

# Fouille de Données

# Data Mining

## **Classification - Partie 4**

# Plan du cours

1. Les réseaux de neurones artificiels
2. L'algorithme de Backpropagation

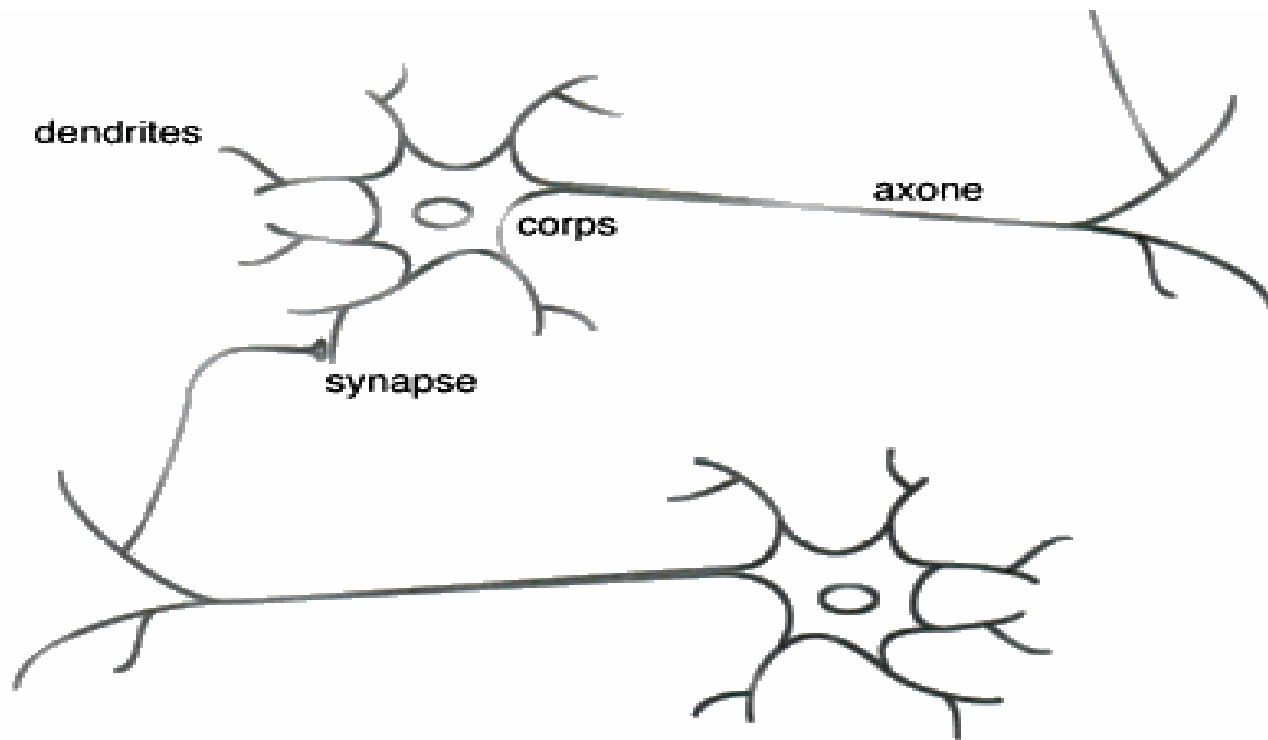
## Réseaux de neurones

- Bio mimétisme.
- *Biomimicry: innovation inspired by nature*, Jeanine Benyus.
- « A chaque fois que vous rencontrez un problème, observez la nature. »



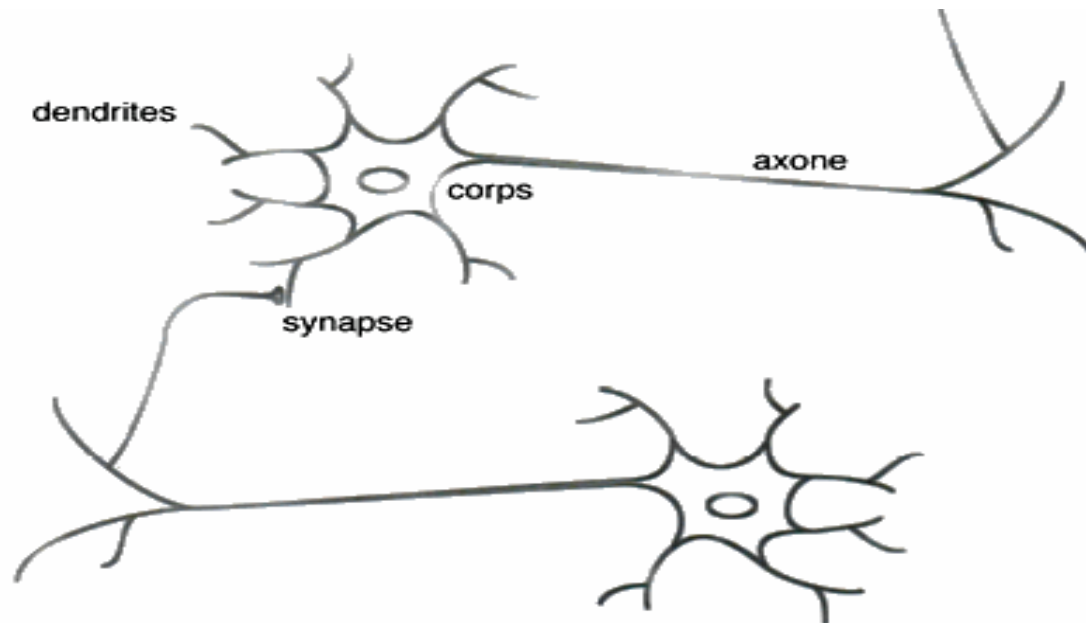
## Réseaux de neurones

- Comment l'homme fait-il pour raisonner, parler, calculer, apprendre, ...?
- S'inspirer du fonctionnement du cerveau humain. *Mais pas tellement.*
- Réseaux de neurones artificiels. ANN / RNA.



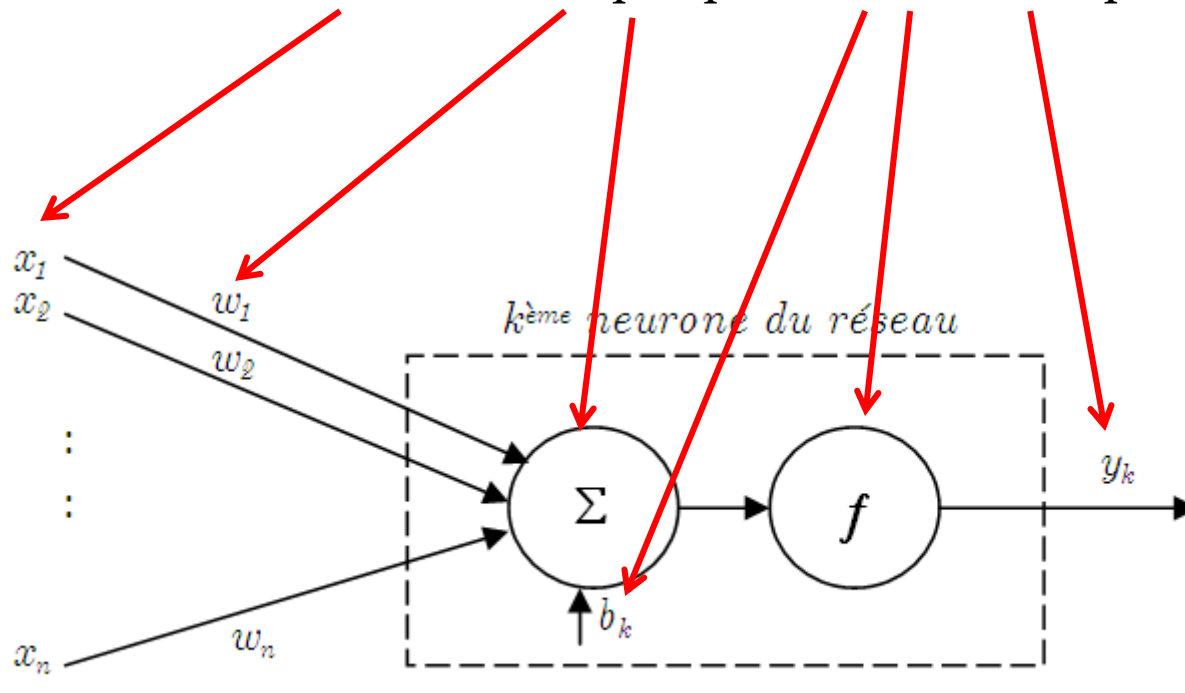
## Réseaux de neurones

- Mac Culloch et Pitts (**1943**) : définition d'un neurone formel.
- Rosenblatt (**1958**), Widrow et Hoff : modèle avec processus d'apprentissage, perceptron.
- Minsky et Papert (**1969**) : limites des perceptrons.
- Kohonen (**1972**) : mémoires associatives.
- Rumelhart – Mc Clelland (**1980**), Werbos – Le Cun : perceptron multi couches, mécanismes d'apprentissage performants (rétropropagation du gradient).
- **Deep Learning.**



# Réseaux de neurones

- Réseaux de neurones artificiels. ANN / RNA.
- D'un modèle biologique à un modèle mathématique. **Perceptron**.
- Réseau : Nœuds interconnectés par des liaisons directionnelles.
- Nœud=Neurone : se constitue de quelques éléments basiques :



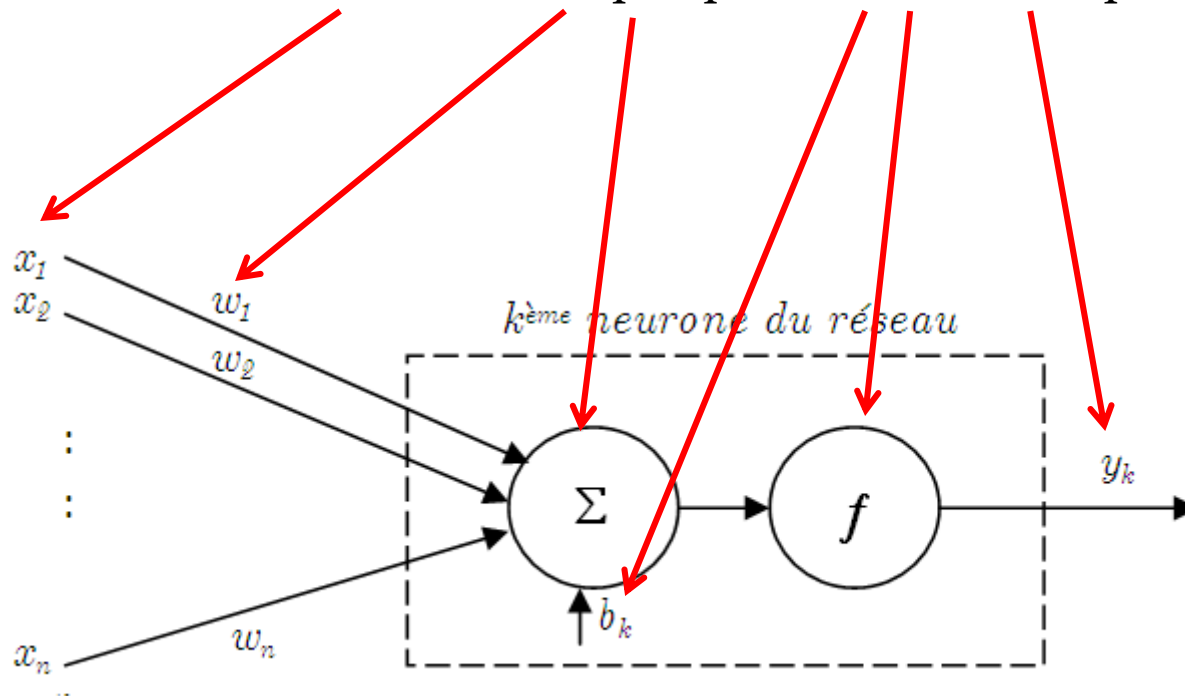
Entrées - **Poids** - Additionneur - Biais - Fonction d'activation - Sortie

## Réseaux de neurones artificiels

- Sortie  $Y$  d'un neurone est exprimée par :

$$y_k = f(w_{k1}x_1 + w_{k2}x_2 + \dots + w_{kn}x_n + b_k)$$

- Nœud=Neurone : se constitue de quelques éléments basiques :

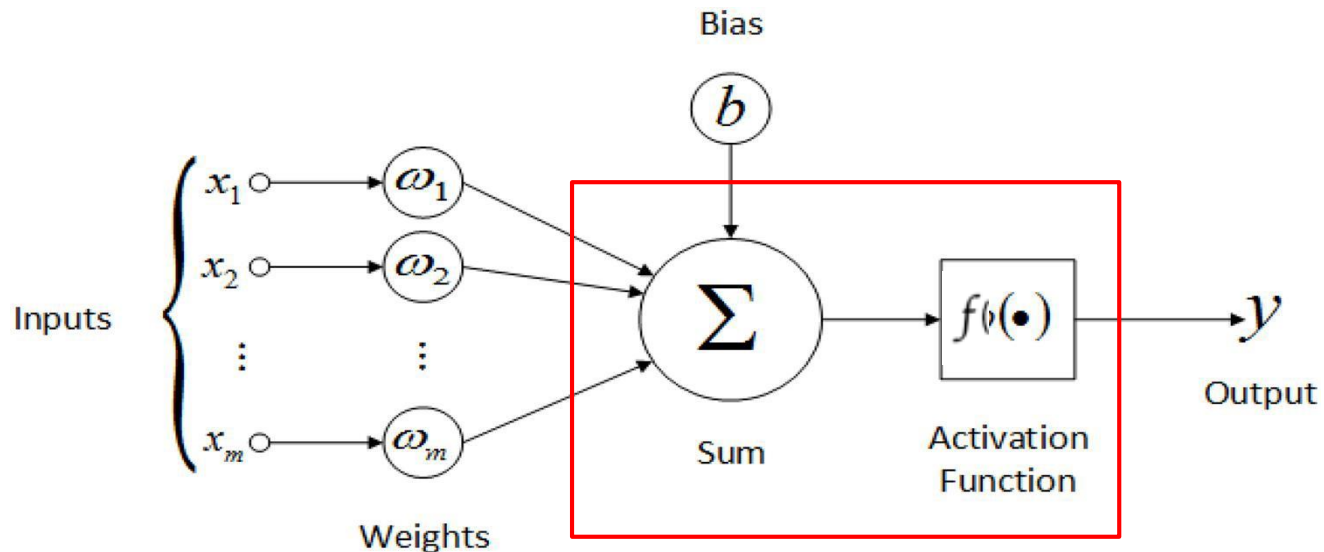


Entrées - **Poids** - Additionneur pondéré - Biais - Fonction d'activation - Sortie

# Réseaux de neurones artificiels

## ➤ Biais – Fonction d'activation, pourquoi ?

- ✓ Introduire la non-linéarité dans la sortie du neurone.
- ✓ La plupart des données du monde réel ne sont pas linéaires >> Apprendre aux neurones ces représentations non linéaires.












**Input** du neurone = Somme ( $x_i \cdot w_i + b$ )    **Output** du neurone =  $f(\text{Input du neurone})$



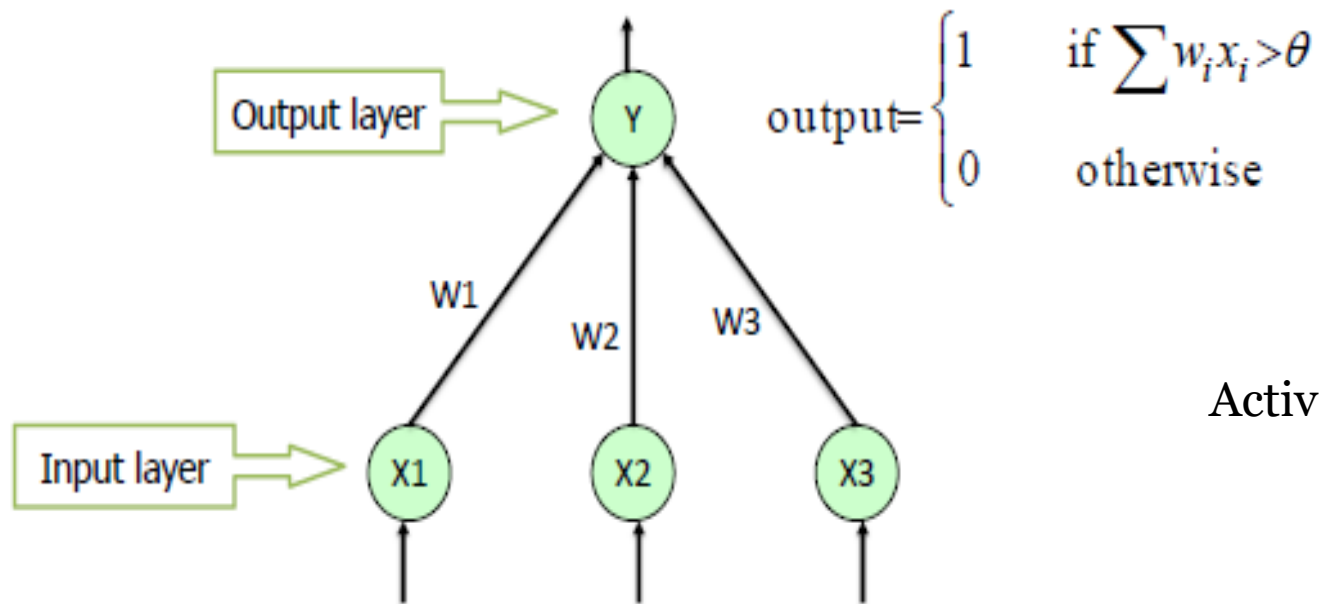
# Réseaux de neurones artificiels

## ➤ Fonction d'activation:

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

# Réseaux de neurones artificiels

## ➤ Feedforward Neural Networks - Single Layer **Perceptron**



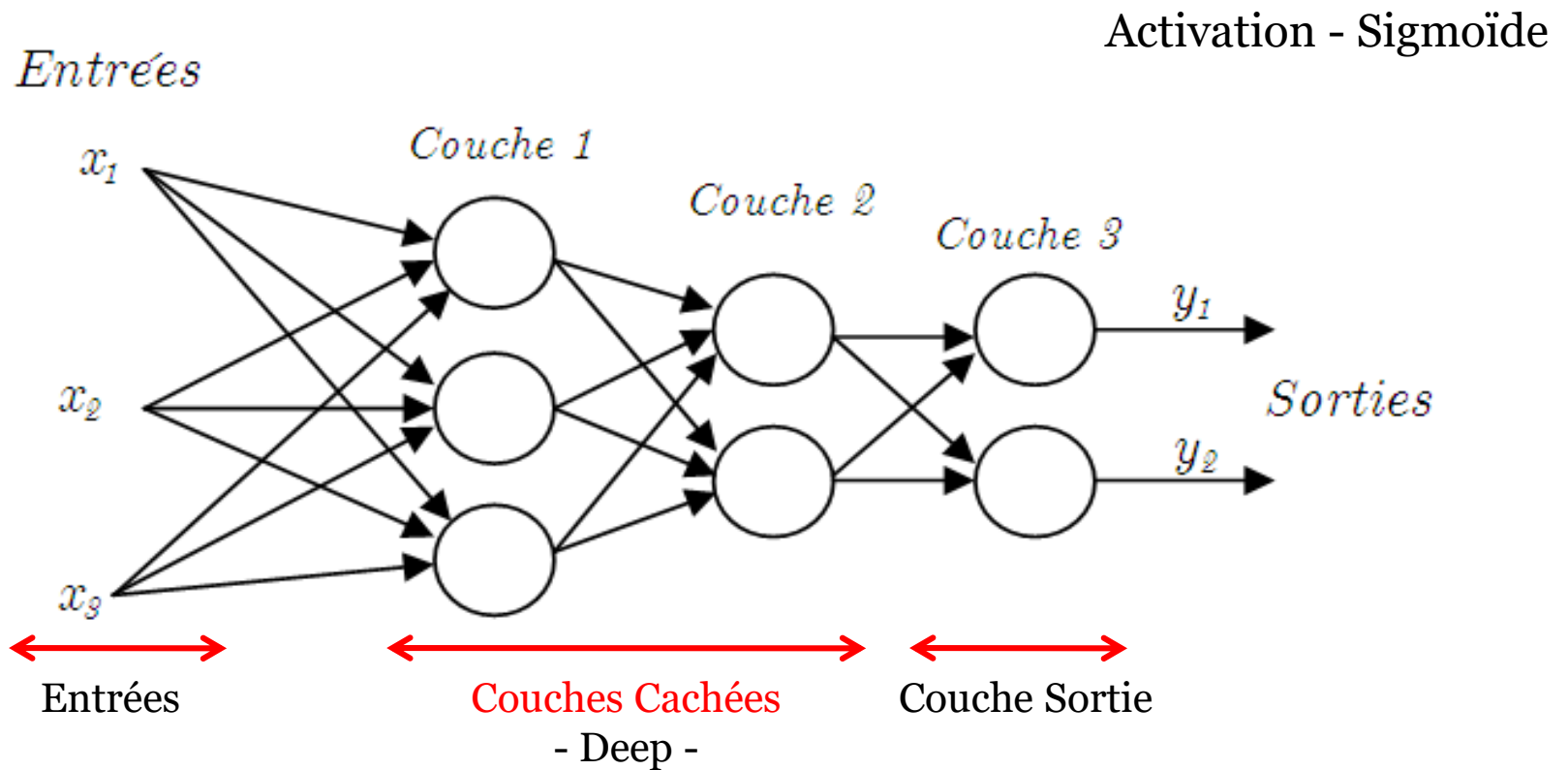
Activation - Seuil

$$w_1 x_1 + w_2 x_2 + \dots + w_n x_n > \theta \quad \text{Output} \quad \rightarrow \quad 1$$

$$w_1 x_1 + w_2 x_2 + \dots + w_n x_n \leq \theta \quad \rightarrow \quad 0$$

# Réseaux de neurones artificiels

## ➤ Feedforward Neural Networks - Multi Layer Perceptron



# Réseaux de neurones artificiels

## ➤ Deux phases – états d'un RNA:

- ✓ Apprentissage / Entrainement
- ✓ Fonctionnement optimal / Utilisation

## ➤ Apprentissage :

- Apprendre via un processus itératif et répétitif d'adaptation des **poids  $W_i$**  ;
- Les valeurs des poids sont initialisés aléatoirement, puis corrigées selon les erreurs entre les valeurs de sortie attendues et obtenues: **Forward propagation**.
- La correction se fait dans un sens inverse du sens de propagation des données : **Backpropagation**.

## ➤ Fonctionnement optimal : Une fois le réseau suffisamment calibré, il atteint un niveau où il n'est plus nécessaire de le superviser.

# Réseaux de neurones artificiels

## L'algorithme Backpropagation:

- À chaque présentation d'un exemple d'apprentissage au réseau, on passe par deux étapes :
  - ✓ Forward Propagation
  - ✓ Back Propagation
- 1. En **Forward Propagation**, on appliquons un ensemble de poids aux données d'entrée et on calcule une sortie. Pour la première propagation, l'ensemble des poids est sélectionné de manière aléatoire.
- 2. Dans la **Back Propagation**, on mesure la marge d'erreur de la sortie et on ajuste les poids en conséquence pour diminuer l'erreur.
- 3. Les réseaux neuronaux répètent les deux propagations jusqu'à ce que les poids soient calibrés pour prédire avec précision une sortie.

# Réseaux de neurones artificiels

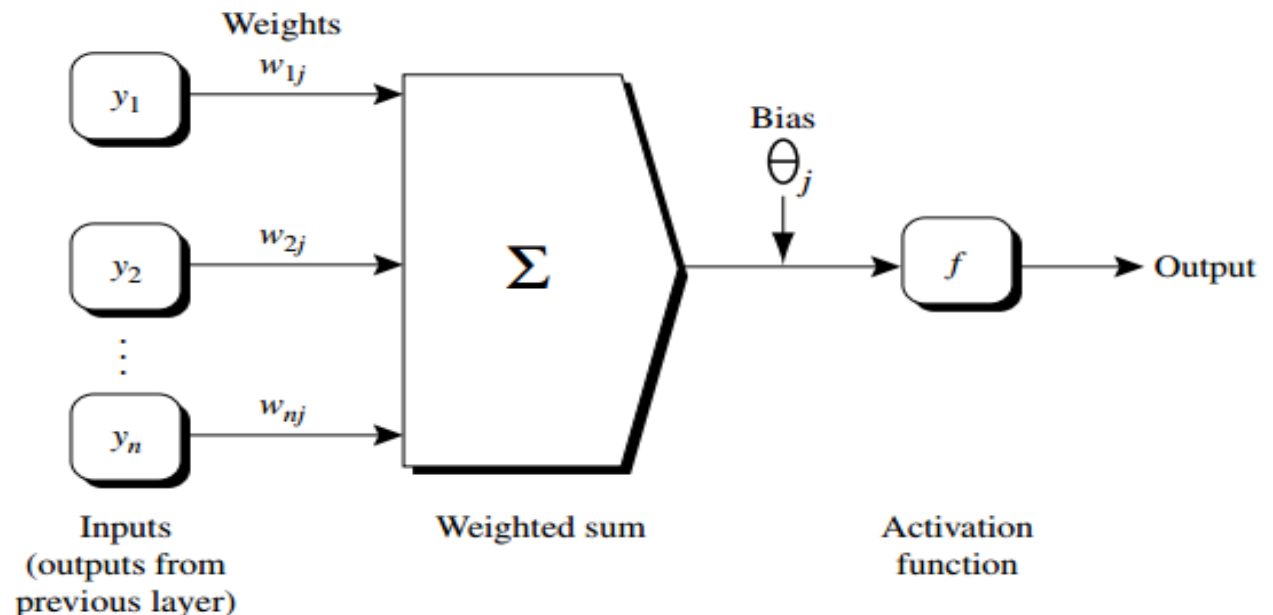
## L'algorithme Backpropagation: Pseudo-code

// Propagate the inputs forward:

**for** each hidden or output layer unit  $j$  {

$I_j = \sum_i w_{ij} O_i + \theta_j$ ; // compute the net input of unit  $j$  with respect to the previous layer,  $i$

$O_j = \frac{1}{1 + e^{-I_j}}$ ; } // compute the output of each unit  $j$



# Réseaux de neurones artificiels

## L'algorithme Backpropagation: Pseudo-code

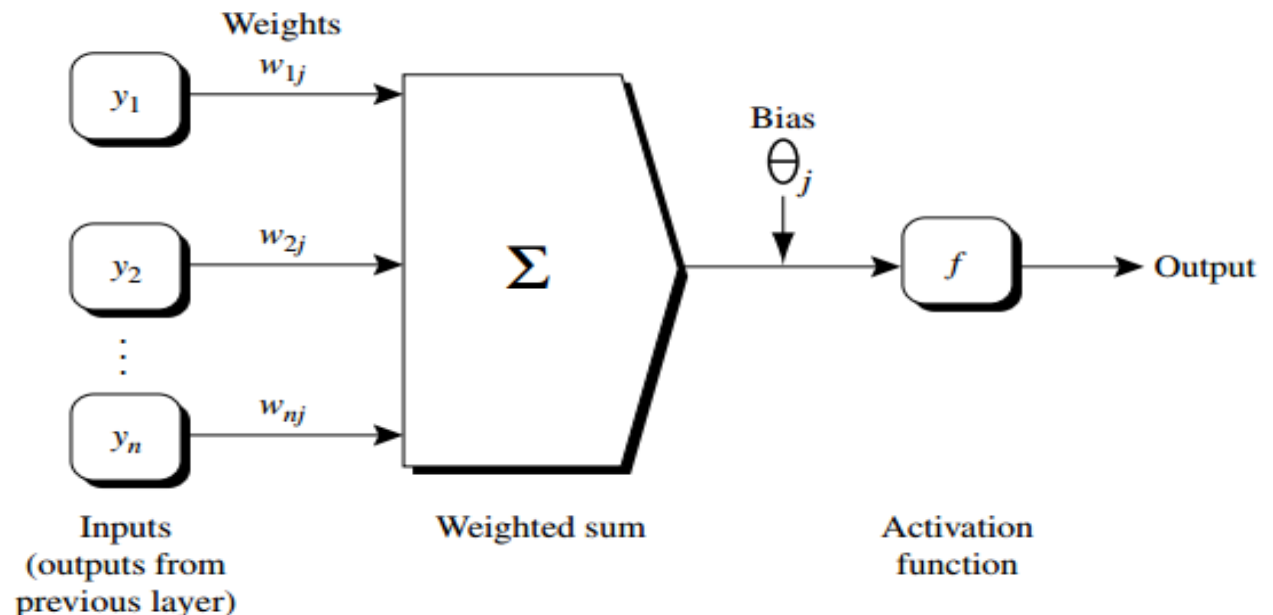
// Backpropagate the errors:

**for** each unit  $j$  in the output layer

$Err_j = O_j(1 - O_j)(T_j - O_j)$ ; // compute the error     T: Target value

**for** each unit  $j$  in the hidden layers, from the last to the first hidden layer

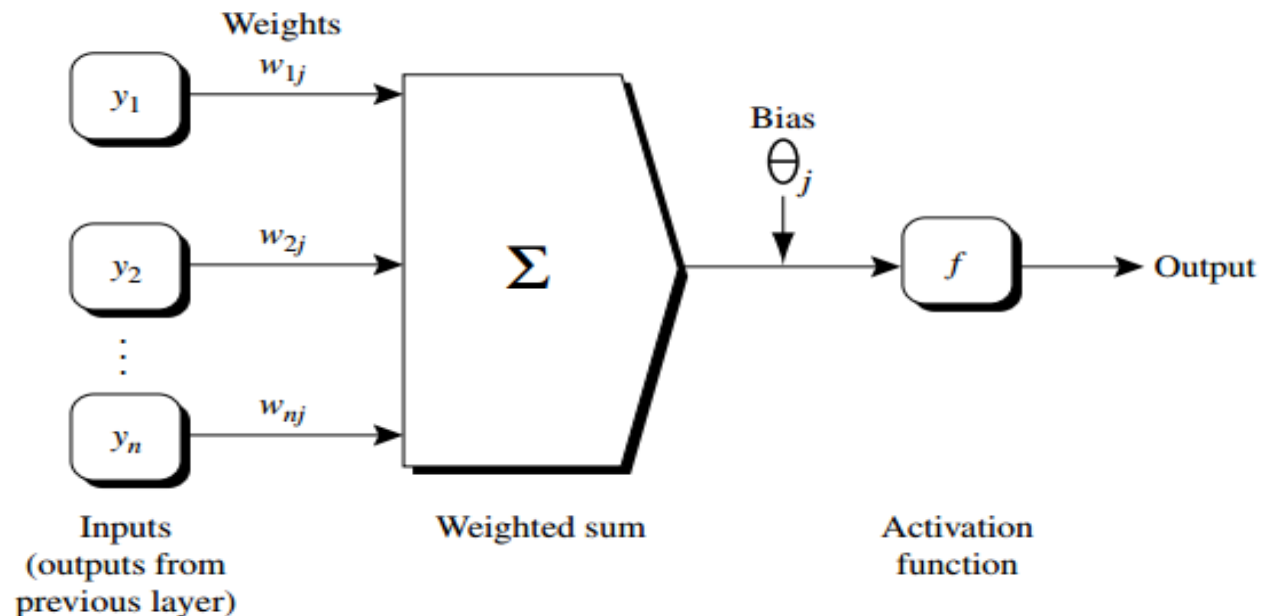
$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$ ; // compute the error with respect to the next higher layer,  $k$



# Réseaux de neurones artificiels

## L'algorithme Backpropagation: Pseudo-code

```
for each weight  $w_{ij}$  in network {  
     $\Delta w_{ij} = (l)Err_j O_i$ ; // weight increment  
     $w_{ij} = w_{ij} + \Delta w_{ij}$ ; } // weight update  
for each bias  $\theta_j$  in network {  
     $\Delta \theta_j = (l)Err_j$ ; // bias increment  
     $\theta_j = \theta_j + \Delta \theta_j$ ; } // bias update  
}
```





# Réseaux de neurones artificiels

## L'algorithme Backpropagation: Déconstruction des étapes de fonctionnement

**1** - Fixer l'architecture du RNA : nombre des neurones (couches cachées et couche sortie (selon le nombre des classes possibles)).

**2** - Initialisation aléatoire de l'ensemble des poids et des biais du RNA.

**3** - Forward propagation : de la couche des entrées vers la couche des sorties

- Calculer les inputs et les outputs pour chaque neurone :

1 Input =  $\sum (\text{weight}_i * \text{input}_i) + \text{bias}$

2 output =  $1 / (1 + \exp(-\text{input}))$

Activation - Sigmoid

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

$$O_j = \frac{1}{1 + e^{-I_j}}$$

# Réseaux de neurones artificiels

## L'algorithme Backpropagation: Déconstruction des étapes de fonctionnement

### 4 – Backpropagate Error :

#### a - Calcul de Transfer Derivative :

1 transfer\_derivative = output \* (1 - output)

#### b - Calcul d'erreur pour chaque neurone : **Loss Function**

-Cas - Neurone couche sorties :

2 error = (expected\_target - output) \* transfer\_derivative(output)

-Cas – Neurone couche cachée : weight et error des neurones précédents

3 error = transfer\_derivative(output) \* somme(weight\_k \* error\_k)

$$Err_j = O_j(1 - O_j)(T_j - O_j);$$

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}; /$$

# Réseaux de neurones artificiels

## L'algorithme Backpropagation: Déconstruction des étapes de fonctionnement

### 4 – Backpropagate Error :

- Le calcul des erreurs est représentée par une fonction qu'on appelle **fonction de coût** (ou fonction de perte) : Loss function en anglais.
- Fonction coût = l'ensemble des erreurs.
- Il s'agit d'une mesure de performance sur la façon dont le RNA parvient à atteindre son objectif de générer des sorties aussi proches que possible des valeurs souhaitées (Expected Target).
- Avoir un **bon modèle** de classification (RNA ou autre), c'est avoir un modèle qui nous donne de **petites erreurs**, donc **une petite Fonction Coût**.
- Différentes fonctions coût existent : Mean Squared Error, Mean Absolut Error, Cross Entropy, Sum of Squared Estimate of Errors, SVM cost, etc.

# Réseaux de neurones artificiels

L'algorithme Backpropagation: Déconstruction des étapes de fonctionnement

## 4 – Backpropagate Error :

- L'objectif central de l'apprentissage est donc de trouver les paramètres du modèle (les poids  $w_i$ ) qui **minimisent la Fonction Coût**.
- Pour cela, on utilise généralement un **algorithme d'optimisation** permettant de converger et de mettre à jour les poids du RNA.
- L'exemple le plus courant étant l'algorithme de **Gradient Descent (Descente de Gradient)**.
- Cet algorithme utilise une vitesse d'apprentissage, appelée **Learning Rate**. La valeur de cet hyperparamètre est fixée au départ.
- Bien choisir sa valeur. Car : Si la vitesse est trop lente (petite), le modèle peut mettre longtemps à être entraîné; si la vitesse est trop grande, alors la distance parcourue est trop longue et le modèle peut ne jamais converger.

# Réseaux de neurones artificiels

## L'algorithme Backpropagation: Déconstruction des étapes de fonctionnement

### 4 – Backpropagate Error :

**c** – Mettre à jour les poids (y compris les biais) selon les erreurs calculées:  
**optimisation de la fonction coût**

1 weight = weight + (learning\_rate \* error \* input)

2 Weight\_bias = weight + (learning\_rate \* error)

$$\begin{aligned}\Delta w_{ij} &= (l) Err_j O_i \\ w_{ij} &= w_{ij} + \Delta w_{ij}\end{aligned}$$

$$\begin{aligned}\Delta \theta_j &= (l) Err_j \\ \theta_j &= \theta_j + \Delta \theta_j\end{aligned}$$

### 5 – Entrainement (Apprentissage) du réseau de neurones :

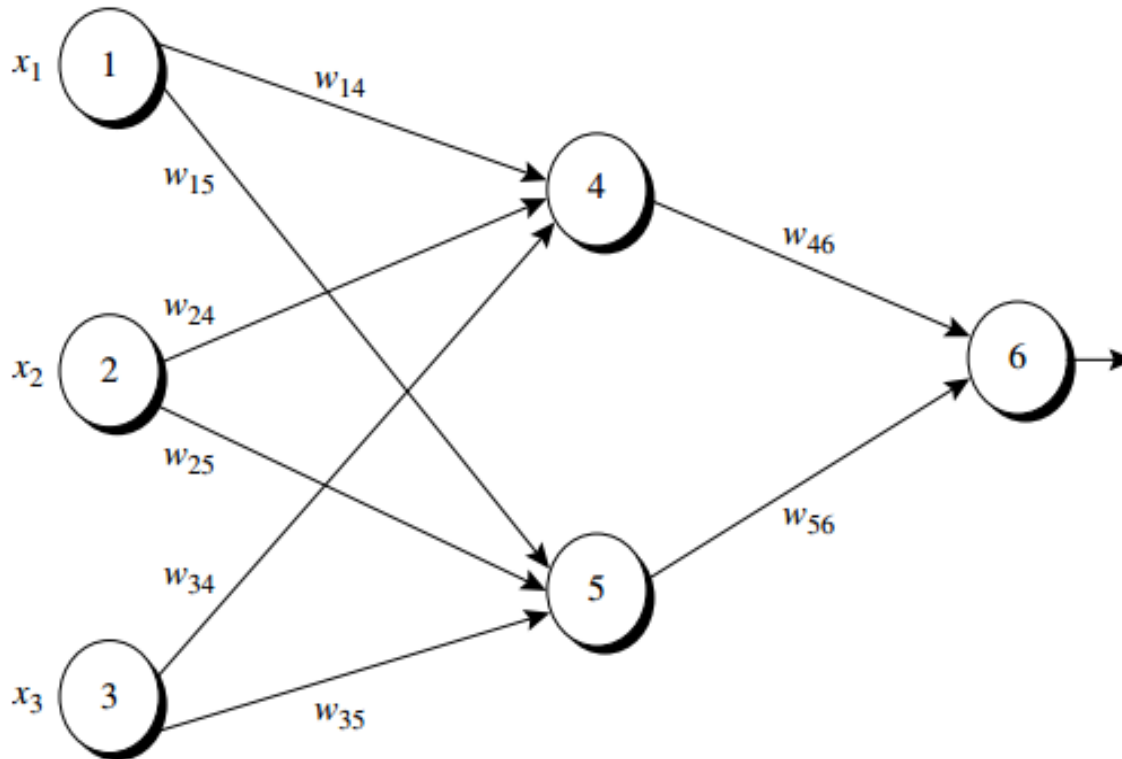
- **Itérer** et Répéter les 4 étapes précédentes pour un nombre d'itérations prédéfini, appelé : **epochs** number.

# Réseaux de neurones artificiels

<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>
----------	----------	----------	----------

L'algorithme Backpropagation: **Exemple**

Learning Rate :  $\eta = 0.9$



Initial Input, Weight, and Bias Values

$x_1$	$x_2$	$x_3$	$w_{14}$	$w_{15}$	$w_{24}$	$w_{25}$	$w_{34}$	$w_{35}$	$w_{46}$	$w_{56}$	$\theta_4$	$\theta_5$	$\theta_6$
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

# Réseaux de neurones artificiels

<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>
----------	----------	----------	----------

L'algorithme Backpropagation: **Exemple**

Learning Rate :  $l = 0.9$

Initial Input, Weight, and Bias Values

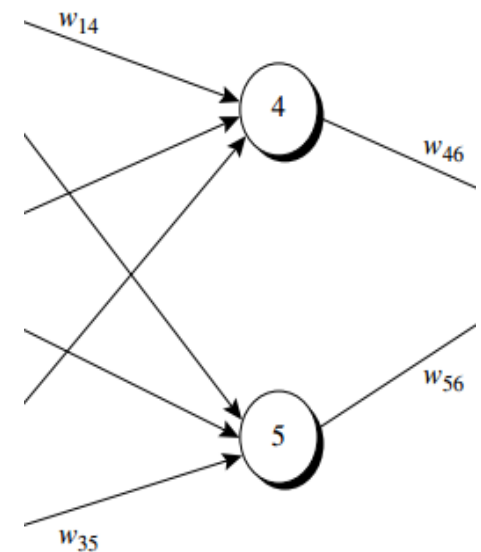
$x_1$	$x_2$	$x_3$	$w_{14}$	$w_{15}$	$w_{24}$	$w_{25}$	$w_{34}$	$w_{35}$	$w_{46}$	$w_{56}$	$\theta_4$	$\theta_5$	$\theta_6$
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

Net Input and Output Calculations

Unit, $j$	Net Input, $I_j$	Output, $O_j$
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1 + e^{0.7}) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1 + e^{-0.1}) = 0.525$
6	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

$$O_j = \frac{1}{1 + e^{-I_j}}$$



# Réseaux de neurones artificiels

<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>
----------	----------	----------	----------

L'algorithme Backpropagation: **Exemple**

Learning Rate :  $\eta = 0.9$

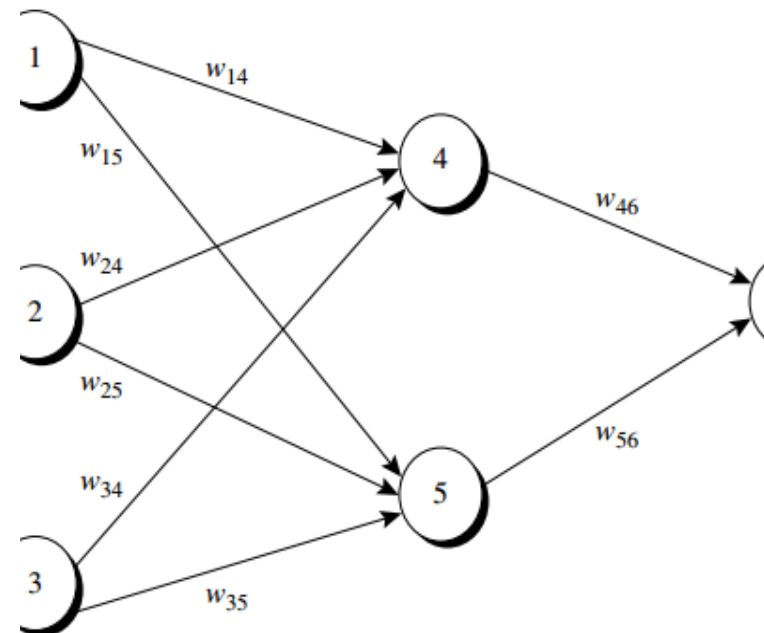
## Net Input and Output Calculations

Unit, $j$	Net Input, $I_j$	Output, $O_j$
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1 + e^{0.7}) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1 + e^{-0.1}) = 0.525$
6	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

## Calculation of the Error at Each Node

Unit, $j$	Err <sub><math>j</math></sub>
6	$(0.474)(1 - 0.474)(1 - 0.474) = 0.1311$
5	$(0.525)(1 - 0.525)(0.1311)(-0.2) = -0.0065$
4	$(0.332)(1 - 0.332)(0.1311)(-0.3) = -0.0087$

$$Err_j = O_j(1 - O_j)(T_j - O_j);$$





# Réseaux de neurones artificiels

<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>
----------	----------	----------	----------

L'algorithme Backpropagation: **Exemple**

Learning Rate :  $l = 0.9$

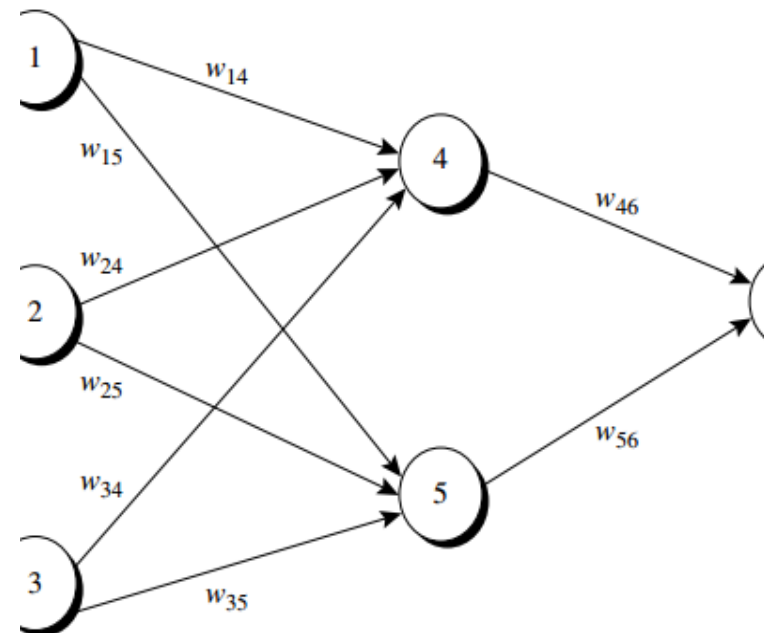
## Net Input and Output Calculations

Unit, $j$	Net Input, $I_j$	Output, $O_j$
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1 + e^{0.7}) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1 + e^{-0.1}) = 0.525$
6	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

## Calculation of the Error at Each Node

Unit, $j$	Err <sub><math>j</math></sub>
6	$(0.474)(1 - 0.474)(1 - 0.474) = 0.1311$
5	$(0.525)(1 - 0.525)(0.1311)(-0.2) = -0.0065$
4	$(0.332)(1 - 0.332)(0.1311)(-0.3) = -0.0087$

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$$



# Réseaux de neurones artificiels

## L'algorithme Backpropagation: Exemple

### Calculations for Weight and Bias Updating

Weight or Bias	New Value
$w_{46}$	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
$w_{56}$	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
$w_{14}$	$0.2 + (0.9)(-0.0087)(1) = 0.192$
$w_{15}$	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
$w_{24}$	$0.4 + (0.9)(-0.0087)(0) = 0.4$
$w_{25}$	$0.1 + (0.9)(-0.0065)(0) = 0.1$
$w_{34}$	$-0.5 + (0.9)(-0.0087)(1) = -0.508$
$w_{35}$	$0.2 + (0.9)(-0.0065)(1) = 0.194$
$\theta_6$	$0.1 + (0.9)(0.1311) = 0.218$
$\theta_5$	$0.2 + (0.9)(-0.0065) = 0.194$
$\theta_4$	$-0.4 + (0.9)(-0.0087) = -0.408$

1	0	1	1
---	---	---	---

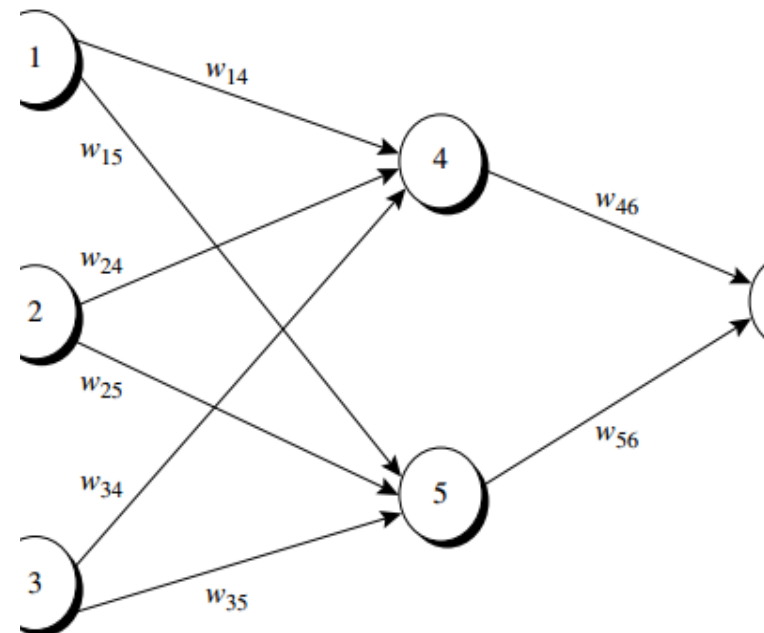
Learning Rate :  $l = 0.9$

$$\Delta w_{ij} = (l) Err_j O_i$$

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

$$\Delta \theta_j = (l) Err_j$$

$$\theta_j = \theta_j + \Delta \theta_j$$



# Réseaux de neurones artificiels

## Observations

- RNA : Boite noire. Difficile d'analyser et comprendre ce qu'il a appris.
- L'ordre de présentation des exemples d'entraînement au réseau influe directement sur les résultats obtenus.
- Répéter l'apprentissage avec un ordre différent des exemples.
- Représentation de connaissance difficile à interpréter pour l'humain.

# Réseaux de neurones artificiels

## Domaines d'application

- Neural networks have been successfully applied to broad spectrum of data-intensive applications. The list below is based on real-world success stories. It will give you an overview of the scope of problems that NeuroIntelligence can address.

**Source** : <http://www.alyuda.com/products/forecaster/neural-network-applications.htm>

# Réseaux de neurones artificiels

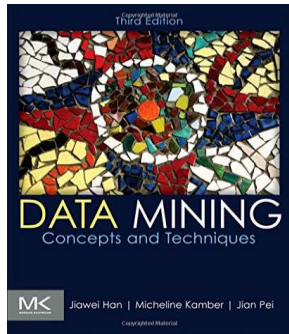
## Domaines d'application

➤ Aussi dans l'Art :

Mike Tyka - TEDx - <https://www.youtube.com/watch?v=oqVOUD76JOg>

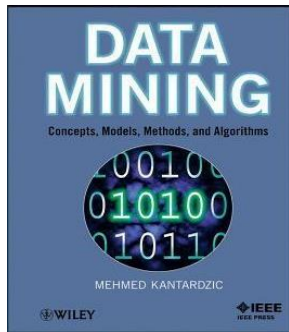


# Ressources



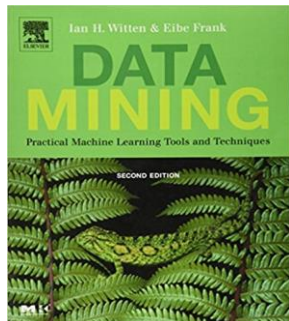
## **Data Mining : concepts and techniques, 3rd Edition**

- ✓ Auteur : Jiawei Han, Micheline Kamber, Jian Pei
- ✓ Éditeur : Morgan Kaufmann Publishers
- ✓ Edition : Juin 2011 - 744 pages - ISBN 9780123814807



## **Data Mining : concepts, models, methods, and algorithms**

- ✓ Auteur : Mehmed Kantardzi
- ✓ Éditeur : John Wiley & Sons
- ✓ Edition : Aout 2011 – 552 pages - ISBN : 9781118029121



## **Data Mining: Practical Machine Learning Tools and Techniques**

- ✓ Auteur : Ian H. Witten & Eibe Frank
- ✓ Éditeur : Morgan Kaufmann Publishers
- ✓ Edition : Juin 2005 - 664 pages - ISBN : 0-12-088407-0

# Ressources

Cours – Abdelhamid DJEFFAL – Fouille de données avancée

✓ [www.abdelhamid-djeffal.net](http://www.abdelhamid-djeffal.net)

WekaMOOC – Ian Witten – Data Mining with Weka

✓ <https://www.youtube.com/user/WekaMOOC/featured>

Cours - PJE : Analyse de comportements avec Twitter Classification supervisée - Arnaud Liefoghe

✓ <http://www.fil.univ-lille1.fr/~liefoghe/PJE/bayes-cours.pdf>

✓ <https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/>