CHAPITRE III

Apprentissage automatique

Machine Learning

Apprentissage automatique

Définition

« Field of study that gives computers the ability to learn without being explicitly programmed »
Arthur Samuel, 1959

- **Un agent apprend s'il améliore sa performance sur des taches futures avec de l'expérience.**

- L'apprentissage automatique fait référence au développement, à l'analyse et à l'implémentation de méthodes qui permettent à une machine d'évoluer grâce à **un processus d'apprentissage.**

Types d'apprentissage

Les algorithmes d'apprentissage peuvent se catégoriser selon le type d'apprentissage qu'ils emploient :

- L'apprentissage supervisé
- L'apprentissage non-supervisé
- L'apprentissage par renforcement

Types d'apprentissage

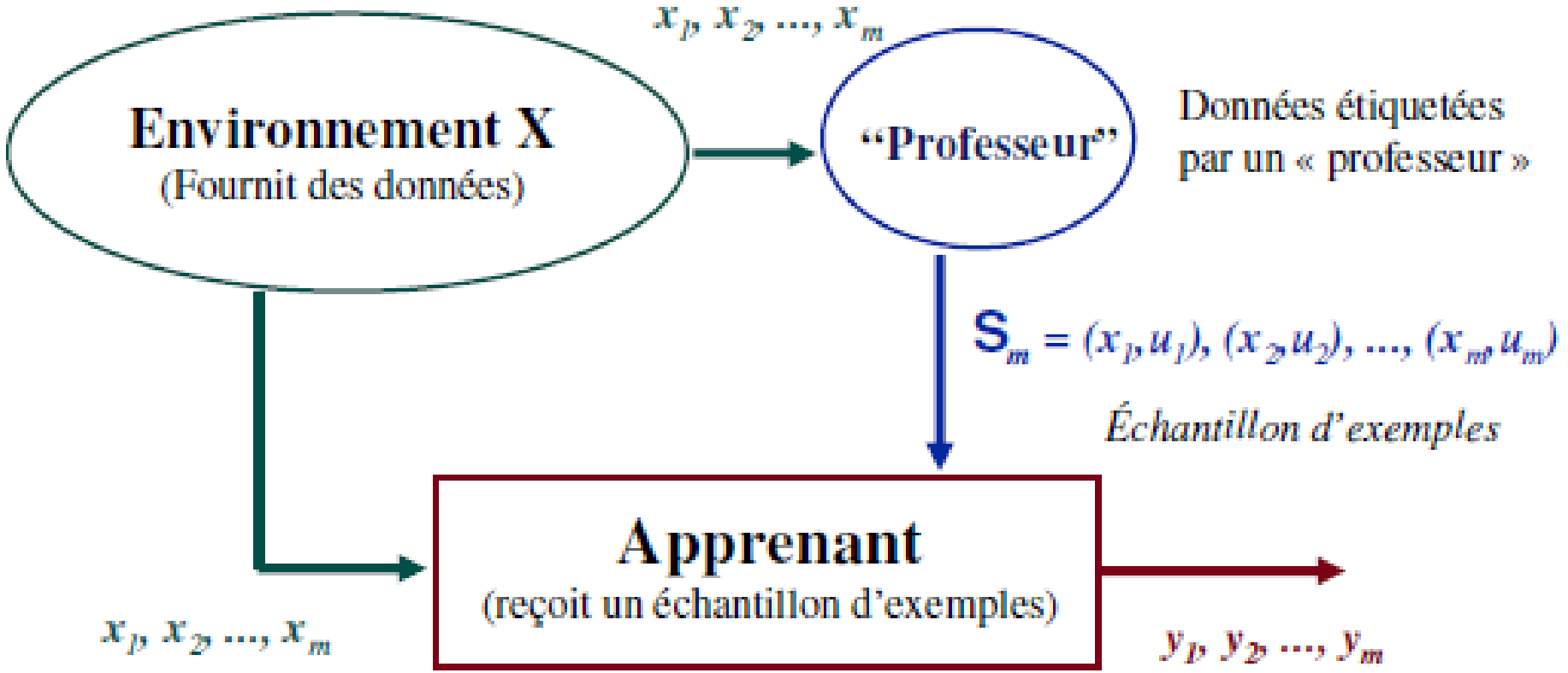
L'apprentissage supervisé

- Un expert est employé pour étiqueter correctement des exemples.
- L'apprenant doit alors trouver ou approximer la fonction qui permet d'affecter la bonne étiquette à ces exemples.

- Exemple : reconnaissance de caractères à l'aide d'un ensemble de paires : (image, identité du caractère)

Types d'apprentissage

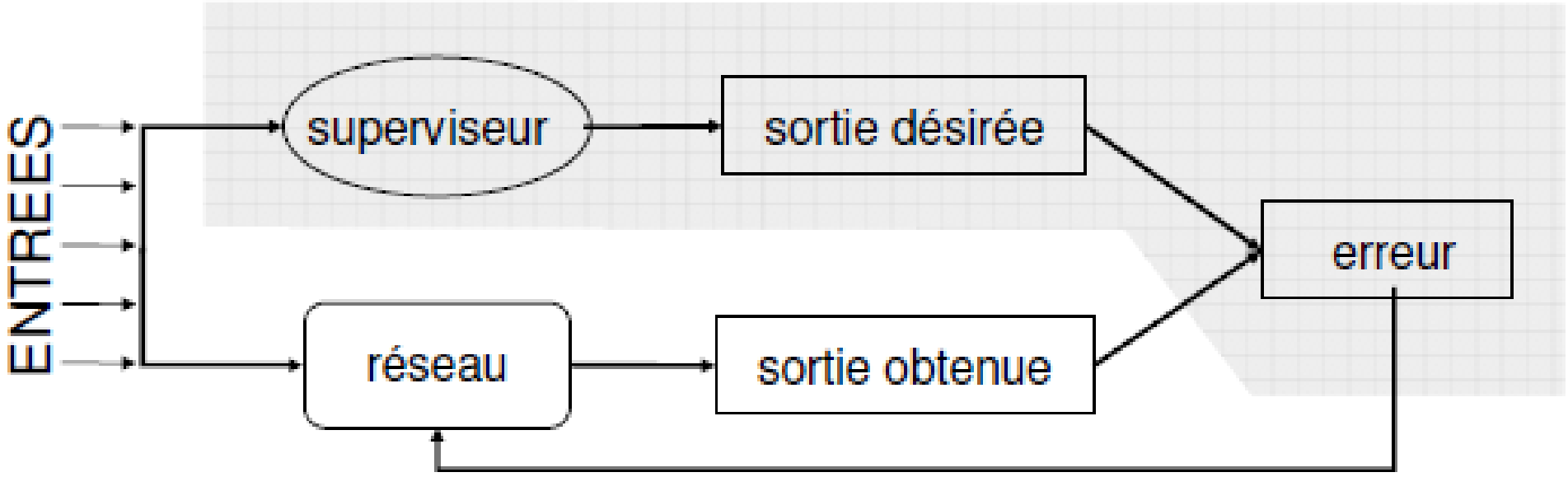
L'apprentissage supervisé



Approximer au mieux la sortie désirée
pour chaque entrée observée

Types d'apprentissage

L'apprentissage supervisé



Types d'apprentissage

L'apprentissage non supervisé

- Aucun expert n'est disponible.
- L'algorithme doit découvrir par lui-même la structure des données.

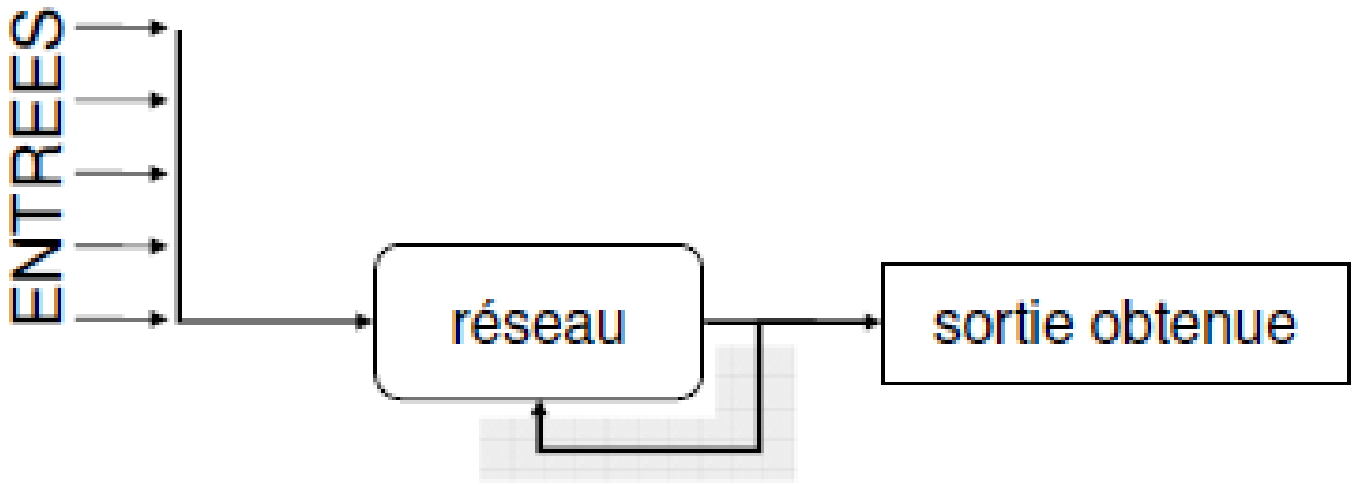
- Exemple : identifier différents thèmes d'articles de journaux en regroupant les articles similaires (clustering)

Types d'apprentissage

L'apprentissage non supervisé

A partir de l'*échantillon d'apprentissage* $S = \{(x_i)\}_{1,m}$ non étiqueté, on cherche des régularités sous-jacentes :

- Sous forme d'une fonction
- Sous forme d'un modèle complexe



Types d'apprentissage

L'apprentissage par renforcement

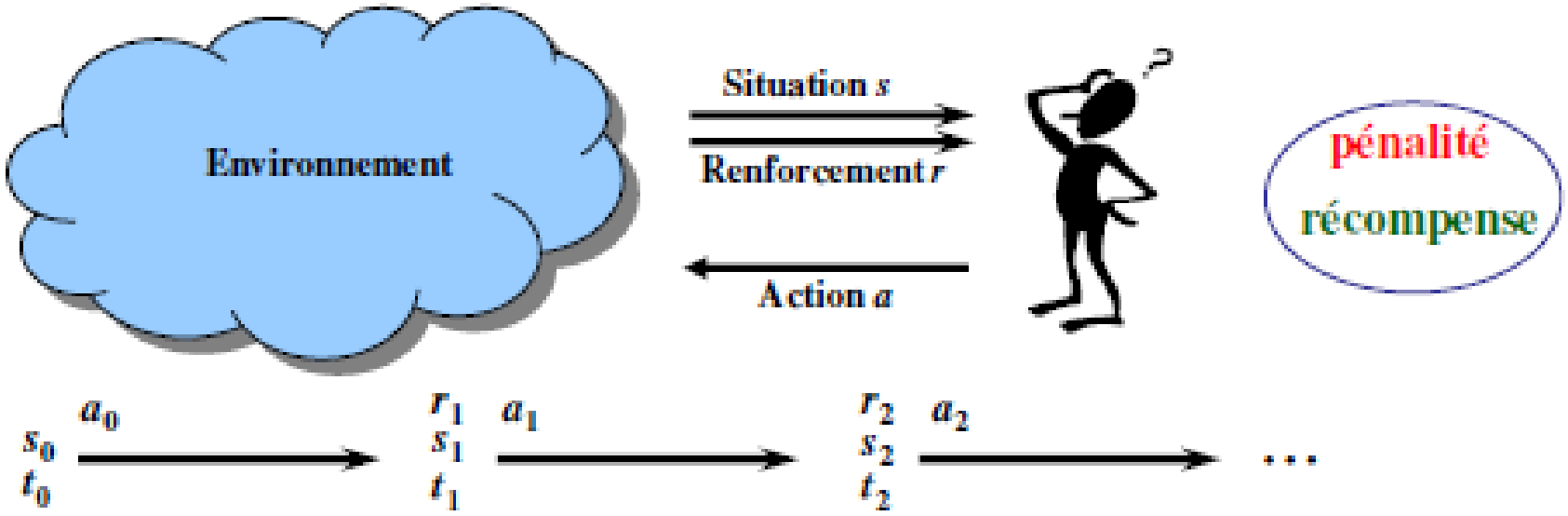
- L'algorithme apprend un comportement étant donné une observation.
- L'action de l'algorithme sur l'environnement produit une valeur de retour qui guide l'algorithme d'apprentissage.

- Exemple : attribuer des récompenses pour un modèle qui joue aux échecs

+1 si le modèle gagne une partie
-1 si le modèle perd une partie

Types d'apprentissage

L'apprentissage par renforcement



- L'agent apprend à se rapprocher d'une **stratégie comportementale optimale** par des interactions répétitives avec l'environnement
- Les décisions sont prises séquentiellement à des intervalles de temps discrets

Algorithmes d'apprentissage

- K-plus proches voisins (KNN)
- Réseaux de neurones artificiels
- Méthodes bayésiennes
- Modèles de Markov Cachés (HMM)
- Algorithmes génétiques
- Arbres de décision
- …

K plus proches voisins

Principe

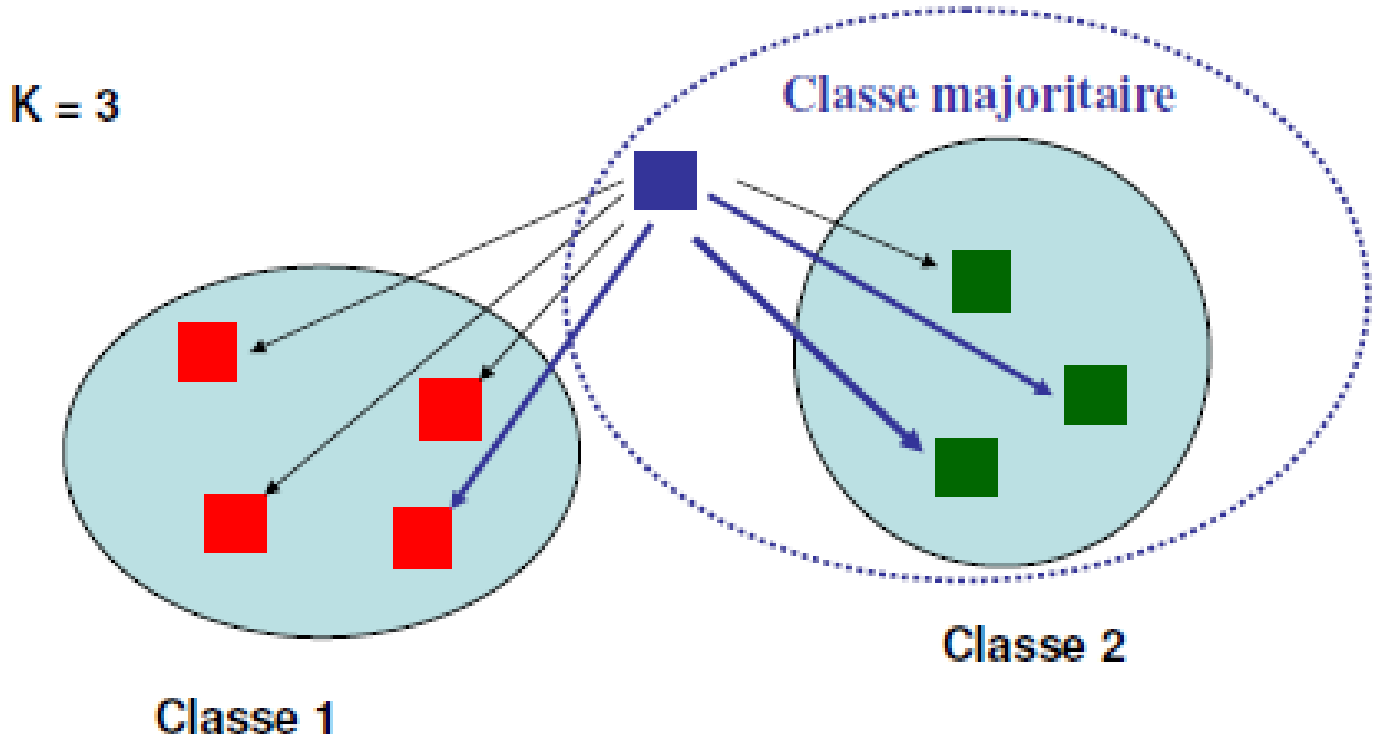
- On dispose d'une base de données d'apprentissage constituée de *m* couples « entrée-sortie ».

- Pour estimer la sortie associée à une nouvelle entrée x, la méthode consiste à prendre en compte les *k* échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée x, selon une distance à définir.

K plus proches voisins

Exemple 1 : Classification de formes géométriques

On retiendra **la classe la plus représentée** parmi les *k* sorties associées aux *k* entrées les plus proches de la nouvelle entrée x.



K plus proches voisins

Exemple 2 : reconnaissance de caractères

e ou o ?

- Exemple d'entrainement (100 exemples d'apprentissage par classe)

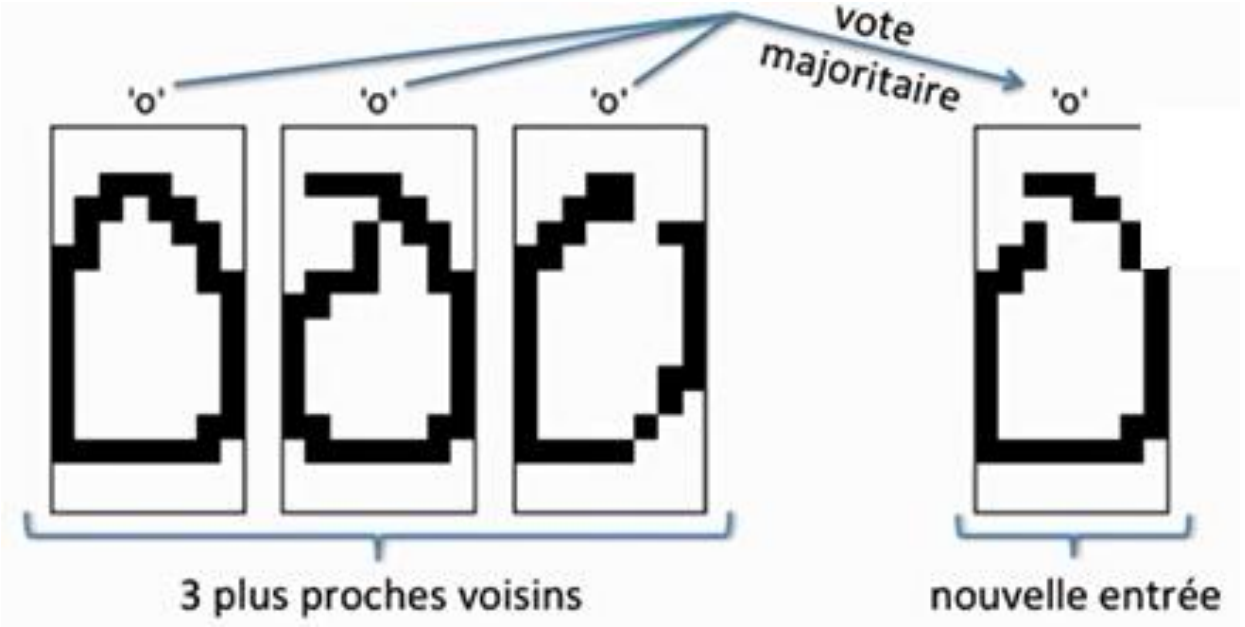
Classe : e

Classe : o

K plus proches voisins

Exemple 2 : reconnaissance de caractères

e ou o ?

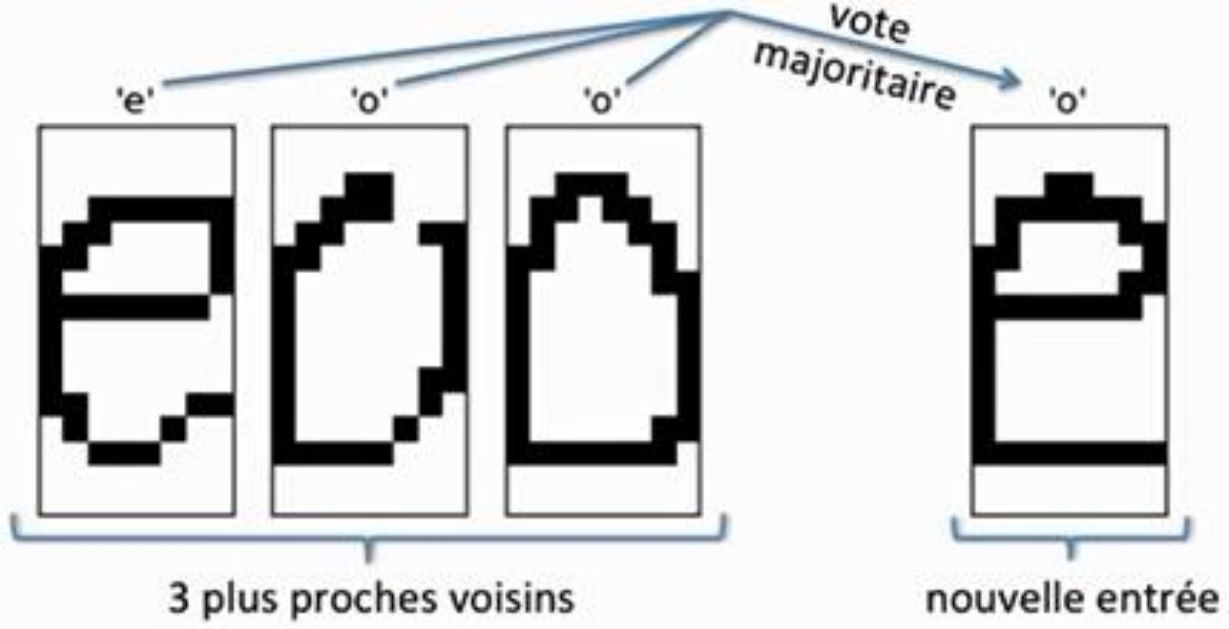


✓ vrai

K plus proches voisins

Exemple 2 : reconnaissance de caractères

e ou o ?



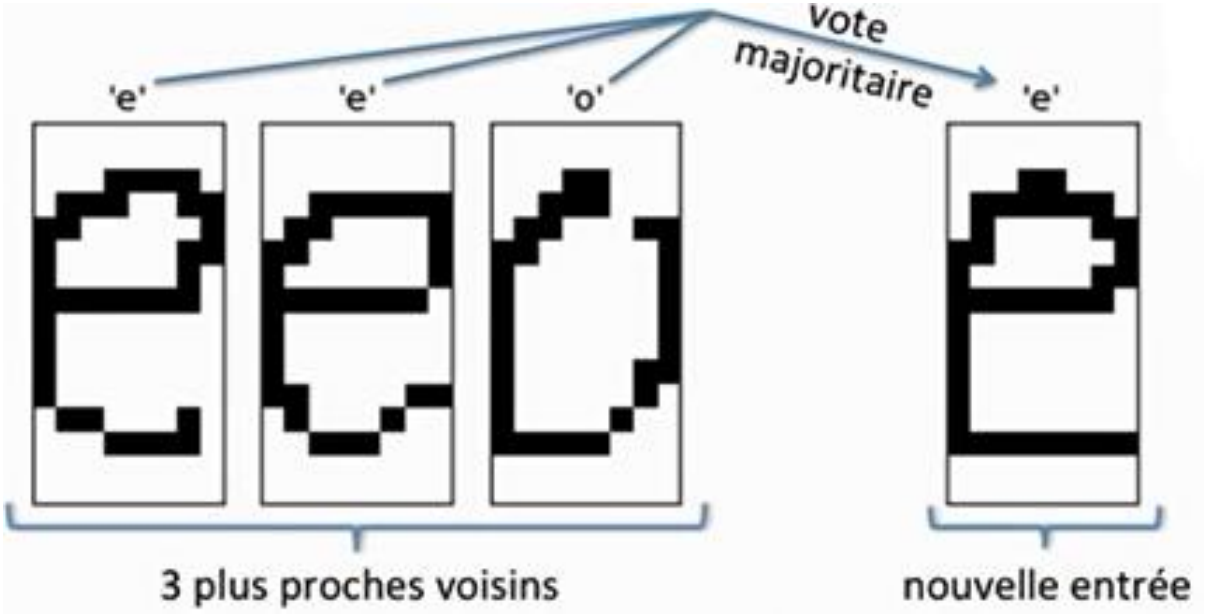
✘ faux

K plus proches voisins

Exemple 2 : reconnaissance de caractères

e ou o ?

Si on ajoute
200 exemples
par classe



✓ vrai

K plus proches voisins

Algorithme

- **Paramètre :** le nombre k de voisins
- **Donnée :** un échantillon de m exemples et leurs classes
  - La classe d'un exemple X est c(X)
- **Entrée :** un enregistrement Y
- Déterminer les k plus proches exemples de Y en calculant les distances
- Combiner les classes de ces k exemples en une classe c
- **Sortie :** la classe de Y est c(Y)=c

K plus proches voisins

Distance

- Le choix de la distance est primordial au bon fonctionnement de la méthode
- Les distances les plus simples permettent d'obtenir des résultats satisfaisants
- Propriétés de la distance:
  - $d(A,A) = 0$
  - $d(A,B) = d(B,A)$
  - $d(A,B) \leq d(A,C) + d(B,C)$

K plus proches voisins

Calcul de la distance

- $d(x,y) = |x-y|$
- $d(x,y) = |x-y|/d_{max}$, où d_{max} est la distance maximale entre deux numériques du domaine considéré

K plus proches voisins

Exemples de distances

- Données binaires : 0 ou 1.

  On choisit d(0,0)=d(1,1)=0 et d(0,1)=d(1,0)=1.

- Données énumératives :

  La distance vaut 0 si les valeurs sont égales et 1 sinon.

- Données énumératives ordonnées : elles peuvent être considérées comme des valeurs énumératives mais on peut également définir une distance utilisant la relation d'ordre.

  - Exemple: Si un champ prend les valeurs A, B, C, D et E, on peut définir la distance en considérant 5 points de l'intervalle [0,1] avec une distance de 0,25 entre deux points successifs, on a alors d(A,B)=0,25 ; d(A,C)=0,5 ; ...

K plus proches voisins

Distance euclidienne

Soit X = (x_1,..., x_n) et Y = (y_1,..., y_n) deux exemples, la distance euclidienne entre X et Y est:

$$D(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

K plus proches voisins

Exercice 1 (3 plus proches voisins)

Client	Age	Revenu	Fidèle
Ahmed	35	35m	Non
Khadidja	22	50m	Oui
Fatima	63	200m	Non
Abdellah	59	170m	Non
Safia	25	40m	Oui
Abderrahmane	37	50m	?

Déterminer la classe de Abderrahmane (Fidèle ou non) **?**

K plus proches voisins

Exercice 1 (3 plus proches voisins)

Client	Age	Revenu	Fidèle	Distance avec Abderrahmane
Ahmed	35	35m	Non	D(Abderrahmane,Ahmed)=Sqrt[(35-37)^2+(35-50)^2]=15.13
Khadidja	22	50m	Oui	D(Abderrahmane,Khadidja)=Sqrt[(22-37)^2+(50-50)^2]=15
Fatima	63	200m	Non	D(Abderrahmane,Fatima)=Sqrt[(63-37)^2+(200-50)^2]=152.23
Abdellah	59	170m	Non	D(Abderrahmane,Abdellah)=Sqrt[(59-37)^2+(170-50)^2]=122
Safia	25	40m	Oui	D(Abderrahmane,Safia)=Sqrt[(25-37)^2+(40-50)^2]=15.62
Abderrahmane	37	50m	Oui	Classe majoritaire

K plus proches voisins

Exercice 2 (3 plus proches voisins)

On dispose d'une base de données d'apprentissage constituée de 5 couples « entrée-sortie ».

(Abdellah, Admis), (Ahmed, Admis), (Khaled, Ajourné), (Salim, Ajourné) et (Salah, Admis).

Pour chaque étudiant, on dispose aussi de 4 notes dans 4 matières différentes :

- Abdellah :14, 12, 8 et 12.
- Ahmed :12, 12, 6 et 10.
- Khaled : 8, 9, 9 et 1.
- Salim : 15, 11, 3 et 5.
- Salah : 12, 9, 14 et 11.

On dispose maintenant d'une nouvelle entrée "Karim" qui a pour notes : 9,14,15 et 6.

En utilisant la méthode des k plus proches voisins (k =3) et en choisissant la distance euclidienne,

Déterminez la classe de Karim **?**.

K plus proches voisins

Exercice 2 (3 plus proches voisins)

Etudiant	Notes	Classe	Distance
Abdellah	14, 12, 8, 12	Admis	$D(Abdellah,Karim)=SQRT[(14-9)^2+(12-14)^2+(8-15)^2+(12-6)^2]$ $=SQRT[25+4+49+36]=SQRT(114)=10.67$
Ahmed	12, 12, 6 et 10	Admis	$D(Ahmed,Karim)=SQRT[(12-9)^2+(12-14)^2+(6-15)^2+(10-6)^2]$ $=SQRT[9+4+81+16]=SQRT(110)=\mathbf{10.48}$
Khaled	8, 9, 9, 1	Ajourné	$D(Khaled,Karim)=SQRT[(8-9)^2+(9-14)^2+(9-15)^2+(1-6)^2]$ $=SQRT[1+25+36+25]=SQRT(87)=\mathbf{9.32}$
Salim	15, 11, 3, 5	Ajourné	$D(Salim,Karim)=SQRT[(15-9)^2+(11-14)^2+(3-15)^2+(5-6)^2]$ $=SQRT[36+9+144+1]=SQRT(190)=13.78$
Salah	12, 9, 14, 11	Admis	$D(Salah,Karim)=SQRT[(12-9)^2+(9-14)^2+(14-15)^2+(11-6)^2]$ $=SQRT[9+25+1+25]=SQRT(60)=\mathbf{7.74}$
Karim	9, 14, 15, 6	**Admis**	

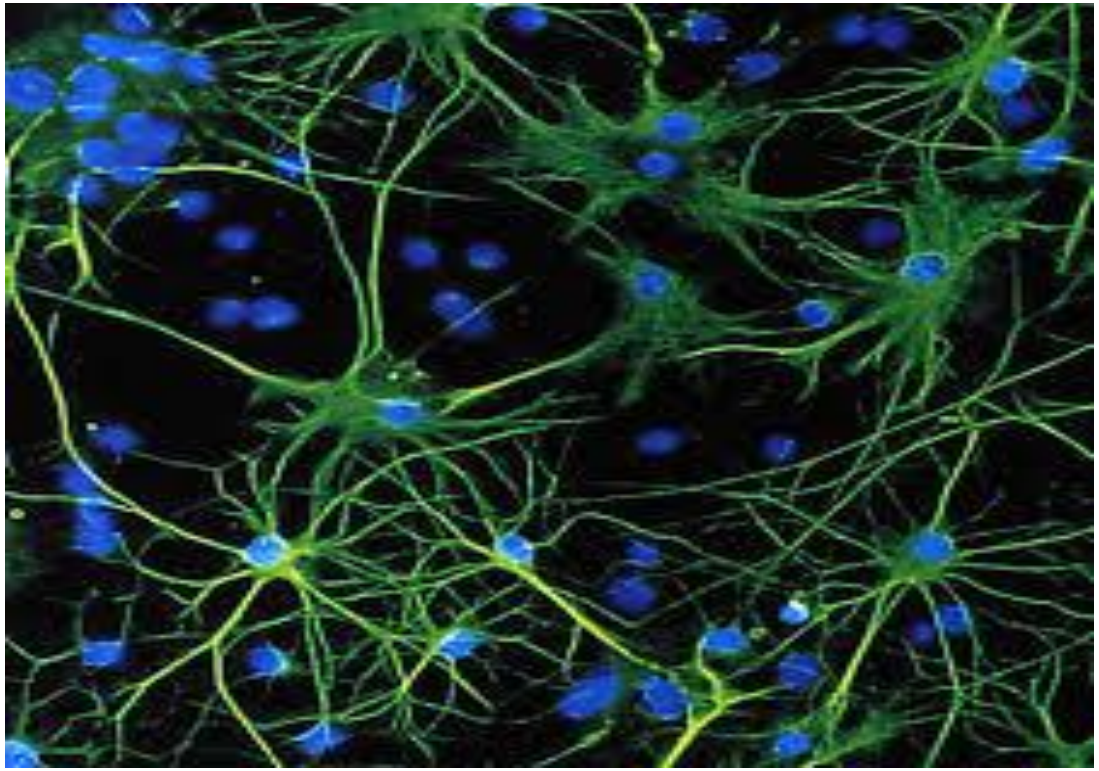
LES RESEAUX DE NEURONES ARTIFICIELS

- **McCulloch et Pitts (1943) :** Naissance du connexionnisme.
- **Rosenblatt (1957) :** Premier modèle opérationnel (Perceptron).
  - RN inspiré du système visuel
  - Apprendre certaines fonctions logiques

LES RESEAUX DE NEURONES

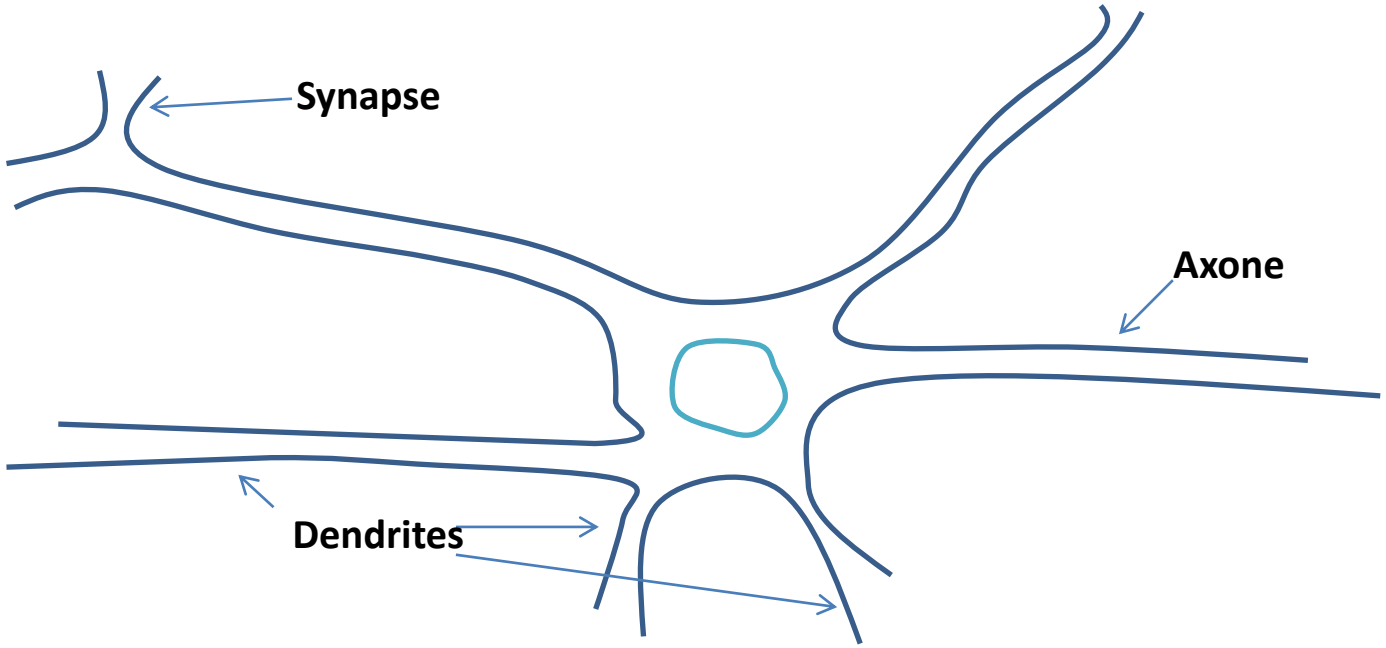
FONDEMENTS BIOLOGIQUES

- **CERVEAU**
  - Centre de contrôle de la perception, de la décision et de l'action.
  - 10^{13} neurones, chacun d'entre eux est connecté en moyenne à 1000 autres neurones.



LES RESEAUX DE NEURONES

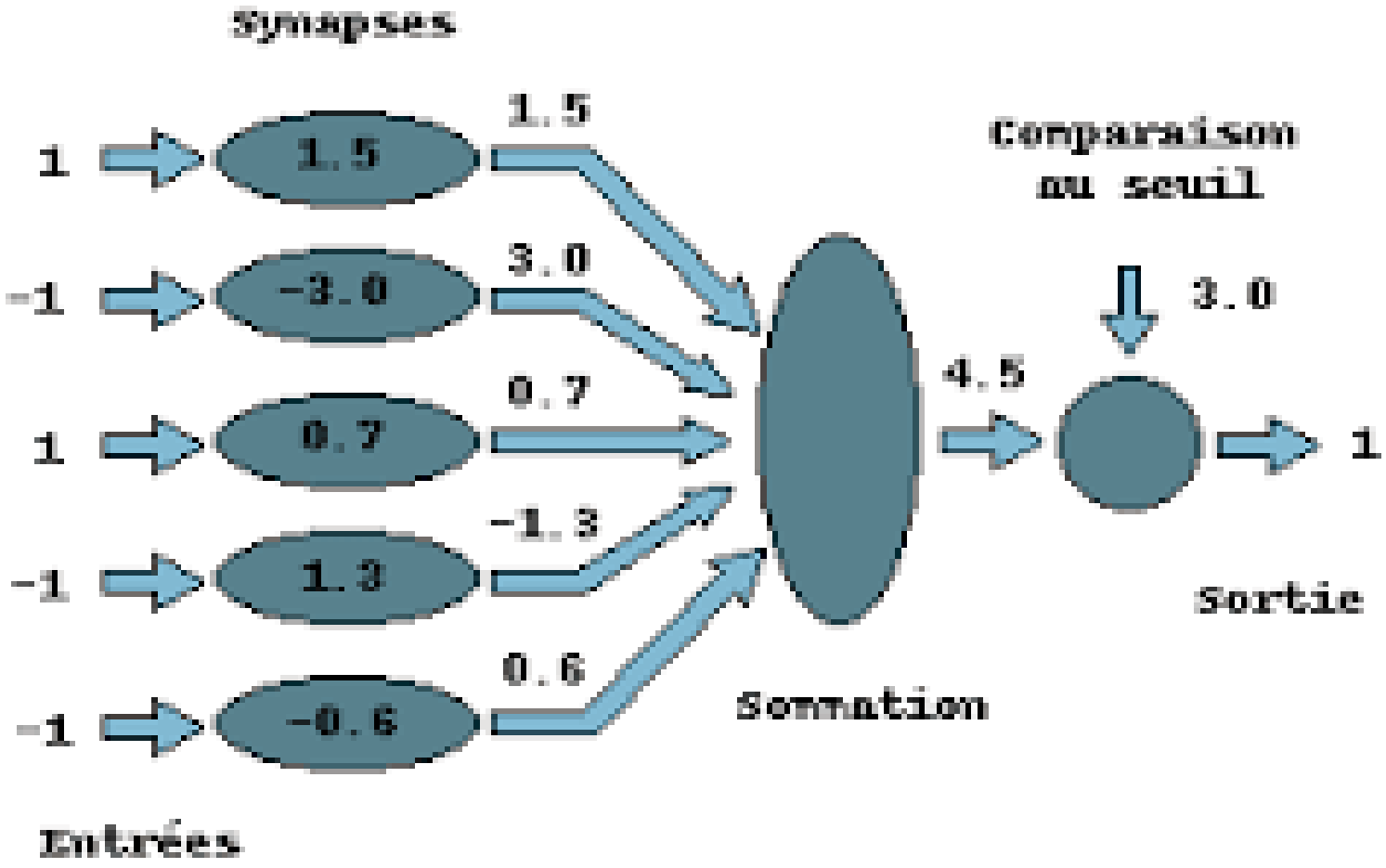
NEURONE BIOLOGIQUE



- Le neurone reçoit des impulsions (informations)des neurones voisins via les **Dendrites**
- Effectuer une **sommation** de ces impulsions
- Distribution de l'activité calculée aux neurones voisins via l'**Axone**.
- **Synapse** : contact entre fibres nerveuses, rôle quantitatif dans la transmission

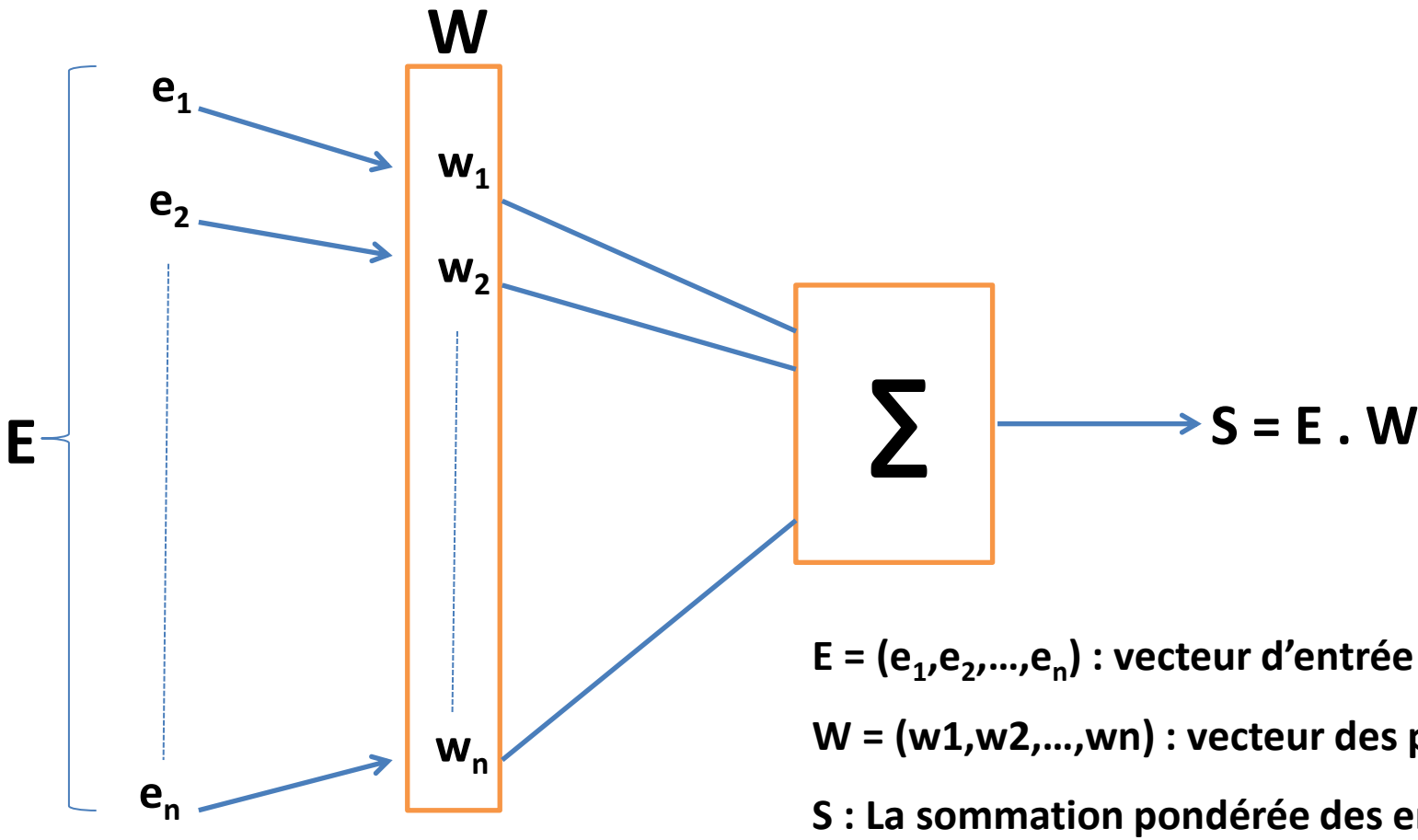
LES RESEAUX DE NEURONES

NEURONE ARTIFICIEL



LES RESEAUX DE NEURONES

Formalisation : neurone formel



E = (e_1,e_2,…,e_n) : vecteur d'entrée du neurone

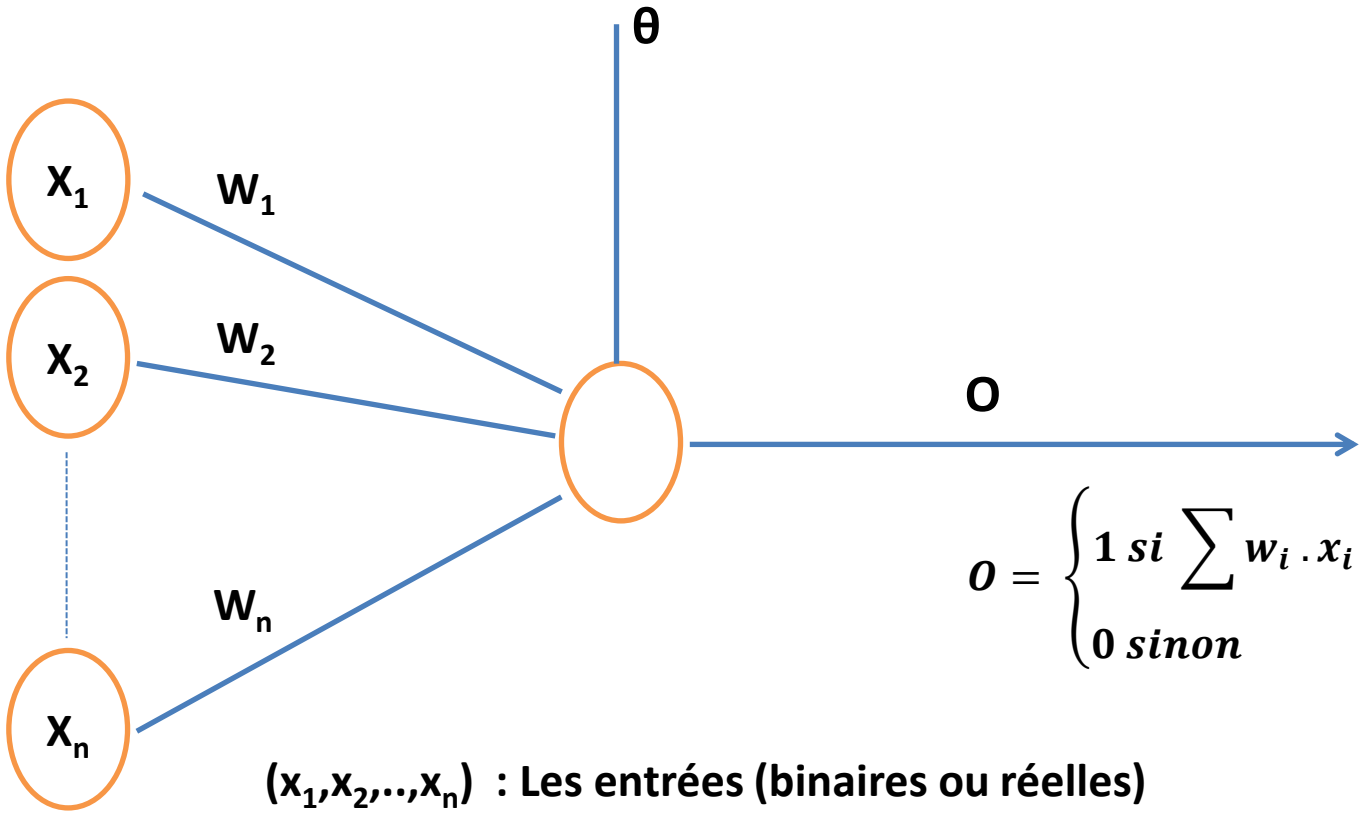
W = (w1,w2,…,wn) : vecteur des poids

S : La sommation pondérée des entrées

$S = \sum_{i=1}^{n} e_i \cdot w_i$

LES RESEAUX DE NEURONES

Modèle de perceptron : Schéma de base



$$O = \begin{cases} 1 \; si \; \sum w_i \, . \, x_i > \theta \\ 0 \; sinon \end{cases}$$

$(x_1,x_2,..,x_n)$: Les entrées (binaires ou réelles)

$(w_1,w_2,..,w_n)$: Vecteur des poids (Coefficients synaptiques)

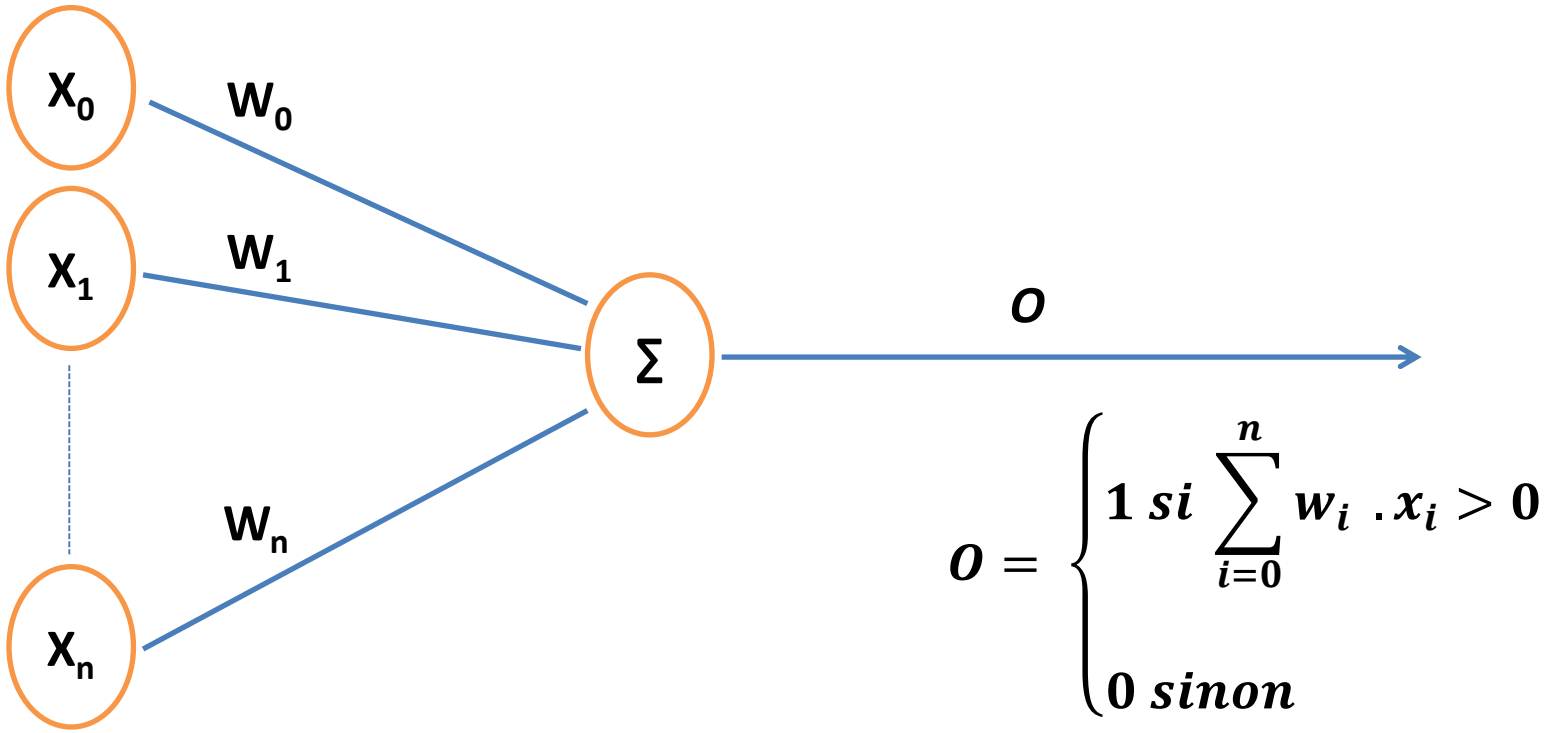
θ : Seuil (Biais)

O : Sortie

LES RESEAUX DE NEURONES

Modèle de perceptron : Schéma simplifié

Remplacer le seuil **θ** par une entrée supplémentaire $\mathbf{x_0}$ qui prend toujours comme valeur 1, à cette entrée est associé un coefficient $\mathbf{w_0}$



$$O = \begin{cases} 1 \; si \sum_{i=0}^{n} w_i \; . x_i > 0 \\ 0 \; sinon \end{cases}$$

LES RESEAUX DE NEURONES

Modèle de perceptron

ALGORITHME D'APPRENTISSAGE PAR CORRECTION D'ERREUR

Etant donné un échantillon d'apprentissage **S** de $\mathbf{R^n \times \{0,1\}}$ ou bien de $\mathbf{\{0,1\}^n \times \{0,1\}}$

- Un ensemble d'exemples dont les descriptions sont sur n attributs réels ou binaires et la classe résultat est binaire.
  - Trouver un algorithme qui infère à partir de S, un échantillon qui classifie correctement les éléments de S en vue de leurs descriptions.

LES RESEAUX DE NEURONES

ALGORITHME D'APPRENTISSAGE PAR CORRECTION D'ERREUR

PROCEDURE

- Initialiser les poids $\mathbf{W_i}$ du perceptron à des valeurs quelconques.
- A chaque fois qu'on présente un nouvel exemple, on ajuste les poids selon que le perceptron a correctement classé l'exemple ou non.
- Arrêt lorsque tous les exemples ont étés présentés sans modification d'aucun poids.

LES RESEAUX DE NEURONES

ALGORITHME D'APPRENTISSAGE PAR CORRECTION D'ERREUR

NOTATION

- On note $\vec{x}$ une description qui sera un élément de l'échantillon. La **i^{ème}** composante de $\vec{x}$ sera notée $\mathbf{x_i}$.

- Un échantillon **S** sera donc un ensemble de couples ($\vec{x}$,**c**) où **c** est la classe de $\vec{x}$.

- Notant que : $\mathbf{x_0=1}$ à qui correspond $\mathbf{w_0}$

LES RESEAUX DE NEURONES

ALGORITHME D'APPRENTISSAGE PAR CORRECTION D'ERREUR

Entrée : un échantillon S de R^n x {0,1} ou bien de $\{0,1\}^n$ x {0,1}
Début
- Initialisation aléatoire des poids w_i (w_0,w_1,..,w_n).
Répéter
-Prendre un exemple ($\vec{x}$,**c**) de **S**
-Calculer la sortie **O** du perceptron pour l'entrée $\boldsymbol{x}$
-Mise à jour des poids (Ajustement)
Pour i := 0 jusqu'à n
Faire
$w_i \leftarrow w_i + (c-o) \times x_i$
Fait
Fin_Répéter
Fin

LES RESEAUX DE NEURONES

Modèle de perceptron : Exemple 1

- On souhaite construire un perceptron qui calcule le ET logique en utilisant un algorithme d'apprentissage par correction d'erreur.
- Nous avons comme échantillon S={(00,0),(01,0),(10,0),(11,1)}.
- Les poids initiaux sont : -1, 1, 1.

- Critère d'arrêt : présentation de tous les exemples de l'échantillon.

➢ Reproduire la trace d'exécution de l'algorithme dans un tableau :

LES RESEAUX DE NEURONES

Modèle de perceptron : Exemple 1

Sachant que : $O = \begin{cases} 1 \; si \; \sum_{i=0}^{n} w_i \; . x_i > 0 \\ 0 \; sinon \end{cases}$ et $w_i := w_i + (C\text{-}O) \; . \; x_i$

Etape	W_0	W_1	W_2	X_0	X_1	X_2	$\sum w_i.x_i$	O	C	W_0	W_1	W_2
1	-1	1	1	1	0	0	-1	0	0	-1	1	1
2	-1	1	1	1	0	1	0	0	0	-1	1	1
3	-1	1	1	1	1	0	0	0	0	-1	1	1
4	-1	1	1	1	1	1	1	1	1	-1	1	1

On remarque une stabilisation des poids dès la première itération. On dit alors que le perceptron a pu apprendre le calcul du ET logique.

LES RESEAUX DE NEURONES

Modèle de perceptron : Exemple 2

On souhaite construire un perceptron qui calcule le **XOR** en utilisant un algorithme d'apprentissage par correction d'erreur.

Les poids (w_i) initiaux sont : -1, 1, 1.

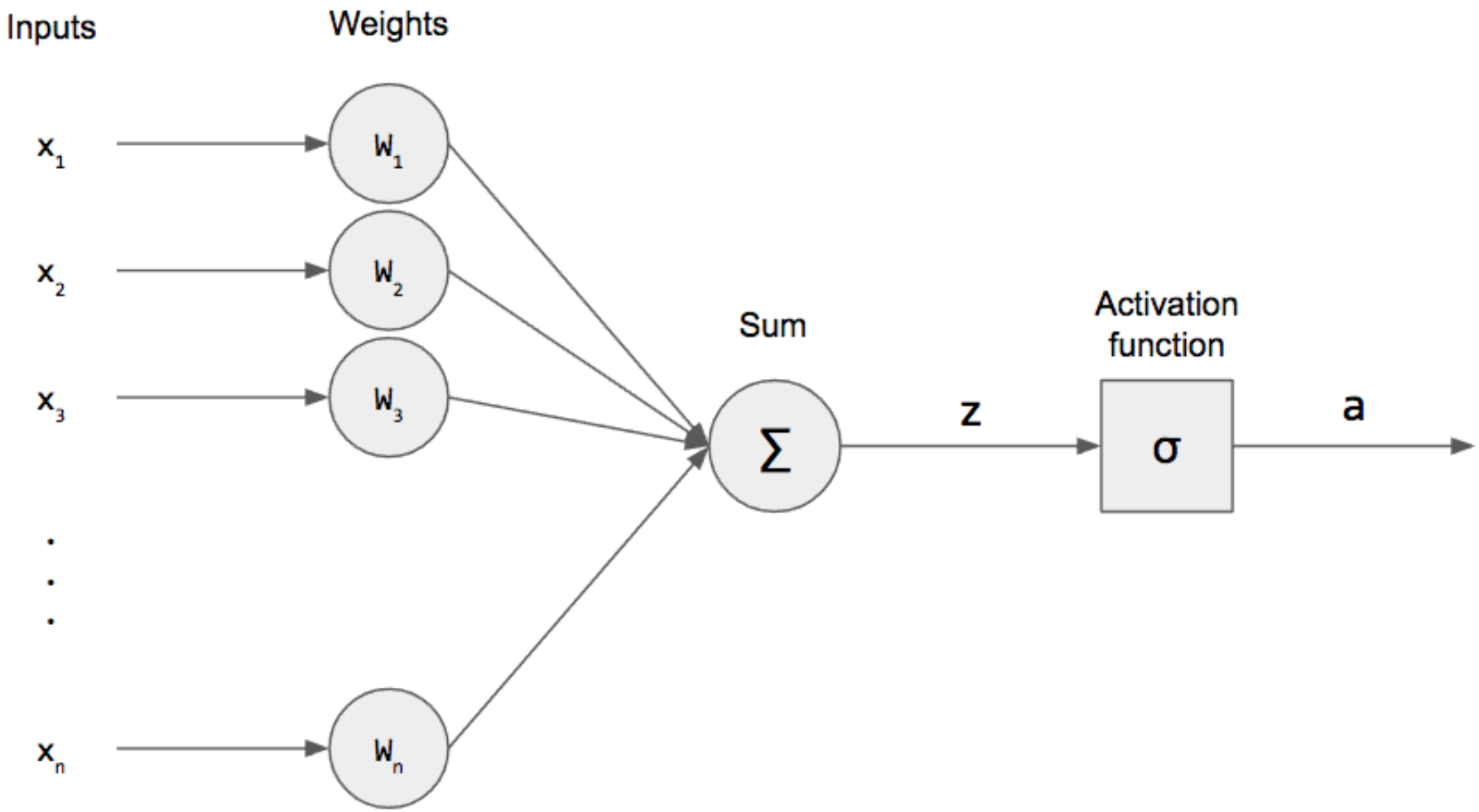
Critère d'arrêt : présentation de tous les exemples de l'échantillon.

$$O = \begin{cases} 1 \; si \; \sum_{i=0}^{n} w_i \; . x_i > 0 \\ 0 \; sinon \end{cases} \quad \text{et} \quad w_i := w_i + (C\text{-}O) \; . \; x_i$$

- Donner la structure de l'échantillon d'entrée (les couples (entrée,sortie))
- Reproduire la trace d'exécution de l'algorithme dans un tableau
- Que peut-on déduire **?**

LES RESEAUX DE NEURONES

Perceptron simple

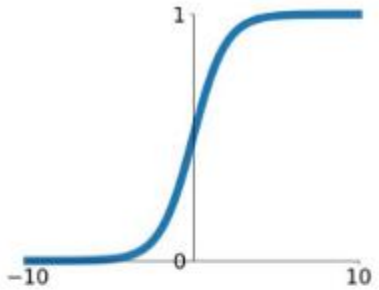


LES RESEAUX DE NEURONES

Fonctions d'activation

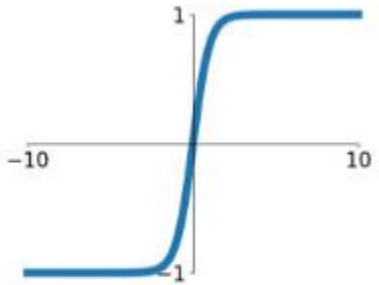
Sigmoid

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



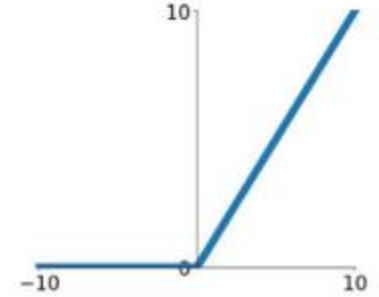
tanh

$$\tanh(x)$$



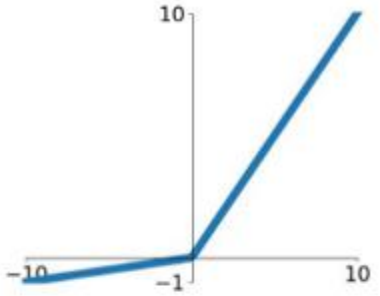
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

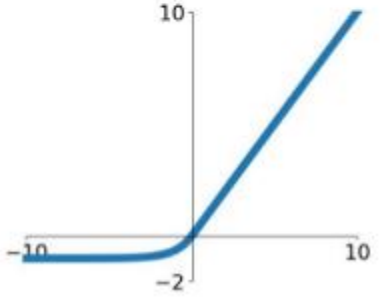


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

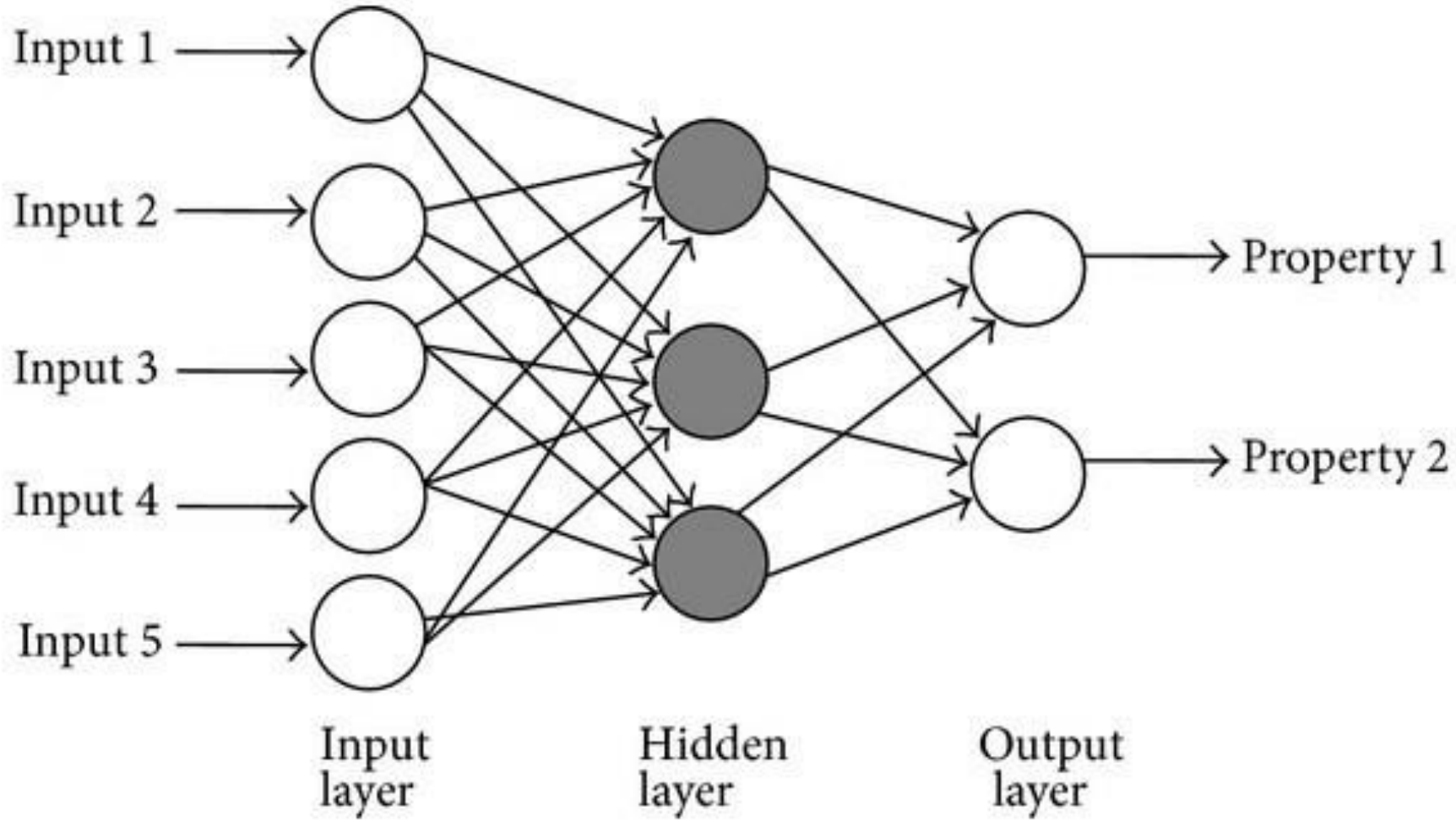
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



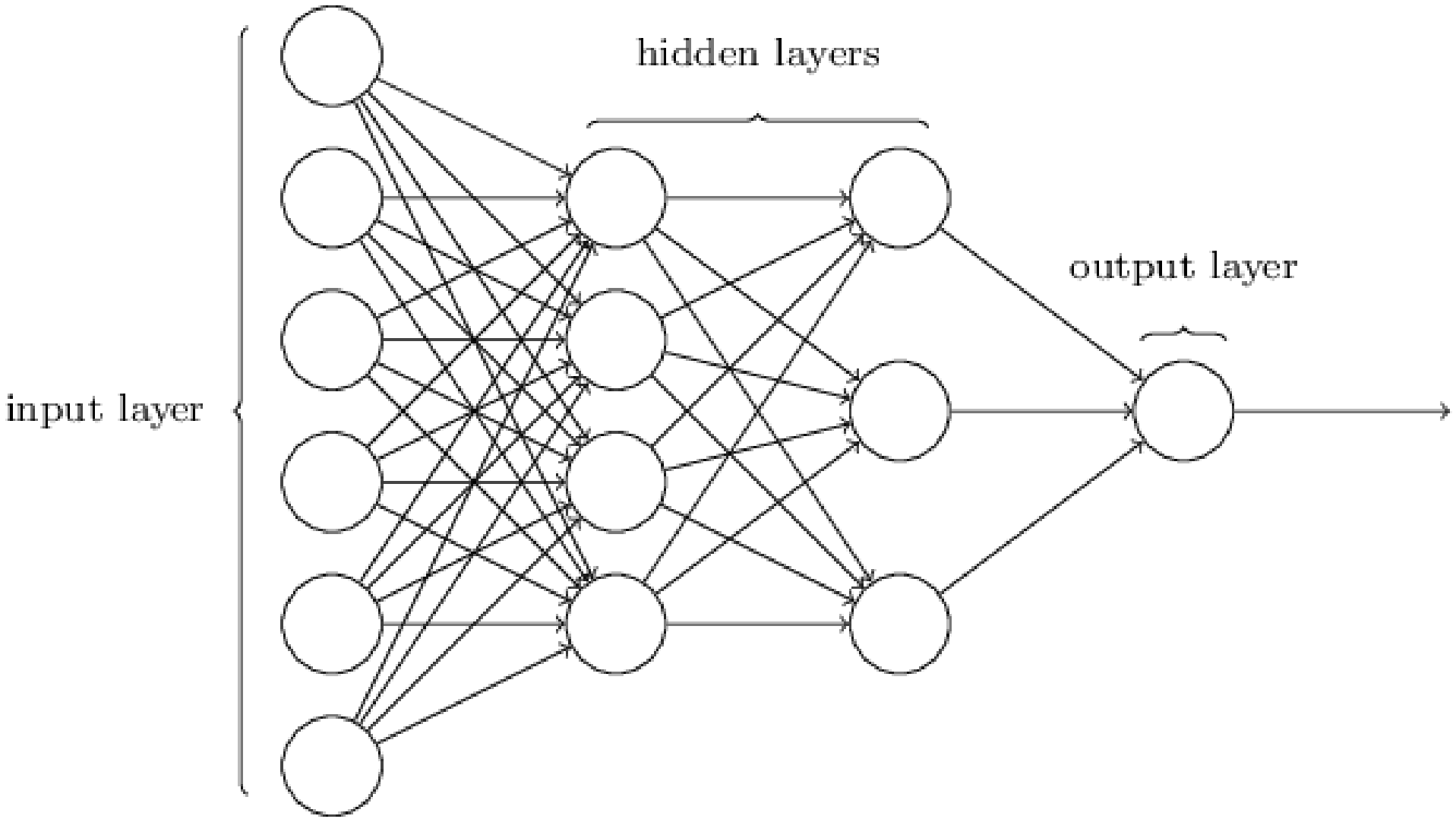
LES RESEAUX DE NEURONES

Perceptron (à couche cachée)



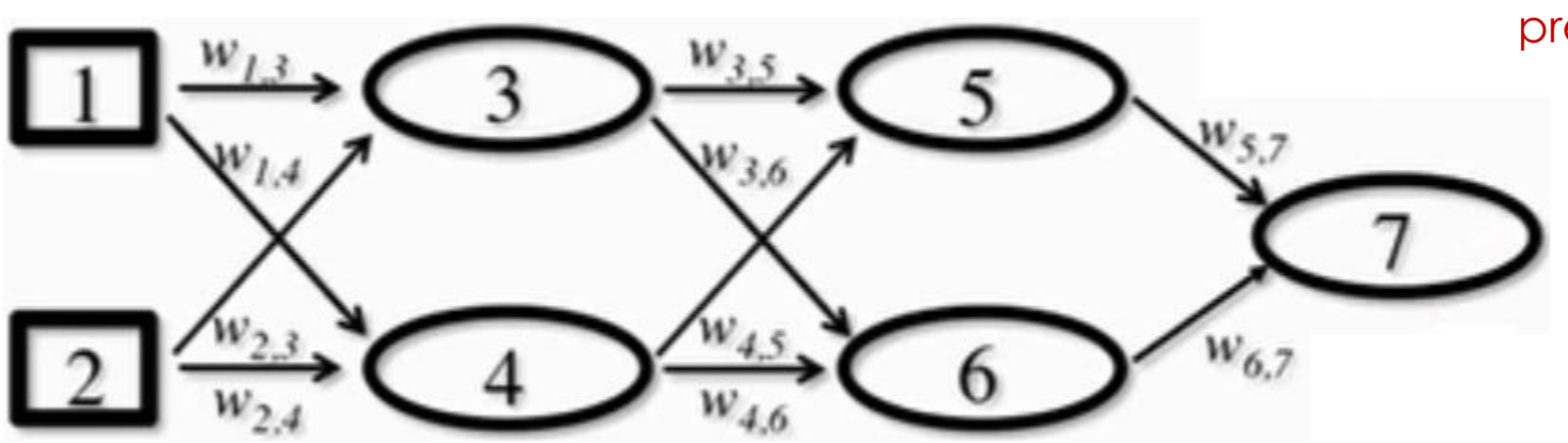
LES RESEAUX DE NEURONES

Perceptron multicouches



LES RESEAUX DE NEURONES

Algorithme d'apprentissage par rétropropagation



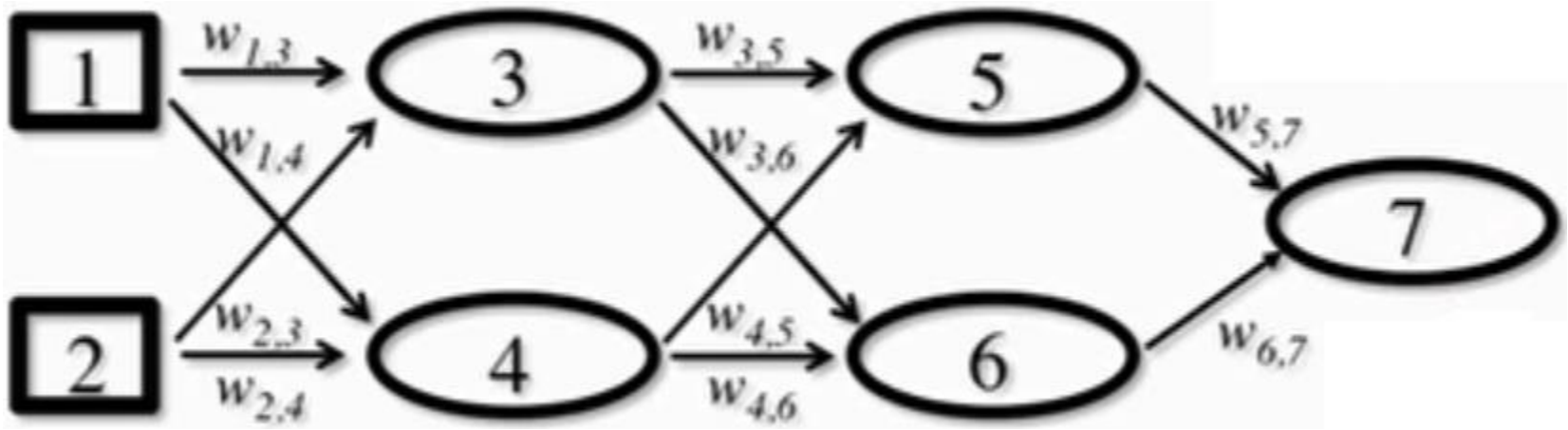
On note :

- a_j l'activité du neurone j
- in_j l'activité du neurone avant l'application de la fonction d'activation
  - $a_j = \sigma(in_j) = \sigma(\sum_i w_{i,j} a_i)$
- $h(x)$ la prédiction

LES RESEAUX DE NEURONES

Algorithme d'apprentissage par rétropropagation

Propagation avant

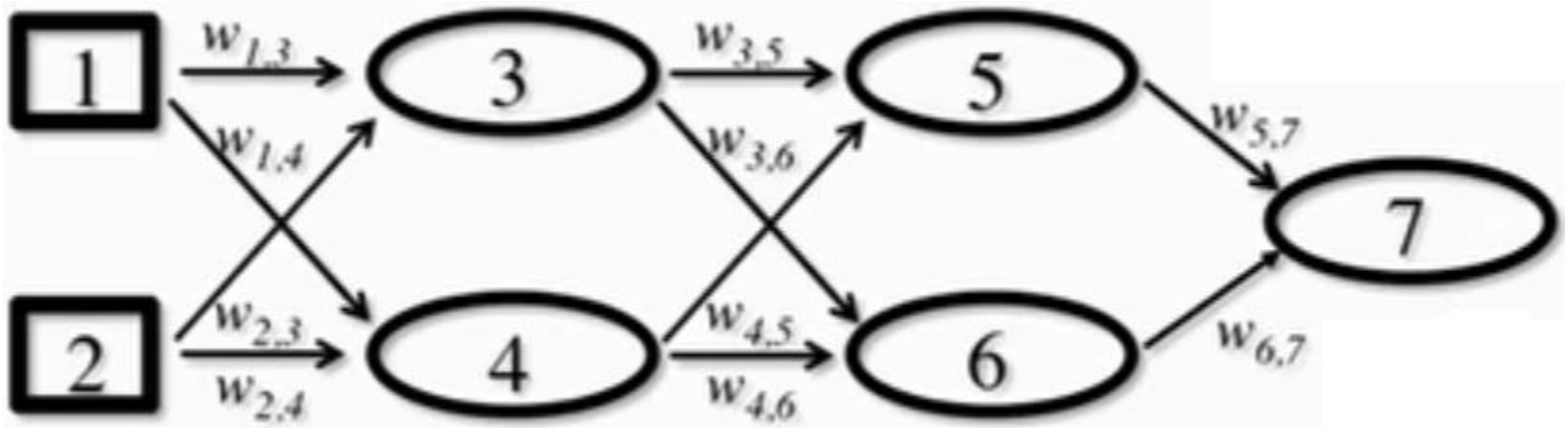


Propagation arrière

LES RESEAUX DE NEURONES

Algorithme d'apprentissage par rétropropagation

Propagation avant

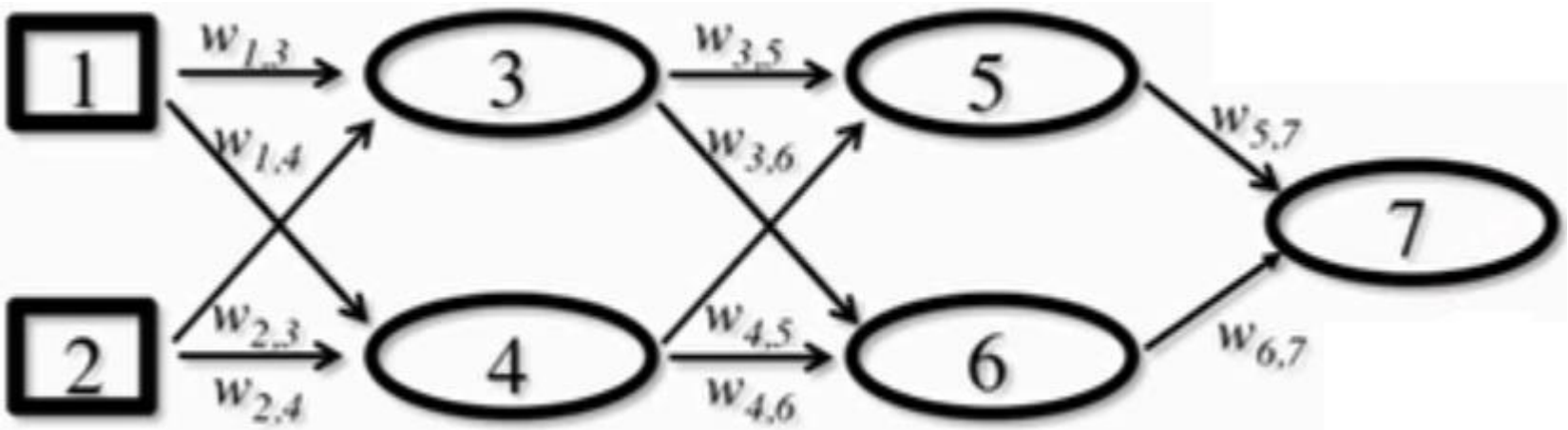


$$a_k = \sigma(\sum_j w_{j,k} a_j)$$

LES RESEAUX DE NEURONES

Algorithme d'apprentissage par rétropropagation

Mise à jour des poids



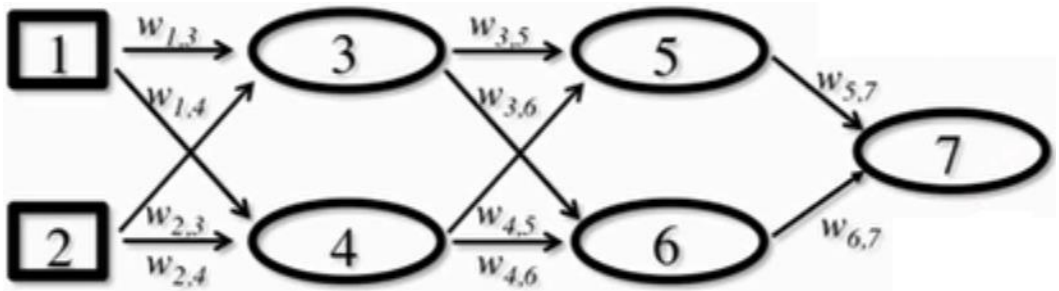
$$w_{i,j} = w_{i,j} - \alpha \frac{\partial}{\partial w_{i,j}} Loss(y_t, h_w(x_t))$$

- α : taux d'apprentissage
- $Loss$: taux de perte associée à la prédiction courante

LES RESEAUX DE NEURONES

Algorithme d'apprentissage par rétropropagation

Mise à jour des poids (simplification)



$$w_{i,j} = w_{i,j} - \alpha \underbrace{\frac{\partial}{\partial in_j} Loss(y_t, h_w(x_t))}_{-\Delta[j]} \underbrace{\frac{\partial}{\partial w_{i,j}} in_j}_{a_i}$$

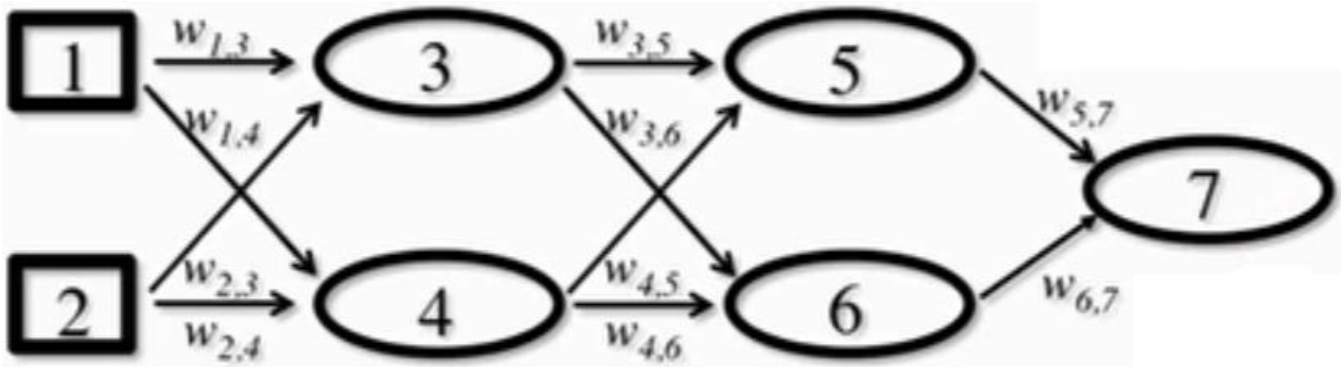
- **Réécriture de la règle de mise à jour :**

$$w_{i,j} = w_{i,j} + \alpha\, a_i \Delta[j]$$

LES RESEAUX DE NEURONES

Algorithme d'apprentissage par rétropropagation
Mise à jour des poids (simplification)

Propagation arrière



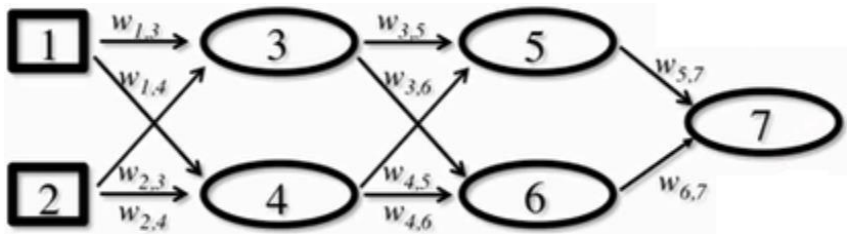
- **Simplification de** $\Delta[j]$**:**

$$\Delta[j] = \sigma(in_j)(1 - \sigma(in_j)) \sum_k w_{j,k} \Delta[k]$$

Algorithme d'apprentissage par rétropropagation

function BACK-PROP-LEARNING(*examples*, *network*) **returns** a neural network
inputs: *examples*, a set of examples, each with input vector **x** and output vector **y**
network, a multilayer network with L layers, weights $w_{i,j}$, activation function g
local variables: Δ, a vector of errors, indexed by network node

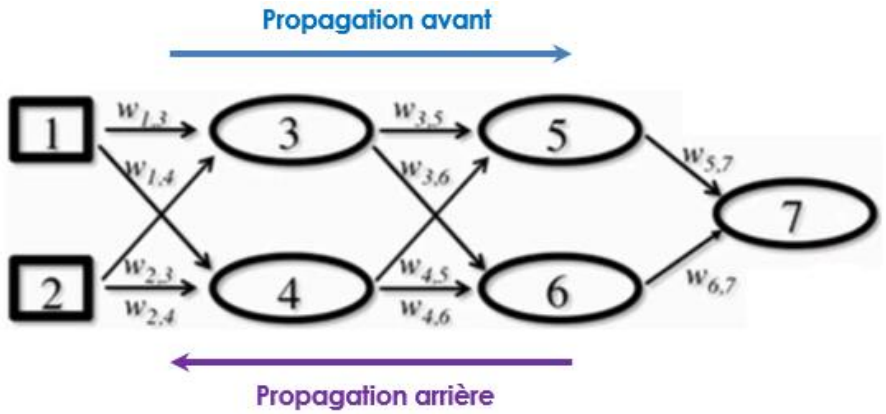
for each weight $w_{i,j}$ in *network* **do**
$w_{i,j} \leftarrow$ a small random number
repeat
for each example $(\mathbf{x}, \mathbf{y})$ **in** *examples* **do**
/* *Propagate the inputs forward to compute the outputs* */
for each node i in the input layer **do**
$a_i \leftarrow x_i$
for $\ell = 2$ **to** L **do**
for each node j in layer ℓ **do**
$in_j \leftarrow \sum_i w_{i,j}\, a_i$
$a_j \leftarrow g(in_j)$
/* *Propagate deltas backward from output layer to input layer* */
for each node j in the output layer **do**
$\Delta[j] \leftarrow y_j - a_j \quad (= -\partial Loss/\partial in_j)$
for $\ell = L - 1$ **to** 1 **do**
for each node i in layer ℓ **do**
$\Delta[i] \leftarrow g(in_i)(1 - g(in_i)) \sum_j w_{i,j}\Delta[j]$
/* *Update every weight in network using deltas* */
for each weight $w_{i,j}$ in *network* **do**
$w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta[j]$
until some stopping criterion is satisfied
return *network*



$g = sigmoid$

LES RESEAUX DE NEURONES

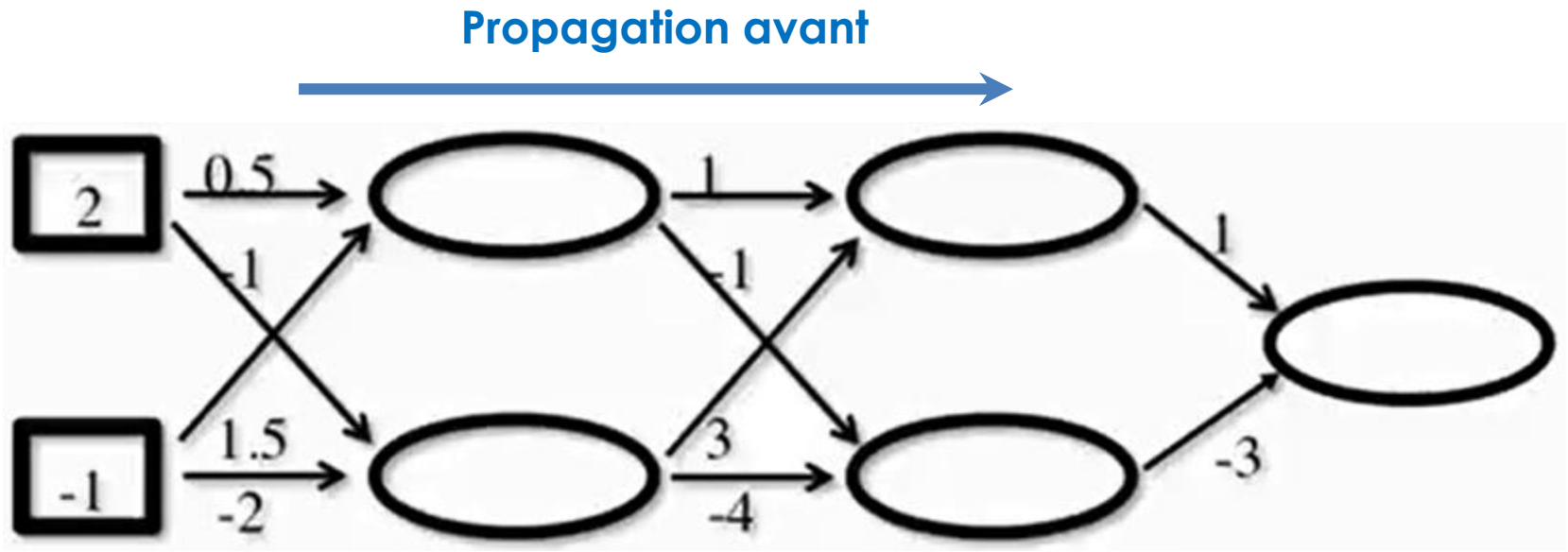
Algorithme d'apprentissage par rétropropagation



- **Propagation avant :** $a_k = \sigma(\sum_j w_{j,k} a_j)$
- **Mise à jour des poids :** $w_{i,j} = w_{i,j} + \alpha\, a_i \Delta[j]$
- **Ecart de la perte :** $\Delta[j] = \sigma(in_j)(1 - \sigma(in_j)) \sum_k w_{j,k} \Delta[k]$

Algorithme d'apprentissage par rétropropagation

Exemple : $x = [2, -1], \quad y = 1, \quad \alpha = 0.1$

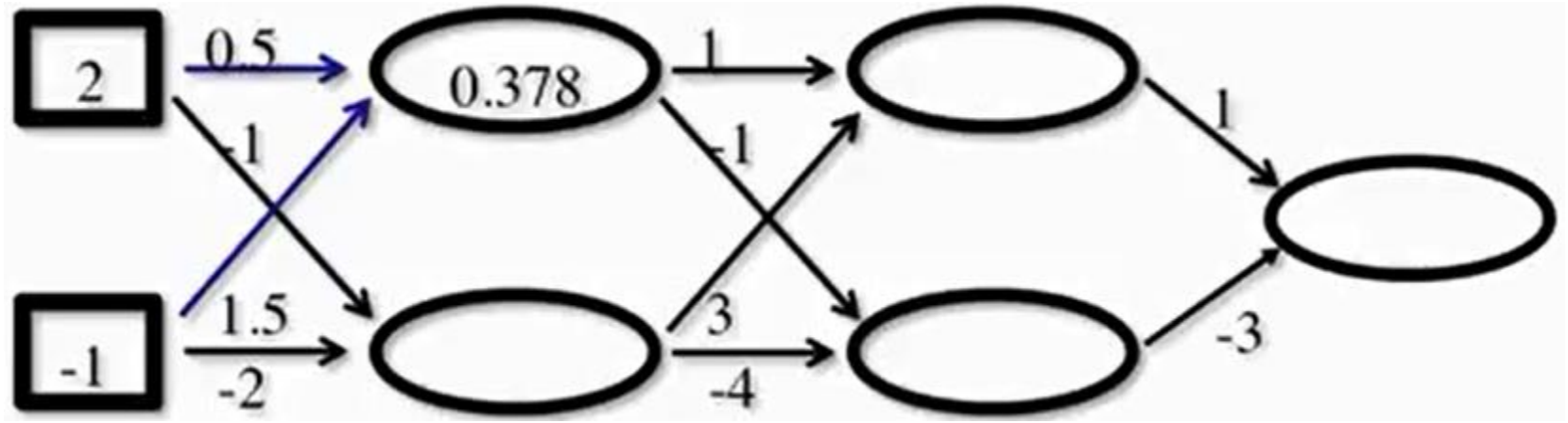


$$a_k = \sigma(\sum_j w_{j,k} a_j)$$

Algorithme d'apprentissage par rétropropagation

Exemple : $x = [2, -1], \quad y = 1, \quad \alpha = 0.1$

Propagation avant



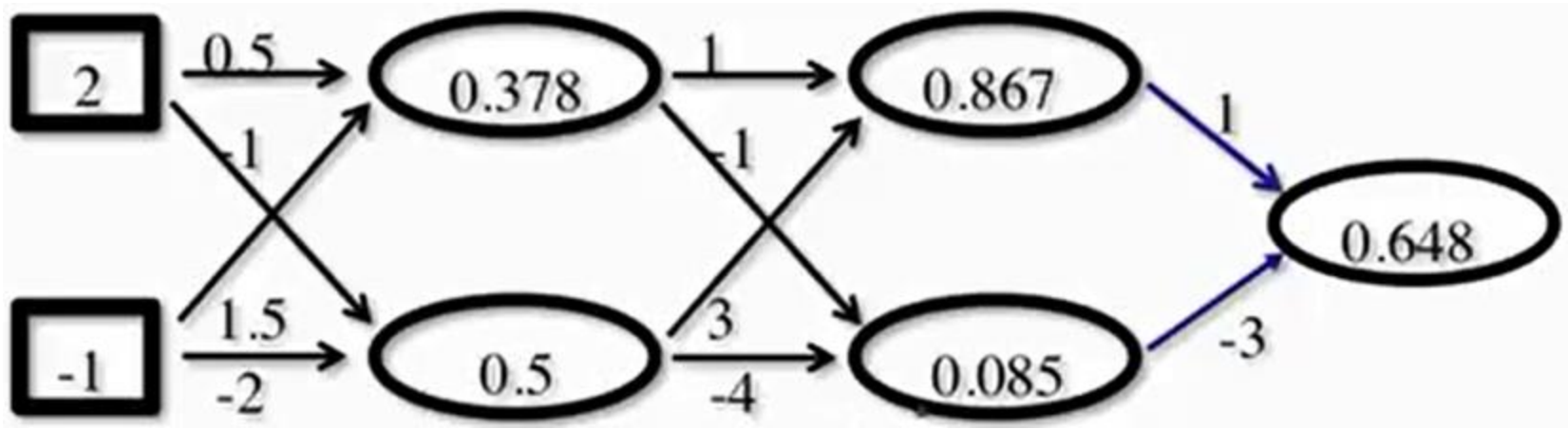
$$a_k = \sigma\left(\sum_j w_{j,k} a_j\right)$$

$$a_3 = \sigma(0.5 \times 2 + 1.5 \times -1) = \sigma(-0.5) = 0.378$$

Algorithme d'apprentissage par rétropropagation

Exemple : $x = [2, -1], \quad y = 1, \quad \alpha = 0.1$

Propagation avant



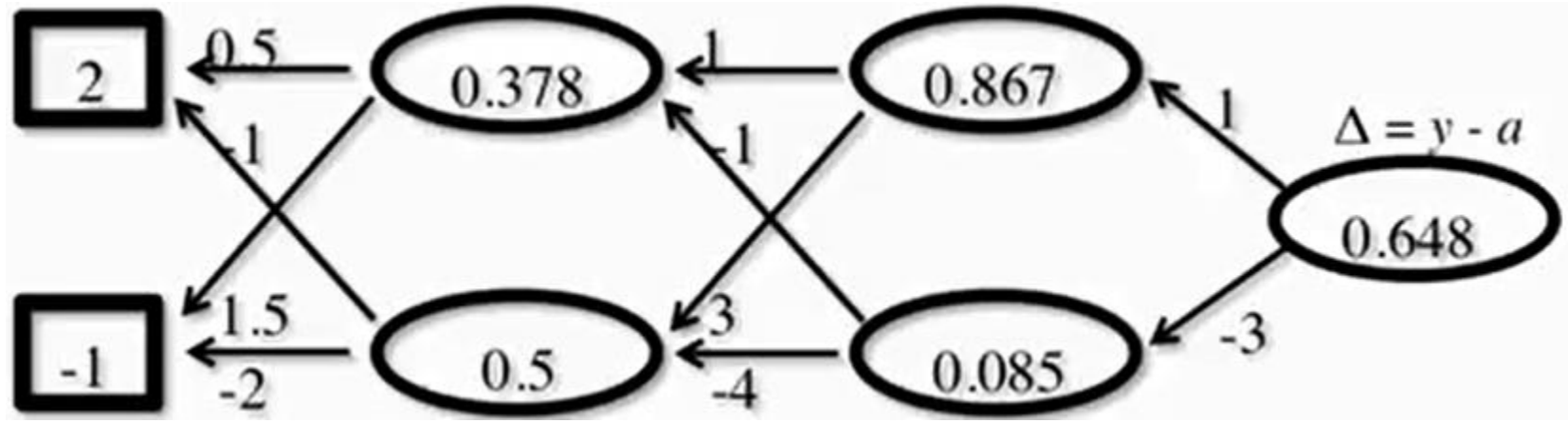
$$a_k = \sigma\left(\sum_j w_{j,k} a_j\right)$$

$$a_7 = \sigma(1 \times 0.867 + (-3) \times 0.085) = \sigma(0.612) = 0.648$$

Algorithme d'apprentissage par rétropropagation

Exemple : $x = [2, -1], \quad y = 1, \quad \alpha = 0.1$

Propagation arrière

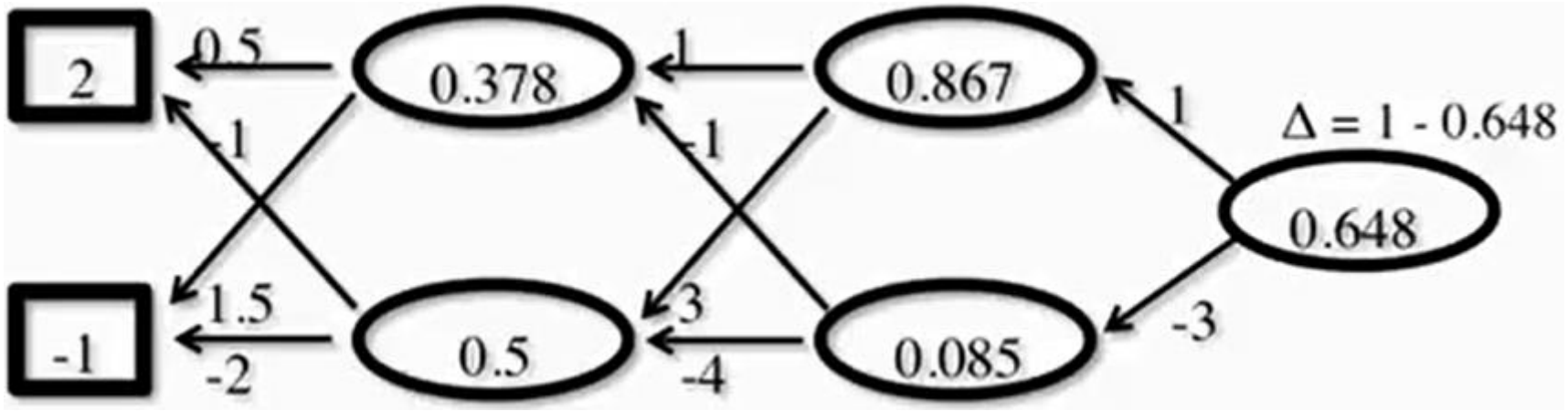


$$\Delta[j] = \sigma(in_j)(1 - \sigma(in_j)) \sum_k w_{j,k} \Delta[k]$$

Algorithme d'apprentissage par rétropropagation

Exemple : $x = [2, -1], \quad y = 1, \quad \alpha = 0.1$

Propagation arrière

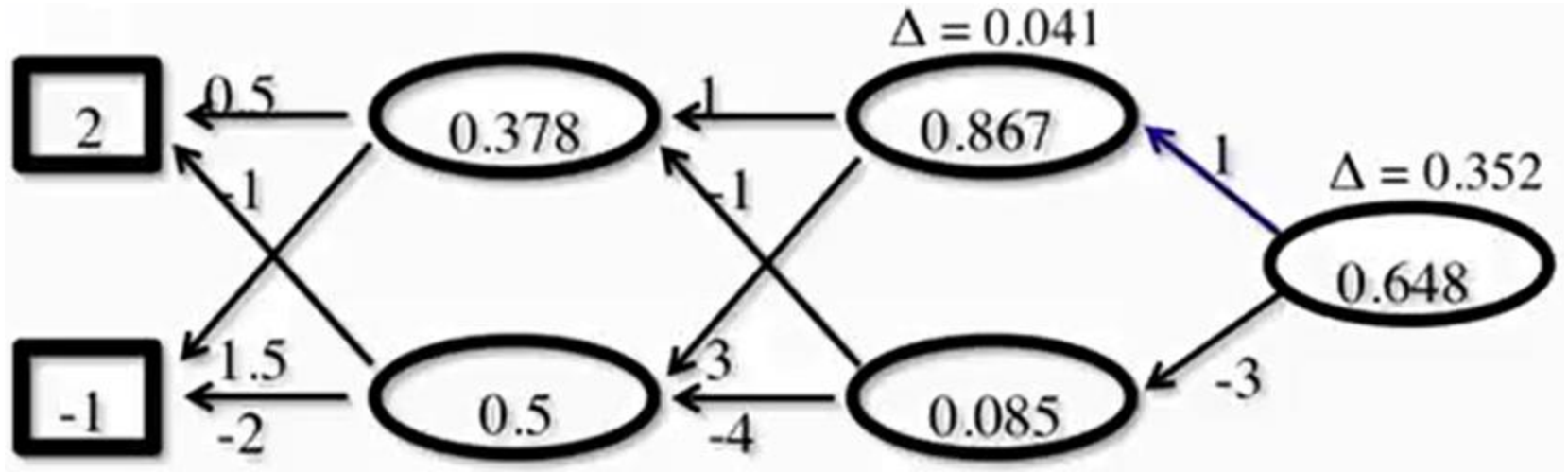


$$\Delta[j] = \sigma(in_j)(1 - \sigma(in_j)) \sum_k w_{j,k} \Delta[k]$$

Algorithme d'apprentissage par rétropropagation

Exemple : $x = [2, -1], \quad y = 1, \quad \alpha = 0.1$

Propagation arrière



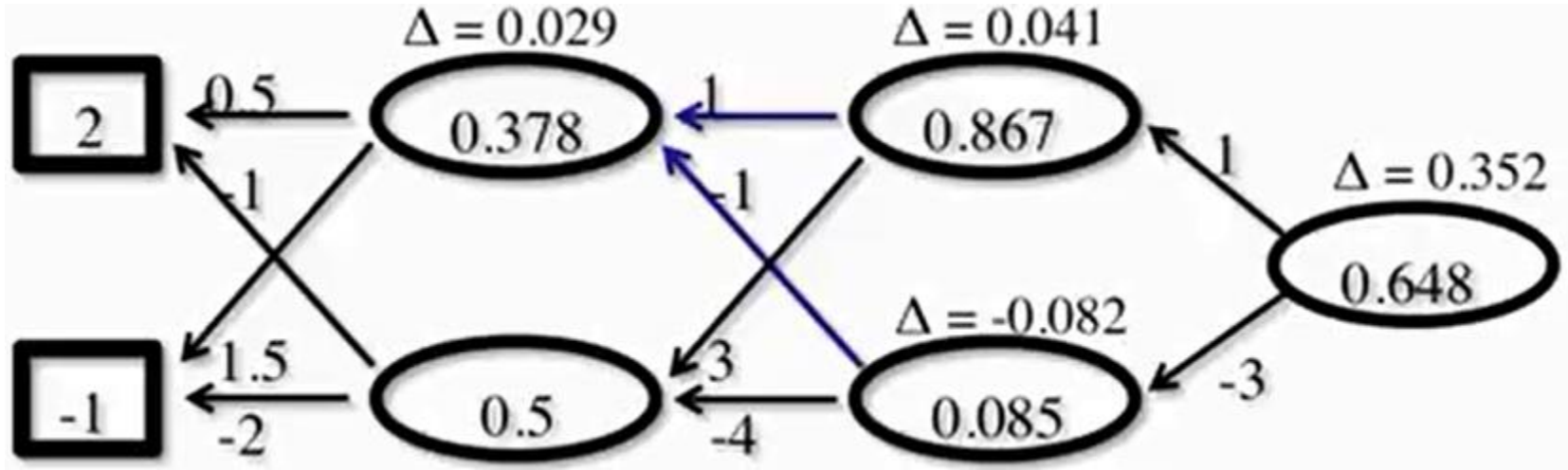
$$\Delta[j] = \sigma(in_j)(1 - \sigma(in_j)) \sum_k w_{j,k} \Delta[k]$$

$$\Delta[5] = 0.867 \times (1 - 0.867) \times 1 \times 0.352 = 0.041$$

Algorithme d'apprentissage par rétropropagation

Exemple : $x = [2, -1], \quad y = 1, \quad \alpha = 0.1$

Propagation arrière



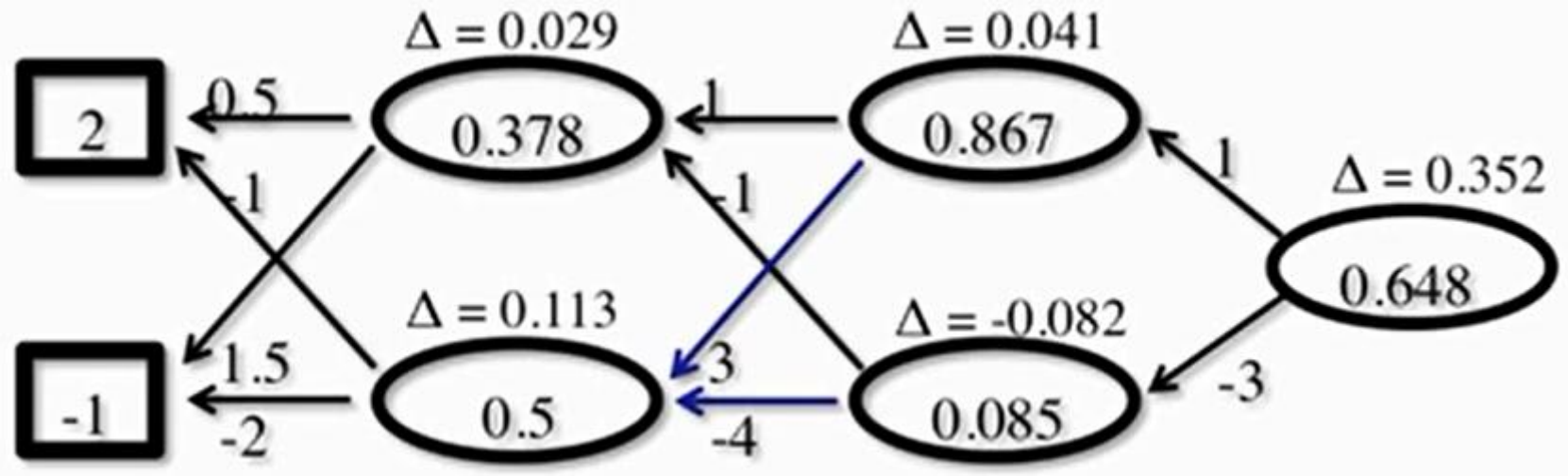
$$\Delta[j] = \sigma(in_j)(1 - \sigma(in_j)) \sum_k w_{j,k} \Delta[k]$$

$$\Delta[3] = 0.378 \times (1 - 0.378) \times (1 \times 0.041 + (-1) \times (-0.082) = 0.029$$

Algorithme d'apprentissage par rétropropagation

Exemple : $x = [2, -1], \quad y = 1, \quad \alpha = 0.1$

Propagation arrière



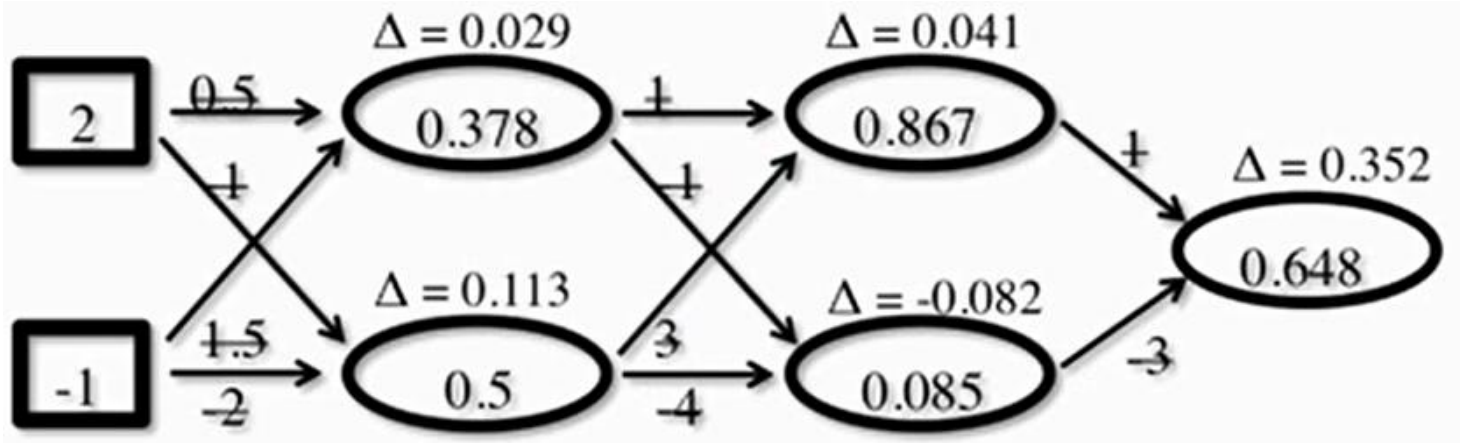
$$\Delta[j] = \sigma(in_j)(1 - \sigma(in_j)) \sum_k w_{j,k} \Delta[k]$$

$$\Delta[4] = 0.5 \times (1 - 0.5) \times (3 \times 0.041 + (-4) \times (-0.082) = 0.113$$

Algorithme d'apprentissage par rétropropagation

Exemple : $x = [2, -1], \quad y = 1, \quad \alpha = 0.1$

Mise à jour des poids



$$w_{i,j} = w_{i,j} + \alpha\, a_i \Delta[j]$$

$w_{1,3} \leftarrow 0.5 + 0.1 * 2 * 0.029 = 0.506$
$w_{1,4} \leftarrow -1 + 0.1 * 2 * 0.113 = -0.977$
$w_{2,3} \leftarrow 1.5 + 0.1 * -1 * 0.029 = 1.497$
$w_{2,4} \leftarrow -2 + 0.1 * -1 * 0.113 = -2.011$

$w_{3,5} \leftarrow 1 + 0.1 * 0.378 * 0.041 = 1.002$
$w_{3,6} \leftarrow -1 + 0.1 * 0.378 * -0.082 = -1.003$
$w_{4,5} \leftarrow 3 + 0.1 * 0.5 * 0.041 = 3.002$
$w_{4,6} \leftarrow -4 + 0.1 * 0.5 * -0.082 = -4.004$

$w_{5,7} \leftarrow 1 + 0.1 * 0.867 * 0.352 = 1.031$
$w_{6,7} \leftarrow -3 + 0.1 * 0.085 * 0.352 = -2.997$

A mostly complete chart of

Neural Networks

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

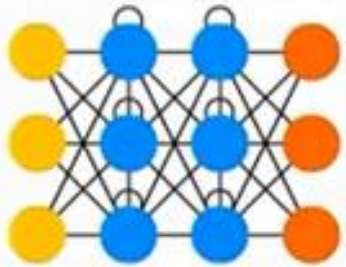
Perceptron (P)

Feed Forward (FF)

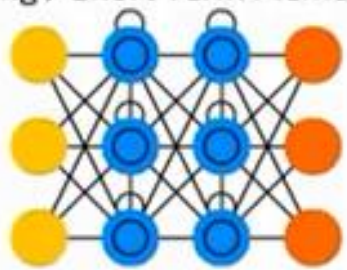
Radial Basis Network (RBF)

Deep Feed Forward (DFF)

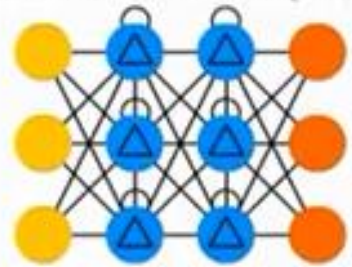
Recurrent Neural Network (RNN)



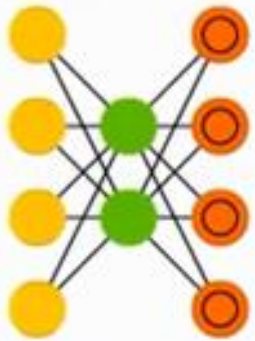
Long / Short Term Memory (LSTM)



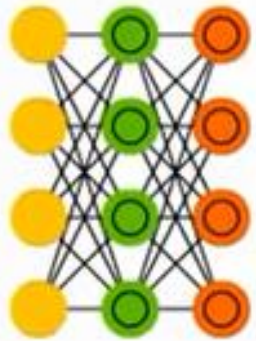
Gated Recurrent Unit (GRU)



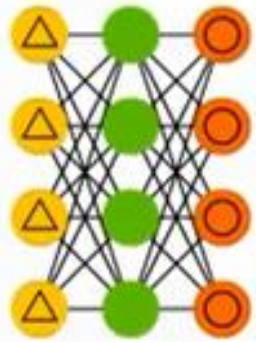
Auto Encoder (AE)



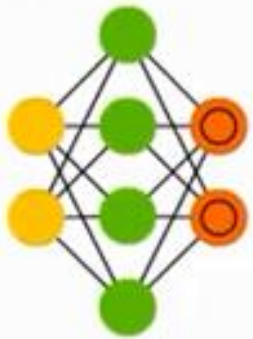
Variational AE (VAE)



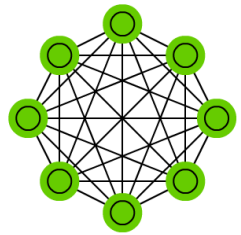
Denoising AE (DAE)



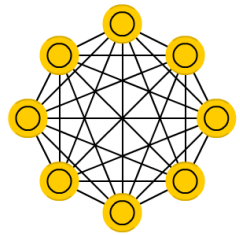
Sparse AE (SAE)



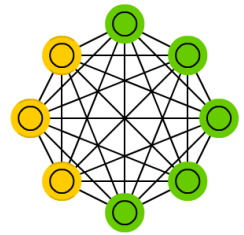
Markov Chain (MC)



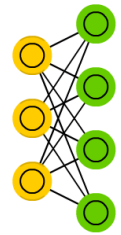
Hopfield Network (HN)



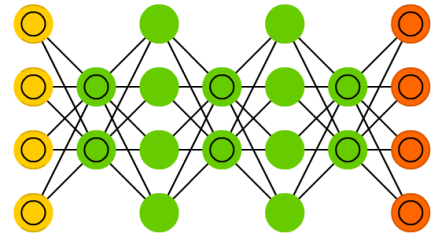
Boltzmann Machine (BM)



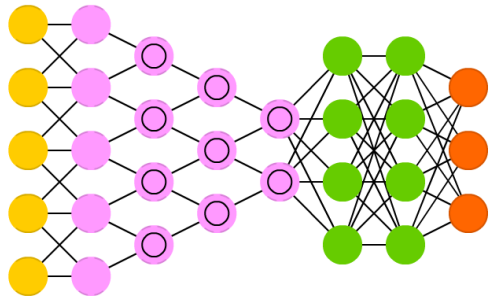
Restricted BM (RBM)



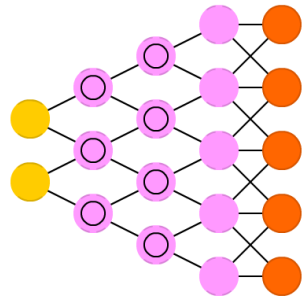
Deep Belief Network (DBN)



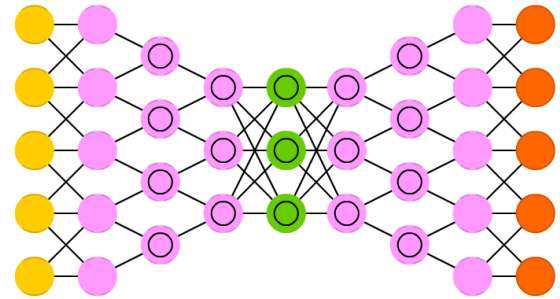
Deep Convolutional Network (DCN)



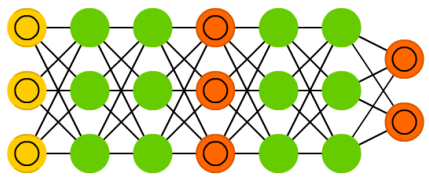
Deconvolutional Network (DN)



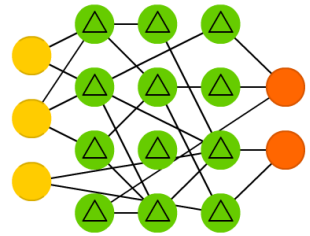
Deep Convolutional Inverse Graphics Network (DCIGN)



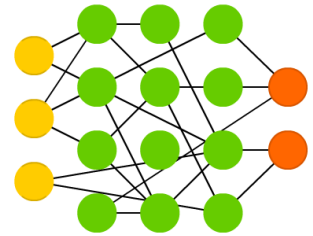
Generative Adversarial Network (GAN)



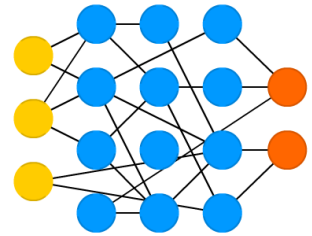
Liquid State Machine (LSM)



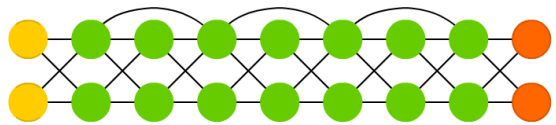
Extreme Learning Machine (ELM)



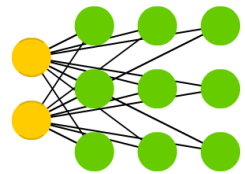
Echo State Network (ESN)



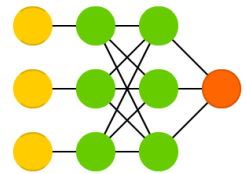
Deep Residual Network (DRN)



Kohonen Network (KN)



Support Vector Machine (SVM)



Neural Turing Machine (NTM)