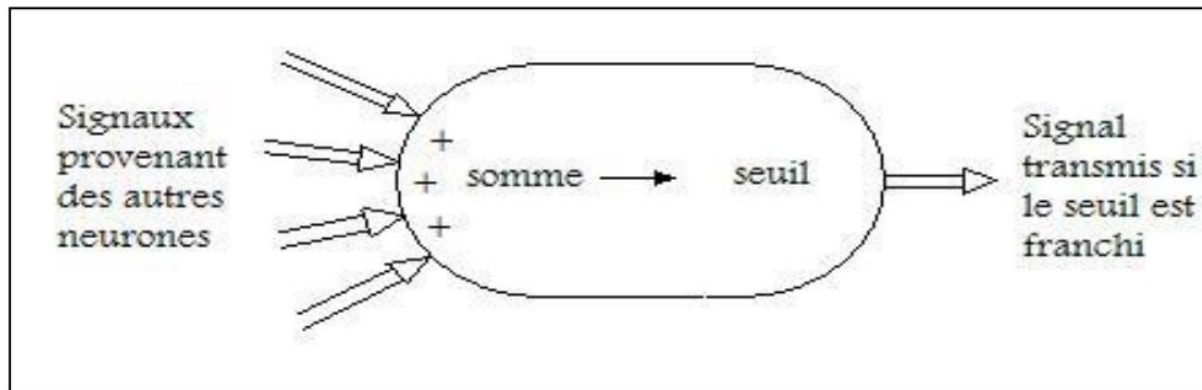
- $\approx 10^{11}$ neurones dans le cerveau humain
- $\approx 10^4$ connexions (synapses + axones) / neurone
- Signaux excitateurs / inhibiteurs



- Les réseaux de Neurones artificiels



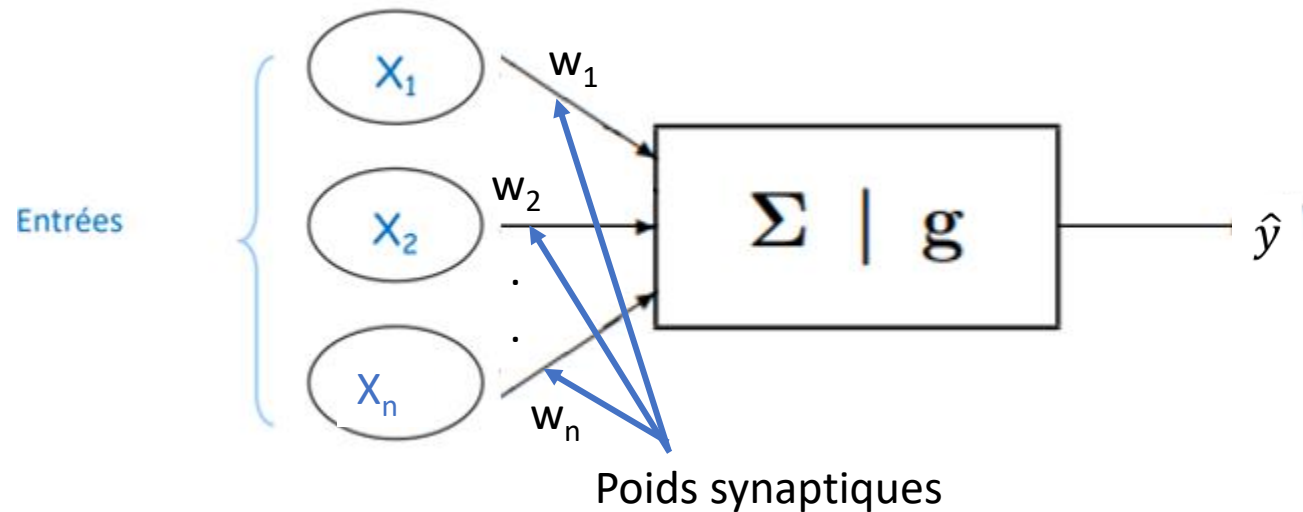
- Un neurone artificiel



Chapitre 5 Les réseaux de neurones artificiels



- Le perceptron de Rosenblatt (1957) (basé sur le modèle de Mc Colluch et Pitts 1943)



Un ensemble d'entrées, x_i

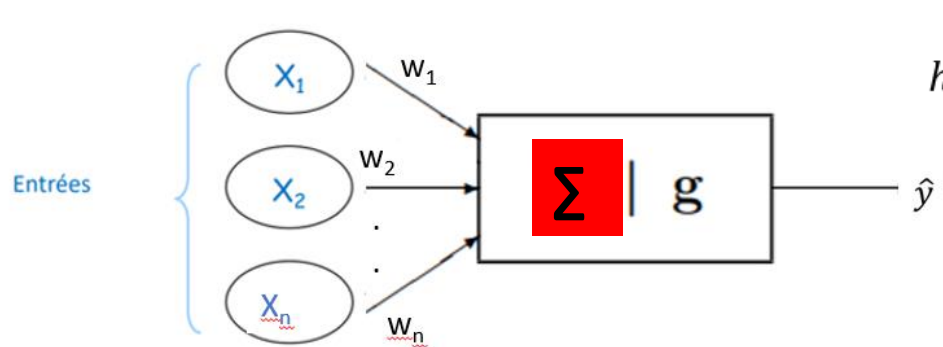
Un ensemble de poids, w_i

Chapitre 5 Les réseaux de neurones artificiels



- Le perceptron de Rosenblatt (1957) (basé sur le modèle de Mc Colluch et Pitts 1943)

Σ : Fonction **somme pondérée**

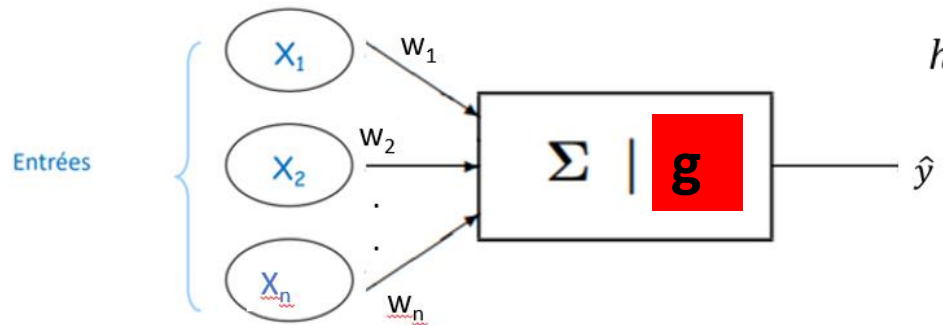


$$h = \sum_{i=1}^n W_i * x_i = W_1 * x_1 + W_2 * x_2 + \dots W_n * x_n$$

Chapitre 5 Les réseaux de neurones artificiels



- Le perceptron de Rosenblatt (1957) (basé sur le modèle de Mc Colluch et Pitts 1943)



Σ : Fonction **somme pondérée**

$$h = \sum_{i=1}^n W_i * x_i = W_1 * x_1 + W_2 * x_2 + \dots + W_n * x_n$$

g : Fonction **d'activation**

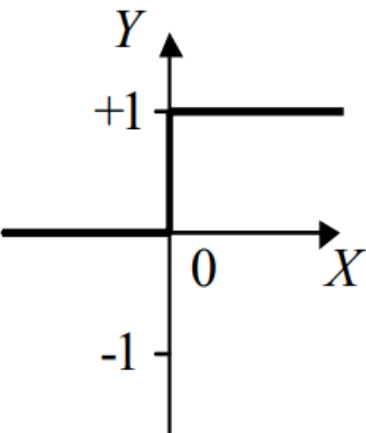
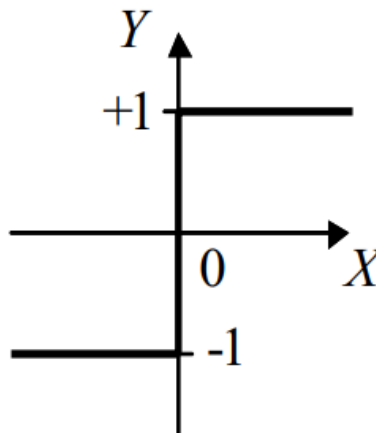
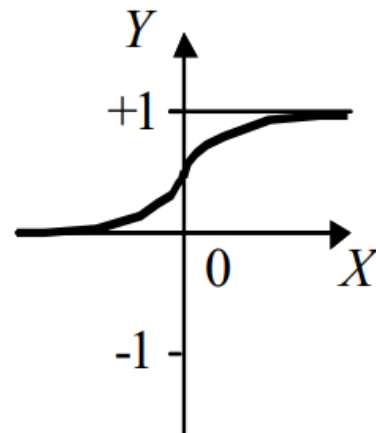
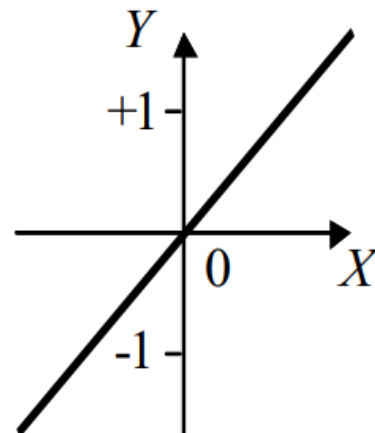
$$g(h) = g\left(\sum_{i=1}^n W_i * x_i\right)$$

$$g(h) = \begin{cases} +1 & \text{si } \sum_{i=1}^n W_i * x_i > \text{seuil} \\ -1 \text{ (ou 0)} & \text{si } \sum_{i=1}^n W_i * x_i \leq \text{seuil} \end{cases}$$

Chapitre 5 Les réseaux de neurones artificiels



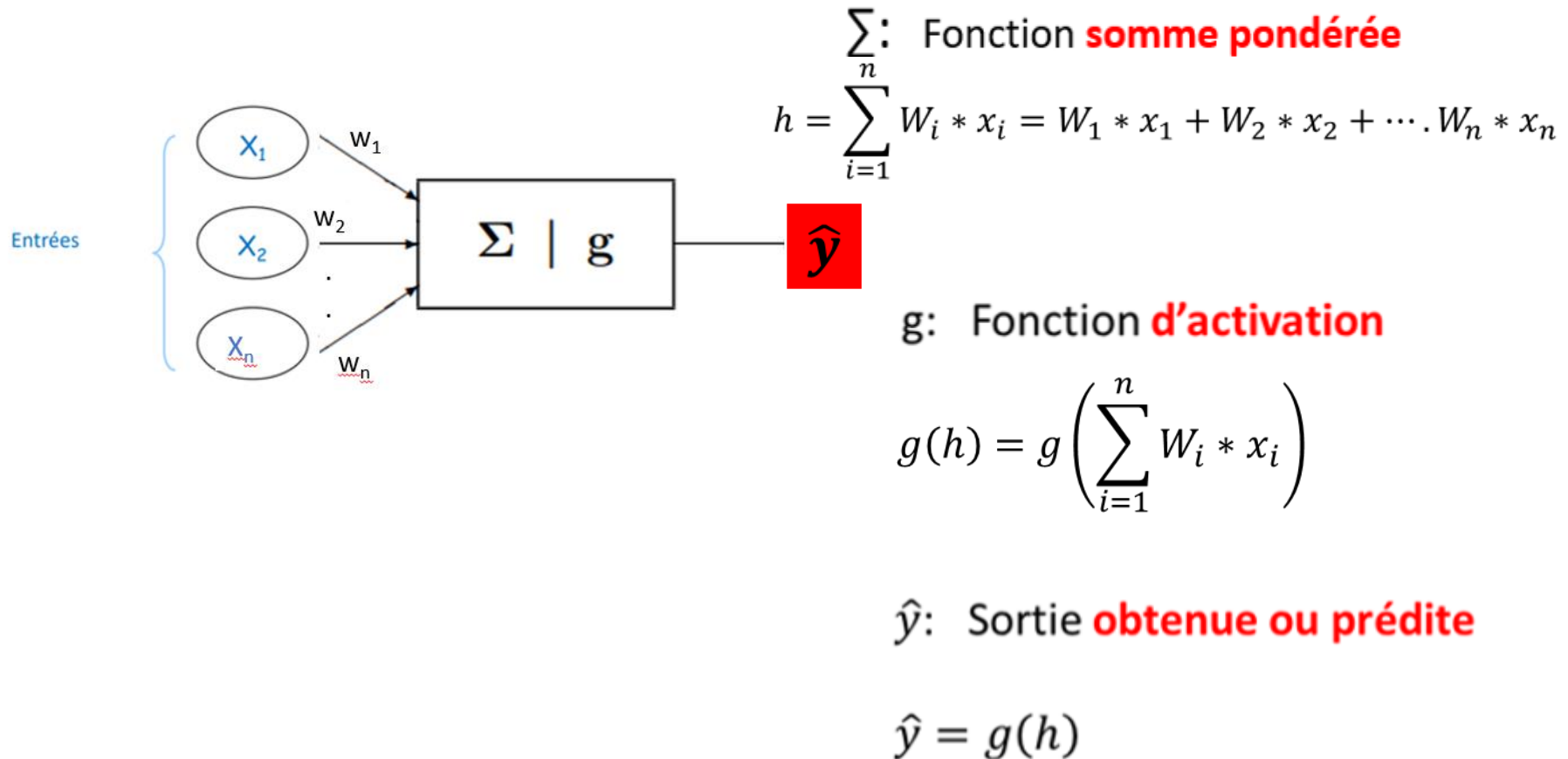
- Types de fonctions de transfert ou d'activation

<i>Step function</i>	<i>Sign function</i>	<i>Sigmoid function</i>	<i>Linear function</i>
			
$Y^{step} = \begin{cases} 1, & \text{if } X > 0 \\ 0, & \text{if } X \leq 0 \end{cases}$	$Y^{sign} = \begin{cases} +1, & \text{if } X > 0 \\ -1, & \text{if } X \leq 0 \end{cases}$	$Y^{sigmoid} = \frac{1}{1 + e^{-X}}$	$Y^{linear} = X$

Chapitre 5 Les réseaux de neurones artificiels



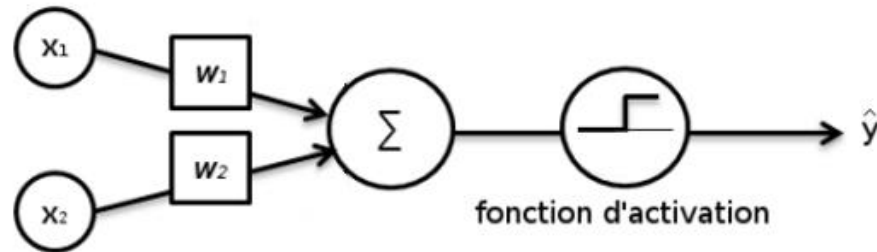
- Le perceptron de Rosenblatt (1957) (basé sur le modèle de McColluch et Pitts 1943)



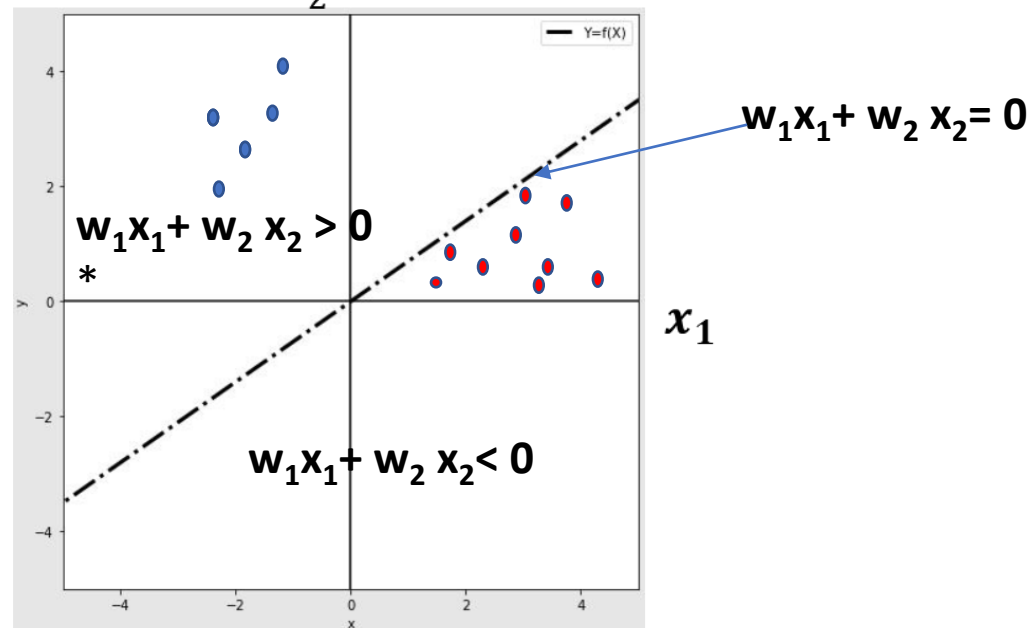
Chapitre 5 Les réseaux de neurones artificiels



- Le perceptron est un modèle linéaire



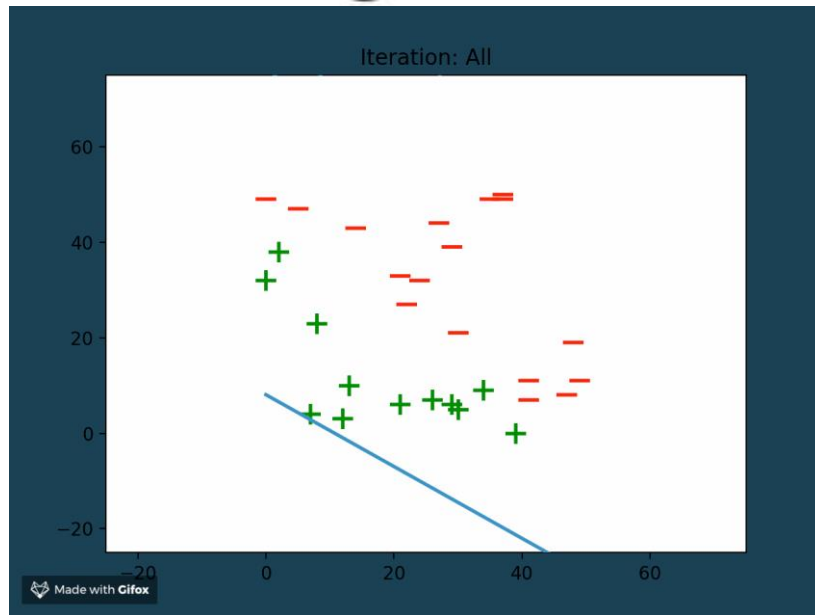
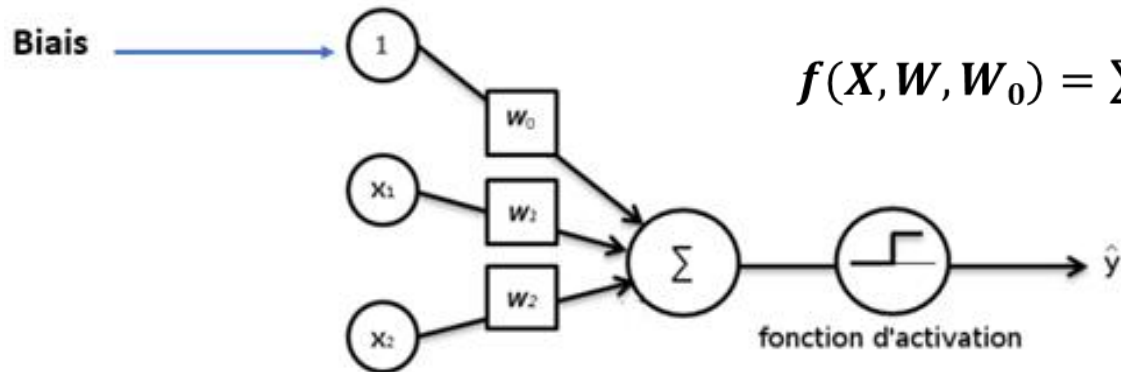
Le perceptron calcule la fonction $f(X, W) = \sum_{i=1}^2 w_i x_i = w_1 x_1 + w_2 x_2$



Chapitre 5 Les réseaux de neurones artificiels



Biais: apporte plus de flexibilité au perceptron



Le perceptron peut donc séparer les données en deux classes.

La droite de séparation est déterminée par les poids synaptiques W et le biais W_0 .

Comment trouver les poids corrects?



Apprentissage



● L'apprentissage

L'apprentissage est un processus par lequel les poids synaptiques du réseau sont modifiés dans le but d'effectuer la tâche demandée (séparation des données en des classes).

Il consiste en un entraînement du réseau sur des exemples.

Deux types d'apprentissage:

Apprentissage supervisé: On présente au réseau des entrées avec leur sorties. Le réseau commence avec des poids initiaux puis ils sont corrigés afin que la sortie obtenue par le réseau soit égale ou très proche de la sortie désirée.

Apprentissage non supervisé: On présente au réseau des entrées sans leur sorties. Le réseau commence avec des poids initiaux puis ils sont corrigés afin de retrouver les structures sous-jacentes à ces entrées non étiquetées.

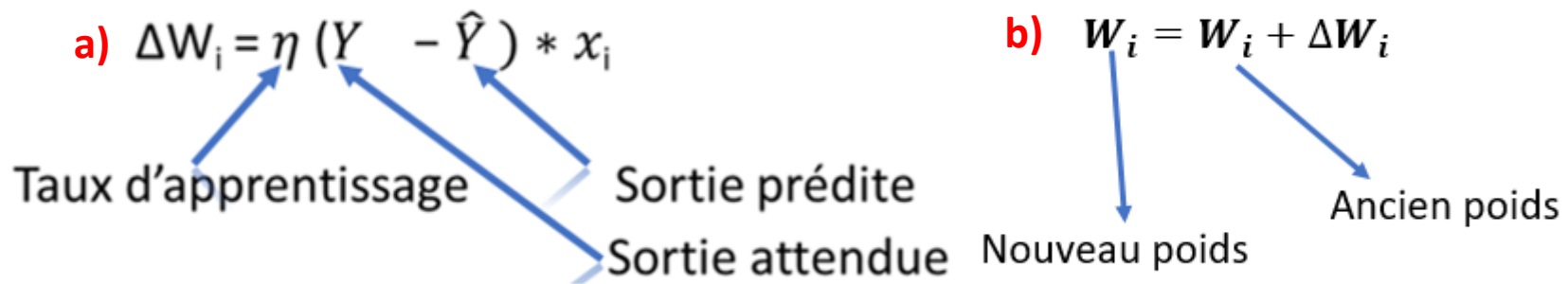


Algorithme d'apprentissage du perceptron

1. Initialiser aléatoirement les poids synaptiques W_i
2. Pour chaque donnée x_i

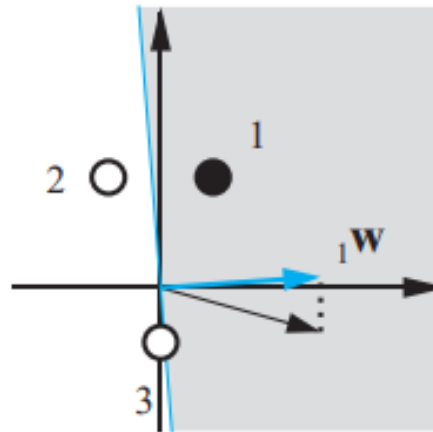
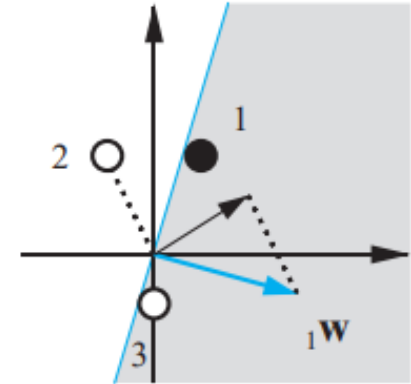
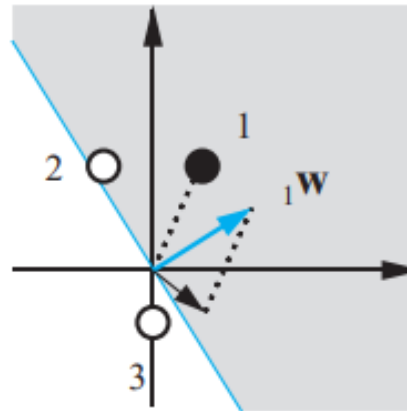
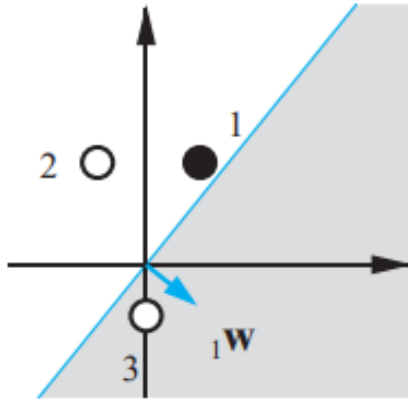
Calculer la sortie prédite $\hat{y} = g(w_0 * \text{biais} + \sum w_i * x_i)$

Mettre à jour les poids synaptiques



3. Revenir à l'étape 2 tant que **tous les exemples ne sont pas classés correctement**

Chapitre 5 Les réseaux de neurones artificiels



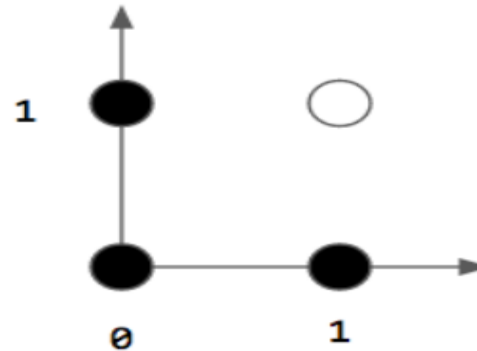
Chapitre 5 Les réseaux de neurones artificiels



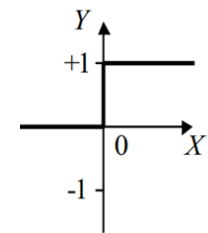
X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Données

AND

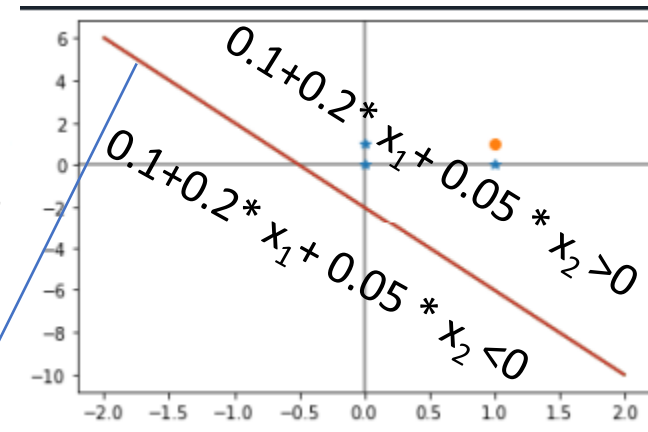
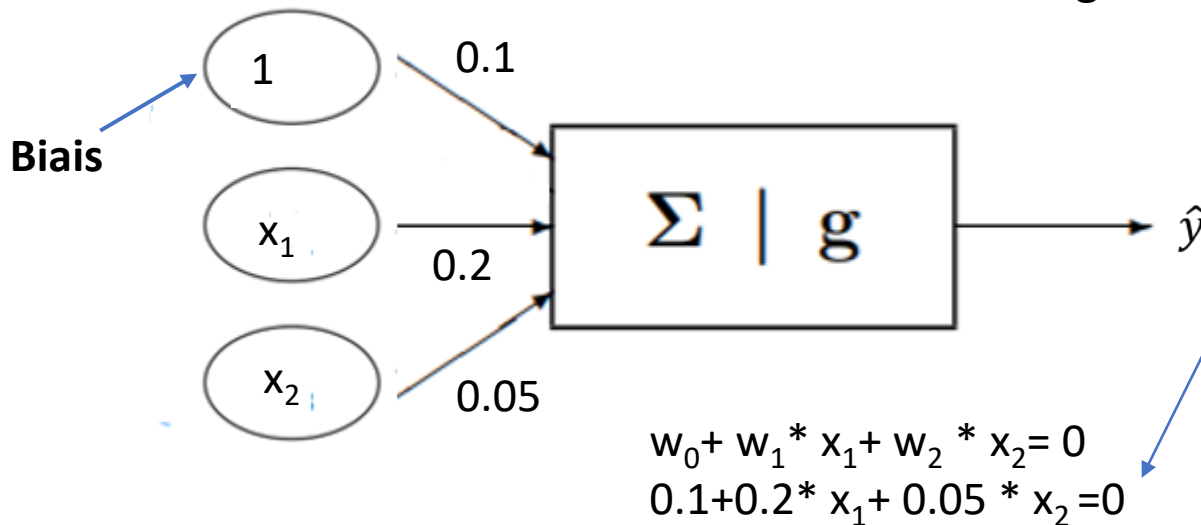


Step function



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

g est la fonction step

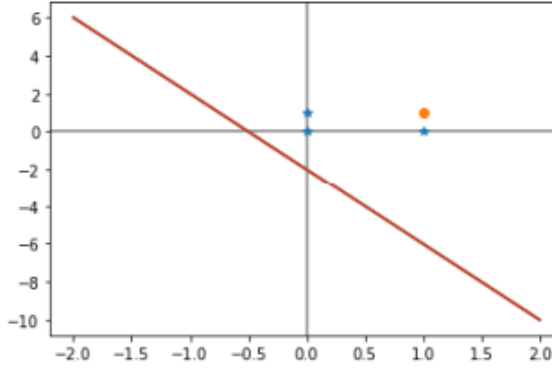


Chapitre 5 Les réseaux de neurones artificiels

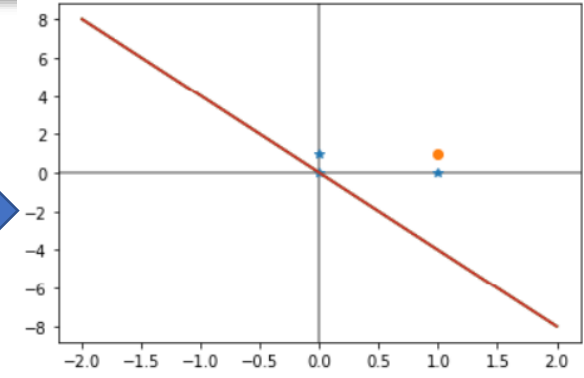


X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

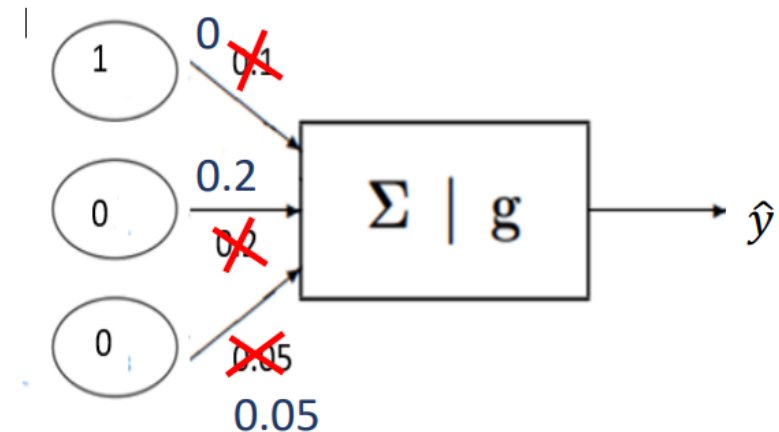
Données



Après mise à jour des poids



1. **Faire passer la donnée 1: (0,0), sortie attendue $y=0$**
2. **Calculer la somme pondérée $\sum W_i * x_i = 0.1 * 1 + 0.2 * 0 + 0.05 * 0 = 0.1$**
3. **Appliquer la fonction de transfert g : $0.1 > 0 \Rightarrow \hat{y}=1$**
4. **Calculer l'erreur de prédiction: $y - \hat{y} = -1$**
5. **Calculer Δw_i :**
 - $\Delta w_0 = 0.1 * (-1) * 1 = -0.1$
 - $\Delta w_1 = 0.1 * (-1) * 0 = 0$
 - $\Delta w_2 = 0.1 * (-1) * 0 = 0$
6. **Mettre à jour les poids**
 - $w_0 = 0.1 - 0.1 = 0$
 - $w_1 = 0.2 + 0 = 0.2$
 - $w_2 = 0.05 + 0 = 0.05$

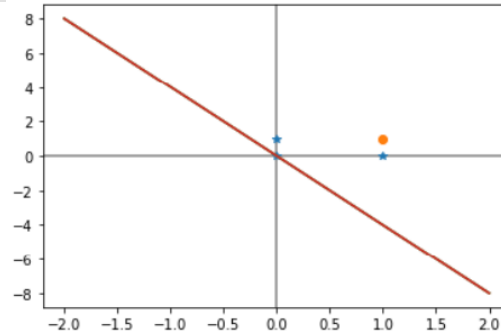


Chapitre 5 Les réseaux de neurones artificiels

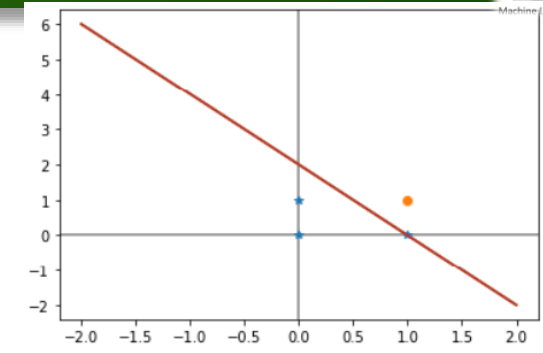


X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

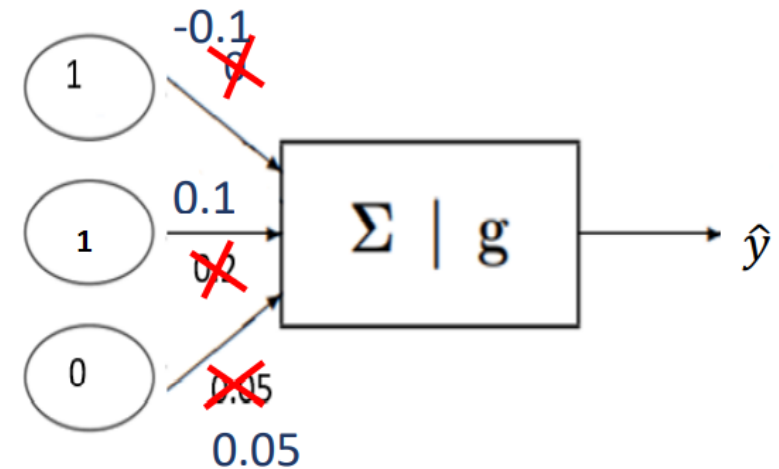
Données



Après mise à jour des poids



1. **Faire passer la donnée 2: (1,0), sortie attendue y=0**
2. **Calculer la somme pondérée** $\sum W_i * x_i = 0.0 * 1 + 0.2 * 1 + 0.05 * 0 = 0.2$
3. **Appliquer la fonction de transfert g:** $0.2 > 0 \Rightarrow \hat{y} = 1$
4. **Calculer l'erreur de prédiction:** $y - \hat{y} = -1$
5. **Calculer Δw_i :**
 - $\Delta w_0 = 0.1 * (-1) * 1 = -0.1$
 - $\Delta w_1 = 0.1 * (-1) * 1 = -0.1$
 - $\Delta w_2 = 0.1 * (-1) * 0 = 0$
6. **Mettre à jour les poids**
 - $w_0 = 0.0 - 0.1 = -0.1$
 - $w_1 = 0.2 - 0.1 = 0.1$
 - $\Delta w_2 = 0.05 + 0 = 0.05$

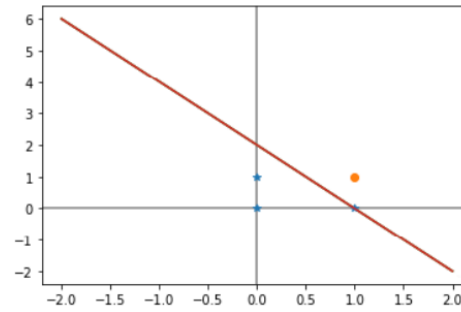


Chapitre 5 Les réseaux de neurones artificiels

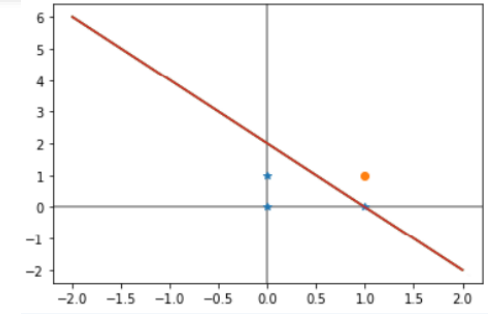


X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Données



Après mise à
jour des poids



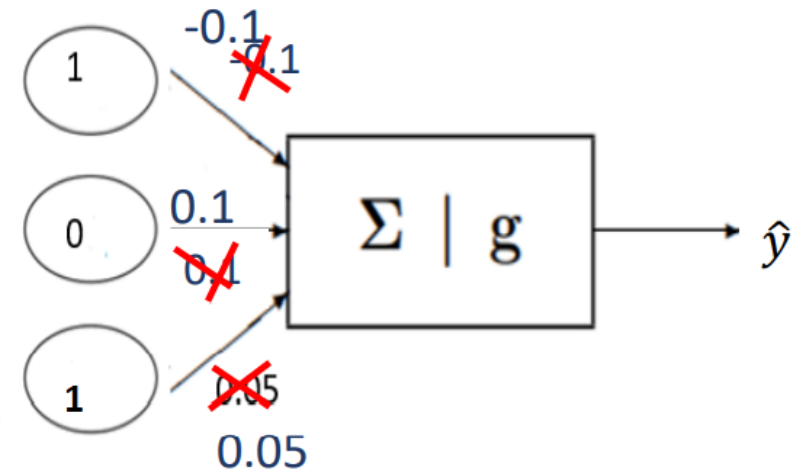
1. **Faire passer la donnée 3: (0,1), sortie attendue y=0**
2. **Calculer la somme pondérée $\sum W_i * x_i = -0.1 * 1 + 0.1 * 0 + 0.05 * 1 = -0.05$**
3. **Appliquer la fonction de transfert g: $-0.05 < 0 \Rightarrow \hat{y} = 0$**
4. **Calculer l'erreur de prédiction: $y - \hat{y} = 0$**

5. **Calculer Δw_i :**

- $\Delta w_0 = 0.1 * (0) * 1 = 0$
- $\Delta w_1 = 0.1 * (0) * 0 = 0$
- $\Delta w_2 = 0.1 * (0) * 1 = 0$

6. **Mettre à jour les poids**

- $w_0 = -0.1 + 0 = -0.1$
- $w_1 = 0.1 + 0 = 0.1$
- $w_2 = 0.05 + 0 = 0.05$

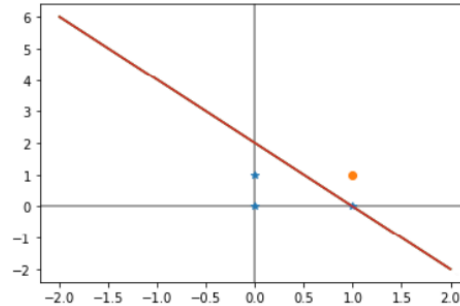


Chapitre 5 Les réseaux de neurones artificiels

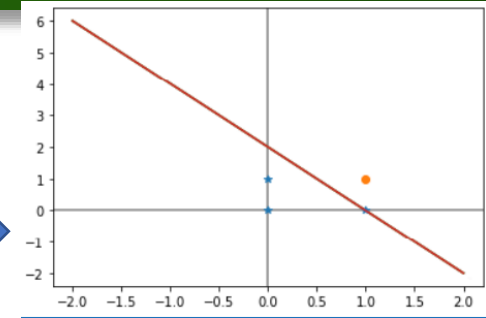


X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

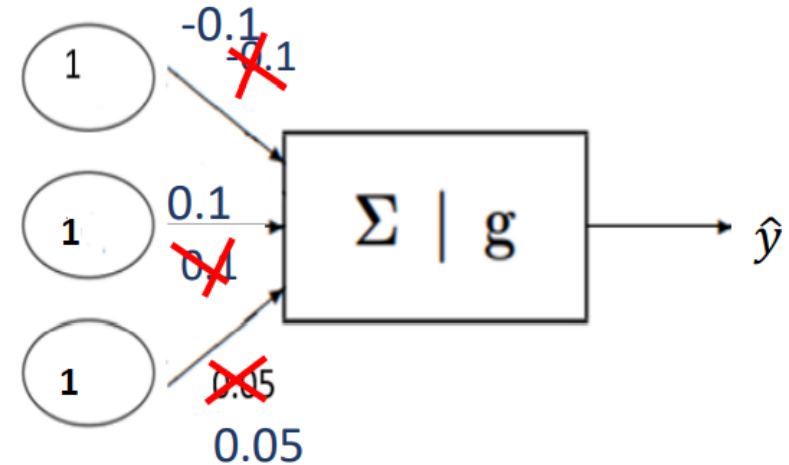
Données



Après mise à jour des poids



1. **Faire passer la donnée 4: (1,1), sortie attendue $y=1$**
2. **Calculer la somme pondérée $\sum W_i * x_i = -0.1 * 1 + 0.1 * 1 + 0.05 * 1 = 0.05$**
3. **Appliquer la fonction de transfert $g: 0.05 > 0 = \hat{y}=1$**
4. **Calculer l'erreur de prédiction: $y - \hat{y}=0$**
5. **Calculer Δw_i :**
 - $\Delta w_0 = 0.1 * (0) * 1 = 0$
 - $\Delta w_1 = 0.1 * (0) * 1 = 0$
 - $\Delta w_2 = 0.1 * (0) * 1 = 0$
6. **Mettre à jour les poids**
 - $w_0 = -0.1 + 0 = -0.1$
 - $w_1 = 0.1 + 0 = 0.1$
 - $w_2 = 0.05 + 0 = 0.05$





● Convergence?

L'algorithme du perceptron est répété un certain nombre de fois tant que:

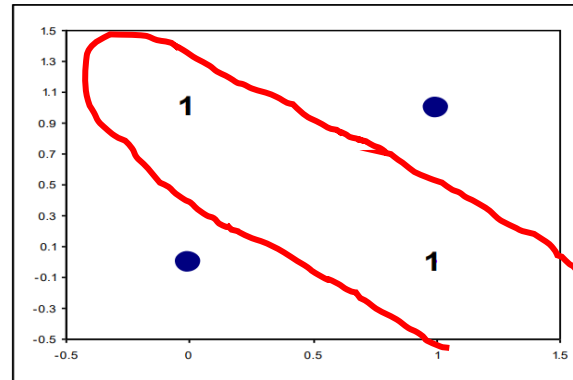
- Aucune nouvelle correction n'est obtenue
- Ou l'erreur globale ne diminue pas considérablement ou bien on atteint une erreur préfixée
- Ou on atteint le nombre maximum d'itérations



● Limite du PERCEPTRON: modèle linéaire

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Données



Non séparable linéairement
(Minsky & Papert, 1969)

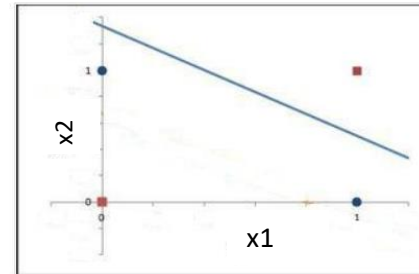
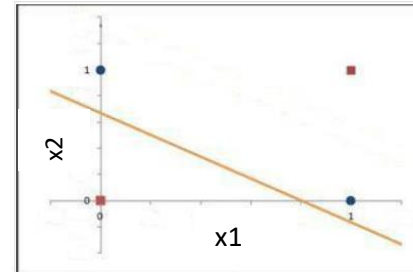
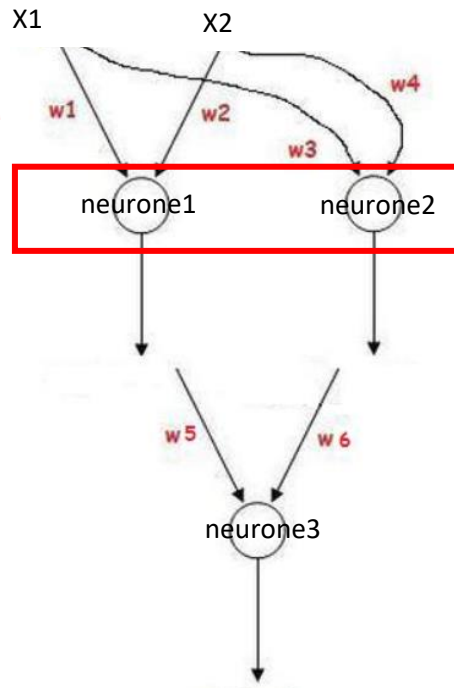
Besoin d'un modèle non linéaire

Chapitre 5 Les réseaux de neurones artificiels



Solution

Ajouter une couche de neurones entre la couche d'entrée et la couche de sortie



x1 OR x2

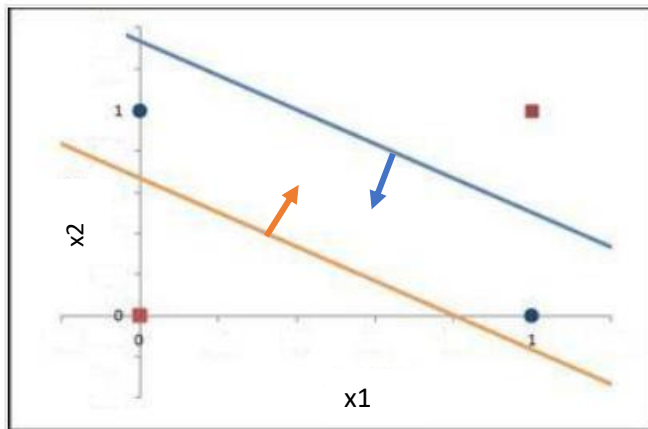
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

neurone1

Not (x1 AND x2)

x1	x2	y
0	0	1
0	1	1
1	0	1
1	1	0

neurone2



x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

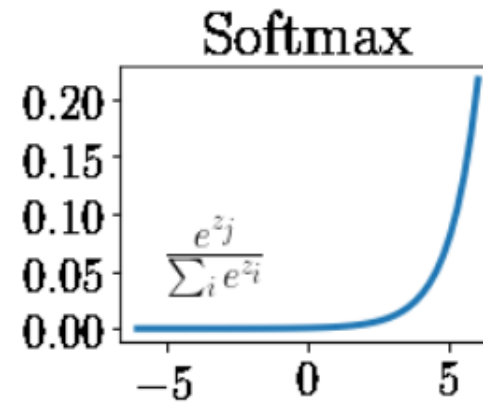
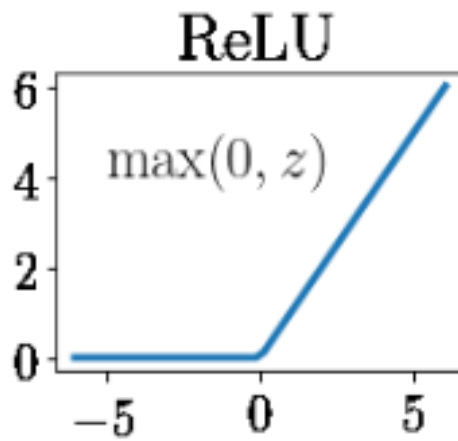
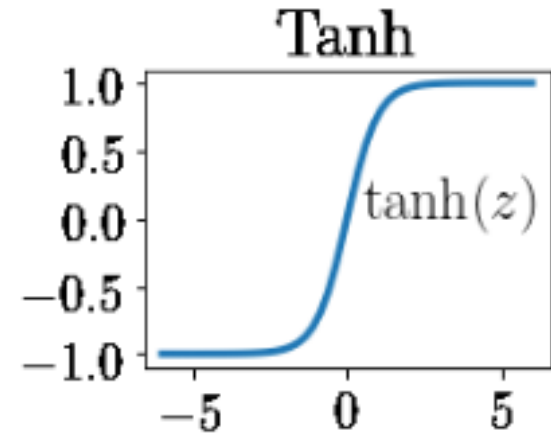
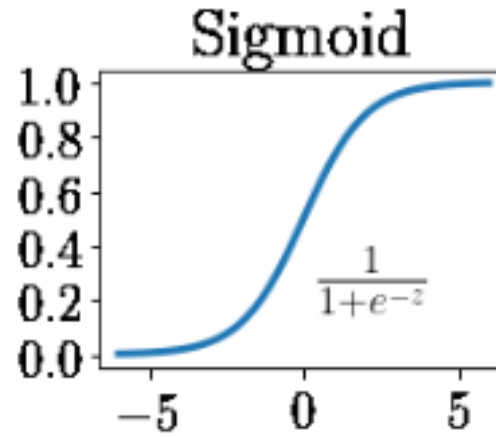
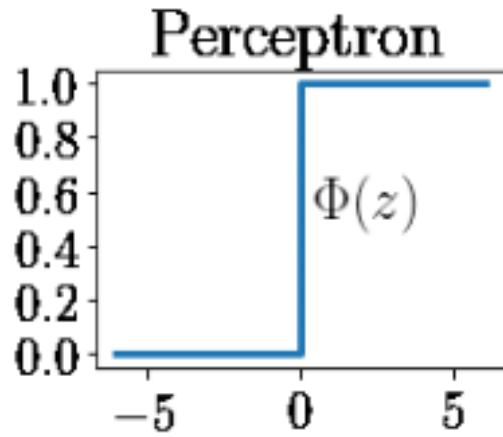
x1	x2	y
0	0	1
0	1	1
1	0	1
1	1	0

x1	x2	y
0	0	0
1	0	1
1	0	1
1	1	0

neurone3

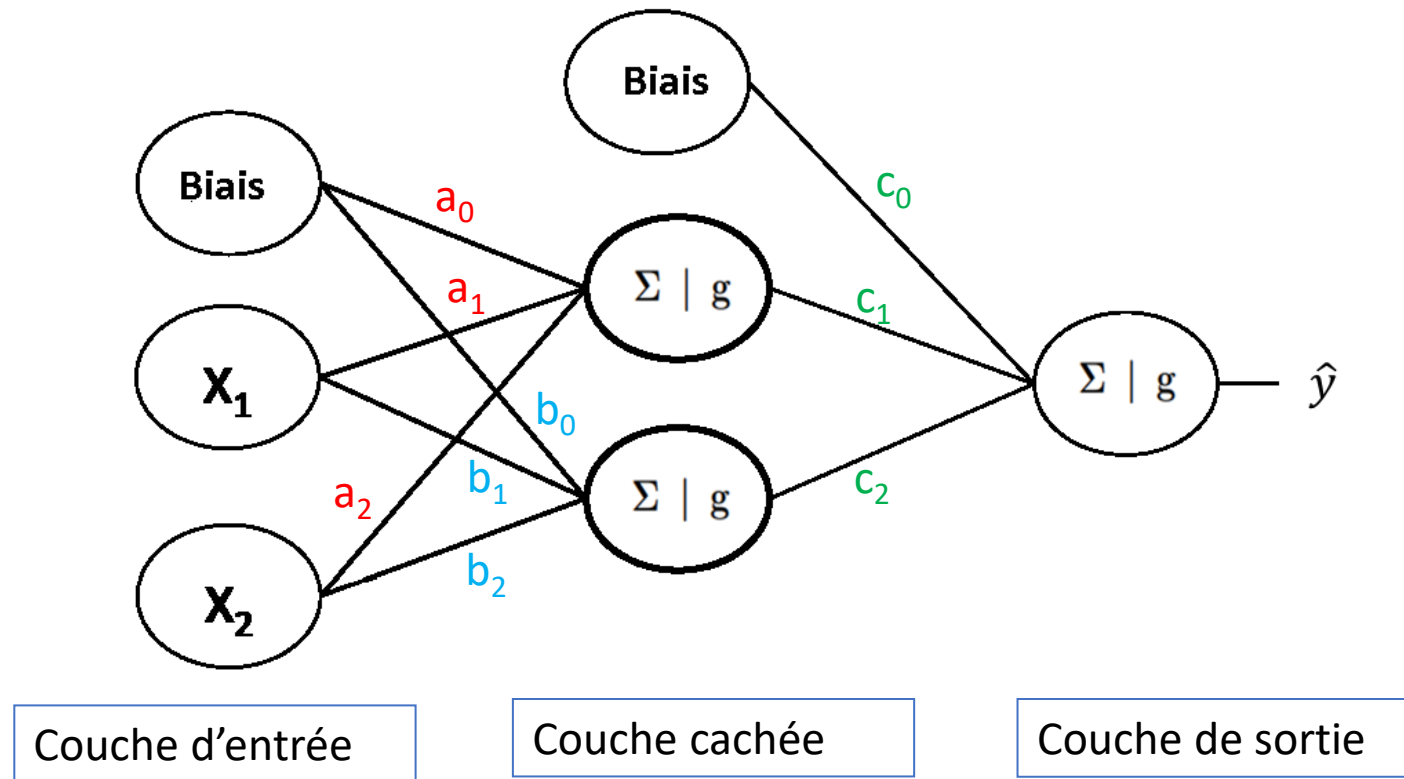


Les fonctions d'activation





● Perceptron multi-couches



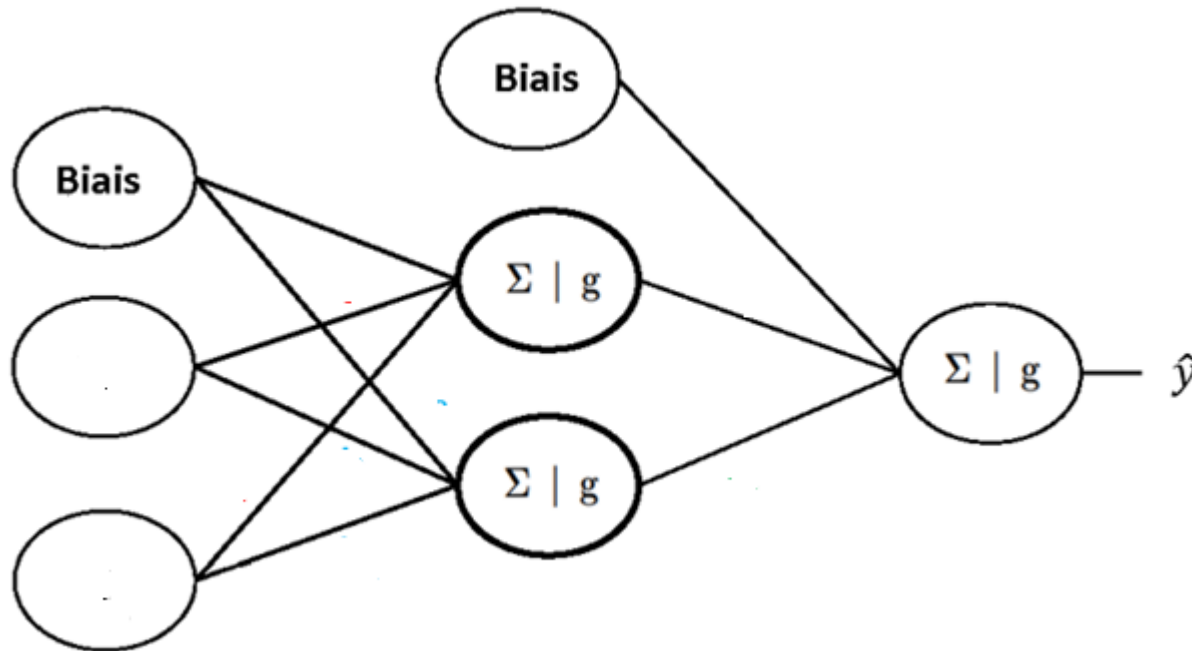


- **L'apprentissage** se fait par l'algorithme par rétropropagation des erreurs
- L'algorithme consiste dans un premier temps à propager vers l'avant les entrées jusqu'à obtenir une sortie calculée par le réseau.
- La seconde étape compare la sortie calculée à la sortie réelle connue.
- On rétro-propage après l'erreur commise vers l'arrière jusqu'à la couche d'entrée
- On modifie les valeurs des poids.
- On répète ce processus sur tous les exemples jusqu'à ce que l'on obtienne une erreur de sortie considérée comme négligeable ou bien on a atteint le nombre maximum des itérations.



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

1. Initialisation aléatoire des poids synaptiques

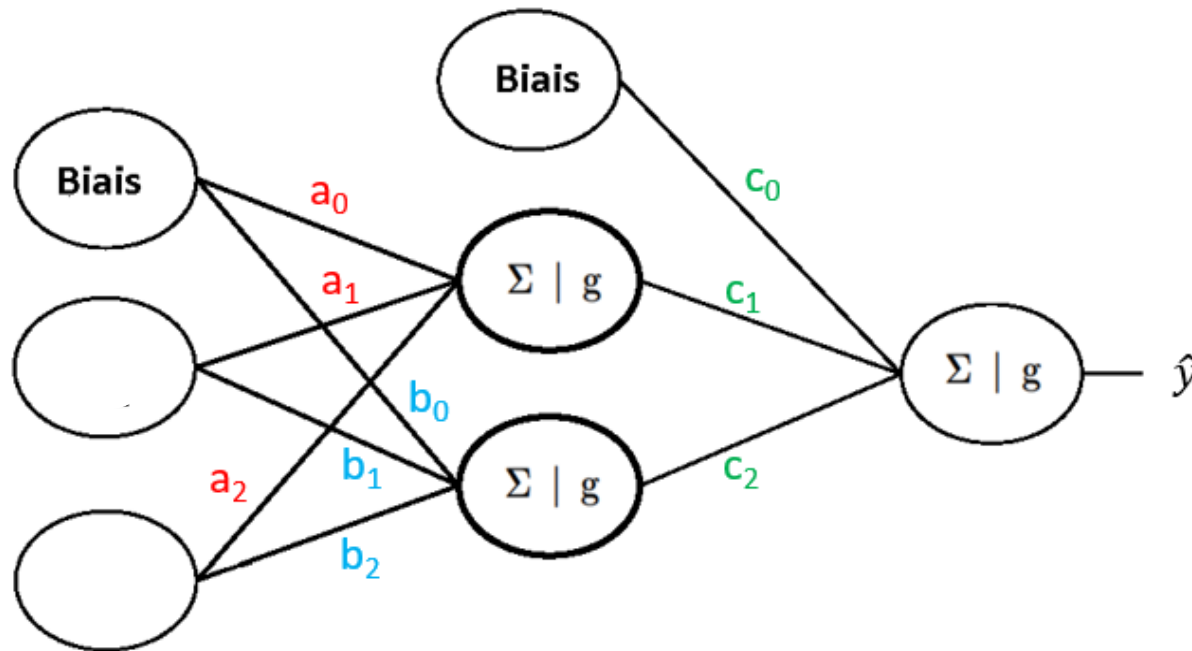


Chapitre 5 Les réseaux de neurones artificiels



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

1. Initialisation aléatoire des poids synaptiques

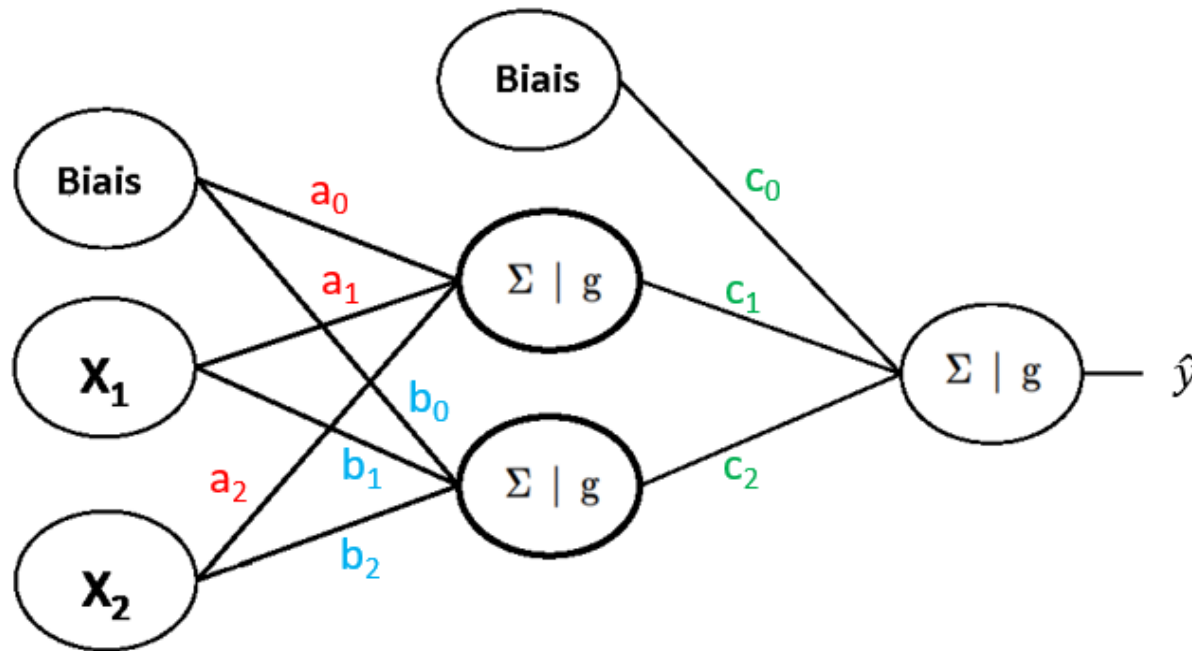


Chapitre 5 Les réseaux de neurones artificiels



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

2. Présentation d'une donnée au réseau (X_1, X_2)

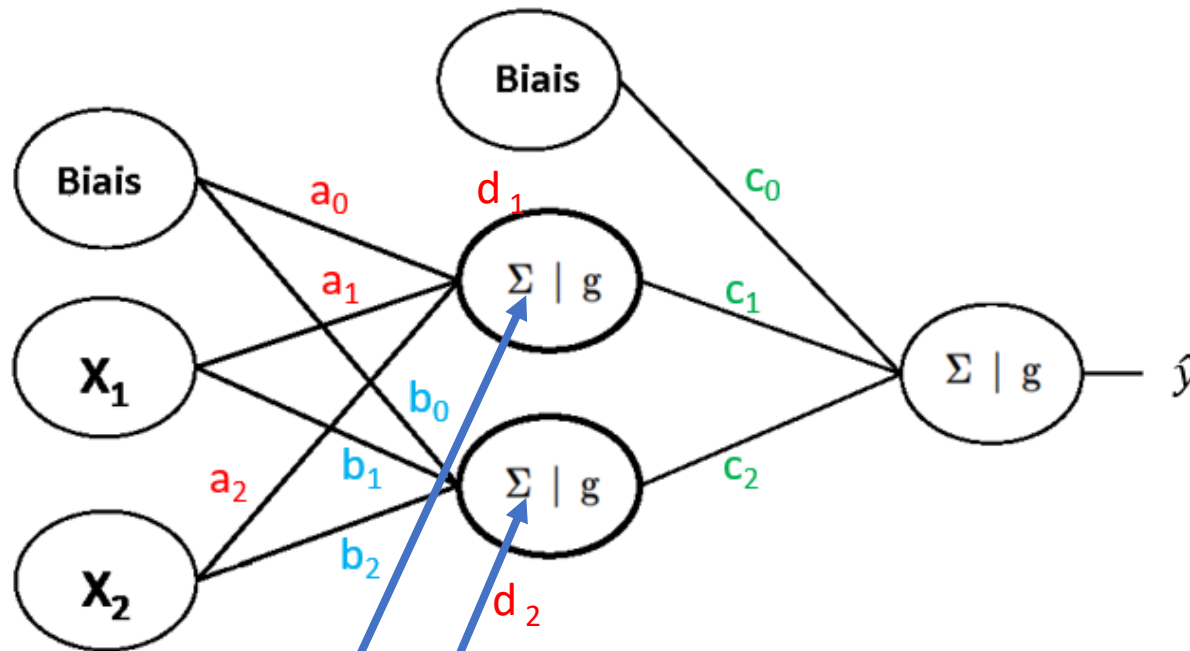


Chapitre 5 Les réseaux de neurones artificiels



● Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs

3. Passage de la couche d'entrée vers la couche cachée



$$d_1 = a_0 * \text{Biais} + a_1 * x_1 + a_2 * x_2$$

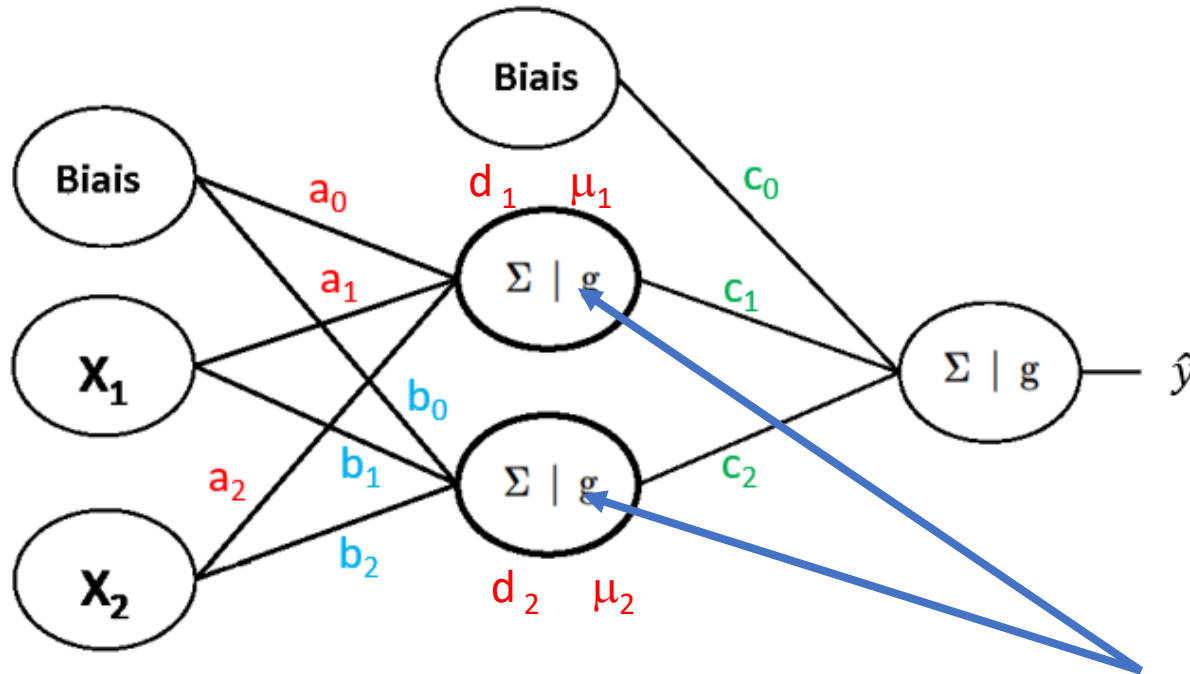
$$d_2 = b_0 * \text{Biais} + b_1 * x_1 + b_2 * x_2$$

Chapitre 5 Les réseaux de neurones artificiels



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

4. Calcul de la sortie de la couche cachée

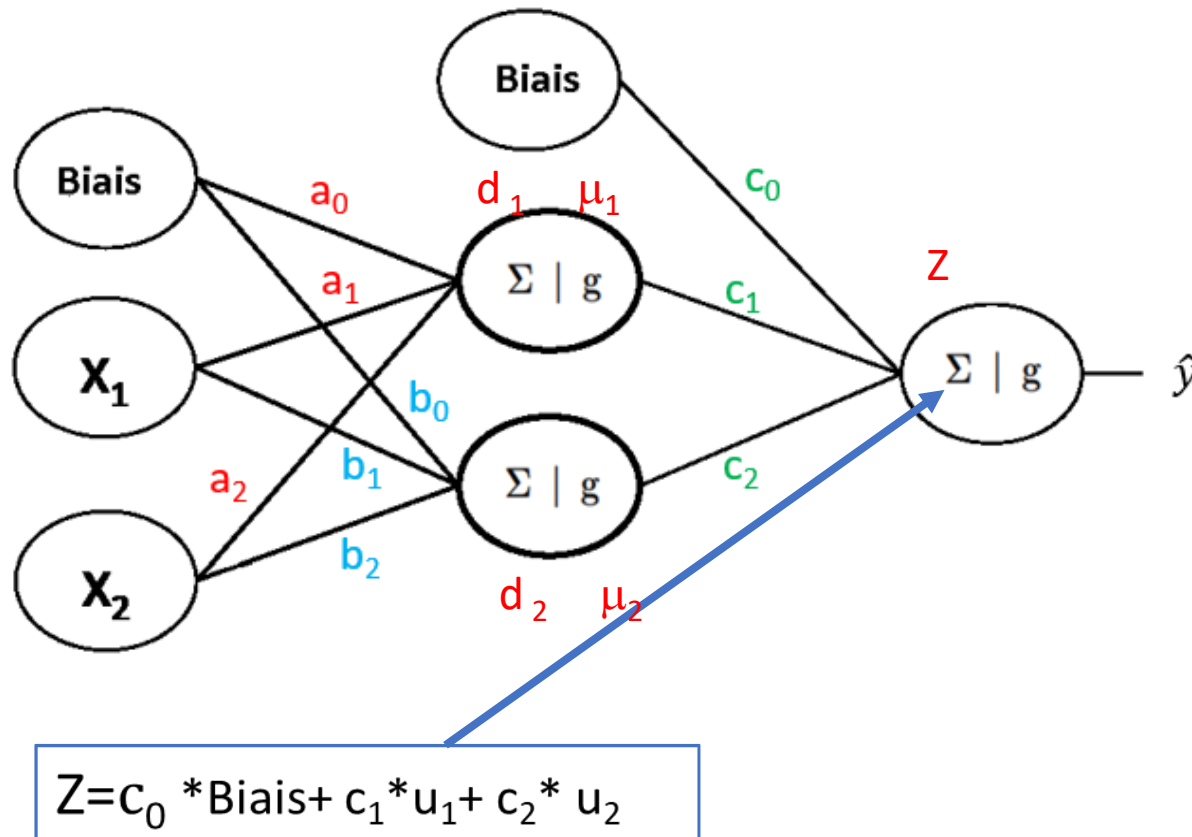


$$u_1 = g(d_1) = \frac{1}{1 + e^{-d_1}}$$
$$u_2 = g(d_2) = \frac{1}{1 + e^{-d_2}}$$



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

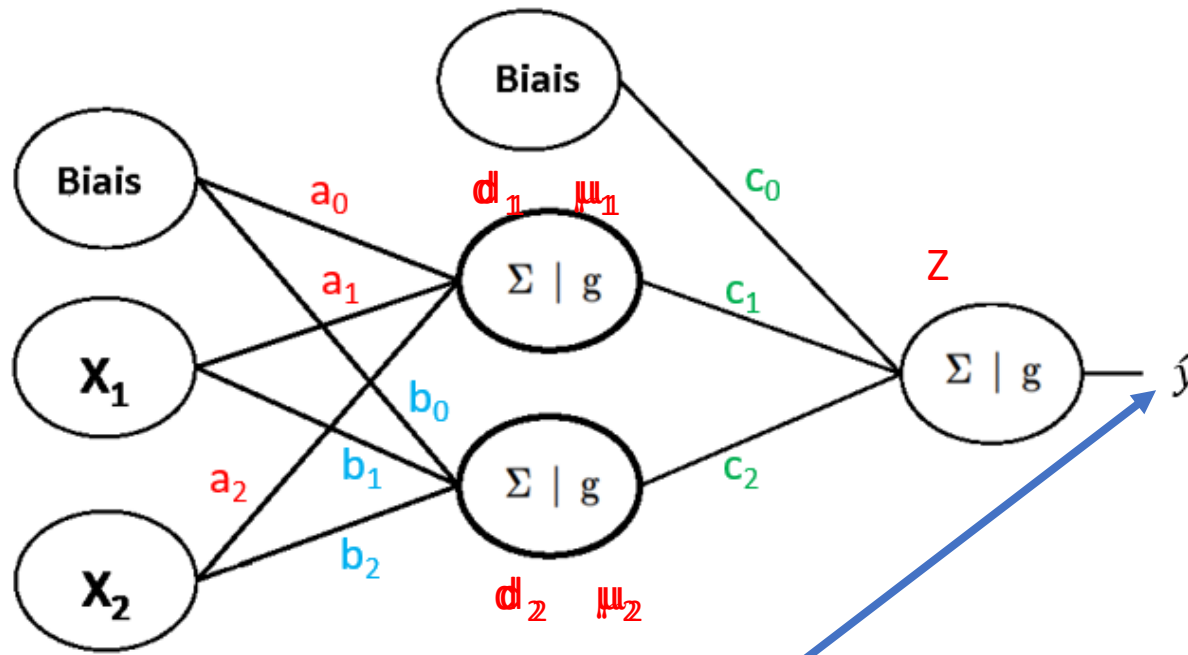
5. Passage de la couche cachée vers la couche de sortie





- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

6. Calcul de la sortie de la couche de sortie

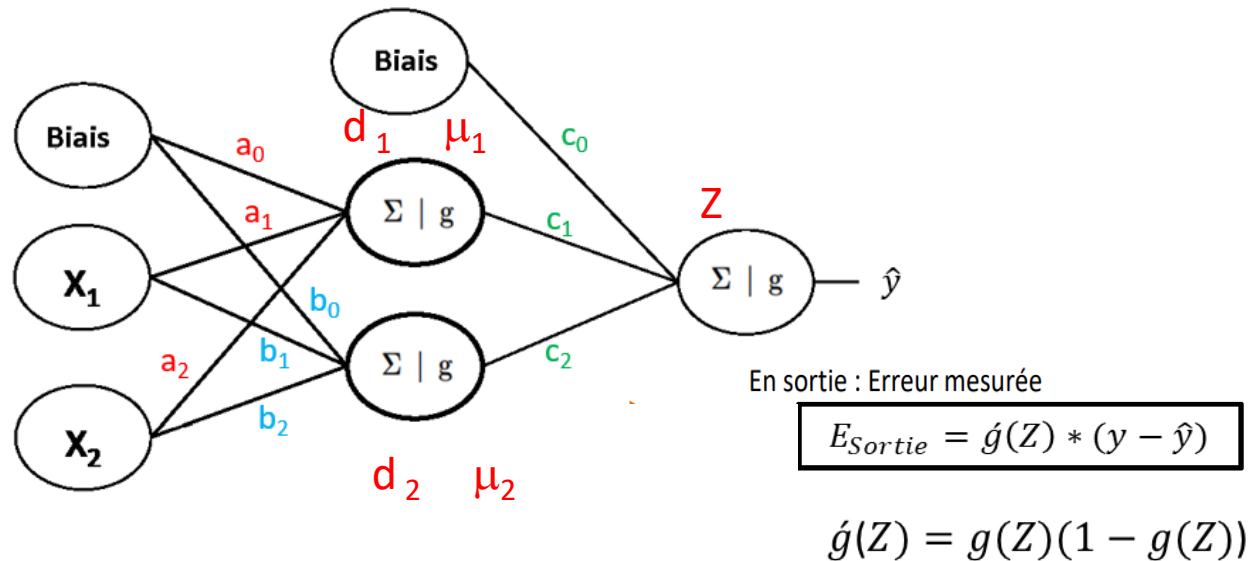


$$\hat{y} = g(Z) = \frac{1}{1 + e^{-Z}}$$



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

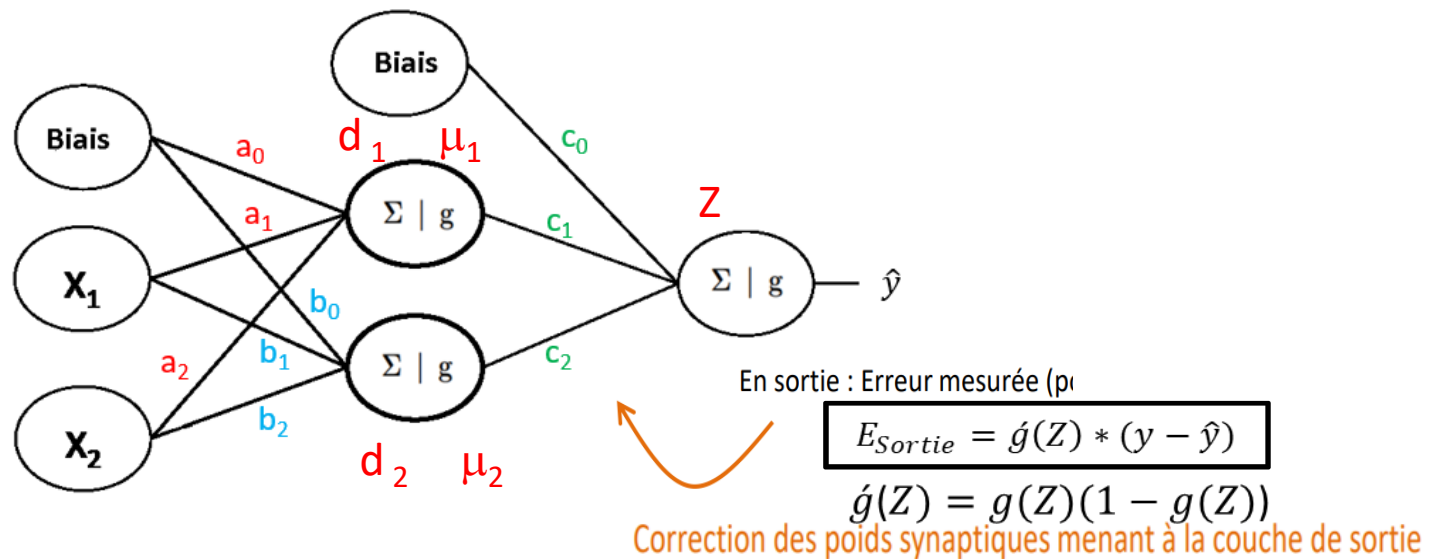
7. Calcul de l'erreur de la couche de sortie





- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

7. Correction des poids synaptiques menant à la couche de sortie

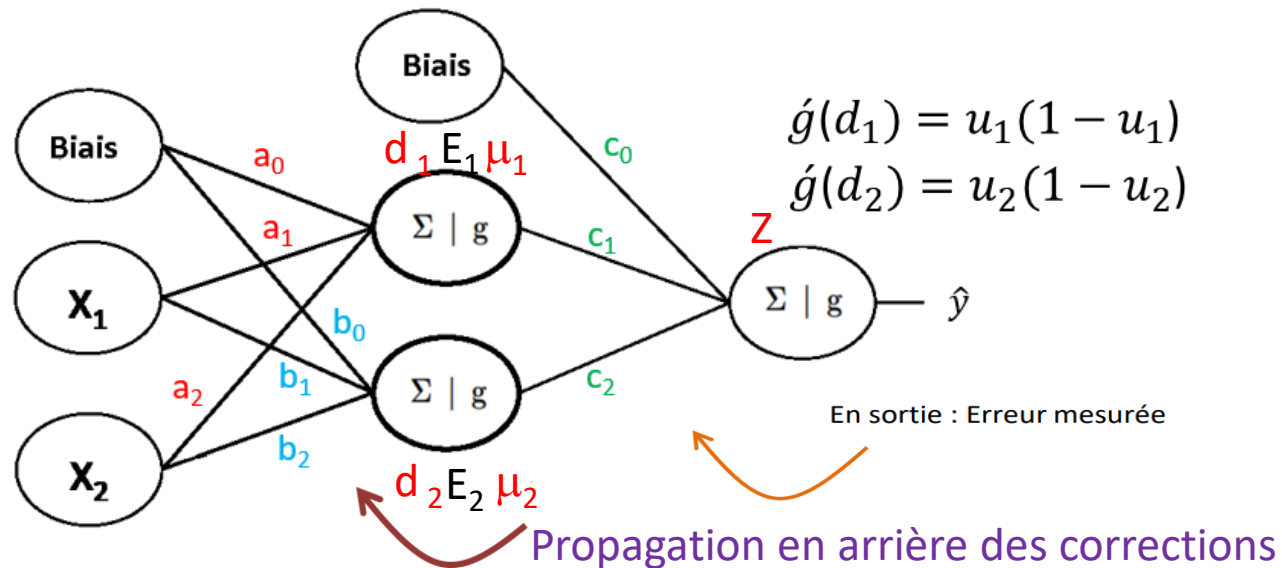


$$\begin{aligned}\Delta c_0 &= \tau * E_{Sortie} * Biais \\ \Delta c_1 &= \tau * E_{Sortie} * u_1 \\ \Delta c_2 &= \tau * E_{Sortie} * u_2\end{aligned}$$



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

8. Correction des poids synaptiques menant à la couche d'entrée



$$E_1 = \dot{g}(d_1) * (c_1 * E_{Sortie}), \quad E_2 = \dot{g}(d_2) * (c_2 * E_{Sortie})$$

$$\begin{aligned} \Delta a_0 &= \tau * E_1 * Biais \\ \Delta a_1 &= \tau * E_1 * X_1 \\ \Delta a_2 &= \tau * E_1 * X_2 \end{aligned}$$

$$\begin{aligned} \Delta b_0 &= \tau * E_2 * Biais \\ \Delta b_1 &= \tau * E_2 * X_1 \\ \Delta b_2 &= \tau * E_2 * X_2 \end{aligned}$$

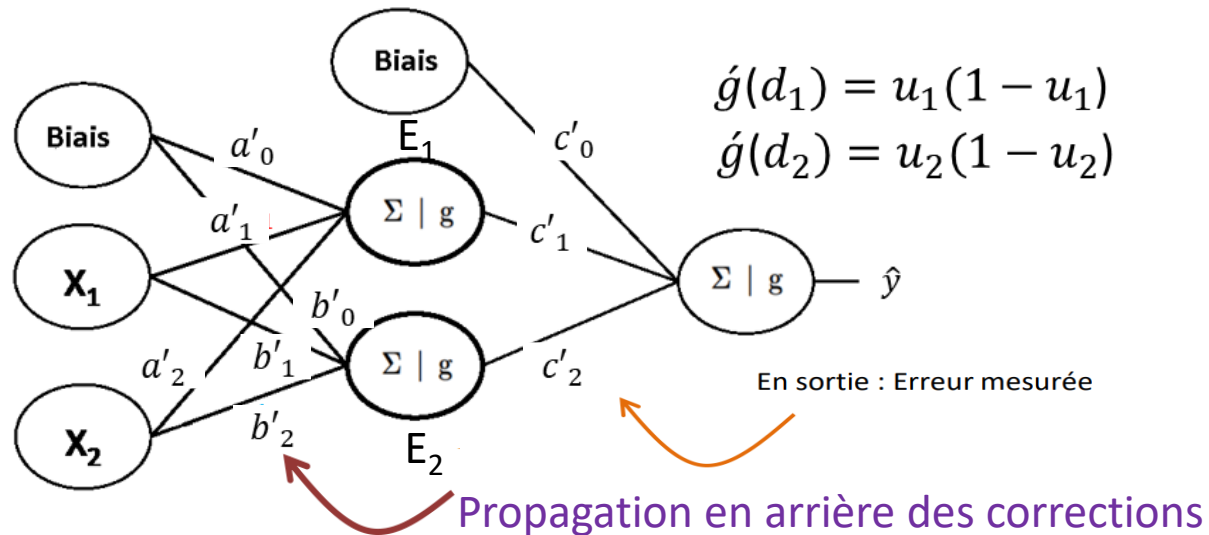


● Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs

8. Ajustement des poids synaptiques

Tant qu'il y a des données, revenir à l'étape 2 pendant un certain nombre d'itération

Voir slide 21



$$E_1 = \dot{g}(d_1) * (c_1 * E_{Sortie}), \quad E_2 = \dot{g}(d_2) * (c_2 * E_{Sortie})$$

$$\begin{aligned} \dot{c}_0 &= c_0 + \Delta c_0 \\ \dot{c}_1 &= c_1 + \Delta c_1 \\ \dot{c}_2 &= c_2 + \Delta c_2 \end{aligned}$$

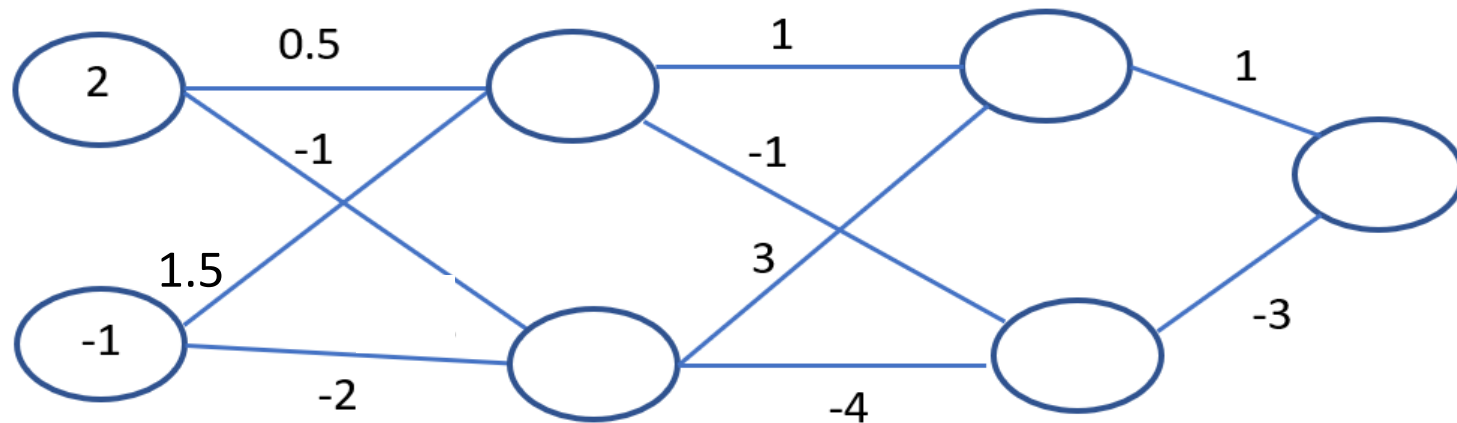
$$\begin{aligned} \dot{a}_0 &= a_0 + \Delta a_0 \\ \dot{a}_1 &= a_1 + \Delta a_1 \\ \dot{a}_2 &= a_2 + \Delta a_2 \end{aligned}$$

$$\begin{aligned} \dot{b}_0 &= b_0 + \Delta b_0 \\ \dot{b}_1 &= b_1 + \Delta b_1 \\ \dot{b}_2 &= b_2 + \Delta b_2 \end{aligned}$$



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

Exemple: $\mathbf{x} = [2, -1]$, $y = 1$

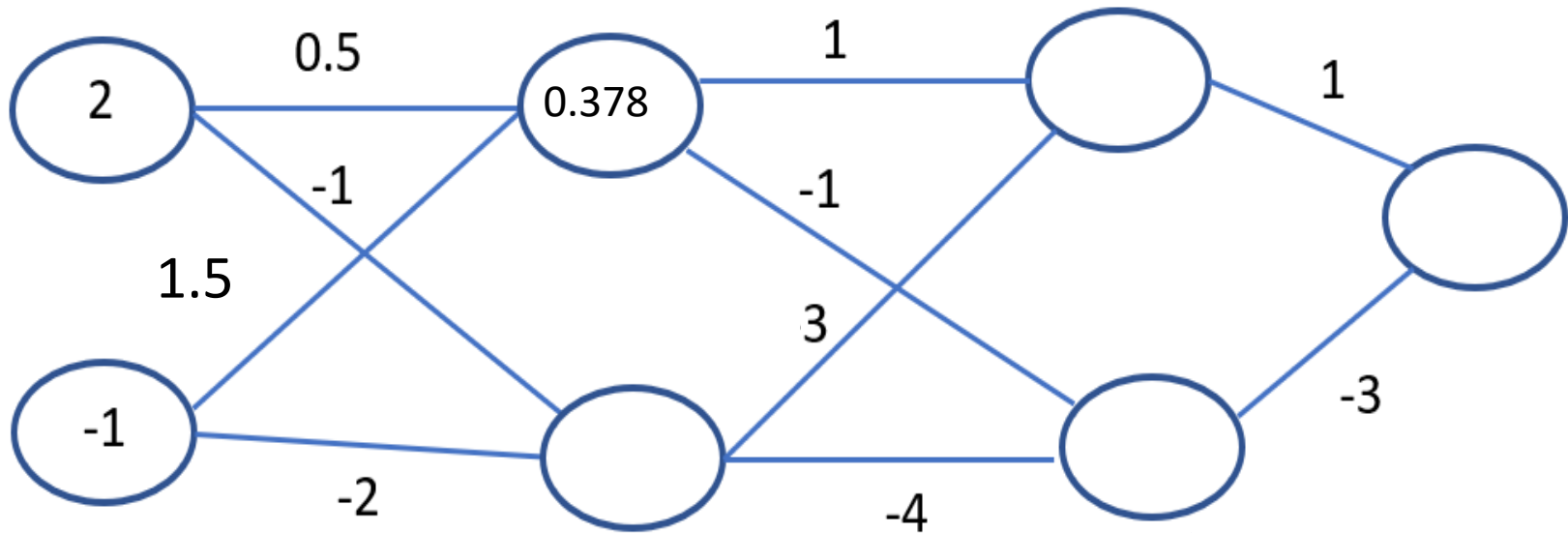




● Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs

Exemple: $\mathbf{x} = [2, -1]$, $y = 1$

Une itération



$$d_1 = 0.5 * 2 + 1.5 * -1 = -0.5$$

$$g(d_1) = g(-0.5) = \frac{1}{1 + e^{0.5}}$$

$$g(-0.5) = 0.378$$

Propagation

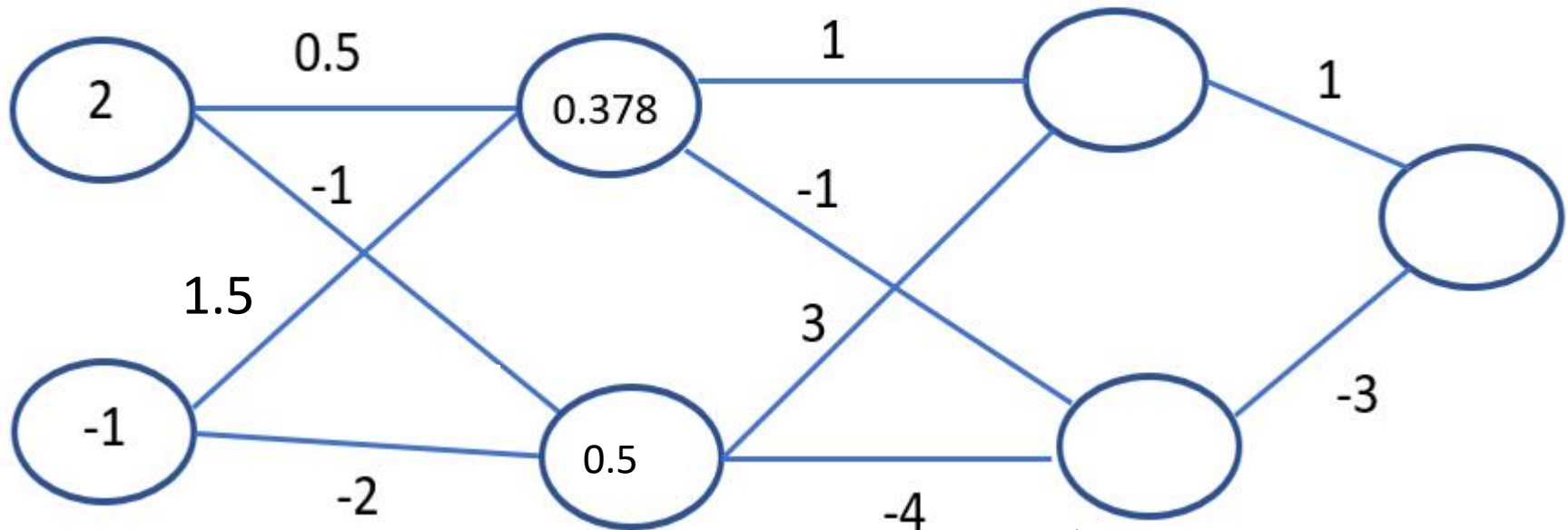
Chapitre 5 Les réseaux de neurones artificiels



● Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs

Exemple: $x = [2, -1]$, $y = 1$

Une itération



$$d_2 = -1 * 2 - 2 * -1 = 0$$

$$g(d_2) = g(0) = \frac{1}{1 + e^0}$$

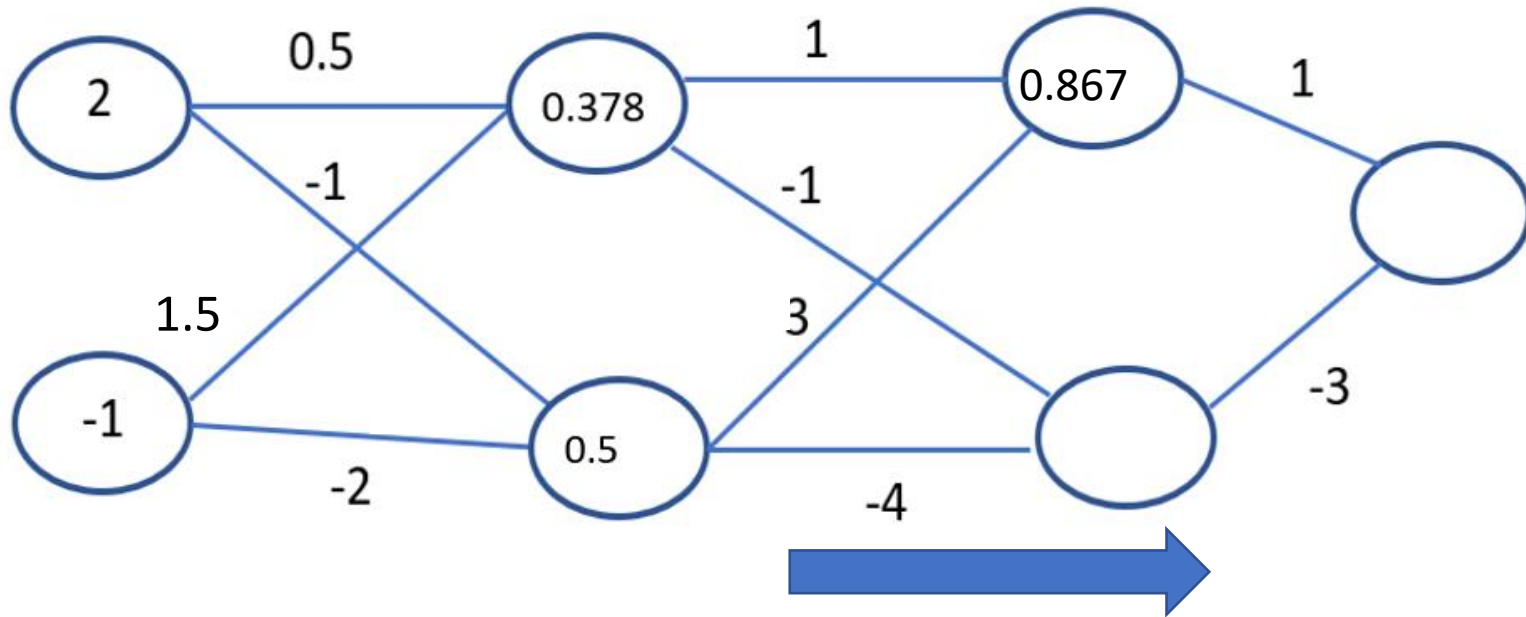
$$g(0) = 0.5$$



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

Exemple: $x = [2, -1]$, $y = 1$

Une itération



$$d_3 = 1 * 0.378 + 3 * 0.5 = 1.878$$

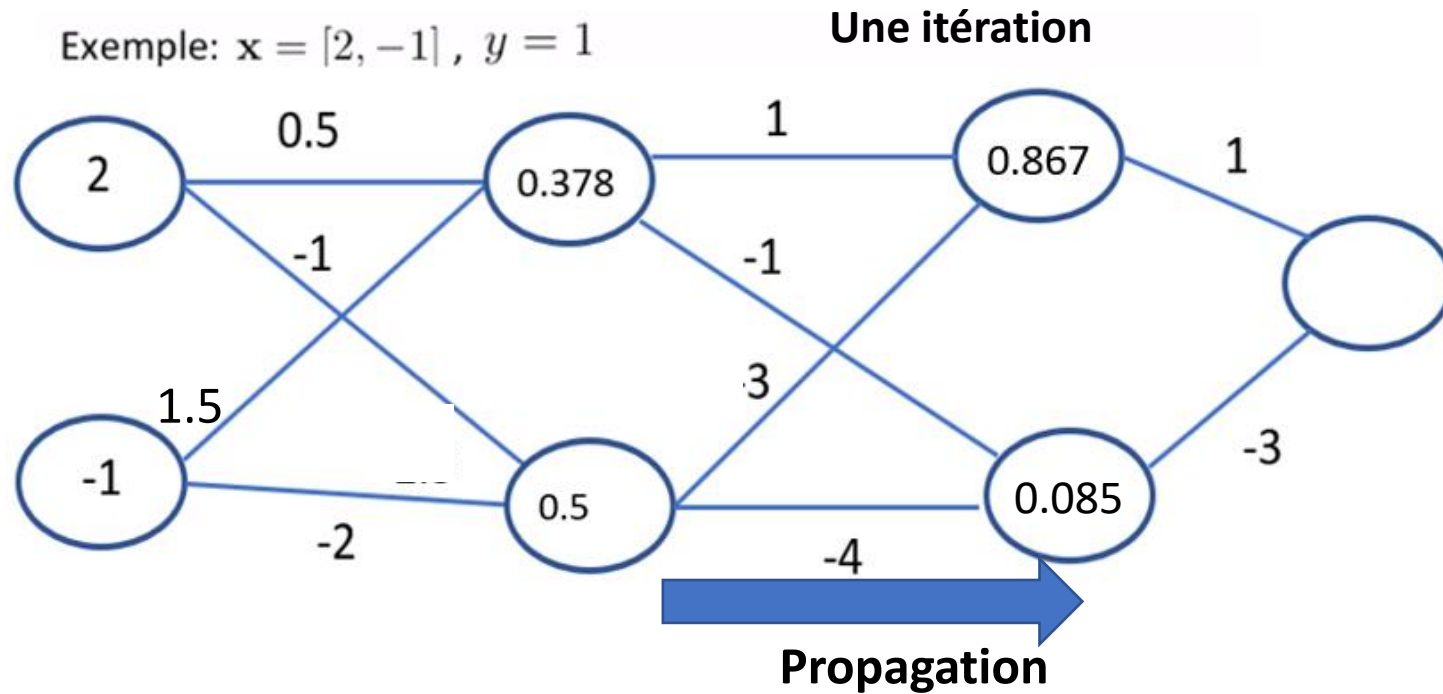
$$g(d_3) = g(1.878) = \frac{1}{1 + e^{-1.878}}$$

$$g(1.878) = 0.867$$

Propagation



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**



$$d_4 = -1 * 0.378 - 4 * 0.5 = -2.378$$

$$g(d_4) = g(-2.378) = \frac{1}{1 + e^{2.378}}$$

$$g(-2.378) = 0.085$$

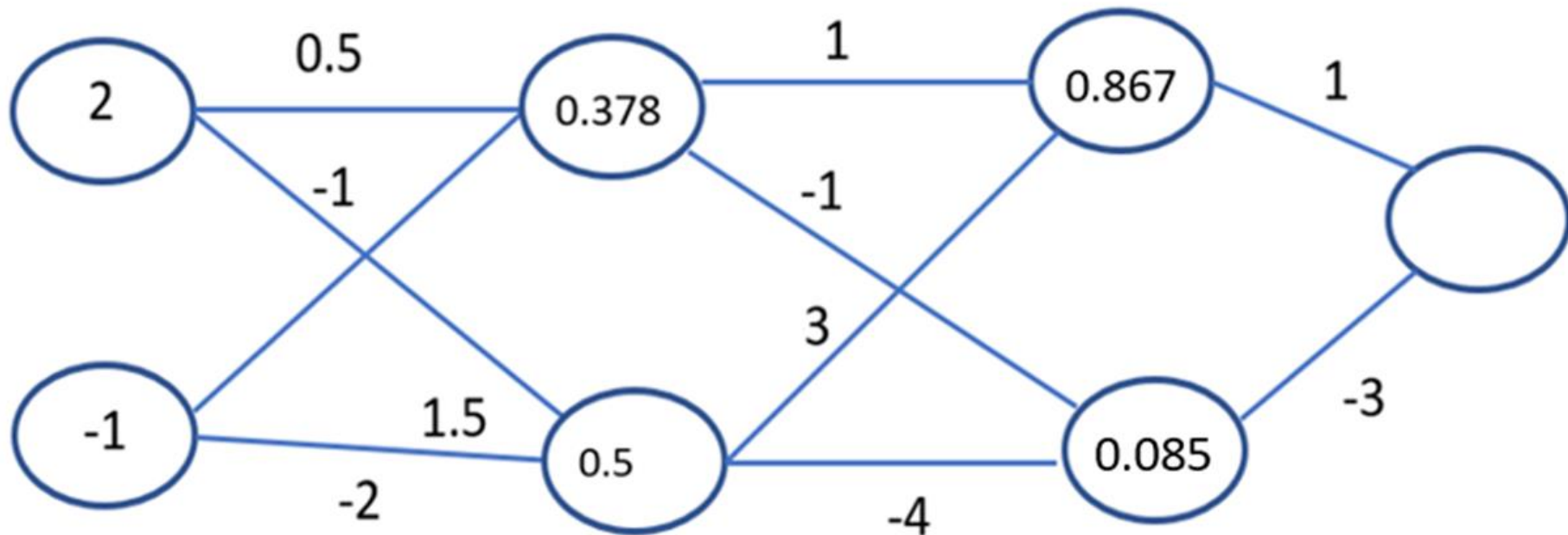
Chapitre 5 Les réseaux de neurones artificiels



● Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs

Exemple: $x = [2, -1]$, $y = 1$

Une itération



$$Z = 1 * 0.867 - 3 * 0.085 = 0.612$$

$$g(Z) = g(0.612) = \frac{1}{1 + e^{0.612}}$$

$$g(0.612) = 0.648$$

$$\hat{Y} = 0.648$$

Propagation

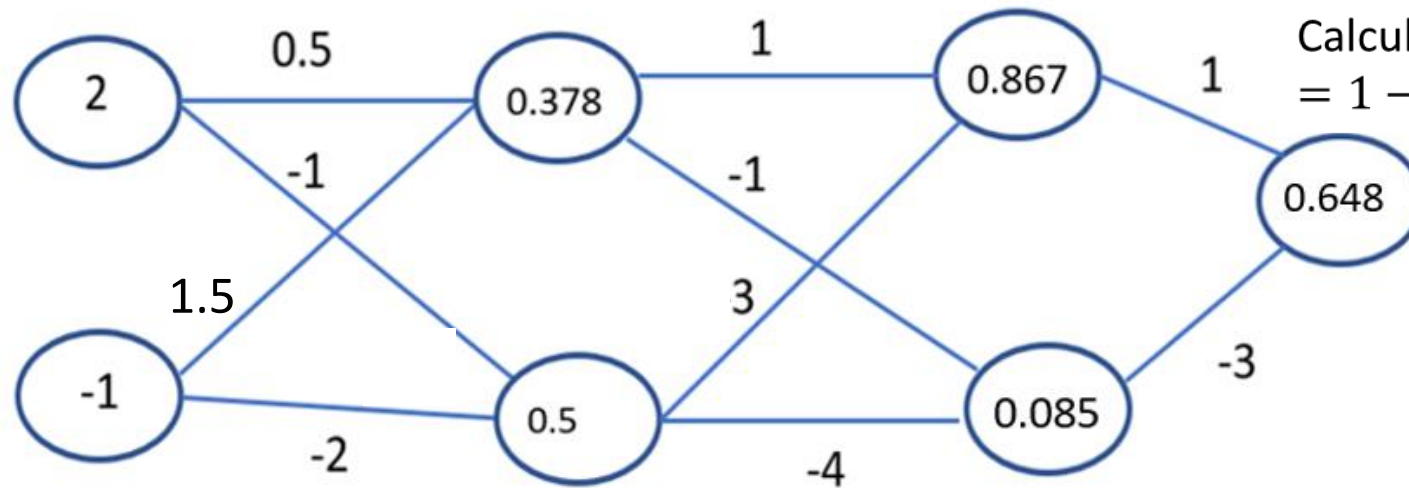
Chapitre 5 Les réseaux de neurones artificiels



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

Une itération

Exemple: $x = [2, -1]$, $y = 1$



Calcul de l'erreur $y - \hat{y}$
 $= 1 - 0.648 = 0.352$



Propagation

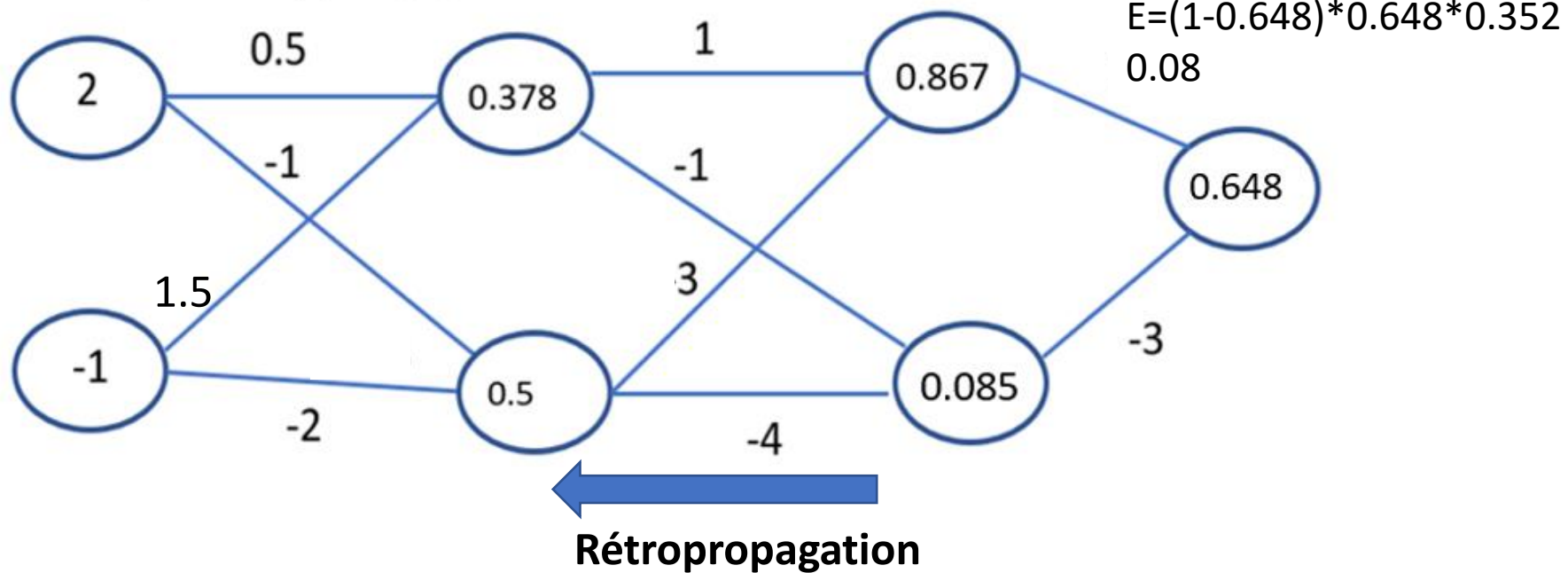
Chapitre 5 Les réseaux de neurones artificiels



- **Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs**

Une itération

Exemple: $x = [2, -1]$, $y = 1$



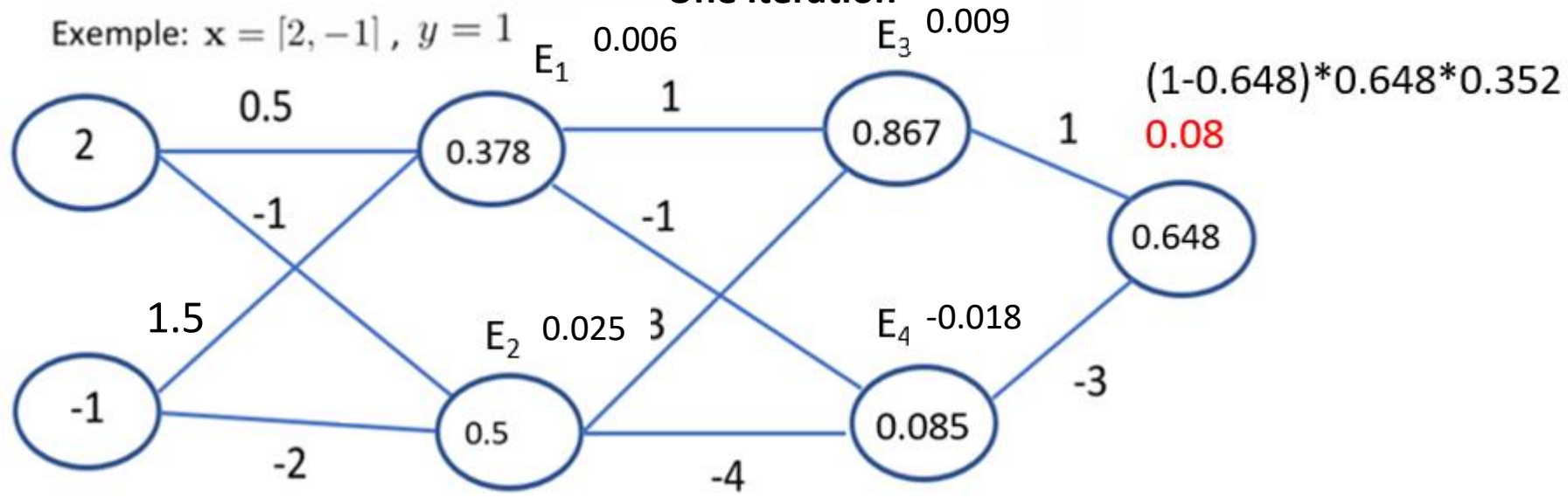
Chapitre 5 Les réseaux de neurones artificiels



● Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs

Une itération

Exemple: $x = [2, -1]$, $y = 1$



← **Rétropropagation** Calcul des erreurs de chaque neurone

$$E_1 = (1 - 0.378) * 0.378 * (1 * 0.009 + -1 * -0.018) \\ 0.006$$

$$E_3 = (1 - 0.867) * 0.867 * 1 * 0.08 \\ 0.009$$

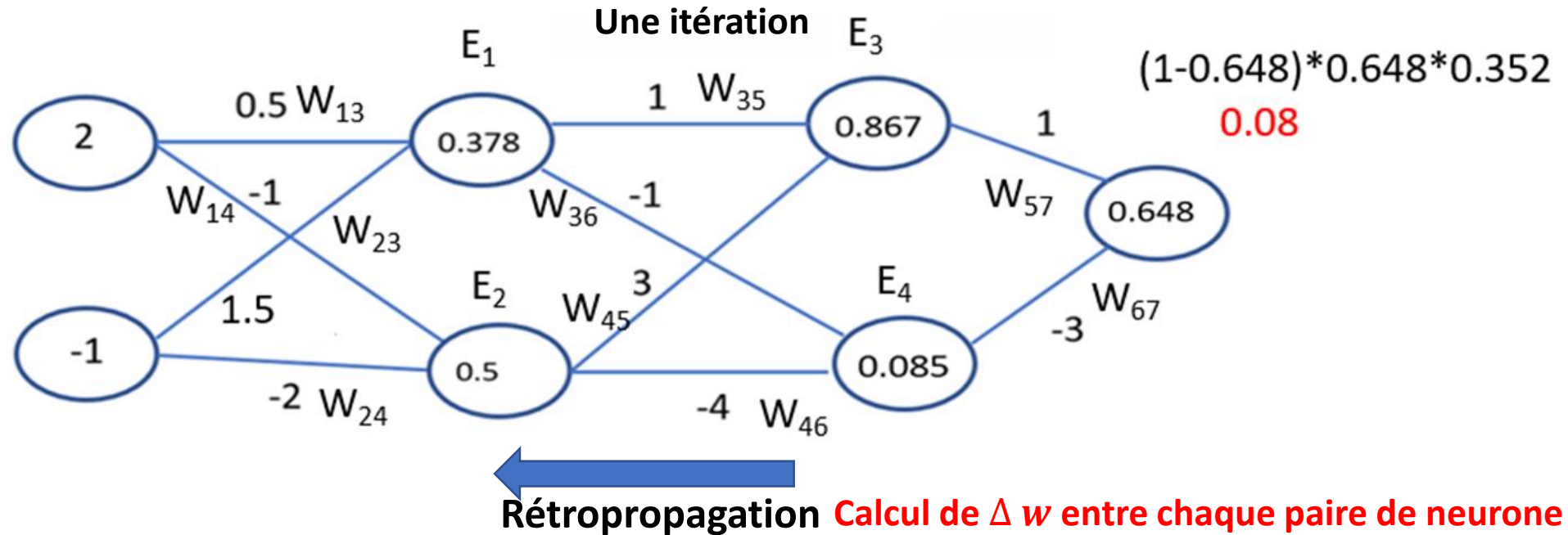
$$E_2 = (1 - 0.5) * 0.5 * (3 * 0.009 + -4 * -0.018) \\ 0.025$$

$$E_4 = (1 - 0.085) * 0.085 * -3 * 0.08 \\ -0.018$$

Chapitre 5 Les réseaux de neurones artificiels



● Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs



$$\Delta W_{13} = 0.01 * 2 * 0.006$$

$$\Delta W_{14} = 0.01 * 2 * 0.025$$

$$\Delta W_{23} = 0.01 * -1 * 0.006$$

$$\Delta W_{24} = 0.01 * -1 * 0.025$$

$$\Delta W_{35} = 0.01 * 0.378 * 0.009$$

$$\Delta W_{36} = 0.01 * 0.378 * 0.018$$

$$\Delta W_{45} = 0.01 * 0.5 * 0.009$$

$$\Delta W_{46} = 0.01 * 0.5 * 0.018$$

$$\Delta W_{57} = 0.01 * 0.867 * 0.08$$

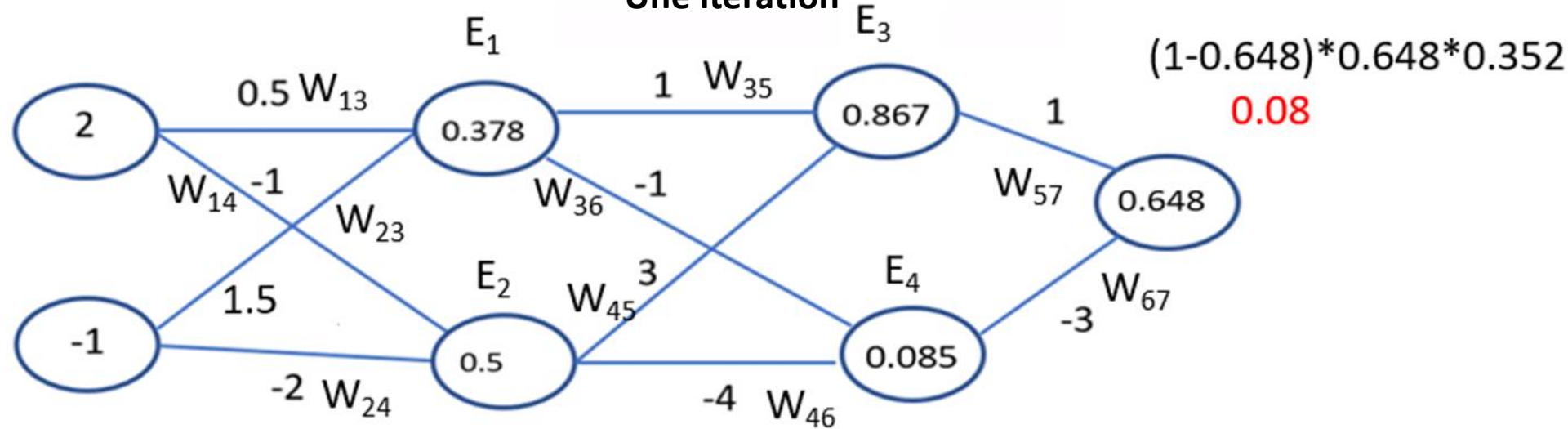
$$\Delta W_{67} = 0.01 * 0.085 * 0.08$$

Chapitre 5 Les réseaux de neurones artificiels



● Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs

Une itération



Rétropropagation Calcul de Δw entre chaque paire de neurone

$$\Delta W_{13} = 0.0012$$

$$\Delta W_{14} = 0.0005$$

$$\Delta W_{23} = -0.0006$$

$$\Delta W_{24} = -0.00025$$

$$\Delta W_{35} = 0.000068$$

$$\Delta W_{36} = 0.000068$$

$$\Delta W_{45} = 0.000045$$

$$\Delta W_{46} = 0.00009$$

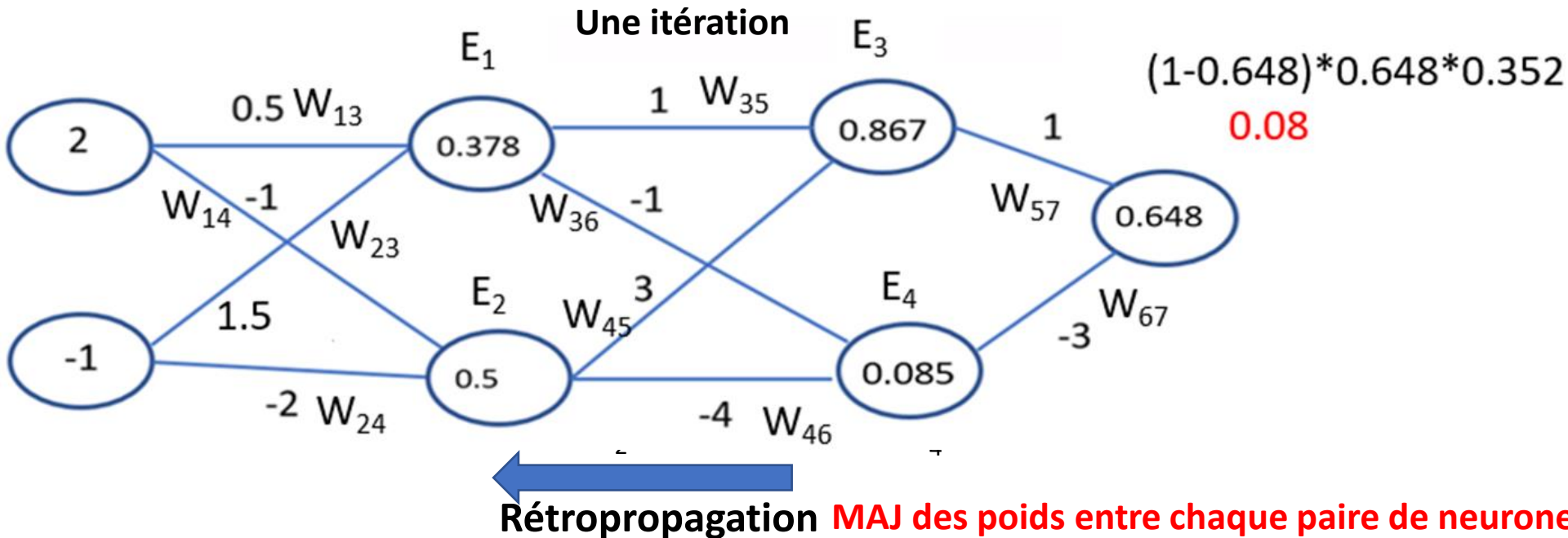
$$\Delta W_{57} = 0.0007$$

$$\Delta W_{67} = 0.000068$$

Chapitre 5 Les réseaux de neurones artificiels



● Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs



$$W_{13} = W_{13} + \Delta W_{13}$$

$$W_{14} = W_{14} + \Delta W_{14}$$

$$W_{23} = W_{23} + \Delta W_{23}$$

$$W_{24} = W_{24} + \Delta W_{24}$$

$$W_{35} = W_{35} + \Delta W_{35}$$

$$W_{36} = W_{36} + \Delta W_{36}$$

$$W_{45} = W_{45} + \Delta W_{45}$$

$$W_{46} = W_{46} + \Delta W_{46}$$

$$W_{57} = W_{57} + \Delta W_{57}$$

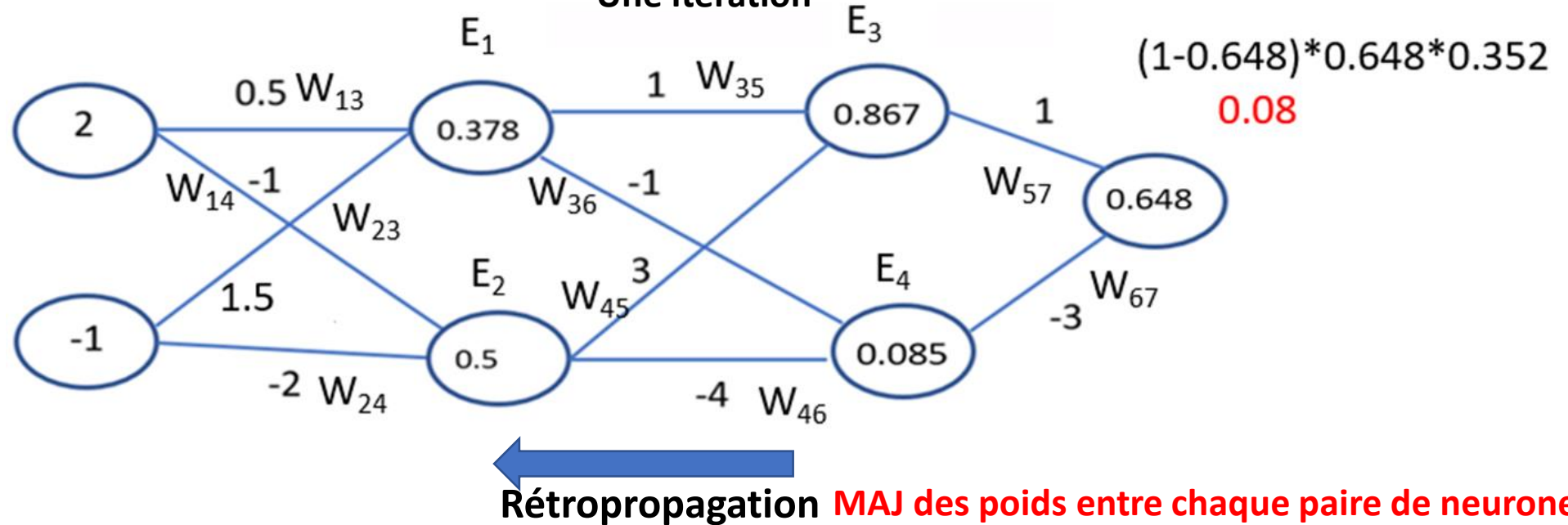
$$W_{67} = W_{67} + \Delta W_{67}$$

Chapitre 5 Les réseaux de neurones artificiels



● Perceptron Multicouches: Apprentissage par rétropropagation d'erreurs

Une itération



$$\begin{aligned} W_{13} &= 0.5012 \\ W_{14} &= -0.9995 \\ W_{23} &= 1.4994 \\ W_{24} &= -2.00025 \end{aligned}$$

$$\begin{aligned} W_{35} &= 1.000068 \\ W_{36} &= -0.999932 \\ W_{45} &= 3.000045 \\ W_{46} &= -3.99991 \end{aligned}$$

$$\begin{aligned} W_{57} &= 1.0007 \\ W_{67} &= -2.999932 \end{aligned}$$