Cours-TD d’introduction
à l’Intelligence Artificielle
Partie IV

Les réseaux multi-couches

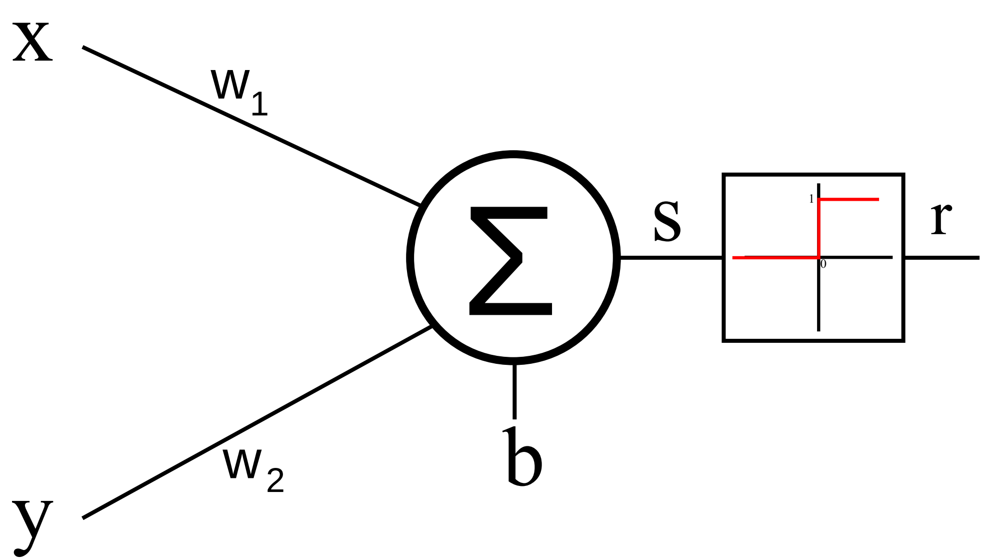
Simon Gay

Introduction à l’Intelligence Artificielle

- **Menu :**
  - Théorie :
    - Les limites du perceptron
    - Vers des réseaux multi-couches
    - Propagation du calcul
    - Descente de gradient et backpropagation
    - Mise à jour des poids synaptiques
    - Exemple

Introduction à l'Intelligence Artificielle

- **Les limites du perceptron**
  - Petite étude de cas : les portes logiques
    - On veut créer des portes logiques ET, OU et XOR avec des neurones formels
    - Un neurone formel par porte
      - Deux entrées x et y et un biais b
      - Fonction d'activation à seuil : 1 si somme >0, 0 sinon



Introduction à l’Intelligence Artificielle

- Porte OU

Table de vérité de OU		
0	0	0
0	1	1
1	0	1
1	1	1

- Une solution (parmi d’autres) : w1=3, w2=3, b=-1
  - $0 . 3 + 0 . 3 - 1 = -1 \rightarrow 0$
  - $0 . 3 + 1 . 3 - 1 = 2 \rightarrow 1$
  - $1 . 3 + 0 . 3 - 1 = 2 \rightarrow 1$
  - $1 . 3 + 1 . 3 - 1 = 5 \rightarrow 1$

Introduction à l'Intelligence Artificielle

- Porte ET

Table de vérité de ET		
0	0	0
0	1	0
1	0	0
1	1	1

- Une solution (parmi d'autres) : w1=2, w2=2, b=-3
  - 0 . 2 + 0 . 2 - 3 = -3 → 0
  - 0 . 2 + 1 . 2 - 3 = -1 → 0
  - 1 . 2 + 0 . 2 - 3 = -1 → 0
  - 1 . 2 + 1 . 2 - 3 = 1 → 1

Introduction à l'Intelligence Artificielle

- **Porte OU exlusif**

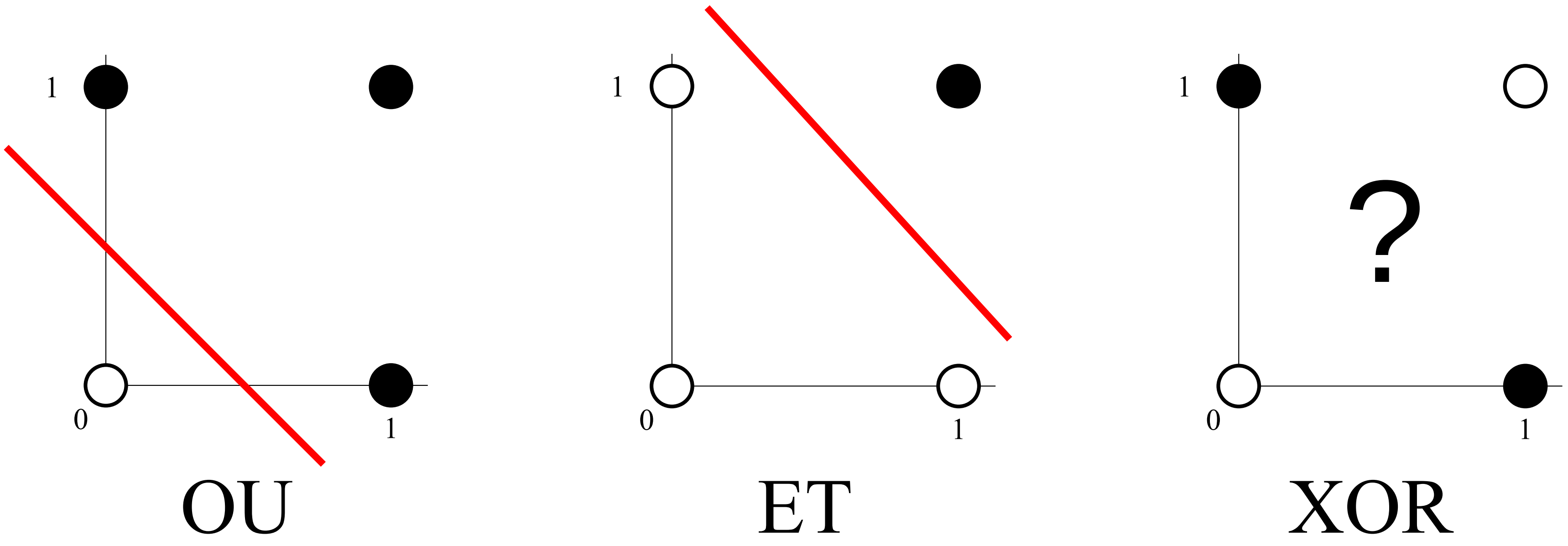
Table de vérité de XOR		
0	0	0
0	1	1
1	0	1
1	1	0

- **Solution :**

?

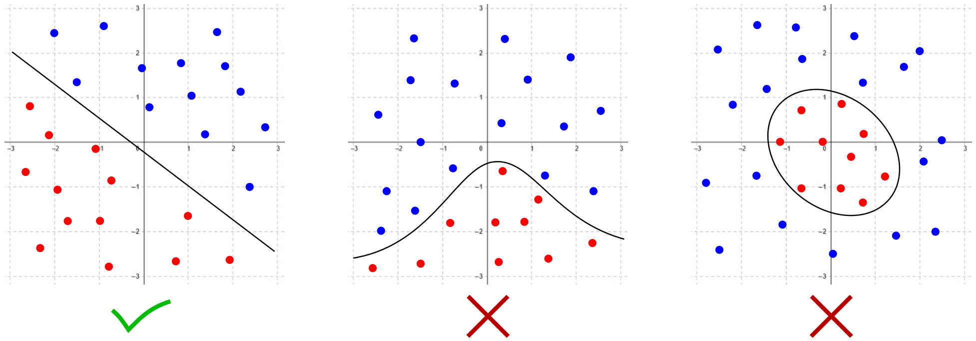
Introduction à l'Intelligence Artificielle

- **Explication**



Introduction à l’Intelligence Artificielle

- **Le perceptron est limité aux problèmes linéaires**



- En 1969, sortie du livre *Perceptrons* (de Marvin Minsky et Seymour Papert) sous-entendant que les perceptrons ne pourront jamais dépasser cette limite

  → baisse des activités de recherche sur le sujet (époque des IA symboliques)

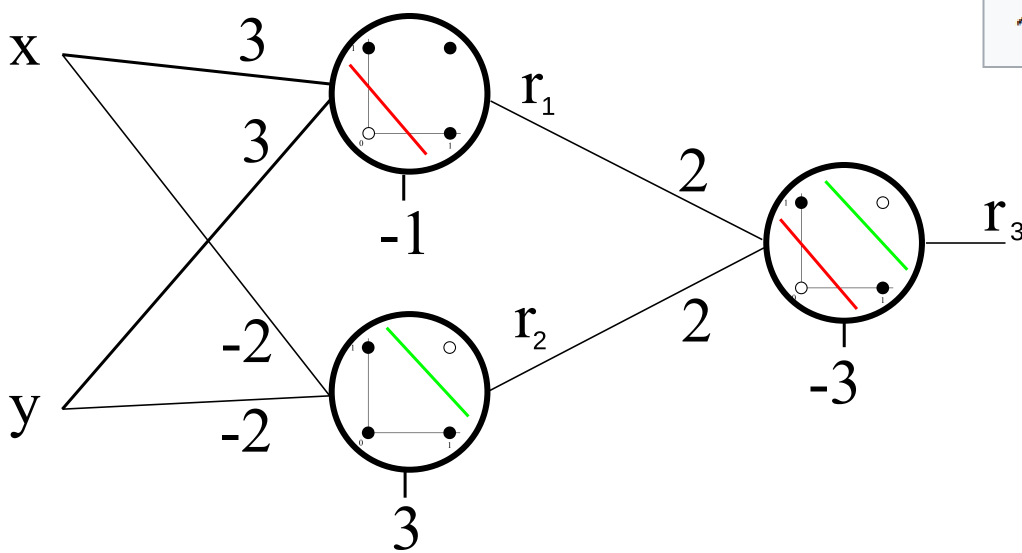
Introduction à l’Intelligence Artificielle

- **Solution pour le OU exlusif**

  → avec plusieurs couches

  de neurones

Table de vérité de XOR		
0	0	0
0	1	1
1	0	1
1	1	0

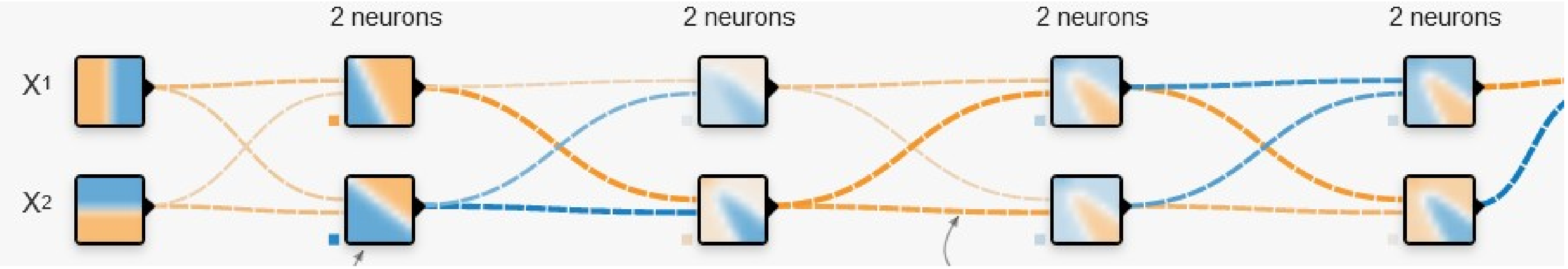


- Les neurones des couches suivantes peuvent combiner les « séparations » des couches précédentes

  → on peut créer des réseaux avec plusieurs couches pour résoudre des problèmes non linéaires !

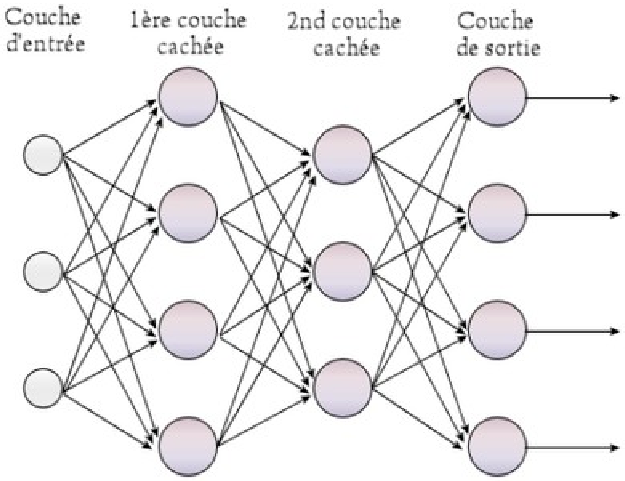
Introduction à l’Intelligence Artificielle

- **En combinant les « régions » de chaque neurone, on peut définir des séparations plus complexes**



Introduction à l'Intelligence Artificielle

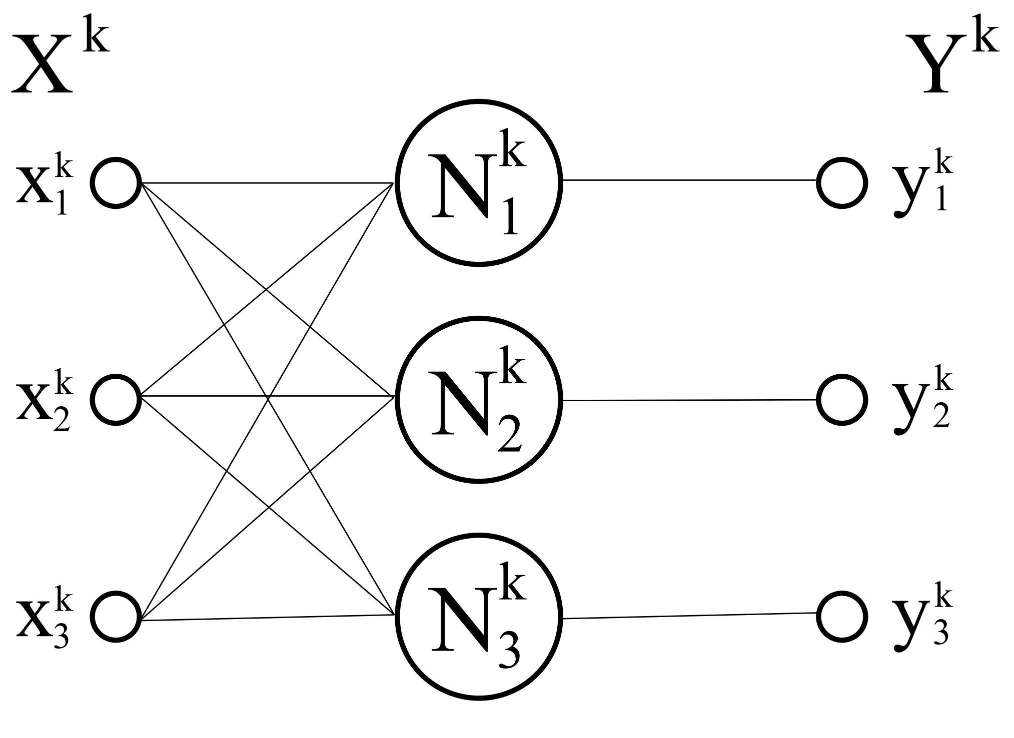
- **Réseaux mutli-couches**
  - Premier réseau multi-couche (perceptron multi-couche) créé en 1986 par David Rumelhart (presque 30 ans après le perceptron!)
  - Composés de :
    - Une couche d'entrées (vecteur d'entrées)
    - Une ou plusieurs couches de neurones cachés
    - Une couche de neurone de sortie



  - Comment calculer les valeurs de sortie ?
  - Comment mettre à jour les poids ?

Introduction à l'Intelligence Artificielle

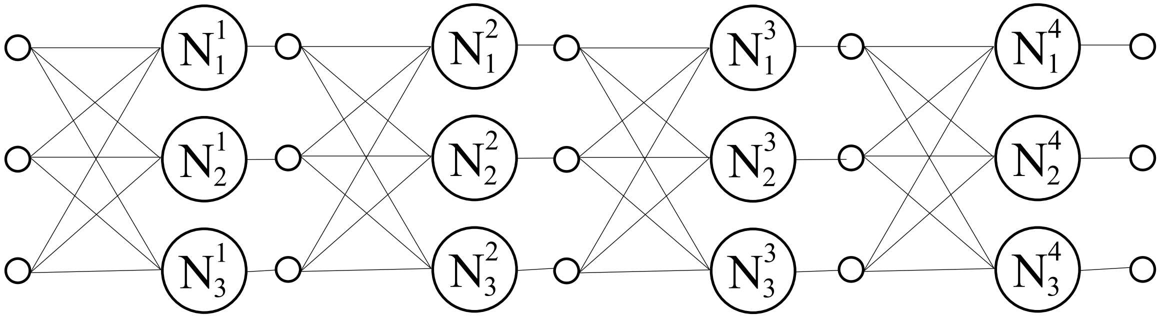
- **Comment calculer les valeurs de sortie ? Normalisons les notations**
  - On note N_i^k le i^ème^ neurone de la couche k
  - Un neurone prend en entrée un vecteur $X^k = \{x_1^k, x_2^k, \dots, x_n^k\}$ et a une sortie y_i^k
    - Chaque couche a en entrée un vecteur X^k et un vecteur de sortie Y^k



Introduction à l'Intelligence Artificielle

- **Comment calculer les valeurs de sortie ?**
  - Le réseau prend en entrée le vecteur X^1
  - La première couche cachée calcule les sorties y_i^1 de ses neurones pour définir le vecteur de sortie Y^1
  - La couche suivante prend en entrée la couche de sortie de la couche précédente
    - Ainsi, $X^k = Y^{k-1}$ (et $x_i^k = y_i^{k-1}\ \forall i$)
  - On répète pour chaque couche jusqu'à la couche de sortie

X^1



Y^4

$Y^1 = X^2$ $Y^2 = X^3$ $Y^3 = X^4$

→ pas de changements par rapport au perceptron

Introduction à l'Intelligence Artificielle

- **Comment mettre à jour les poids du réseau ?**

  → là, c'est plus compliqué !

  - Il faut, pour chaque poids, déterminer de combien il faut modifier le poids, mais également définir l'importance d'un tel changement
    - Chaque poids va influencer l'ensemble des neurones des couches suivantes
  - On va chercher à déterminer le changement d'un poids sur les sorties
    - On pourra ainsi réduire l'erreur par rapport à la sortie voulue

$$\frac{\partial y_i^L}{\partial w_{ij}^k}$$

  - Algorithme de la *descente de gradient*

Introduction à l'Intelligence Artificielle

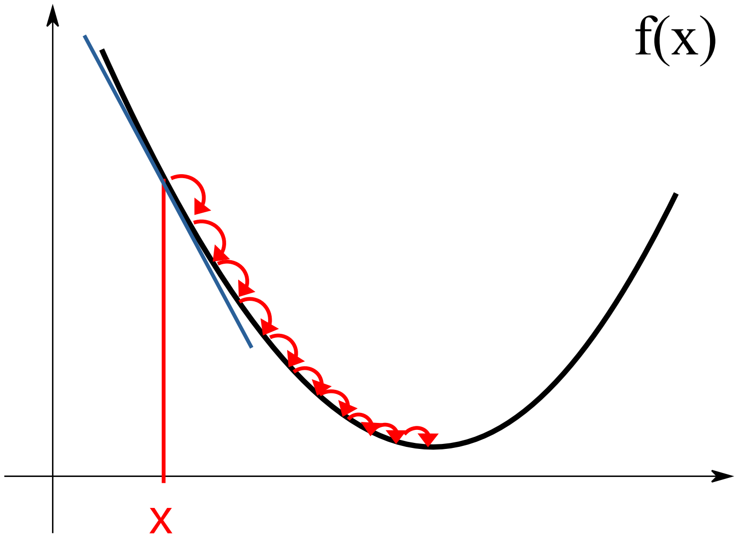
- **La descente de gradient**

  - Petite illustration : randonnée en montagne
    - Soudain : un blizzard, visibilité réduite à quelques mètres
    - Le refuge est au point le plus bas de la vallée
    - Vous êtes toujours dans la vallée

  - Comment revenir au refuge ?
    - Déterminer la direction de la plus grande descente
    - Avancer d'une centaine de mètre
    - Recommencer jusqu'à arriver en bas de la vallée

Introduction à l'Intelligence Artificielle

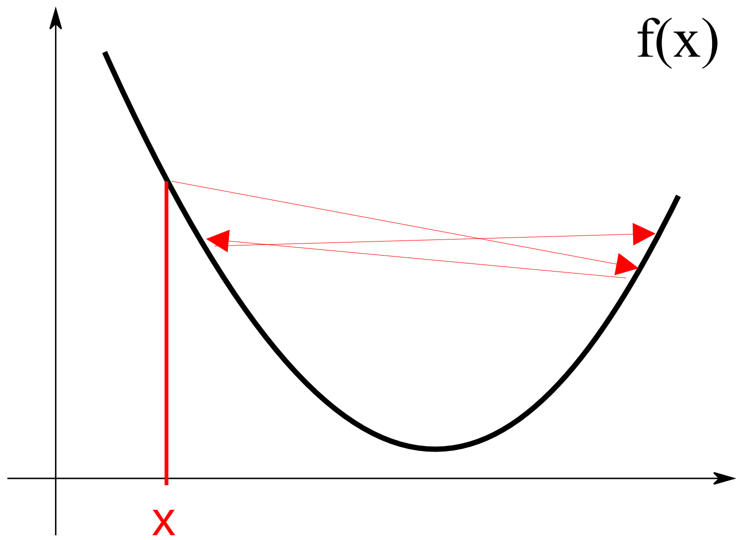
- **La descente de gradient**

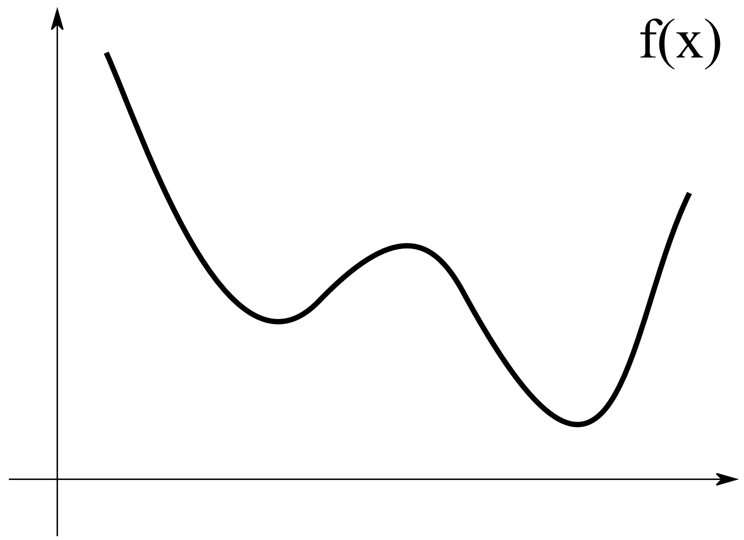


  - Une fonction f à minimiser (min($f_{(x)}$))
  - On initialise un x^0 au hasard
  - On calcule le gradient de f en x (soit $f'_{(x)}$)
  - On applique une mise à jour de x définie par : $x^{t+1} = x^t - \alpha . \frac{df(x)}{dx}$
  - On répète jusqu'à convergence

Introduction à l’Intelligence Artificielle

- **La descente de gradient**
  - Quel α choisir ?
    - Trop grand, on risque d’ociller autours du minimum
    - Trop petit, la convergence nécessitera un nombre trop important d’itérations
  - Problème des minimums locaux
    - En pratique, avec un grand nombre de dimension, il est très rare de trouver un point qui soit minimum sur l’ensemble des dimensions





Introduction à l'Intelligence Artificielle

- **La descente de gradient appliquée aux réseaux de neurones**

  – On définit une fonction d'erreur :

$$E(y) = \frac{1}{2} . (r - y)^2$$

  – Pour chaque poids w_{ij}^{l} , on va chercher à déterminer son impact sur E

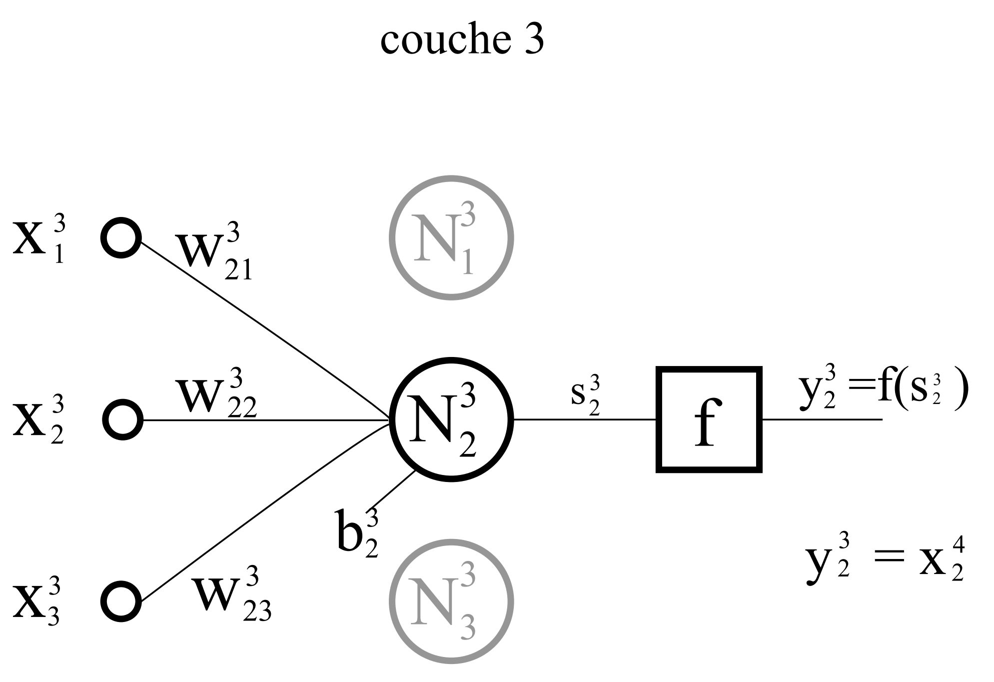
    - Descente de gradient sur w_{ij}^{l}

$$w_{ij}^{l} \Leftarrow w_{ij}^{l} - \alpha . \frac{\partial E(y)}{\partial w_{ij}^{l}}$$

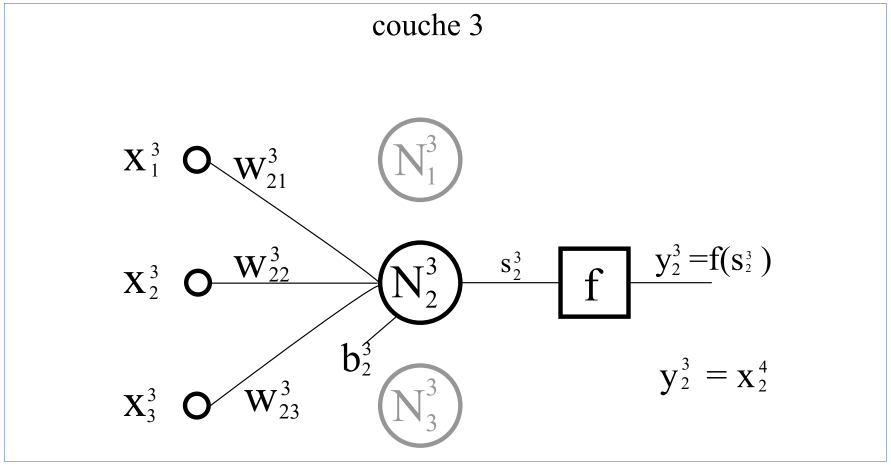
    - On doit donc calculer $\frac{\partial E(y)}{\partial w_{ij}^{l}}$

Introduction à l’Intelligence Artificielle

- **Calcul du gradient : petite mise au point sur les notations**
  - On note l une couche, i l’index d’un neurone, L la couche de sortie.
  - Un poids w_{ij}^l connecte le neurone N_i^l et le neurone N_j^{l-1} ($N_j^{l-1} \rightarrow N_i^l$)
  - On note $s_i^l = s(W_i^l, X^l, b_i^l) = \Sigma\, w_{ij}^l \,.\, x_i^l + b_i^l$ la somme pondérée du neurone N_i^l de la couche l.
  - On note $y_i^l = f(s_i^l)$ la sortie du neurone N_i^l après la fonction d’activation f



Introduction à l'Intelligence Artificielle



- **Calcul du gradient :**
  cas de la dernière couche L
  - On cherche à calculer

$$\frac{\partial E(y_i^L)}{\partial w_{ij}^L}$$

  - On décompose avec les variables intermédiaires
    - Théorème de dérivation des fonctions composées

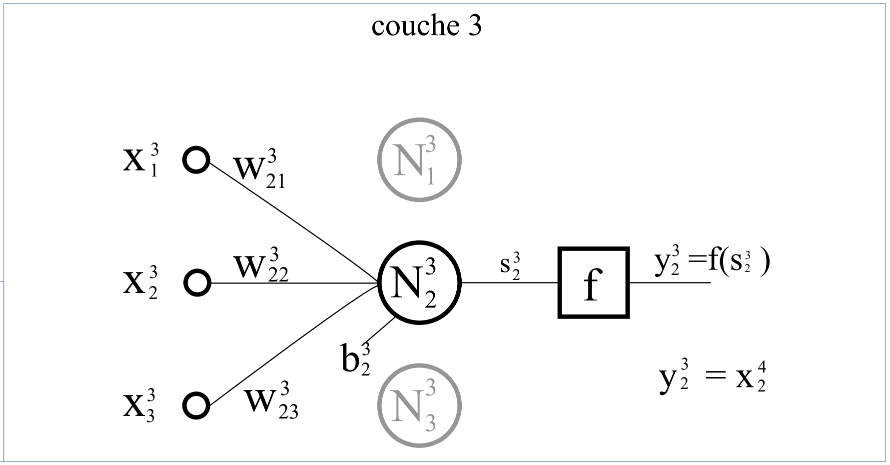
$$\frac{\partial E(y_i^L)}{\partial w_{ij}^L} = \frac{\partial E(y_i^L)}{\partial y_i^L} \cdot \frac{\partial y_i^L}{\partial s_i^L} \cdot \frac{\partial s_i^L}{\partial w_{ij}^L}$$

    - Trois dérivées à calculer

Introduction à l'Intelligence Artificielle

- **Calcul du gradient :**
  cas de la dernière couche L

$$\frac{\partial E(y_i^L)}{\partial w_{ij}^L} = \frac{\partial E(y_i^L)}{\color{red}\partial y_i^L} \cdot \frac{\color{red}\partial y_i^L}{\color{green}\partial s_i^L} \cdot \frac{\color{green}\partial s_i^L}{\partial w_{ij}^L}$$



- Premier terme : $\dfrac{\partial E(y_i^L)}{\partial y_i^L}$

- Fonction d'erreur : $E(y) = \dfrac{1}{2}.(r - y)^2$

Dérivation des fonctions Composées :
(f o g)' = (f' o g).g'

-1

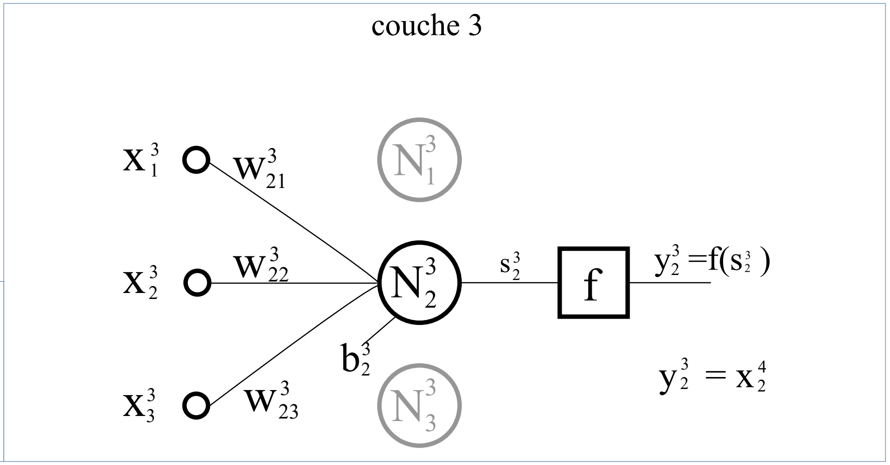
$$\frac{\partial E(y)}{\partial y} = \frac{1}{2}.\frac{\partial (r-y)^2}{\partial y} = \cancel{\frac{2}{2}}.(r-y).\frac{\partial (r-y)}{\partial y}$$

$$\boxed{\frac{\partial E(y)}{\partial y} = -(r-y)}$$

Introduction à l’Intelligence Artificielle

- **Calcul du gradient :**
  cas de la dernière couche L

$$\frac{\partial E\left(y_i^L\right)}{\partial w_{ij}^L} = \underline{\frac{\partial E\left(y_i^L\right)}{\partial y_i^L}} \cdot \frac{\partial y_i^L}{\partial s_i^L} \cdot \frac{\partial s_i^L}{\partial w_{ij}^L}$$



- second terme : $\dfrac{\partial y_i^L}{\partial s_i^L} = \dfrac{\partial f\left(s_i^L\right)}{\partial s_i^L}$

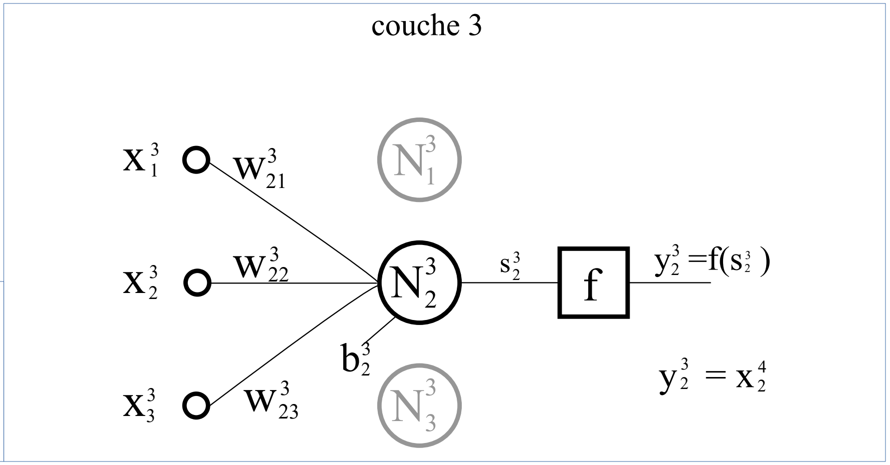
- Simplement la dérivée de la fonction d’activation !

$$\boxed{\frac{\partial y_i^L}{\partial s_i^L} = f'_{\left(s_i^L\right)}}$$

Introduction à l’Intelligence Artificielle

- **Calcul du gradient :**
  cas de la dernière couche L

$$\frac{\partial E(y_i^L)}{\partial w_{ij}^L} = \underline{\frac{\partial E(y_i^L)}{\partial y_i^L}} \cdot \underline{\frac{\partial y_i^L}{\partial s_i^L}} \cdot \frac{\partial s_i^L}{\partial w_{ij}^L}$$



- troisième terme : $\dfrac{\partial s_i^L}{\partial w_{ij}^L}$

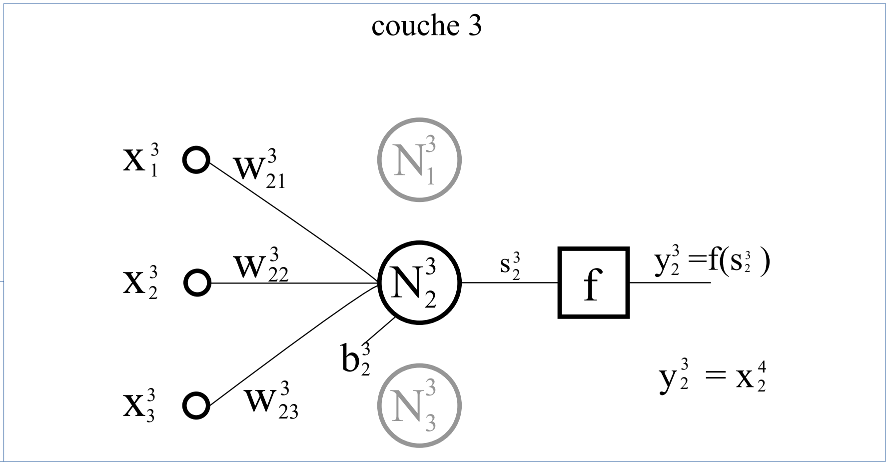
‘constantes’

$$\frac{\partial s_i^L}{\partial w_{ij}^L} = \frac{\partial (w_{i1}^L . x_1^L + w_{i2}^L . x_2^L + \ldots + w_{ij}^L . x_i^L + \ldots + b_i^L)}{\partial w_{ij}^L} = \frac{\partial (w_{ij}^L . x_j^L)}{\partial w_{ij}^L}$$

$$\boxed{\frac{\partial s_i^L}{\partial w_{ij}^L} = x_j^L}$$

Introduction à l’Intelligence Artificielle

- **Calcul du gradient :**
  cas de la dernière couche L



$$\frac{\partial E(y_i^L)}{\partial w_{ij}^L} = \underline{\frac{\partial E(y_i^L)}{\partial y_i^L}} \cdot \underline{\frac{\partial y_i^L}{\partial s_i^L}} \cdot \underline{\frac{\partial s_i^L}{\partial w_{ij}^L}}$$

- On récapitule :

$$\frac{\partial E(y_i^L)}{\partial w_{ij}^L} = -(r - y_i^L) \;.\; f'_{(s_i^L)} \;.\; x_j^L$$

- Descente du gradient : $w_{ij}^L \Leftarrow w_{ij}^L + \alpha . x_j^L . (r - y_i^L) . f'_{(s_i^L)}$

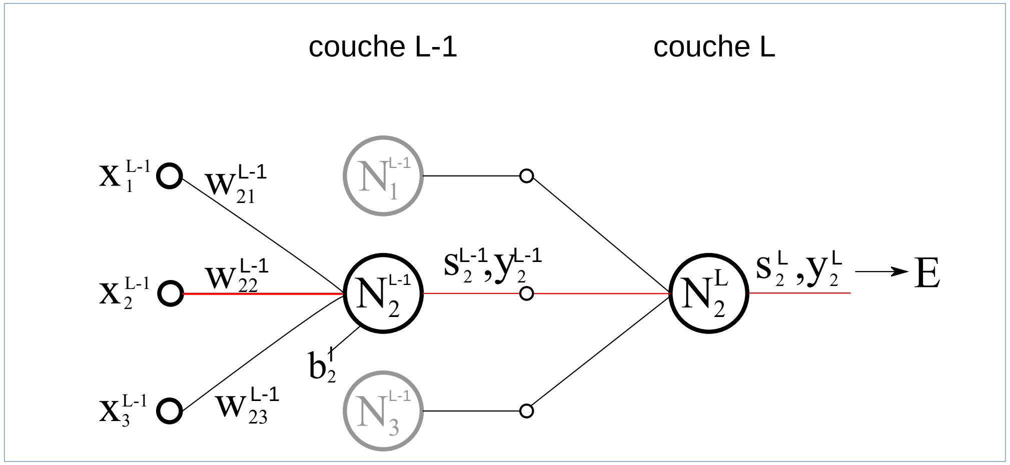
- On pose $\delta_i^L = (r - y_i^L) \;.\; f'_{(s(\ldots))}$

À noter :

$$\delta_i^L = -\frac{\partial E(y_i^L)}{\partial s_i^L}$$

$\rightarrow$ $\boxed{w_{ij}^L \Leftarrow w_{ij}^L + \alpha . x_j^L . \delta_i^L}$

Introduction à l'Intelligence Artificielle



- **Calcul du gradient :**
  cas de la couche cachée L-1
  - On cherche à calculer

$$\frac{\partial E\left(y_m^L\right)}{\partial w_{ij}^{L-1}}$$

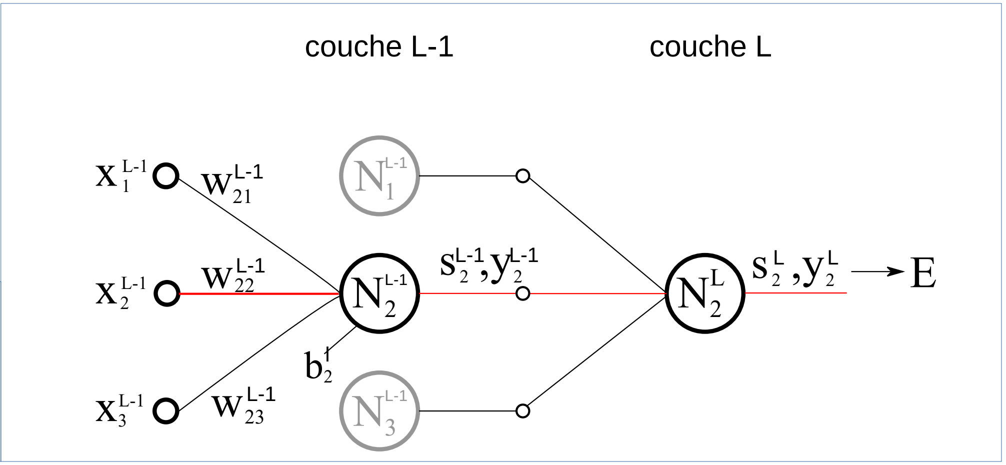
  - On décompose à nouveau :

$$\frac{\partial E\left(y_m^L\right)}{\partial w_{ij}^{L-1}} = \frac{\partial E\left(y_m^L\right)}{\partial y_i^{L-1}} \cdot \frac{\partial y_i^{L-1}}{\partial s_i^{L-1}} \cdot \frac{\partial s_i^{L-1}}{\partial w_{ij}^{L-1}}$$

Introduction à l'Intelligence Artificielle

- **Calcul du gradient :**
  cas de la couche cachée L-1

$$\frac{\partial E(y_m^L)}{\partial w_{ij}^{L-1}} = \frac{\partial E(y_m^L)}{\partial y_i^{L-1}} \cdot \frac{\partial y_i^{L-1}}{\partial s_i^{L-1}} \cdot \frac{\partial s_i^{L-1}}{\partial w_{ij}^{L-1}}$$



- Les deuxième et troisième termes ne changent pas

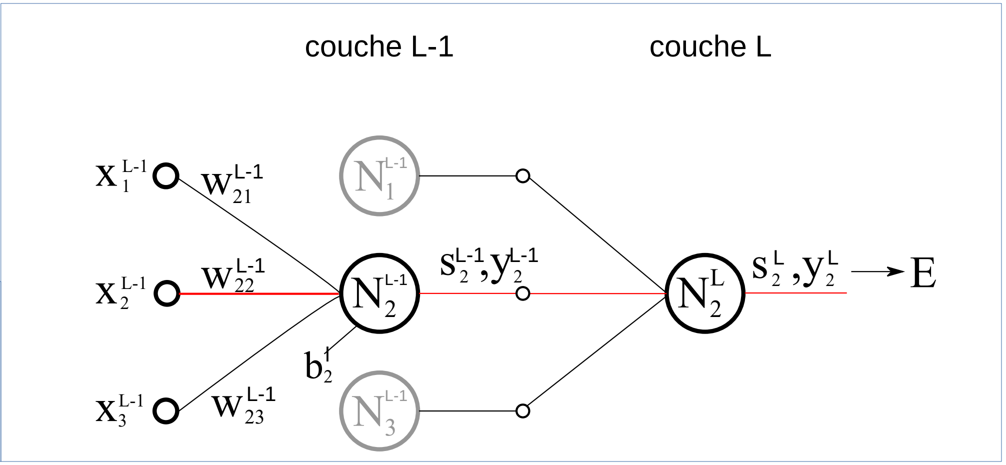
$$\boxed{\frac{\partial y_i^{L-1}}{\partial s_i^{L-1}} = f'_{(s_i^{L-1})}}$$

$$\boxed{\frac{\partial s_i^{L-1}}{\partial w_{ij}^{L-1}} = x_j^{L-1}}$$

Introduction à l'Intelligence Artificielle

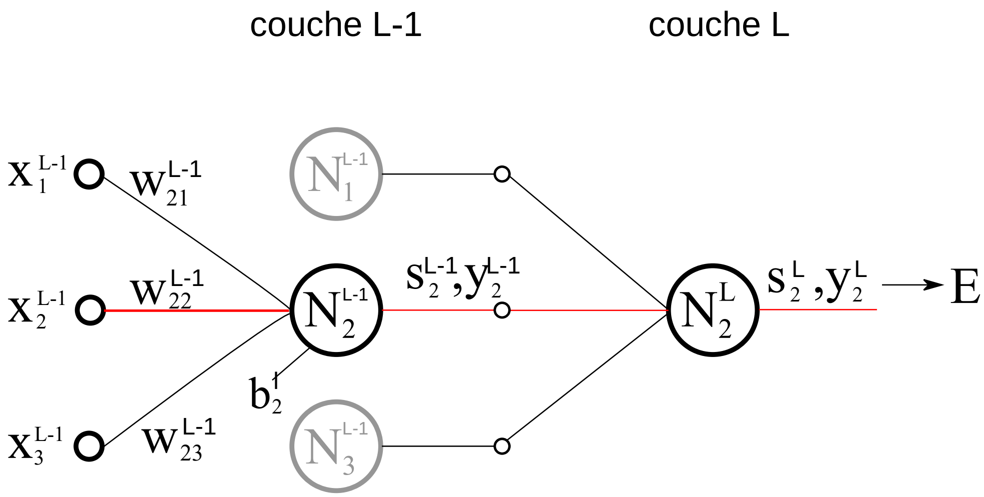
- **Calcul du gradient :**
  cas de la couche cachée L-1

$$\frac{\partial E(y_m^L)}{\partial w_{ij}^{L-1}} = \frac{\partial E(y_m^L)}{\partial y_i^{L-1}} \cdot \frac{\partial y_i^{L-1}}{\partial s_i^{L-1}} \cdot \frac{\partial s_i^{L-1}}{\partial w_{ij}^{L-1}}$$



- Premier terme : $\frac{\partial E(y_m^L)}{\partial y_i^{L-1}}$

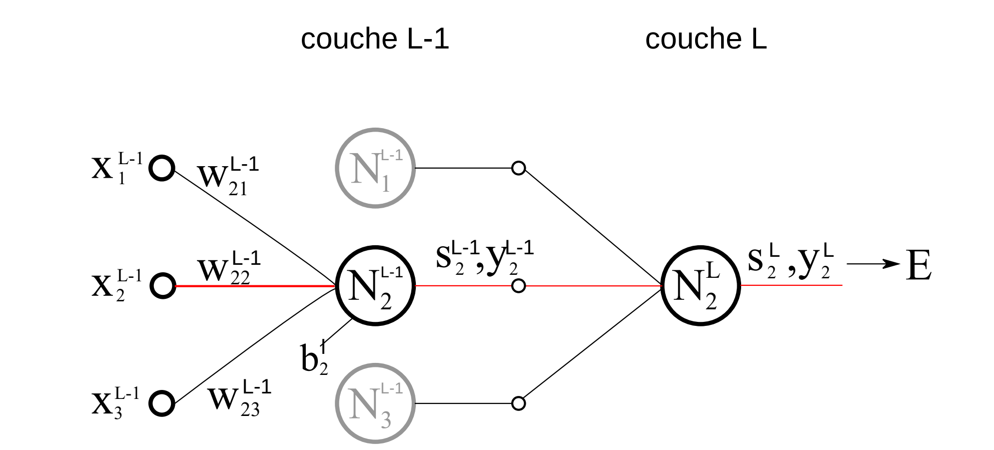
- Un peu plus complexe : si y_i^{L-1} est modifié, on modifie E via s_i^{L-1} (et donc y_j^{L-1}) de la couche L



Introduction à l’Intelligence Artificielle

- **Calcul du gradient :**
  cas de la couche cachée L-1

$$\frac{\partial E(y_m^L)}{\partial w_{ij}^{L-1}} = \frac{\partial E(y_m^L)}{\partial y_i^{L-1}} \cdot \frac{\partial y_i^{L-1}}{\partial s_i^{L-1}} \cdot \frac{\partial s_i^{L-1}}{\partial w_{ij}^{L-1}}$$



$$\frac{\partial E(y_m^L)}{\partial y_i^{L-1}} = \underbrace{\frac{\partial E(y_m^L)}{\partial s_m^L}}_{\text{C'est } -\delta_m^L \text{ !}} \cdot \frac{\partial s_m^L}{\partial y_i^{L-1}}$$

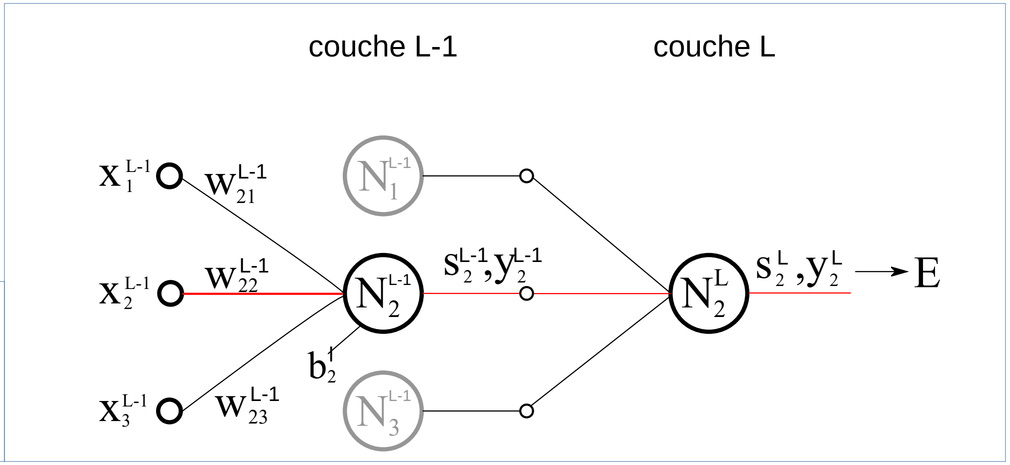
$$\frac{\partial s_m^L}{\partial y_i^{L-1}} = \frac{\partial \left(w_{m1}^L . x_1^L + w_{m2}^L . x_2^L + \ldots + w_{mi}^L . x_i^L + \ldots + b_m^L\right)}{\partial x_i^L} = w_{mi}^L$$

$$\boxed{\frac{\partial E(y_m^l)}{\partial y_i^{L-1}} = -\delta_m^L \; . \; w_{mi}^L}$$

Introduction à l'Intelligence Artificielle

- **Calcul du gradient :**
  cas de la couche cachée L-1

$$\frac{\partial E(y_m^L)}{\partial w_{ij}^{L-1}} = \frac{\partial E(y_m^L)}{\partial y_i^{L-1}} \cdot \frac{\partial y_i^{L-1}}{\partial s_i^{L-1}} \cdot \frac{\partial s_i^{L-1}}{\partial w_{ij}^{L-1}}$$



- On récapitule :

$$\frac{\partial E(y_i^L)}{\partial w_{ij}^{L-1}} = - \; f'_{(s_i^{L-1})} \; . \; x_j^{L-1} \; . \; \delta_m^L \; . \; w_{mi}^L$$

- Descente du gradient : $w_{ij}^{L-1} \Leftarrow w_{ij}^{L-1} + \alpha . x_j^{L-1} . f'_{s_i^{L-1}} \; . \; \delta_m^L \; . \; w_{mi}^L$

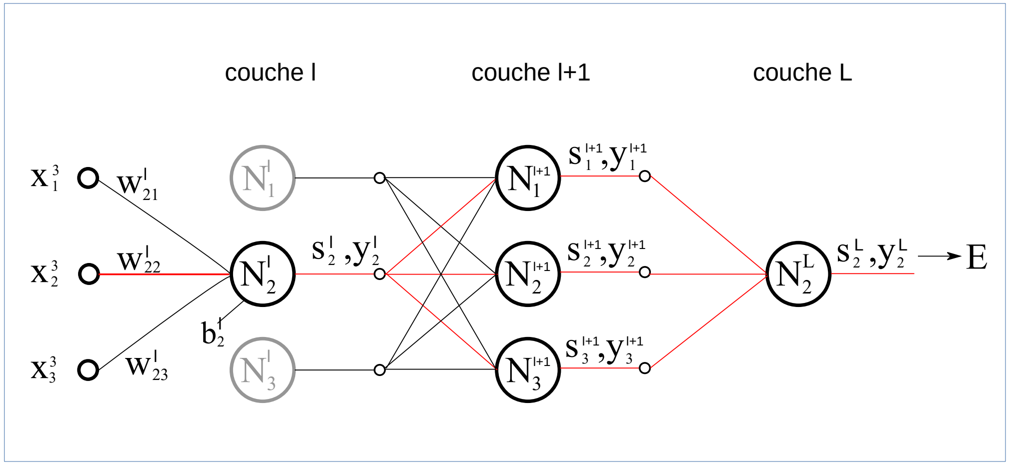
- On pose : $\delta_i^{L-1} = f'_{s_i^{L-1}} \; . \; \delta_m^L \; . \; w_{mi}^L$

À noter :

$$\delta_i^{L-1} = -\frac{\partial E(y_m^L)}{\partial s_i^{L-1}}$$

on a toujours $\boxed{w_{ij}^{L-1} \Leftarrow w_{ij}^{L-1} + \alpha . x_j^{L-1} . \delta_i^{L-1}}$

Introduction à l’Intelligence Artificielle



- **Calcul du gradient :**

  On généralise pour les couches l

  - On cherche à calculer

$$\frac{\partial E(y_m^L)}{\partial w_{ij}^l}$$

  - On décompose à nouveau :

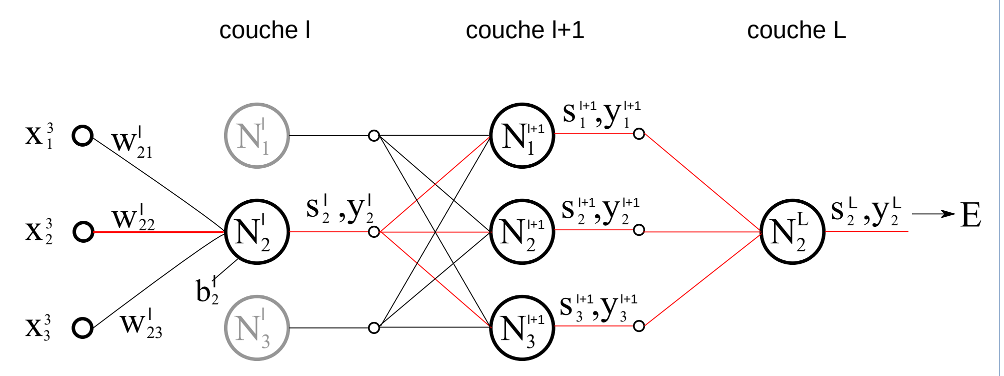
$$\frac{\partial E(y_m^L)}{\partial w_{ij}^l} = \frac{\partial E(y_m^L)}{\partial y_i^l} \cdot \frac{\partial y_i^l}{\partial s_i^l} \cdot \frac{\partial s_i^l}{\partial w_{ij}^l}$$

Introduction à l'Intelligence Artificielle

- **Calcul du gradient :**

  On généralise pour les couches l

$$\frac{\partial E(y_m^L)}{\partial w_{ij}^l} = \frac{\partial E(y_m^L)}{\partial y_i^l} \cdot \frac{\partial y_i^l}{\partial s_i^L} \cdot \frac{\partial s_i^L}{\partial w_{ij}^l}$$



- Toujours pas de changements pour les deuxième et troisième termes

$$\boxed{\frac{\partial y_i^l}{\partial s_i^l} = f'_{(s_i^l)}}$$

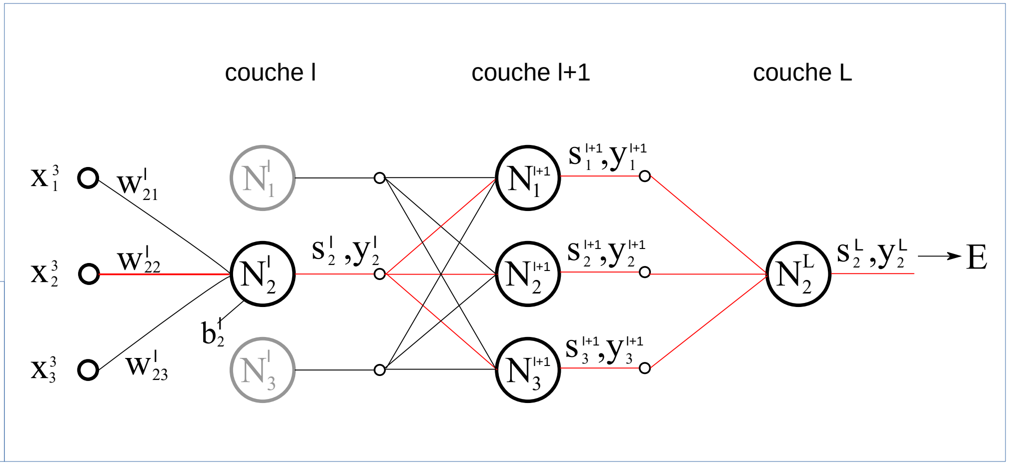
$$\boxed{\frac{\partial s_i^l}{\partial w_{ij}^l} = x_j^l}$$

Introduction à l’Intelligence Artificielle

- **Calcul du gradient :**

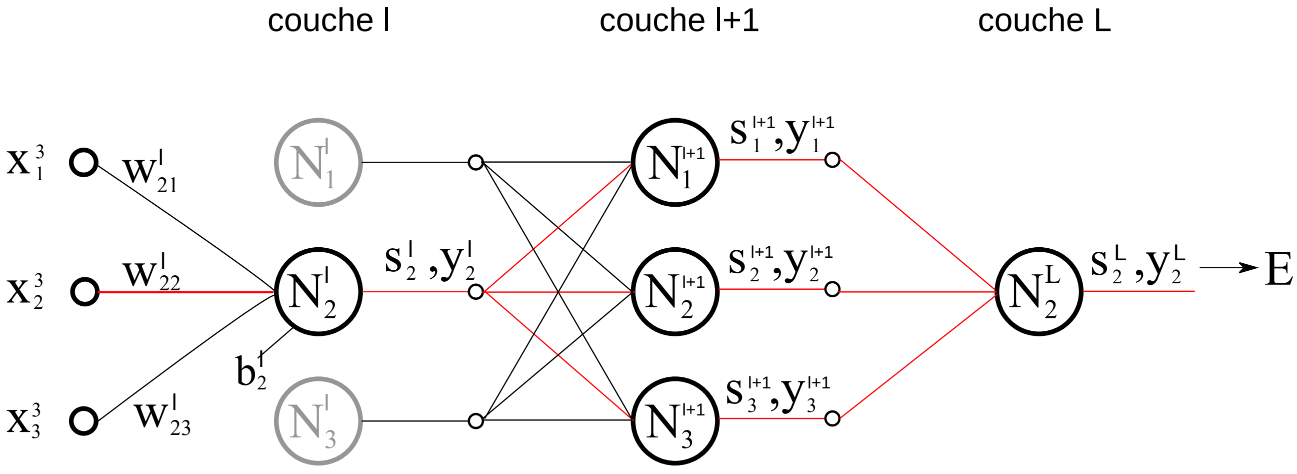
  On généralise pour les couches l

$$\frac{\partial E(y_m^L)}{\partial w_{ij}^l} = \frac{\partial E(y_m^L)}{\partial y_i^l} \cdot \frac{\partial y_i^l}{\partial s_i^L} \cdot \frac{\partial s_i^L}{\partial w_{ij}^l}$$



- Premier terme : $\dfrac{\partial E(y_m^L)}{\partial y_i^l}$

- Cette fois, si y_i^l est modifié, on modifie E via les sorties s (et donc y_j^{l+1}) de la couche suivante l+1

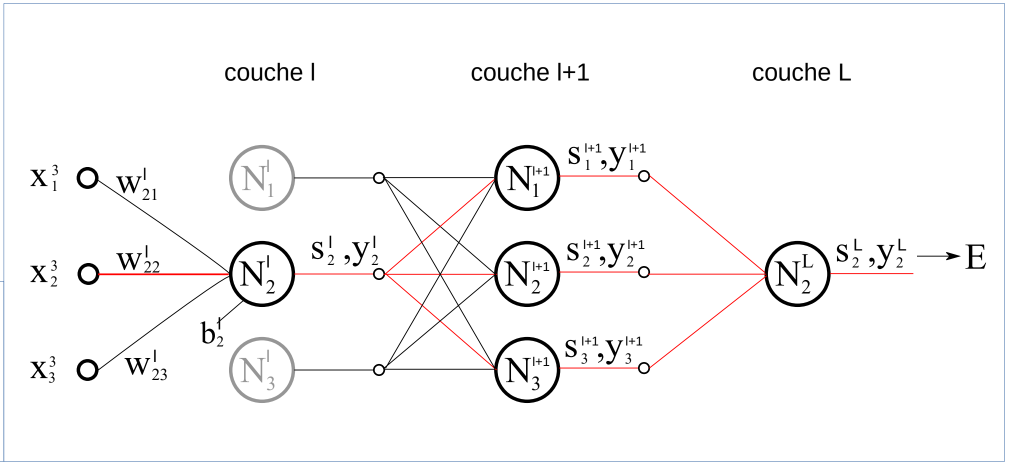


Introduction à l'Intelligence Artificielle

- **Calcul du gradient :**

  On généralise pour les couches l

$$\frac{\partial E\left(y_m^L\right)}{\partial w_{ij}^l} = \frac{\partial E\left(y_m^L\right)}{\partial y_i^l} \cdot \frac{\partial y_i^l}{\partial s_i^L} \cdot \frac{\partial s_i^L}{\partial w_{ij}^l}$$



$$\frac{\partial E\left(y_m^L\right)}{\partial y_i^l} = \frac{\partial E\left(y_m^L\right)}{\partial s_1^{l+1}} \cdot \frac{\partial s_1^{l+1}}{\partial y_i^l} + \frac{\partial E\left(y_m^L\right)}{\partial s_2^{l+1}} \cdot \frac{\partial s_2^{l+1}}{\partial y_i^l} + \dots + \frac{\partial E\left(y_m^L\right)}{\partial s_n^{l+1}} \cdot \frac{\partial s_n^{l+1}}{\partial y_i^l}$$

$$\frac{\partial E\left(y_m^l\right)}{\partial y_i^l} = \sum_{k\in[1,n]} \frac{\partial E\left(y_m^L\right)}{\partial s_k^{l+1}} \cdot \frac{\partial s_k^{l+1}}{\partial y_i^l}$$

w_{ki}^{l+1}

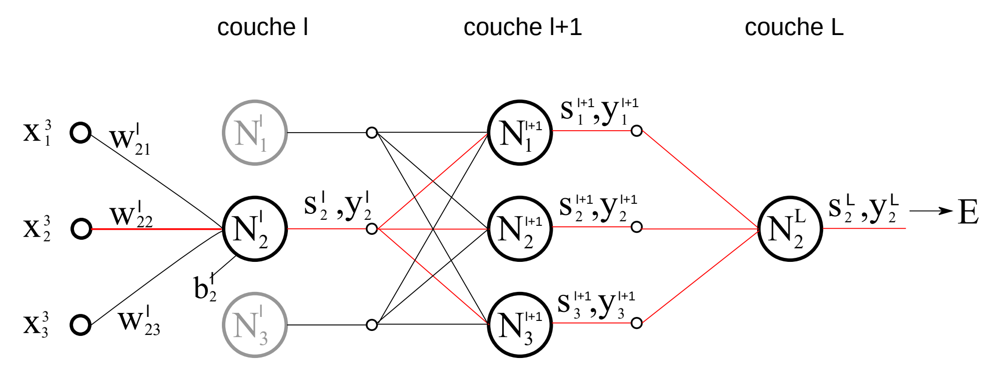
Ce sont les $-\delta_k^{l+1}$!

Introduction à l'Intelligence Artificielle

- **Calcul du gradient :**

  On généralise pour les couches l

$$\frac{\partial E\left(y_m^L\right)}{\partial w_{ij}^l} = \frac{\partial E\left(y_m^L\right)}{\partial y_i^l} \cdot \frac{\partial y_i^l}{\partial s_i^L} \cdot \frac{\partial s_i^L}{\partial w_{ij}^l}$$



- On récapitule (une dernière fois):

$$\frac{\partial E\left(y_i^L\right)}{\partial w_{ij}^l} = -f'_{\left(s_i^l\right)} \cdot x_j^l \cdot \sum_{k\in[1,n]} \delta_k^{l+1} \cdot w_{ki}^{l+1}$$

- Descente du gradient : $w_{ij}^l \Leftarrow w_{ij}^l + \alpha . x_j^l . f'_{\left(s_i^l\right)} \cdot \sum_{k\in[1,n]} \delta_k^{l+1} . w_{ki}^{l+1}$

- On pose : $\delta_i^l = f'_{s_i^l} \cdot \sum_{k\in[1,n]} \delta_k^{l+1} \cdot w_{ki}^{l+1}$

  et encore : $\boxed{w_{ij}^l \Leftarrow w_{ij}^l + \alpha . x_j^l . \delta_i^l}$

Introduction à l’Intelligence Artificielle

- **Apprentissage du réseau : 3 étapes**
  - Forward propagation : on calcule la valeur de sortie des neurones, depuis la couche d’entrée vers la couche de sortie :

$$y_i^l = f\left(\sum_{k \in [0,n]} w_{ik}^l . x_k^l + b_i^l \right)$$

  - Backward propagation : on calcule les valeurs de delta des neurones de la couche de sortie, puis on propage les deltas vers la couche d’entrée
    - Dernière couche $\delta_i^L = \left(r - y_i^L\right) . f'_{(s(\dots))}$
    - Couches cachées $\delta_i^l = f'_{s_i^l} . \sum_{k \in [1,n]} \delta_k^{l+1} . w_{ki}^{l+1}$
  - Mise à jour des poids à l’aide des deltas calculés précédemment

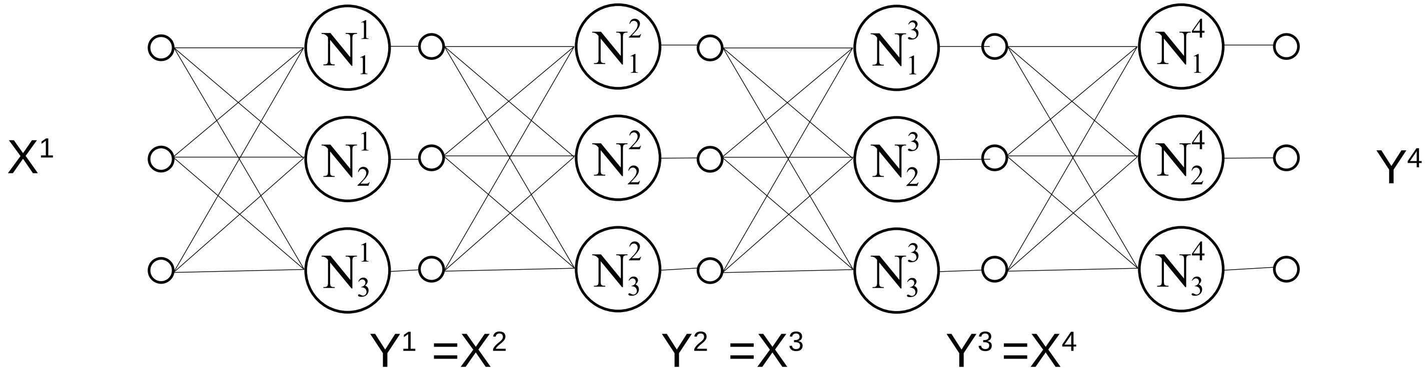
$$w_{ij}^l \Leftarrow w_{ij}^l + \alpha . x_j^l . \delta_i^l$$

$$b_i^l \Leftarrow b_i^l + \alpha . \delta_i^l$$

Introduction à l’Intelligence Artificielle

- **En pratique :**
  - On remarque que si les poids sont initialement à 0,
    - Les vecteurs Y^2 à Y^n resteront tous à 0
    - Pas d’apprentissage possible

  - Il faut initialisez les poids du réseau avec des valeurs aléatoires
    - Chaque apprentissage donnera lieu à un résultat différent
    - Avec une fonction d’activation sigmoïde, il faut éviter des valeurs trop grandes, sinon, risque de gradient trop faible



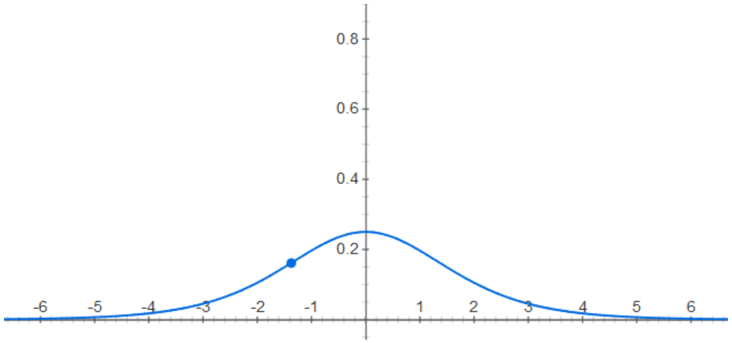
Introduction à l'Intelligence Artificielle

- **En pratique :**

  – Avec une fonction d'activation sigmoïde :

  – Dérivée de la fonction sigmoïde :

$$\sigma'(x) = \sigma(x) \, . \, (1-\sigma(x))$$



  – On a déjà calculé les σ pendant le *forward propagation* : il s'agit des *y*

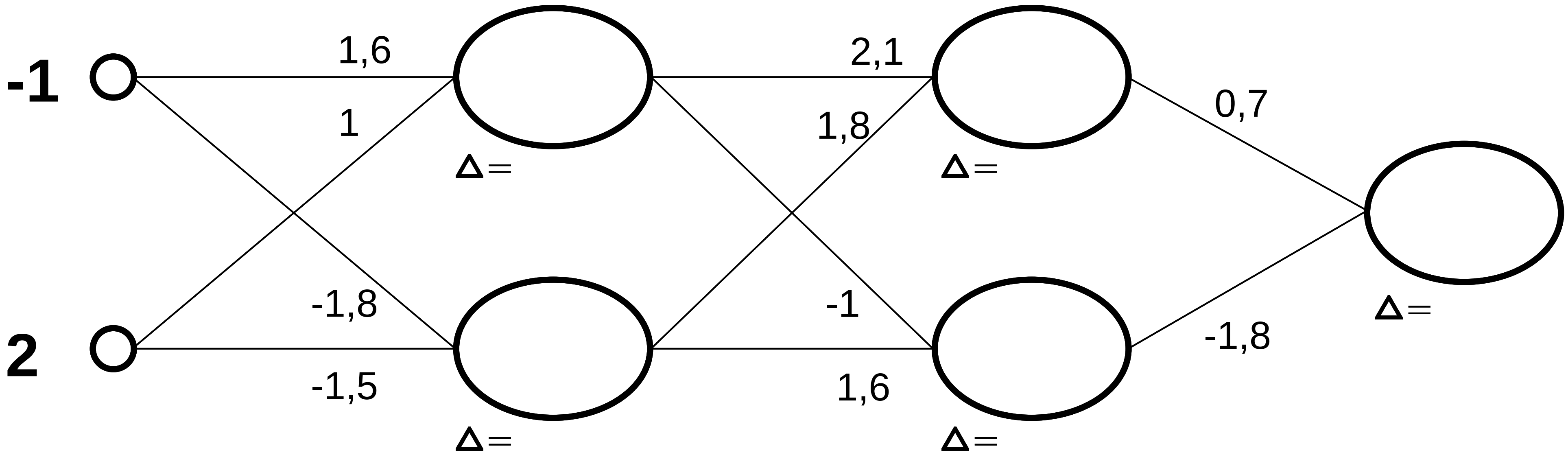
    - On a donc les équations suivantes pour le calcul des deltas :

$$\delta_i^L = (r-y_i^L) \, . \, y_i^L \, . \, (1-y_i^L)$$

$$\delta_i^l = y_i^l \, . \, (1-y_i^l) \, . \sum_{k\in[1,n]} \delta_k^{l+1} \, . \, w_{ki}^{l+1}$$

Introduction à l'Intelligence Artificielle

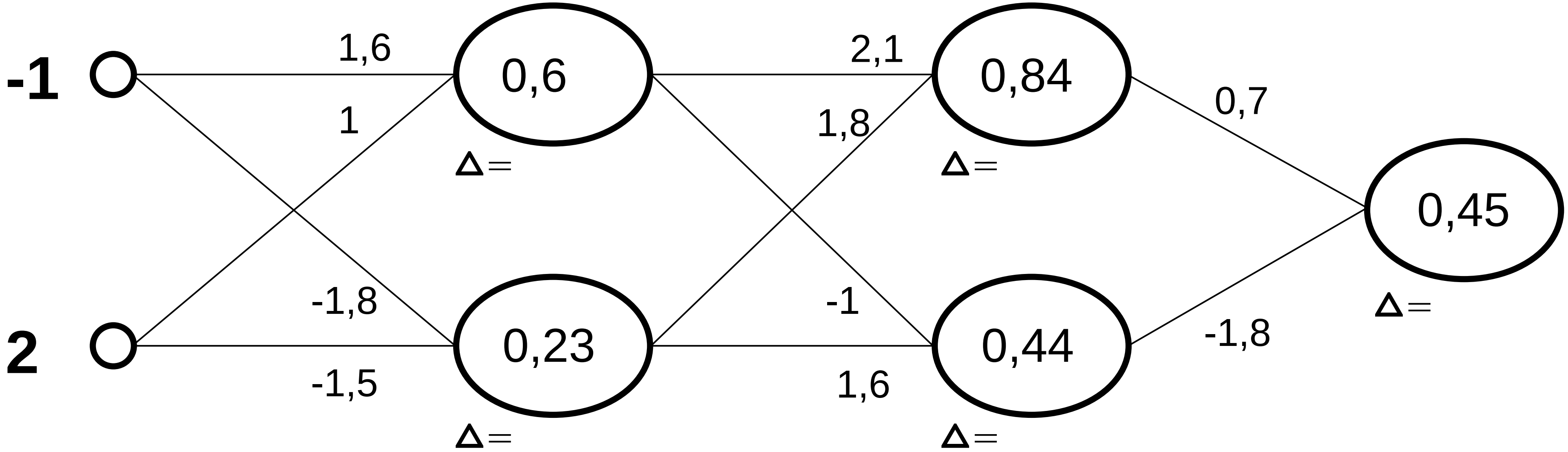
- **Un petit exemple**



- Vecteur d'entrées : [-1;2], valeur de sortie attendue : 1
- Pas de biais pour simplifier les notations
- Poids définis aléatoirement

Introduction à l'Intelligence Artificielle

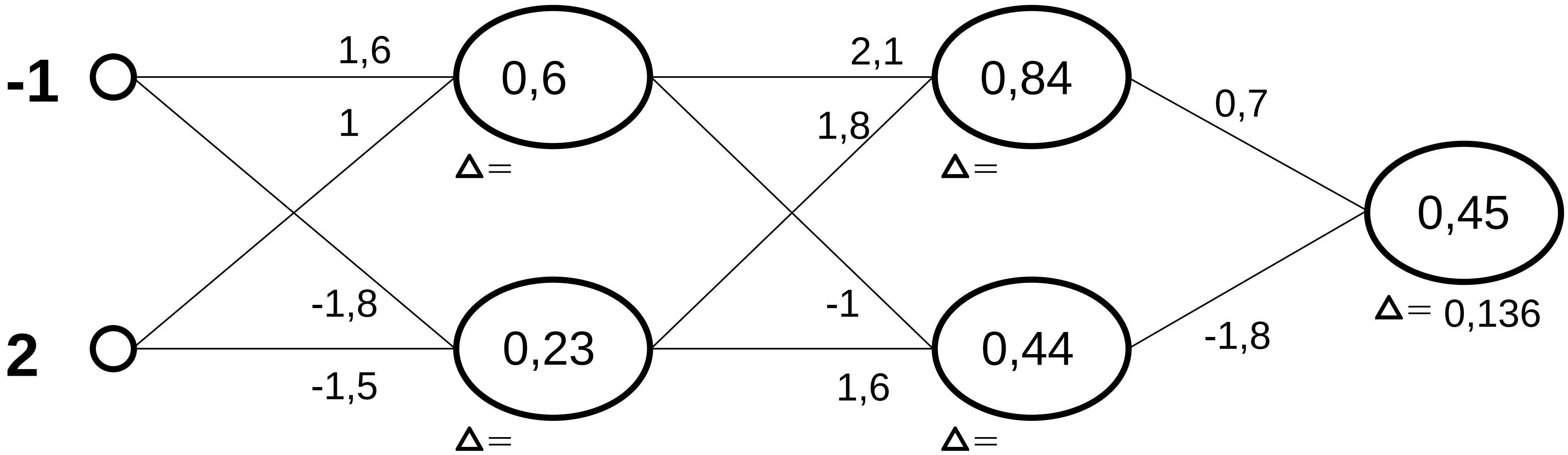
- **Un petit exemple**



- Forward propagation
  - y_1^1 = s (1,6 x -1 + 1 x 2) = s (0,4) = 0,6
  - y_2^1 = s (-1,8 x -1 + -1,5 x 2) = s (-1,2) = 0,23
  - y_1^2 = s (2,1 x 0,6 + 1,8 x 0,23) = s (1,67) = 0,84
  - y_2^2 = s (-1 x 0,6 + 1,6 x 0,23) = s (-0,22) = 0,44
  - y_1^3 = s (0,7 x 0,84 + -1,8 x 0,44) = s (0,21) = 0,45

Introduction à l'Intelligence Artificielle

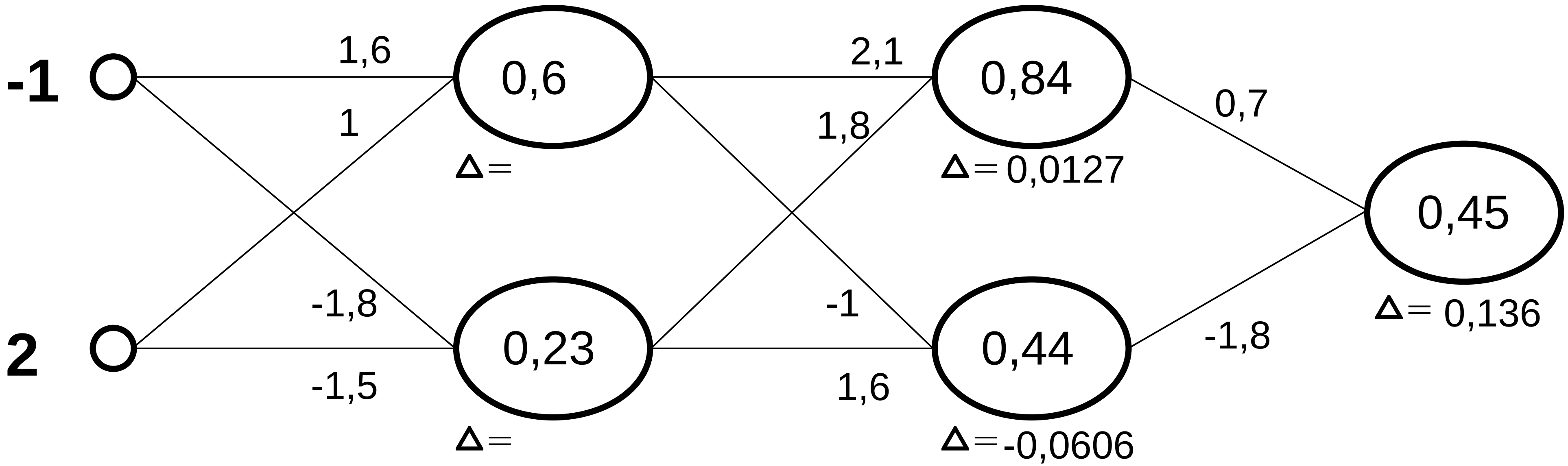
- **Un petit exemple**



- Backward propagation (r=1)
  - Premier delta : $$\delta_i^L \;=\; \left(r - y_i^L\right) \;.\; y_i^L \;.\; \left(1 - y_i^L\right)$$
  - Δ_1^3 = (1 – 0,45) x 0,45 x (1-0,45) = 0,136

Introduction à l'Intelligence Artificielle

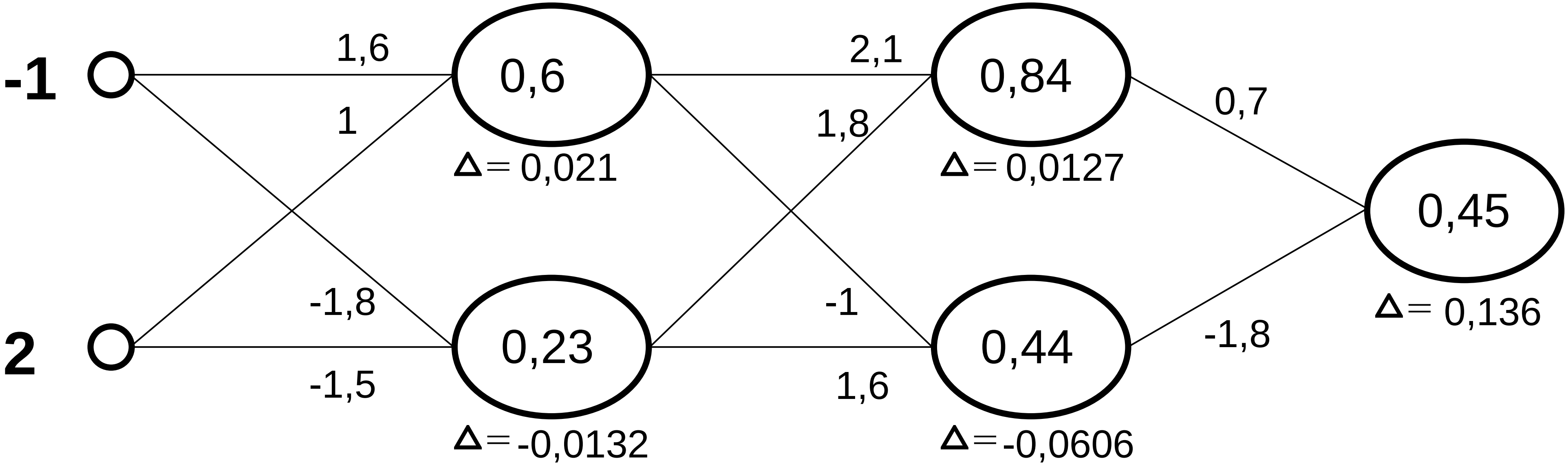
- **Un petit exemple**



- Backward propagation
  - On propage : $$\delta_i^l = y_i^l \, . \, \left(1 - y_i^l\right) \, . \, \sum_{k \in [1,n]} \delta_k^{l+1} \, . \, w_{ki}^{l+1}$$
  - Δ_1^2 = 0,84 x (1 – 0,84) x 0,136 x 0,7 = 0,0127
  - Δ_2^2 = 0,44 x (1 – 0,44) x 0,136 x -1,8 = -0,0606

Introduction à l'Intelligence Artificielle

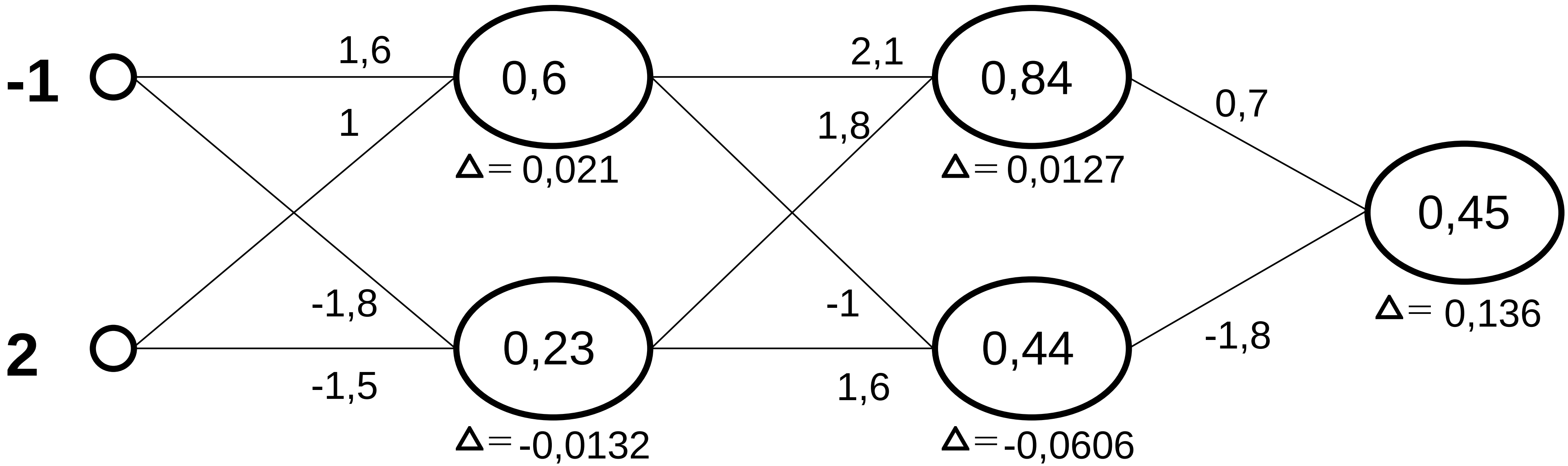
- **Un petit exemple**



- Backward propagation
  - On propage : $$\delta_i^l = y_i^l \;.\; (1-y_i^l) \;.\; \sum_{k\in[1,n]} \delta_k^{l+1} \;.\; w_{ki}^{l+1}$$
  - Δ_1^1 = 0,6 x (1 – 0,6) x (0,0127 x 2,1 + -0,0606 x -1) = 0,021
  - Δ_2^1 = 0,23 x (1 – 0,23) x (0,0127 x 1,8 + -0,0606 x 1,6) = -0,0132

Introduction à l'Intelligence Artificielle

- **Un petit exemple**

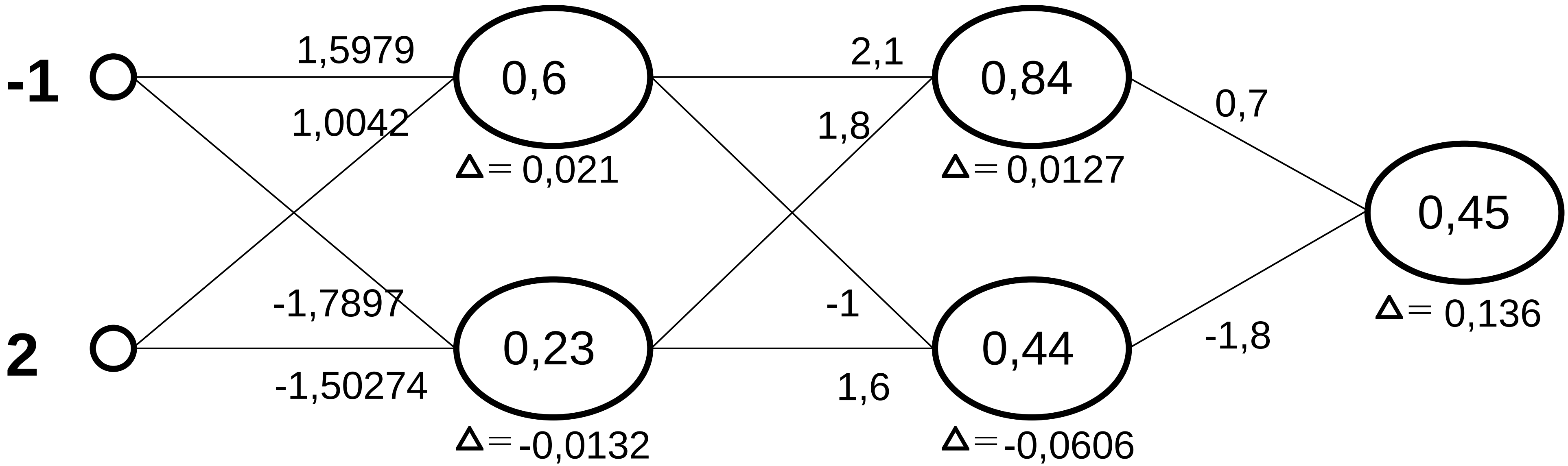


- Mise à jour des poids (α = 0,1) $$w_{ij}^{l} \Leftarrow w_{ij}^{l} + \alpha \, . \, x_{j}^{l} \, . \, \delta_{i}^{l}$$
  - $W_{11}^{1} \rightarrow$ 1,6 + 0,1 x -1 x 0,021 = 1,6 + -0,0021 = 1,5979
  - $W_{12}^{1} \rightarrow$ 1 + 0,1 x 2 x 0,021 = 1 + 0,0042 = 1,0042

  - $W_{21}^{1} \rightarrow$ -1,8 + 0,1 x -1 x -0,0132 = -1,8 + 0,00132 = -1,79868
  - $W_{22}^{1} \rightarrow$ -1,5 + 0,1 x 2 x -0,0132 = 1 + -0,00274 = -1,50274

Introduction à l'Intelligence Artificielle

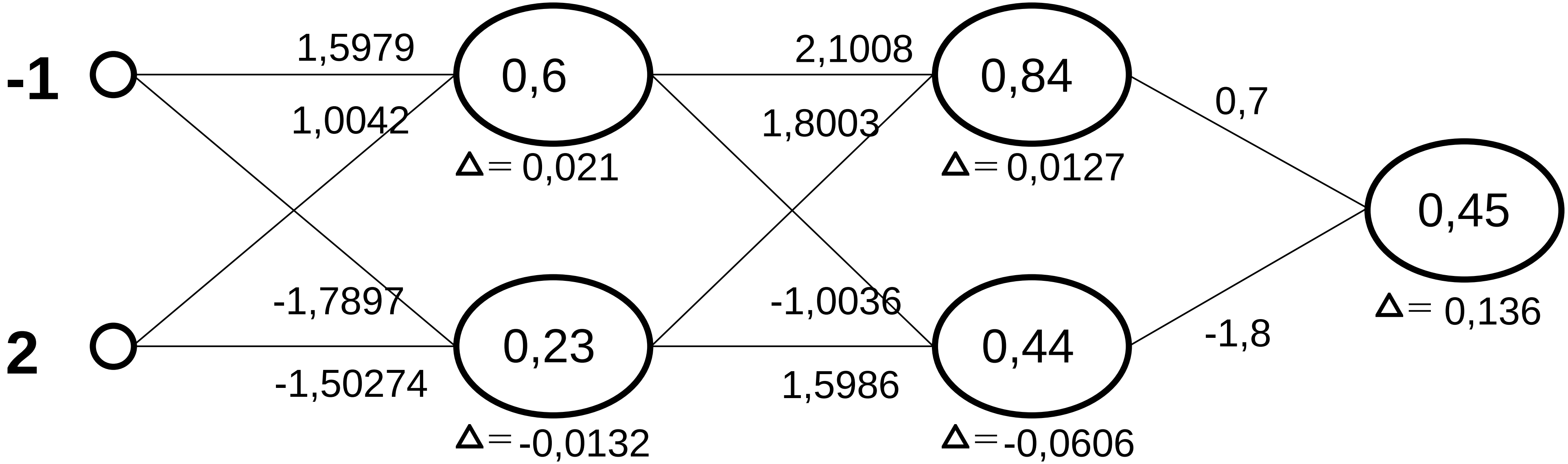
- **Un petit exemple**



- Mise à jour des poids (α = 0,1) $\quad w_{ij}^{l} \Leftarrow w_{ij}^{l} + \alpha . x_{j}^{l} . \delta_{i}^{l}$
  - $W_{11}^{2} \rightarrow$ 2,1 + 0,1 x 0,6 x 0,0127 = 2,1 + 0,0008 = 2,1008
  - $W_{12}^{2} \rightarrow$ 1,8 + 0,1 x 0,23 x 0,0127 = 1,8 + 0,0003 = 1,8003
  - $W_{21}^{2} \rightarrow$ -1 + 0,1 x 0,6 x -0,0606 = -1 + -0,0036 = -1,0036
  - $W_{22}^{2} \rightarrow$ 1,6 + 0,1 x 0,23 x -0,0606 = 1,6 + -0,0014 = -1,5986

Introduction à l’Intelligence Artificielle

- **Un petit exemple**



- Mise à jour des poids (α = 0,1) $$w_{ij}^{l} \Leftarrow w_{ij}^{l} + \alpha . x_{j}^{l} . \delta_{i}^{l}$$
  - $W_{11}^{3} \rightarrow$ 0,7 + 0,1 x 0,84 x 0,0136 = 0,7 + 0,01149 = 0,71149
  - $W_{12}^{3} \rightarrow$ -1,8 + 0,1 x 0,44 x -0,0606 = -1,8 + 0,00605 = -1,79395

- On recommence pour chaque exemple

Introduction à l'Intelligence Artificielle

- **Pour tester : http://playground.tensorflow.org**