Université de Batna 2 (Mostefa Ben Boulaid)
Faculté de Technologie
Département de Génie Industriel

Apprentissage Automatique
Cours 4 (Régression et RNA)

Pr Hassen BOUZGOU

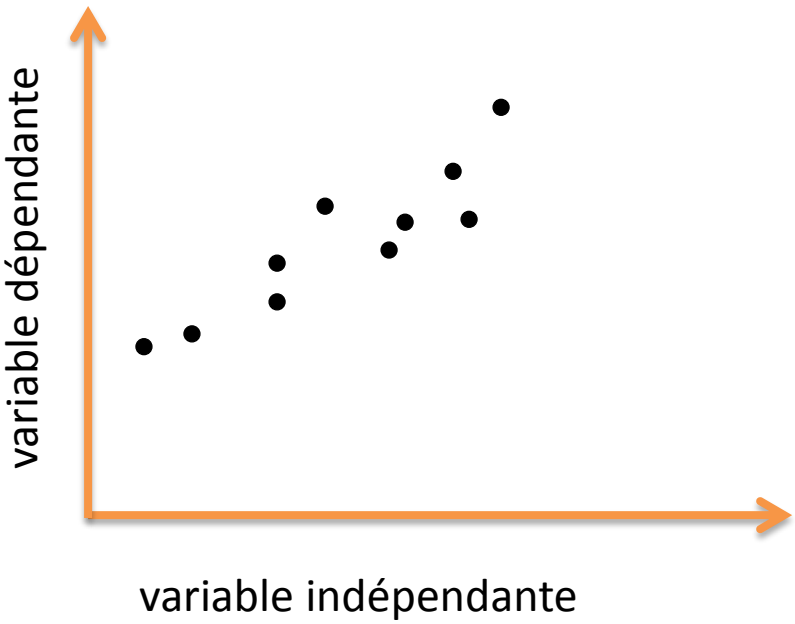
Différents problèmes de ML

	Apprentissage Supervisé	Apprentissage Non-supervisé
Discrète	Classification / Catégorisation	Clustering
Continue	Régression	Réduction de dimensionnalité

La régression

Définition

- La régression est la tentative d'expliquer la variation d'une variable dépendante en utilisant la variation des variables indépendantes.
- La régression est donc une explication de la causalité.
- Si la ou les variables indépendantes expliquent suffisamment la variation de la variable dépendante, le modèle peut être utilisé pour la prédiction.



Variable Indépendante

- Une variable indépendante est une variable qui est autonome et qui n'est pas modifiée par les autres variables que vous essayez de mesurer.

Exemple:

l'âge de quelqu'un pourrait être une variable indépendante. D'autres facteurs (tels que ce qu'ils mangent, combien ils vont à l'école, combien de télévision ils regardent) ne vont pas changer l'âge d'une personne. En fait, lorsque vous cherchez une sorte de relation entre les variables, vous essayez de voir si la variable indépendante provoque une sorte de changement dans les autres variables, ou dans les variables dépendantes.

Variable dépendante

- Une variable dépendante est quelque chose qui dépend d'autres facteurs.
  - Exemple: un score de test (performance) peut être une variable dépendante car il peut varier en fonction de plusieurs facteurs tels que la quantité de sommeil, la nutrition que vous avez eu la nuit précédant le test ou même la faim lorsque vous l'avez fait.
- Habituellement, lorsque vous cherchez une relation entre deux choses, vous essayez de découvrir ce qui fait que la variable dépendante change.

Regresssion Linéaire

But: établir un lien entre une variable dépendante Y et une variable indépendante X pour pouvoir ensuite faire des **prédictions** sur Y lorsque X est mesurée.

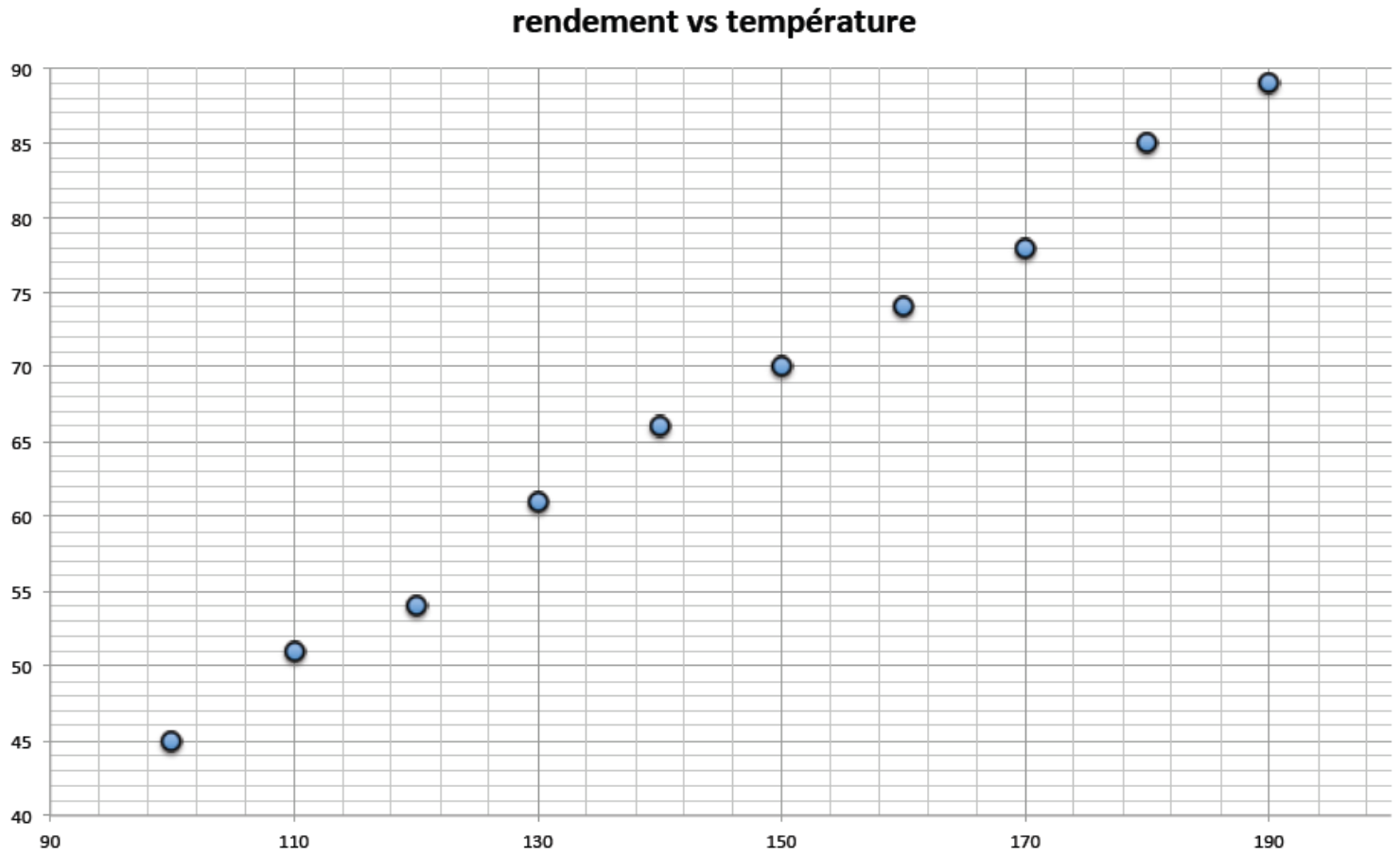
Exemple:

L'effet de la température de fonctionnement d'un procédé industriel sur le rendement du produit a donné les valeurs suivantes pour la température X_i et le rendement correspondant Y_i

Température °C	Rendement %	Température °C	Rendement %
100	45	150	70
110	51	160	74
120	54	170	78
130	61	180	85
140	66	190	89

Exemple

- Le graphe ci-dessous représente les points (X_i; Y_i) pour ces données et **suggère** une relation linéaire entre X et Y .



Regresssion Linéaire Simple

Définition

Un modèle de régression linéaire simple est de la forme:

$$Y=\beta_0+ \beta_1 X+\varepsilon$$

Où:

- Y est la variable dépendante
- β_0 et β_1 sont les coefficients (ordonnée à l'origine et pente)
- X est la variable indépendante
- ε est une erreur aléatoire

Résolution du système

Etant donnés *n* points de données

$(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$ de R^2,

- on essaie de trouver l'équation d'une droite qui passe par les *n* points.
- Cette équation est $Y = \beta_0 + \beta_1 X$ avec $\beta_0; \beta_1 \in R$
- β_0 et β_1 devraient être les solutions du système

Ax = b avec:

Le vecteur des 1 représente la constante

$$A = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$$

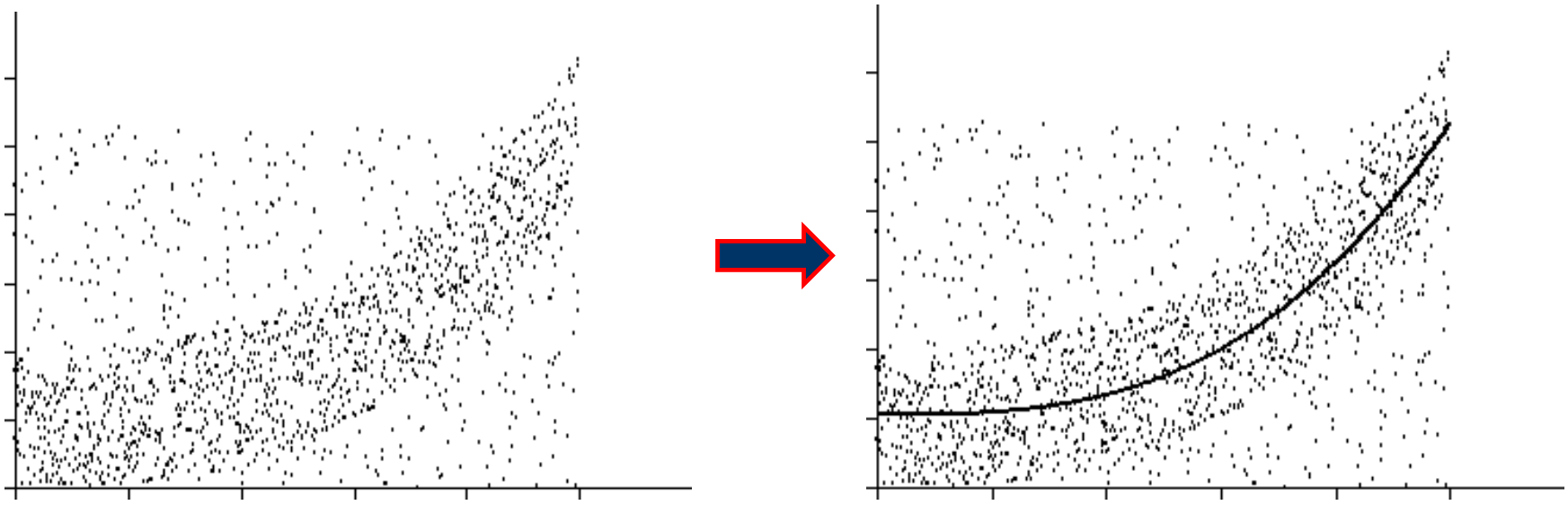
Résolution avec les **moindres carrés** : $(\hat{\beta}_0, \hat{\beta}_1) = \left(A^\top A\right)^{-1} A^\top \mathbf{b}$

Régression non-linéaire

Les fonctions non linéaires peuvent également être modélisées par l'analyse en régression. Différents modèles peuvent êtres utilisés:

polynomiale, Logarithmique, Exponentiel …..

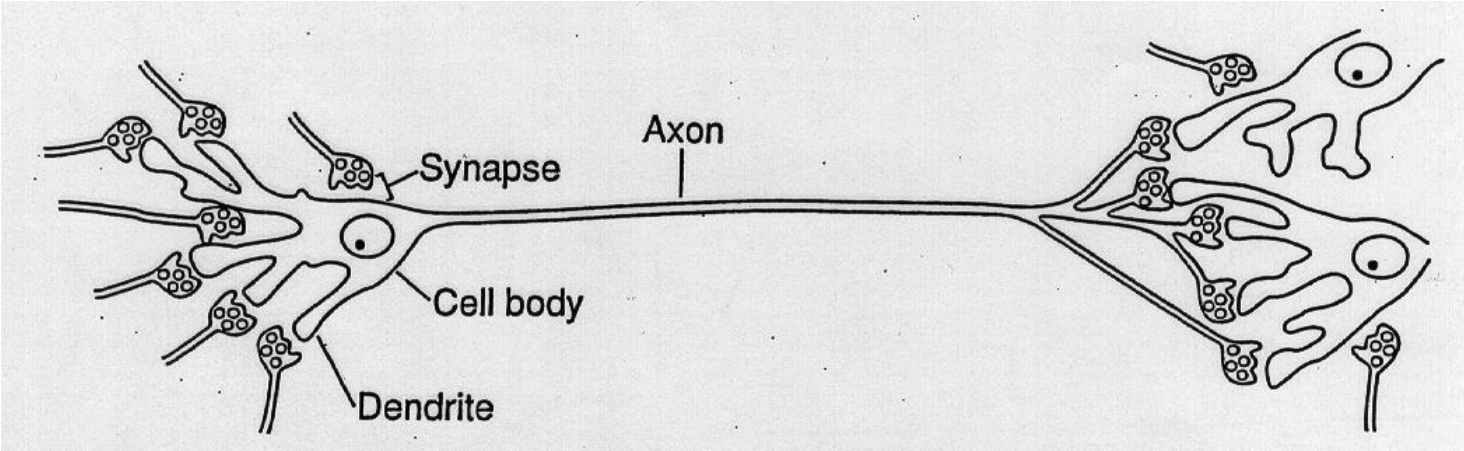
mais toute fonction continue peut être utilisée.



Les Réseaux de Neurones Artificiels

Neurones biologiques

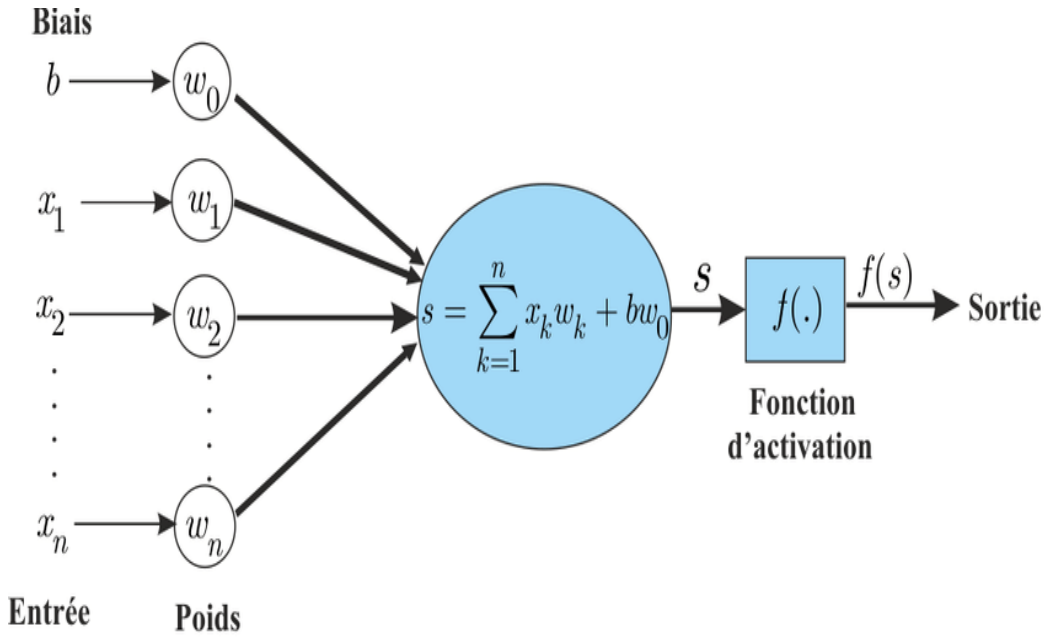
- Cerveau humain :
  - 10 milliards de neurones
  - 60 milliards de connexions (synapses)
  - Un synapse peut être inhibant ou excitant.



Cerveau	RNA
cellule (soma)	neurone
dendrites	entrées
synapses	poids
axon	sortie

Neurones artificiels

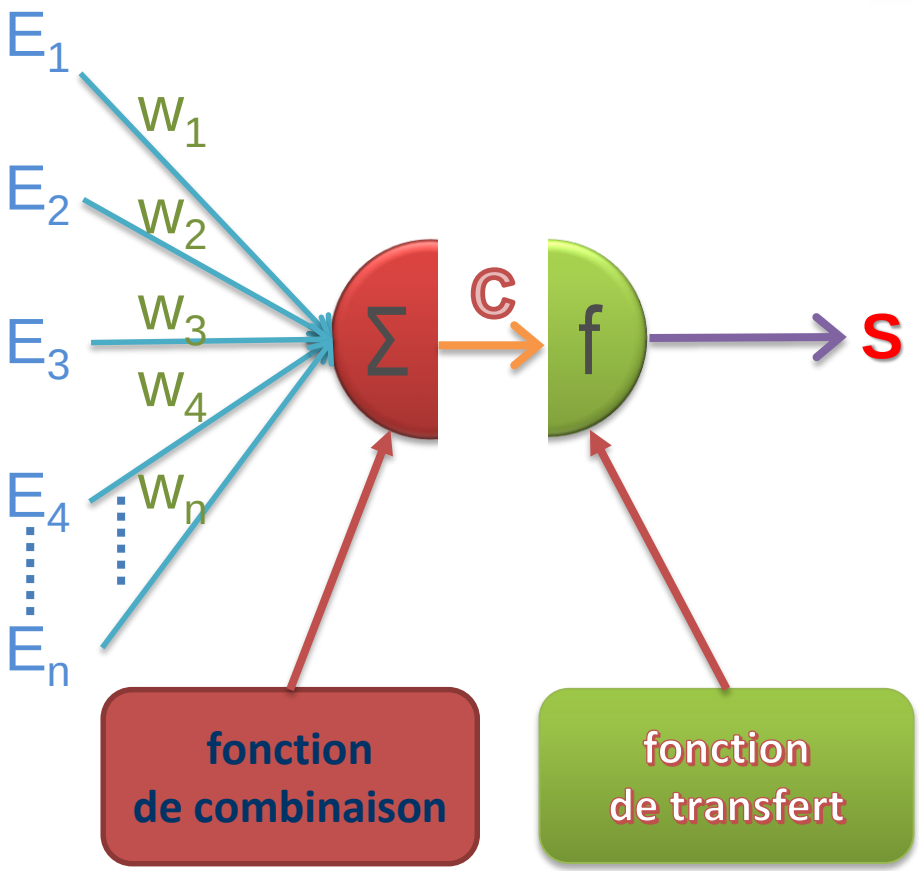
- RNA :
  - Un nombre fini de processeurs élémentaires (neurones).
  - Liens pondérés passant un signal d'un neurone vers d'autres.
  - Plusieurs signaux d'entrée par neurone
  - Un seul signal de sortie



Cerveau	RNA
cellule (soma)	neurone
dendrites	entrées
synapses	poids
axon	sortie

Neurone formel

- Les entrées "$\mathbf{E_i}$" du neurone proviennent soit d'autres éléments "processeurs", soit de l'environnement.
- Les poids "$\mathbf{W_i}$" déterminent l'influence de chaque entrée.
- La fonction de combinaison "**C**" combine les entrées et les poids (produit scalaire).
- La fonction de transfert "**f**" calcule la sortie "**S**" du neurone en fonction de la combinaison en entrée.



Neurone formel

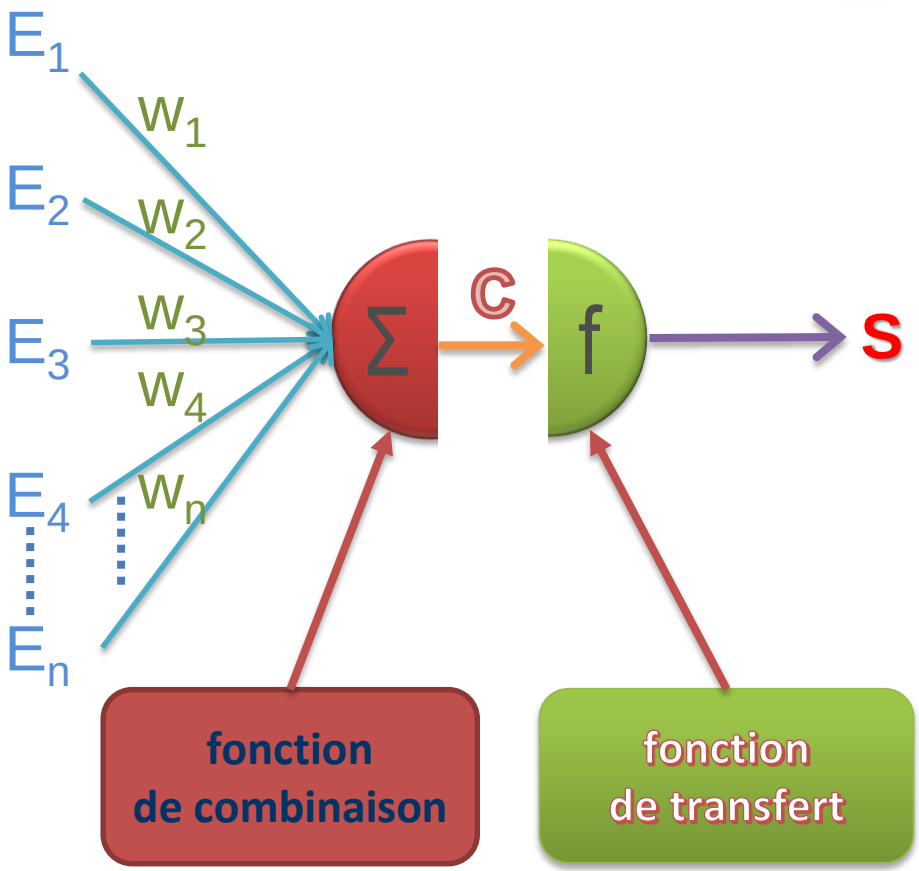
La fonction de transfert détermine l'état du neurone (en sortie)

Calcul de la sortie :

- $S = f(C)$

ou encore :

- $S = f(\sum W_i E_i)$
- La fonction de transfert "**f**" peut avoir plusieurs formes



Fonctions de transfert

Plusieurs possibilités existent, elles sont quasiment empiriques et à adapter en fonction des situations. Les plus courantes et les plus citées dans la littérature sont énumérées dans la figure suivante :

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

Types d'apprentissage

Le but des réseaux neuronaux est d'apprendre à répondre correctement à différentes entrées.

Moyen : modification des poids par apprentissage: *supervisé*, *ou* *non supervisé*.

- **Apprentissage supervisé**: un système "instructeur" corrige les réponses éronnées.
- **Apprentissage non supervisé:** le système neuronal apprend tout seul en formant des classes d'entrées à réponses communes.

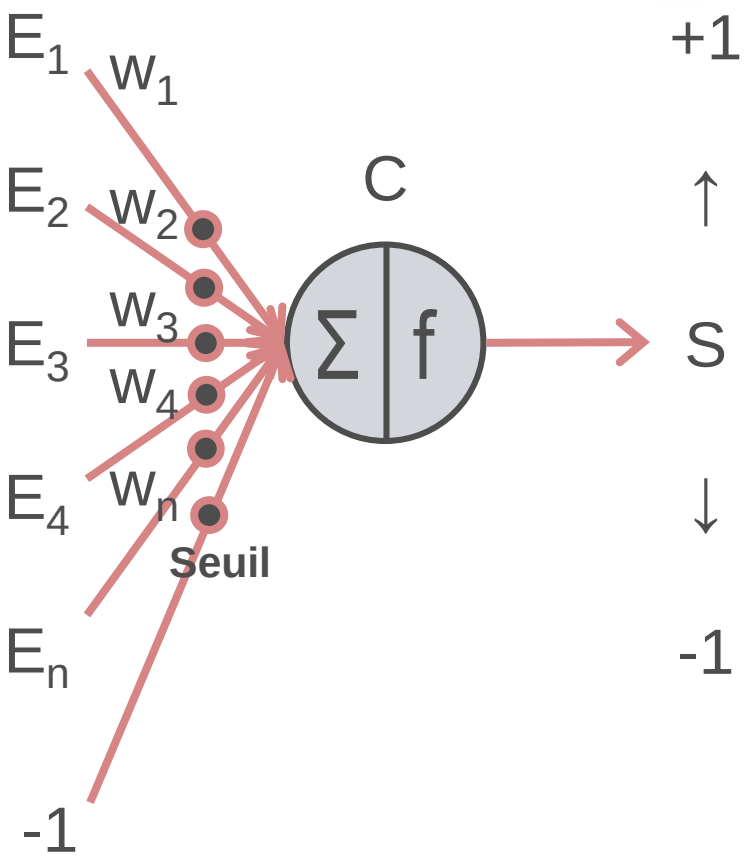
Apprentissage / Test

Phase 1: APPRENTISSAGE, le concept du Perceptron est basé sur un algorithme d'apprentissage dont l'objectif est de corriger les poids de pondération des entrées afin de fournir une sortie la plus proche possible de la sortie désirée (éléments à apprendre).

Phase 2: UTILISATION (Test), une fois les exemples appris, le réseau va essayer de tester ses performances sur des données non-vues dans la phase d'apprentissage pour tester sa généralisation.

Algorithme d'apprentissage

- Le perceptron est un **classificateur linéaire**
- Il réalise une partition de son espace d'entrée (E1,…,En) en deux, ou plusieurs classes S_1, …, S_m. séparables linéairement
- On considère deux classes :
  - S_1 (S = +1)
  - S_2 (S = -1)



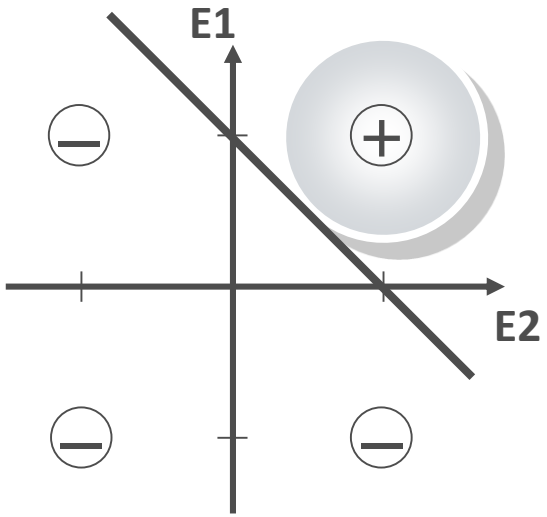
Afin de simplifier les calculs, le SEUIL est considéré une entrée du neurone à valeur constante et fixée à -1.

Algorithme d'apprentissage (1)

1: **INITIALISATION:** W1 et W2:[-1;+1], SEUIL et PAS:[0;+1]
Base d'apprentissage: (E1;E2)-> "Sortie correcte"
ET logique:(+1;+1)→+1;(-1;+1) → -1;(-1;-1) → -1;(+1;-1) → -1
2: **REPETER**
3: **POUR** chaque exemple de la base:**(E1;E2) → "S_correcte"**
4: Calcul de la sortie "S_calculée" pour chaque exemple:
4a: Fonction de combinaison:(« p »)
"p" = (W1 x E1) + (W2 x E2) - SEUIL
4b: Fonction d'activation:
SI("p")>= 0 ALORS "S_calculée" = +1
SINON "S_calculée" = -1
5: **SI** la sortie **"S_calculée" est différente** de la sortie **"S_correcte"** (**ERREUR = "S_correcte" - "S_calculée"**)
ALORS Modification des poids:
5a: **W1(t+1)= W1(t) + (E1 x PAS x ERREUR)**
5b: **W2(t+1)= W2(t) + (E2 x PAS x ERREUR)**
6: **Revenir à 4 pour recalculer la sortie**
7: **TANTQUE** une **ERREUR** subsiste revenir à 4

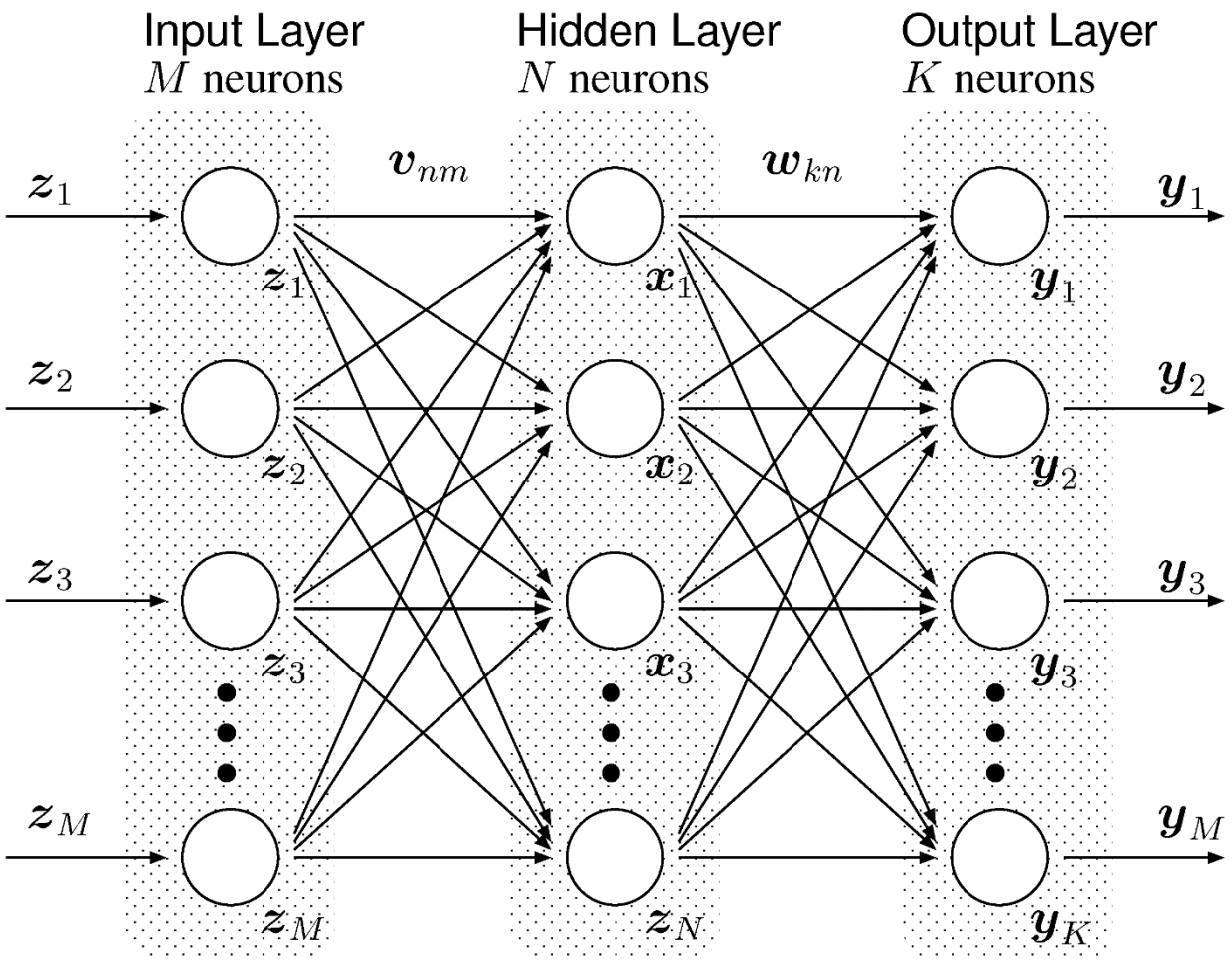
Exemple ET Logique (and)

- Dans le cas de la fonction ET la séparation des deux classes se fait par une ligne droite:
  - $W_1xE_1 + W_2xE_2$ - SEUIL = 0
  - Deux classes (deux réponses):
  - S_1 (S = +1):
    - Pour les entrées: (+1;+1)
  - S_2 (S = -1):
    - Pour les entrées:
    - {(-1;-1); (+1;-1); (-1;+1)}



« ET » logique

Perceptron Multicouches



⇒ Besoin d'un algorithme de rétro-propagation d'erreur!

Rétro-propagation du gradient

- La rétro-propagation du gradient est l'algorithme le plus utilisé. Il consiste à suivre la "ligne de plus grande pente" de la "surface d'erreur", qui doit logiquement conduire à un minimum (local ou global).
- Le principe de l'algorithme est de remonter couche par couche, des neurones de sortie vers les neurones d'entrées et de modifier les poids en amont de chaque couche, de façon à diminuer l'erreur global de sortie.
- Le mécanisme est généralement itératif (problème souvent non-linéaire) : l'erreur globale diminue à chaque itération.

Rétro-propagation du gradient (1)

- La convergence peut varier selon les problèmes posés, l'architecture du réseau, les fonctions d'activation, la base d'apprentissage et les variables d'entrée du réseau.

- L'avantage du réseau MLP est qu'il idéalement adapter à ce genre d'algorithme vue son architecture et ses effets de propagation fortement structurés et maîtrisés.

- Il existe des variantes de la rétro-propagation du gradient, s'inspirant par exemple des travaux réalisés en recherche opérationnelle: Levenberg Marquardt, Quasi Newton, Cojuguate Gradient, Quickprop Rprop et Standard Backprop (rétropropagation du gradient)

Critères d'arrêt

- La variation des paramètres entre les itérations est inferieure à un seuil prédéfini (convergence des paramètres). *Ne pas tomber dans le problème de disparition du gradient.*
- La valeur de la fonction d'erreur est inferieure à un seuil prédéfini.
- Nombre d'itérations atteint.