

**Département d'Informatique Fondamentale et ses
Applications DIFA**
Dr .Esma BENDIAB
Maître de conférences



LES RÉSEAUX DE NEURONES

Objectifs



- **Aborder** une approche radicalement différente des autres modes de programmation.
- **Comprendre** les principaux concepts de simulation informatique d'un neurone.
- **Maîtriser** deux algorithmes majeurs d' "apprentissage".
- **Evaluer** les performances et les limites d'un réseau neuronal.
- **Structurer et tester** des "mini-réseaux" (ex: OCR).

Plan



Voici les parties que nous allons aborder:

- **Notions de base**
- **Le perceptron**
- **Réseaux multicouches à rétro-propagation de l'erreur**

Plan de la partie

4

Voici les points que nous allons aborder:

- Historique et développements
- Modélisation du neurone
- Types d'apprentissage





Historique et développements

5

Approches informatiques pour résoudre un problème :

- Approche algorithmique (programmation complète)
- Création des « moteurs d'inférence » (programme qui raisonne ; règles SI..ALORS.. ; système expert)
- Approche connexionniste : le réseau s'organise par apprentissage (pas de programmation)

Historique et développements

6

Caractéristiques de l'approche connexionniste (réseaux neuronaux) :

- Calcul non-algorithmique
- Information et mémoire distribuée dans le réseau
- Architecture massivement parallèle (processeurs élémentaires interconnectés)
- Apprentissage par entraînement sur des exemples
- Inspiré du fonctionnement du cerveau



Historique et développements

7

Niveaux de modélisation :

CIRCUITS NEURONAUX



MECANISMES MENTAUX



**COMPORTEMENT
INTELLIGENT**

Historique et développements

8

- **1943 J.Mc Culloch et W.Pitts établissent le "modèle logique" du neurone qui ouvre la voie à des modèles techniques.**
- **1949 D.Hebb élabore une théorie formelle de l'apprentissage biologique par modifications des connexions neuronales.**
- **1957 F.Rosenblatt réalise le Perceptron, le premier modèle technique basé sur la modification des poids.**
- **1960 B.Widrow réalise Adaline (Adaptive Linear Element), un réseau adaptatif de type perceptron.**
- **1969 M.Minsky et S.Papert émettent des critiques et démontrent les limites des modèles neuronaux de type perceptron.**
- **La recherche s'arrête durant un peu plus d'une dizaine d'années.**
- **1982 J.Hopfield (physicien) propose une nouvelle approche des réseaux neuronaux basée sur l'analogie avec les milieux à grand nombre de particules. Cela relance l'intérêt pour les réseaux neuronaux**
- **depuis 1984 : développement croissant du domaine connexionniste aussi bien en IA qu'en informatique.**

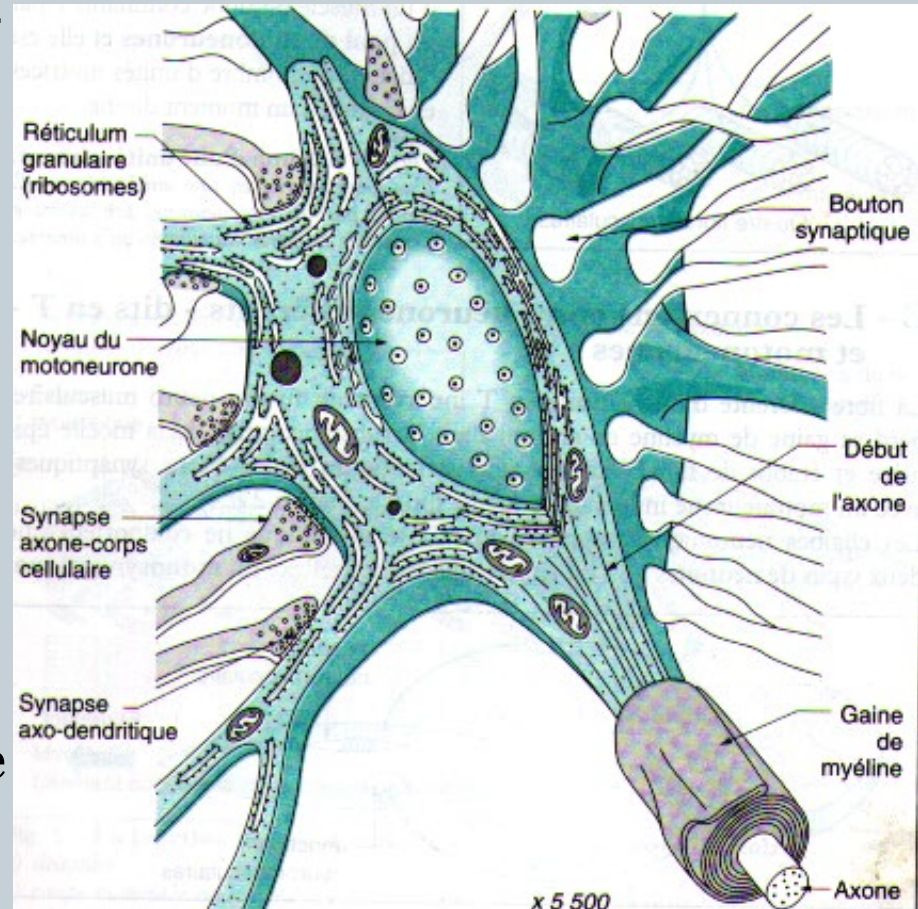
- **Traitement des images**
- **Identification des signatures**
- **Reconnaissance des caractères (dactylos ou manuscrits)**
- **Reconnaissance de la parole**
- **Reconnaissance de signaux acoustiques (bruits sous-marins, ...)**
- **Extraction d'un signal du bruit**
- **Contrôle de systèmes asservis non-linéaires (non modélisables)**
- **Robotique (apprentissage de tâches)**
- **Aide à la décision (domaine médical, bancaire, management, ...)**

Modélisation du neurone






Le neurone réel

10

- Dans un cerveau, il y a 10^{12} neurones avec 10^3 à 10^4 connexions par neurone.
- Dendrite : récepteur des messages
- Corps : génère le potentiel d'action (la réponse)
- Axone : transmet le signal aux cellules suivantes
- Synapse : jonction axone - dendrite (plus ou moins passante)
- Neurone : élément autonome dépourvu d'intelligence



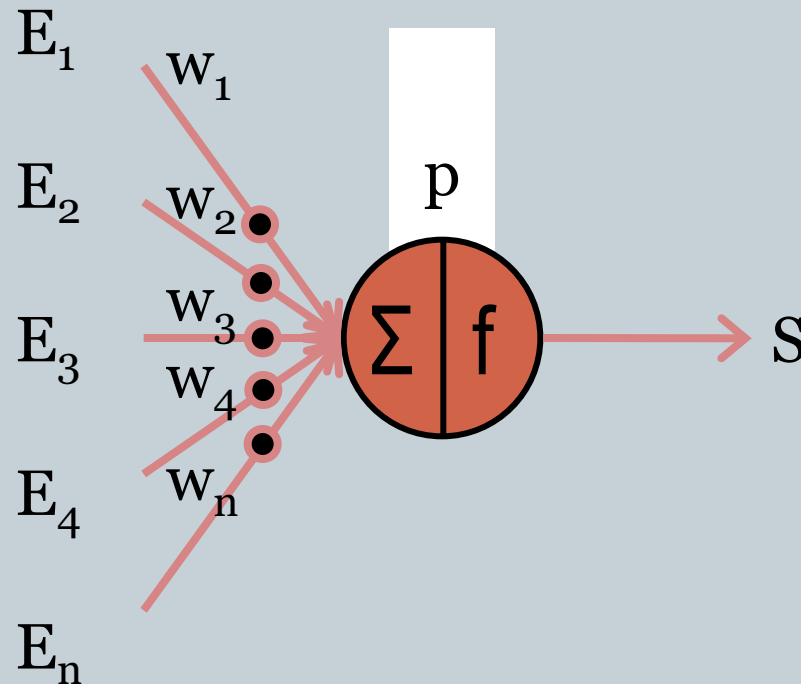
La modélisation du système nerveux biologique repose sur la correspondance suivante

Système nerveux		Système de calcul
Neurone		Processeur
Dendrite		Fonction de combinaison
Corps du neurone		Fonction de transfert
Axone		Élément de sortie
Synapse		Poids

Modélisation du neurone

12

La représentation graphique (conventionnelle) d'un neurone formel modélisé par Mc Culloch et Pitts.

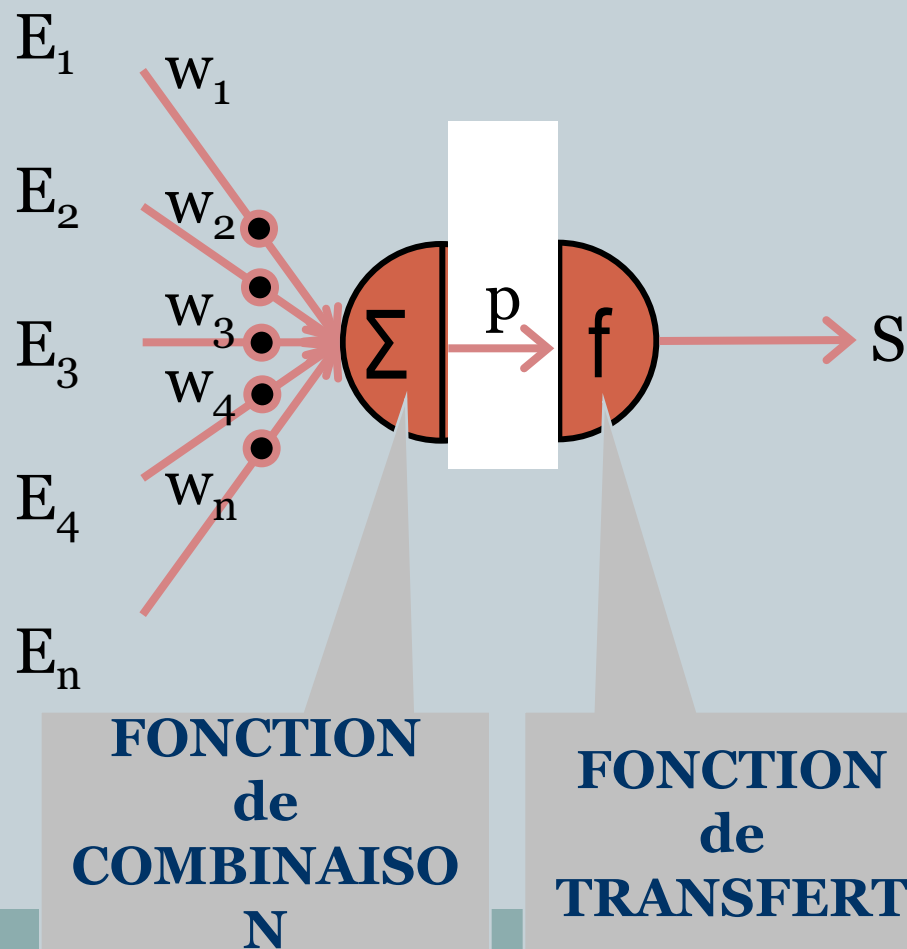


Modélisation du neurone

13

Les éléments constitutifs du neurone artificiel

- Les entrées "**E**" du neurone proviennent soit d'autres éléments "processeurs", soit de l'environnement.
- Les poids "**W**" déterminent l'influence de chaque entrée.
- La fonction de combinaison "**p**" combine les entrées et les poids.
- La fonction de transfert calcule la sortie "**S**" du neurone en fonction de la combinaison en entrée.



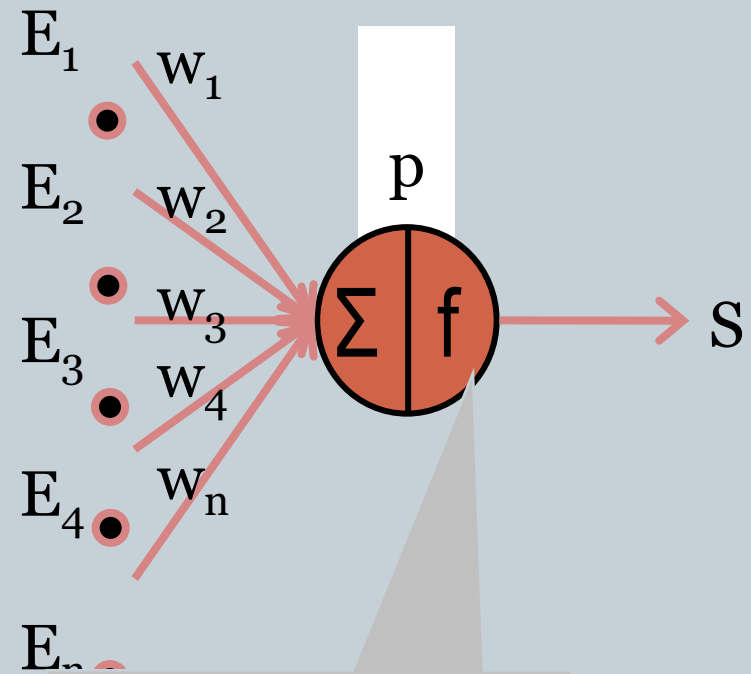
Modélisation du neurone

14

La Fonction de Combinaison calcule l'influence de chaque entrée en tenant compte de son poids. Elle fait la somme des entrées pondérées :

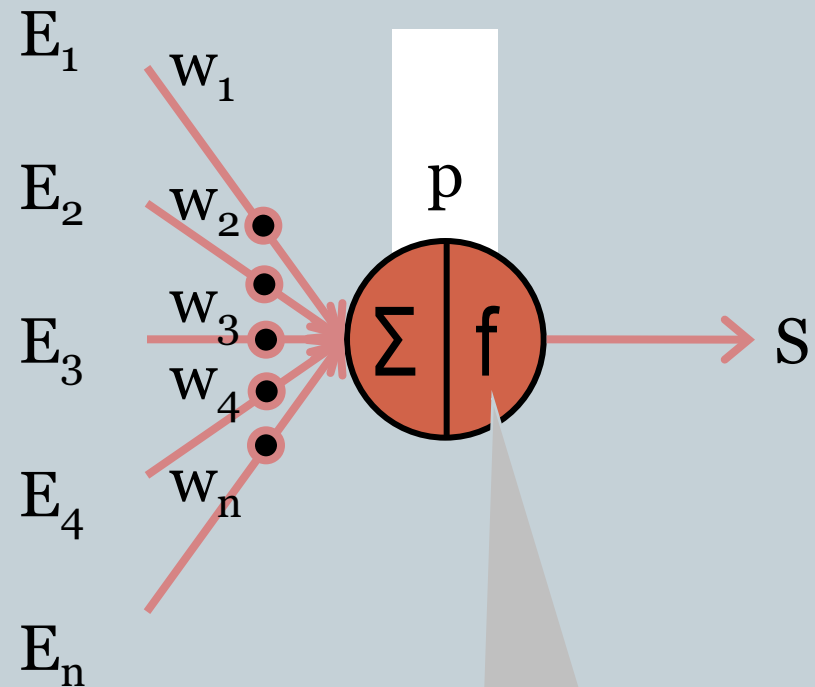
$$p = \sum W_i E_i$$

- W_i :
- Poids de la connexion à l'entrée i .
- E_i :
- Signal de l'entrée i .



La Fonction de Transfert détermine l'état du neurone (en sortie)

- Calcul de la sortie :
- $S = f(p)$
- ou encore :
- $S = f(\sum W_i E_i)$
- La fonction de transfert "f" peut avoir plusieurs formes.



Modélisation du neurone

16

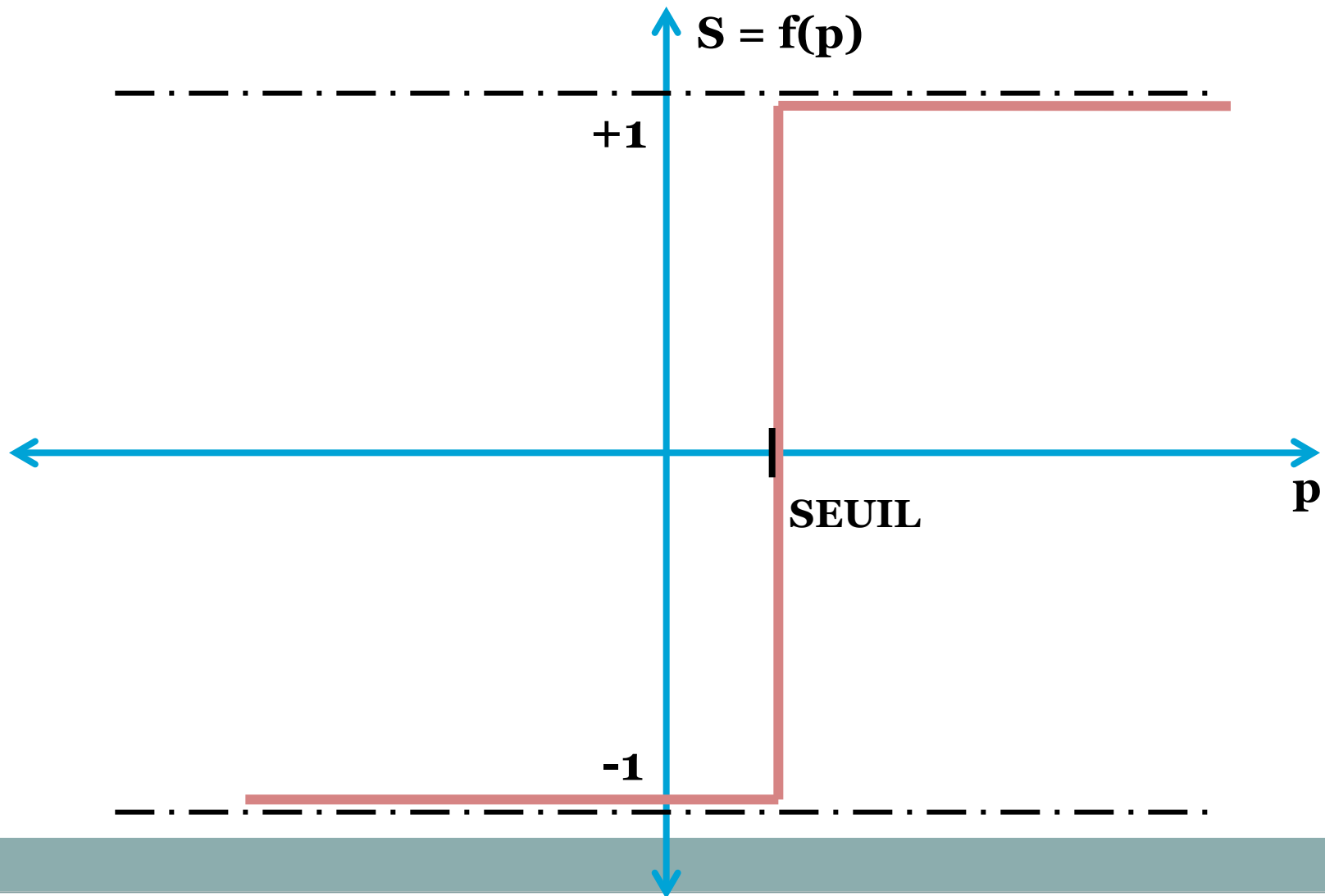
La fonction 'f' peut être de la forme :

- Fonction en échelon.
- Fonction linéaire par morceaux.
- Fonction dérivable (sigmoïde).

Modélisation du neurone

Fonction de transfert en échelon :

17

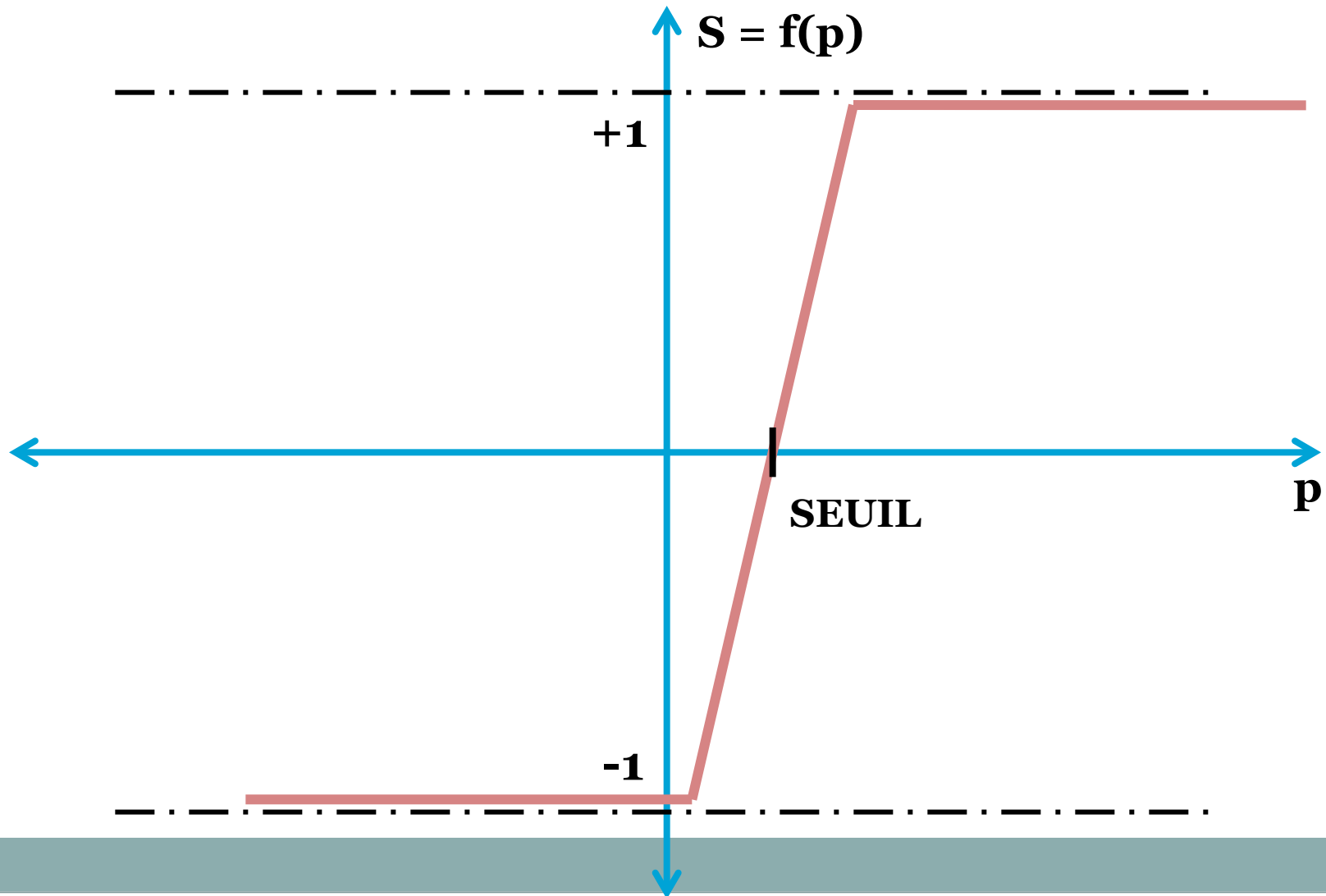




Modélisation du neurone

Fonction de transfert linéaire par morceaux :

18

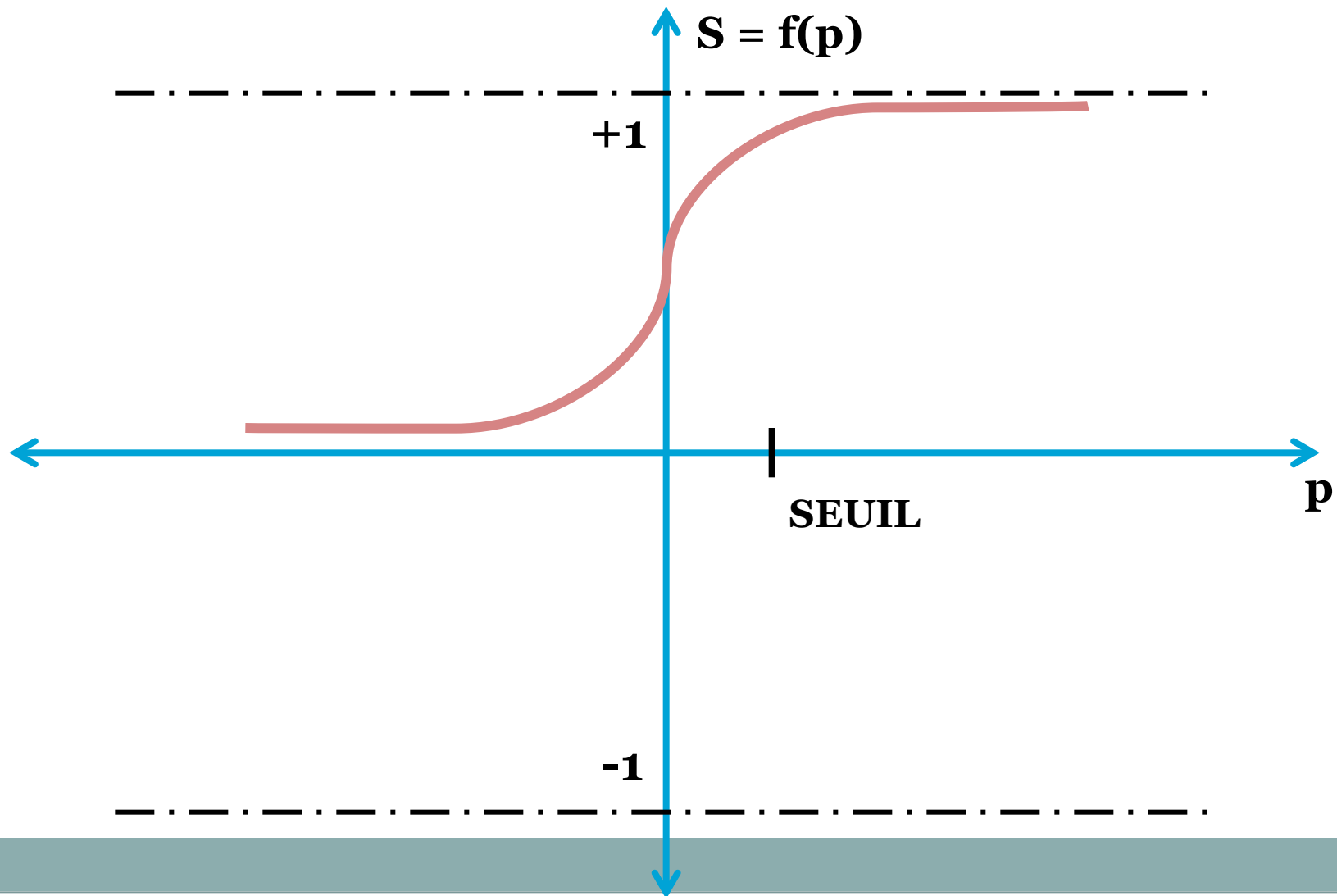




Modélisation du neurone

Fonction de transfert dérivable (sigmoïde) :

19



Modélisation du neurone

20

- Les différents types de neurones se distinguent par la nature g de leur fonction d'activation.
- Les principaux types sont :
 - • linéaire g est la fonction identité,
 - • seuil $g(x) = 1_{[0, +\infty[}(x)$,
 - • sigmoïde $g(x) = 1/(1 + e^{-x})$,
 - • ReLU $g(x) = \max(0, x)$ (rectified linear unit),
 - • softmax $g(x)_j = e^{x_j} / \sum_{k=1}^K e^{x_k}$ pour tout $k \in \{1 \dots K\}$,
 - • radiale $g(x) = \exp(-x^2/2) / \sqrt{2\pi}$,
 - • stochastique $g(x) = 1$ avec la probabilité $1/(1 + e^{-x/H})$, 0 sinon (H intervient comme une température dans un algorithme de recuit simulé), • ...

Modélisation du neurone

21

- Les modèles linéaires, sigmoïdaux, ReLU, softmax sont bien adaptés aux algorithmes d'apprentissage impliquant une rétro-propagation du gradient car leur fonction d'activation est différentiable ; ce sont les plus utilisés.
- Le modèle à seuil est sans doute plus conforme à la réalité biologique mais pose des problèmes d'apprentissage.
- Enfin le modèle stochastique est utilisé pour des problèmes d'optimisation globale de fonctions perturbées ou encore pour les analogies avec les systèmes de particules.

Définition

22

- Déterminer un réseau de neurones = Trouver les coefficients synaptiques.
- On parle de phase d'apprentissage : les caractéristiques du réseau sont modifiées jusqu'à ce que le comportement désiré soit obtenu.
- Base d'apprentissage : exemples représentatifs du comportement ou de la fonction à modéliser. Ces exemples sont sous la forme de couples (entrée ; sortie) connus.
- Base d'essai (test): pour une entrée quelconque (bruitée ou incomplète), calculer la sortie. On peut alors évaluer la performance du réseau.



Types d'apprentissage

23

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.
- les coefficients synaptiques sont évalués en minimisant l'erreur (entre sortie souhaitée et sortie obtenue) sur une base d'apprentissage.

Types d'apprentissage

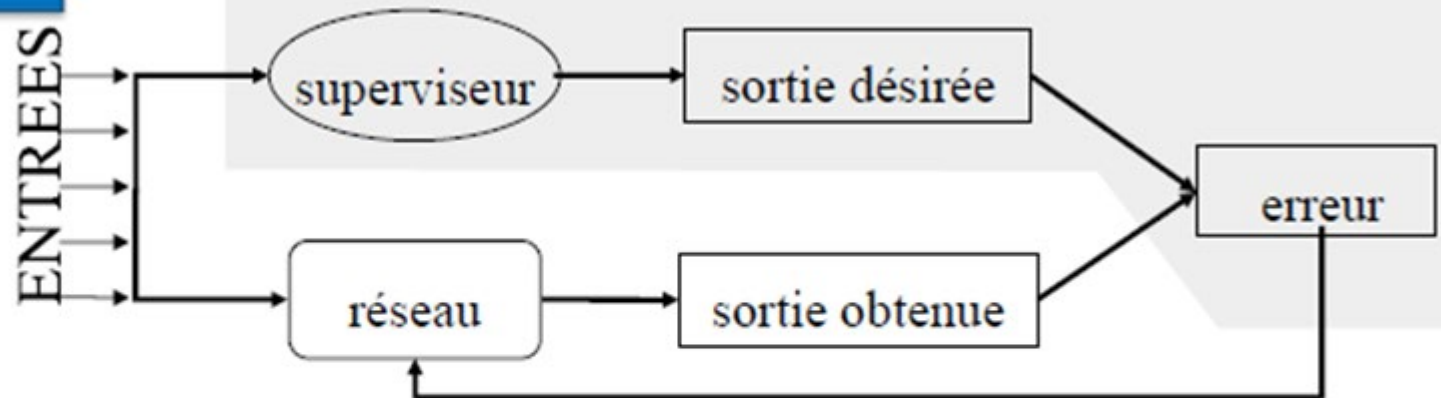
24

- **Apprentissage non supervisé:** le système neuronal apprend tout seul en formant des classes d'entrées à réponses communes.
- On ne dispose pas de base d'apprentissage. Les coefficients synaptiques sont déterminés par rapport à des critères de conformité : spécifications générales.
- **•Sur-apprentissage** : on minimise l'erreur sur la base d'apprentissage à chaque itération mais on augmente l'erreur sur la base d'essai. Le modèle perd sa capacité de généralisation : c'est l'apprentissage par cœur. ⇒ Explication : trop de variables explicatives dans le modèle ; on n'explique plus le comportement global mais les résidus.

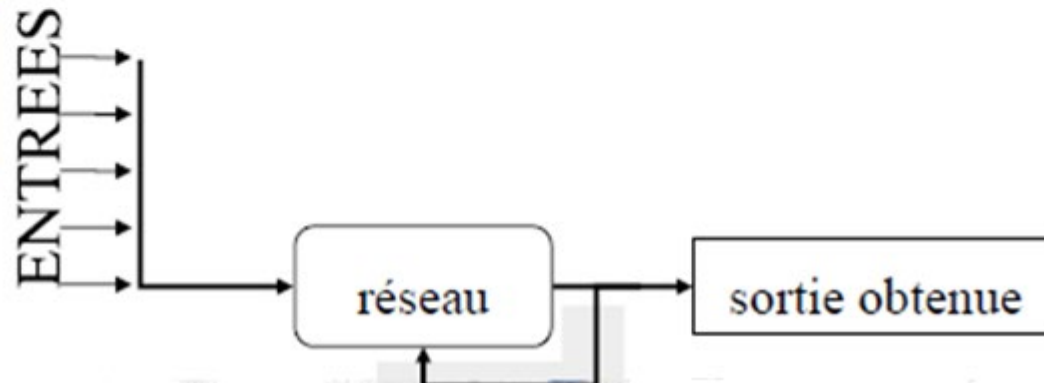
Types d'apprentissage

25

Apprentissage supervisé



Apprentissage non supervisé



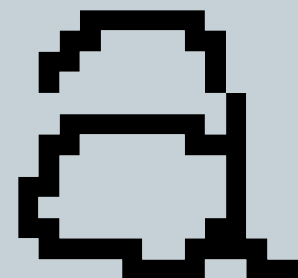


Types d'apprentissage

Apprentissage supervisé (ex: OCR)

26

- Association imposée entre un vecteur d'entrée (forme multidimensionnelle) et un vecteur de sortie (la réponse désirée).
- L'erreur est calculée à chaque essai afin de corriger les poids.
- Les poids sont modifiés jusqu'à l'erreur minimale, voire aucune erreur.



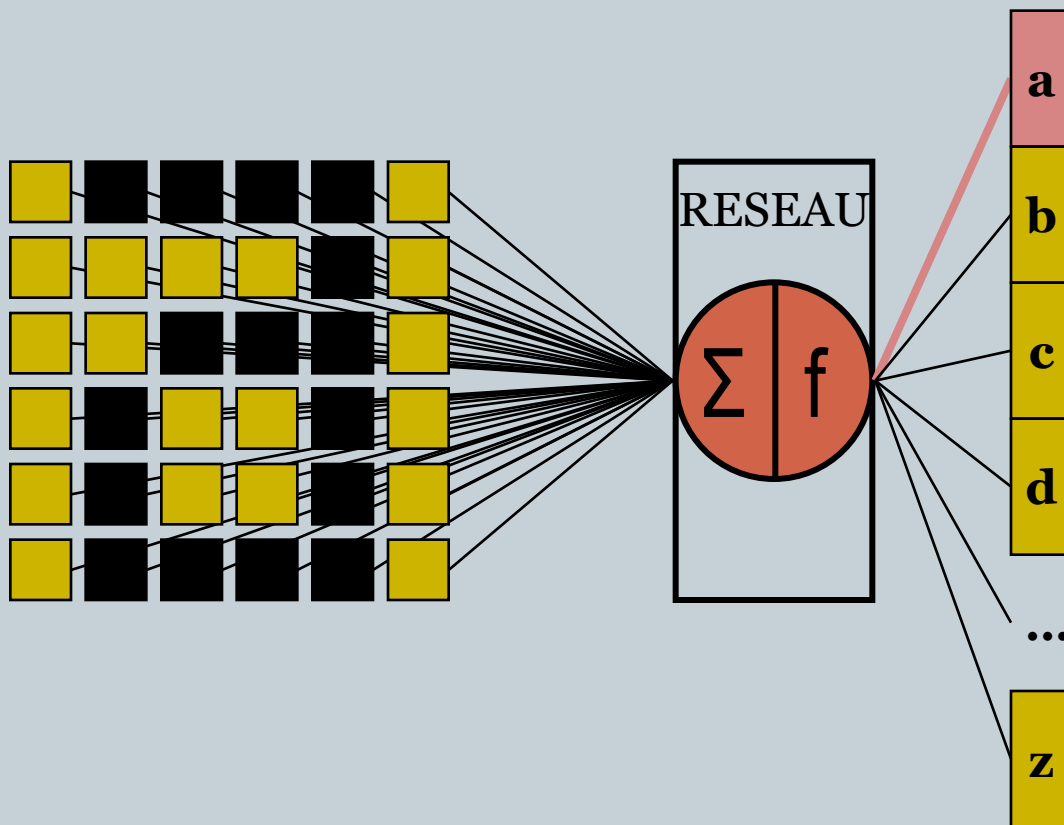
a



Types d'apprentissage

Apprentissage supervisé (ex: OCR)

- La réponse attendue est le "a". Un seul et unique vecteur de sortie doit être activé.





Types d'apprentissage

28

Apprentissage non supervisé

- Pas d'indication sur les erreurs.
- Le réseau détecte les caractéristiques communes des différentes entrées.
- Il tente ainsi de former des « classes » de façon autonome.
- Il apprend à donner des réponses aux classes



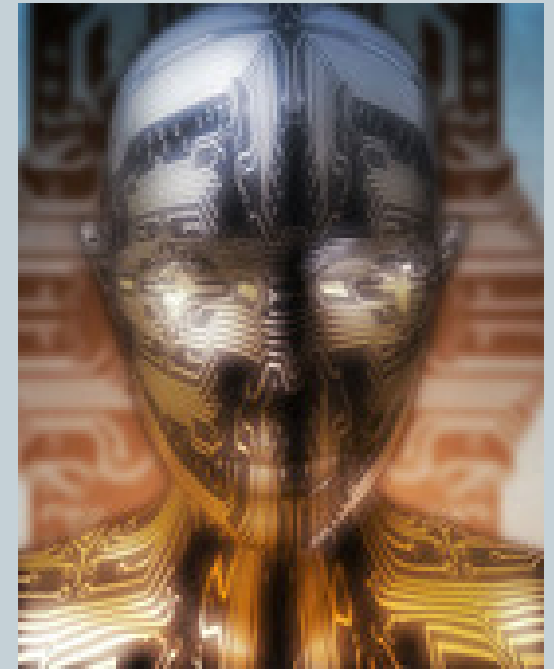
Le perceptron

Plan de la partie

30

Voici les éléments que nous allons aborder:

- Les phases apprentissage-utilisation
- L'algorithme d'apprentissage
- Apprentissage des fonctions logiques
- Les limites du perceptron





Les phases apprentissage-utilisation

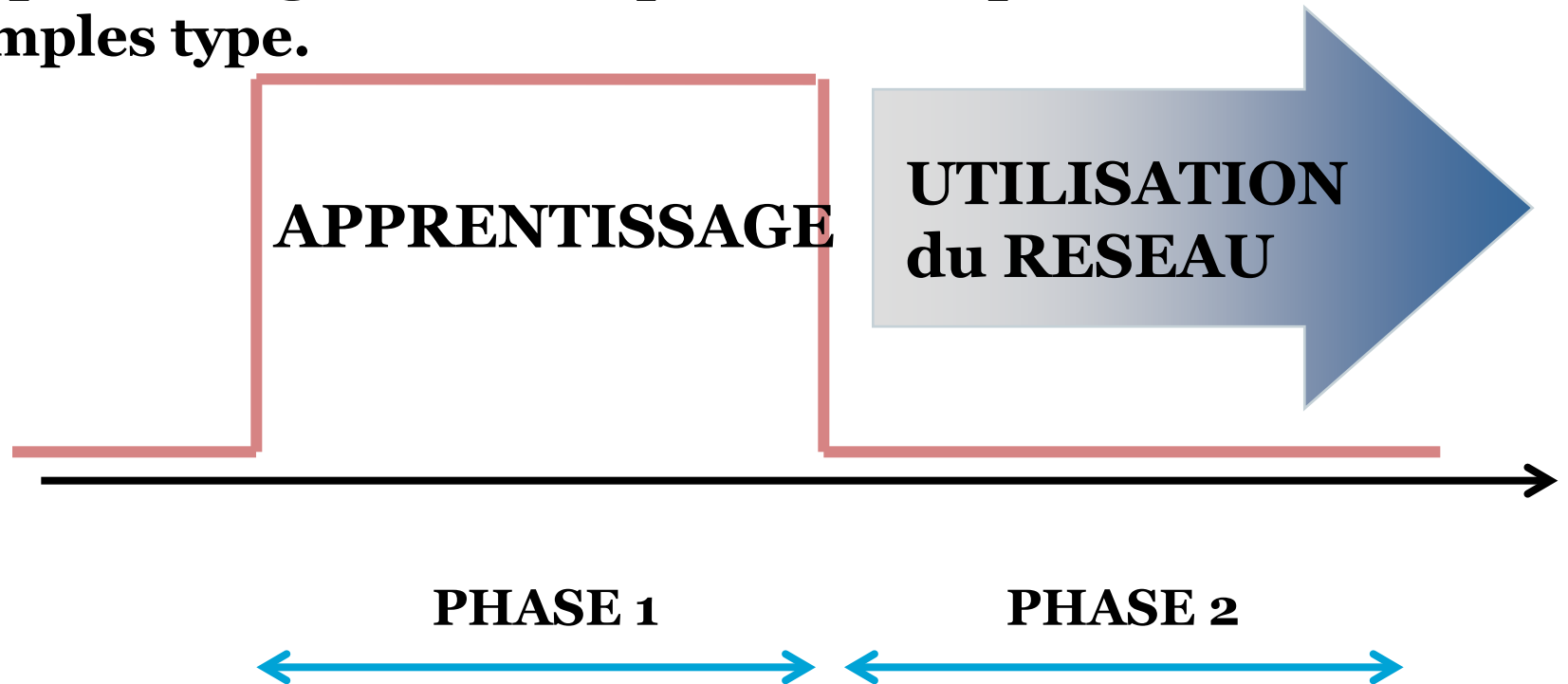
31

- 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 activité (en sortie) en adéquation avec les éléments à apprendre.
- Phase 2: **UTILISATION**, une fois les exemples appris, chaque neurone active ou non sa sortie (en correspondance avec le domaine acquis), en fonction des éléments appliqués en entrée.

Les phases apprentissage-utilisation

32

Toute utilisation du réseau doit être précédée d'une phase d'apprentissage durant laquelle on lui présente des exemples type.

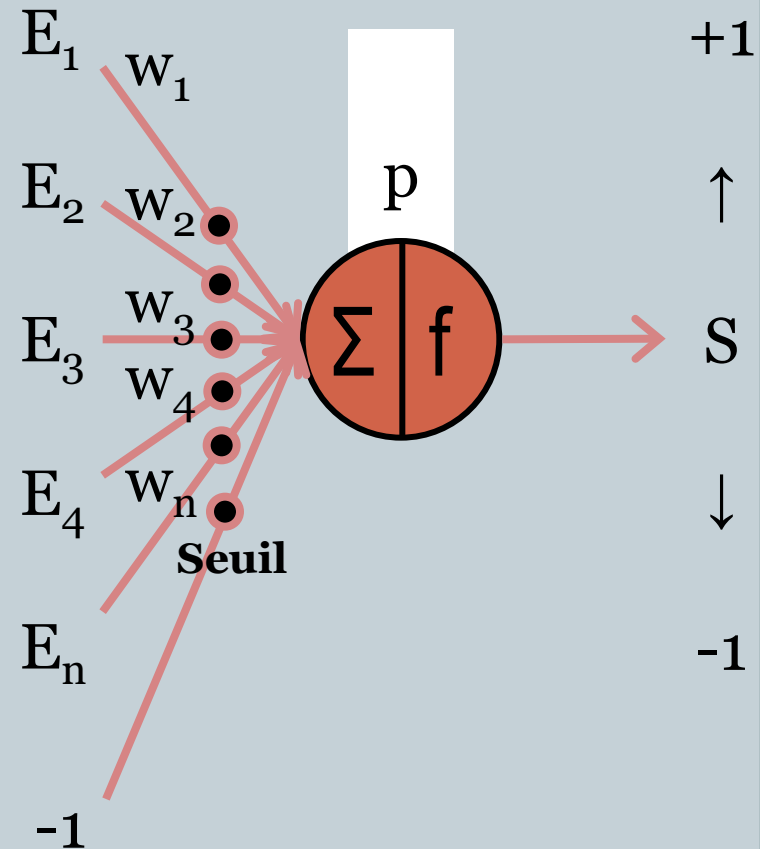


L'algorithme d'apprentissage

33

Classes et séparabilité linéaire

- Le perceptron est un **classificateur linéaire**
- Il réalise une partition de son espace d'entrée (E_1, \dots, E_n) en deux, ou plusieurs classes C_1, \dots, C_m , séparables linéairement
- On considère deux classes :
 - C_1 ($S = +1$)
 - C_2 ($S = -1$)





L'algorithme d'apprentissage

Algorithme du perceptron (algorithme de principe)

34

```
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:(sa sortie "p": potentiel)
       "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
```

Apprentissage des fonctions logiques

Simulation de la fonction ET logique par apprentissage:

35

- Dans le cas de la fonction ET la séparation des deux classes se fait par une ligne droite:

- $W_1 \times E_1 + W_2 \times E_2 - \text{SEUIL} = 0$

Deux classes (deux réponses):

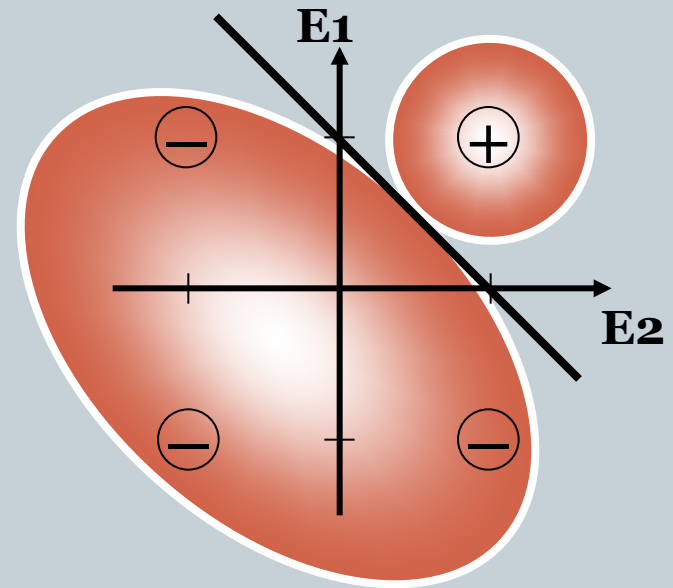
- C1 ($S = +1$):

- ✦ Pour les entrées: $(+1; +1)$

- C2 ($S = -1$):

- ✦ Pour les entrées:

$\{(-1; -1); (+1; -1); (-1; +1)\}$



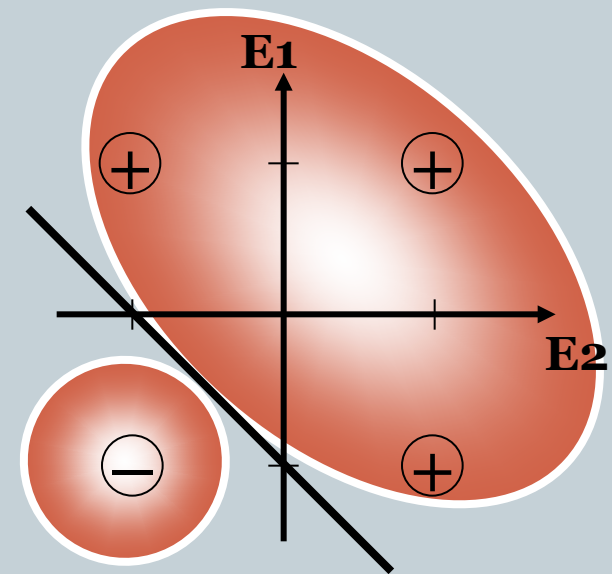
« ET » logique

Apprentissage des fonctions logiques

Simulation de la fonction OU logique par apprentissage:

36

- Dans le cas de la fonction OU, une droite permet toujours la séparation des deux classes.
- Pour deux classes :
 - C1 ($S = +1$):
 - ✦ $\{(+1;+1); (+1;-1); (-1;+1)\}$
 - C2 ($S = -1$):
 - ✦ $(-1;-1)$



« OU » logique

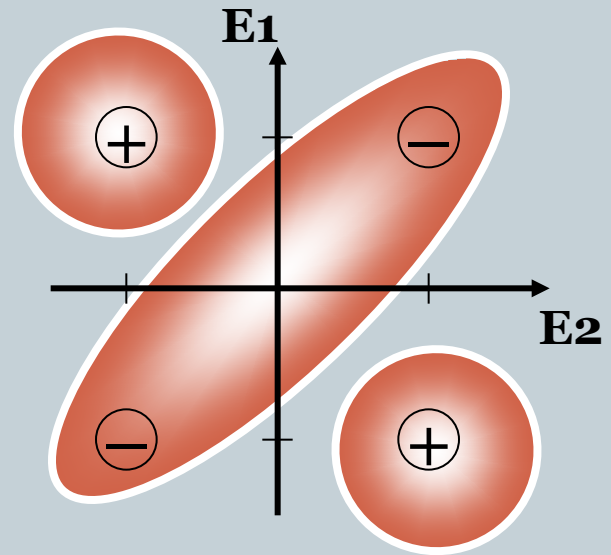
Les limites du perceptron

Les limites du perceptron : la fonction logique OU exclusif.

37

- Dans le cas de la fonction **OU-EXCLUSIF**, la séparation des deux classes ne peut se faire par une droite mais par une courbe.

- C1 ($S = +1$):
 - ✦ $\{(-1; +1); (+1; -1)\}$
- C2 ($S = -1$):
 - ✦ $\{(+1; +1); (-1; -1)\}$



« OU-Exc » logique

Les limites du perceptron

38

- Le perceptron est un classificateur linéaire.
- Il ne peut traiter les problèmes non linéaires du type OU EXCLUSIF, COMPAREUR (et bien d'autres...).
- La structuration d'un réseau neuronal (constitué de plusieurs couches), et l'utilisation conjointe d'un algorithme d'apprentissage approprié vont permettre de pallier les limites identifiées ici.

Résumons

39

- Un réseau de neurones monocouche, aussi appelé perceptron, est caractérisé de la manière suivante.
- Il possède n informations en entrée ;
- Il est composé de p neurones, que l'on représente généralement alignés verticalement. Chacun peut en théorie avoir une fonction d'activation différente. En pratique, ce n'est généralement pas le cas ;
- Chacun des p neurones est connecté aux n informations d'entrée.
- Le réseau de neurones possède ainsi n informations en entrée et p sorties, chaque neurone renvoyant sa sortie.

Résumons

40

- $\mathbf{X} = (x_i)_{1 \leq i \leq n}$ les n informations d'entrée ;
- $w_{i,j}$ pour $1 \leq i \leq n$ et $1 \leq j \leq p$, le poids reliant l'information x_i et le neurone j puis a_j l'**activation** du j -ème neurone ;
- $w_{o,j}$ le **coefficient de biais**, également appelé **seuil**, du j -ème neurone ;
- in_j la donnée d'entrée (somme pondérée) du j -ème neurone.
- Chaque neurone de la couche donnera donc une sortie. Une utilisation courante est que chaque neurone de la couche représente une classe. Pour un exemple \mathbf{X} donné, on obtient la classe de cet exemple en prenant la plus grande des p sorties.
- Il existe 2 types de perceptrons : les perceptrons *feed-forward* et les perceptrons *récurrents*.
- Les perceptrons *récurrents* sont ceux qui alimentent leurs entrées avec leurs sorties, alors que les perceptrons *feed-forward* non.



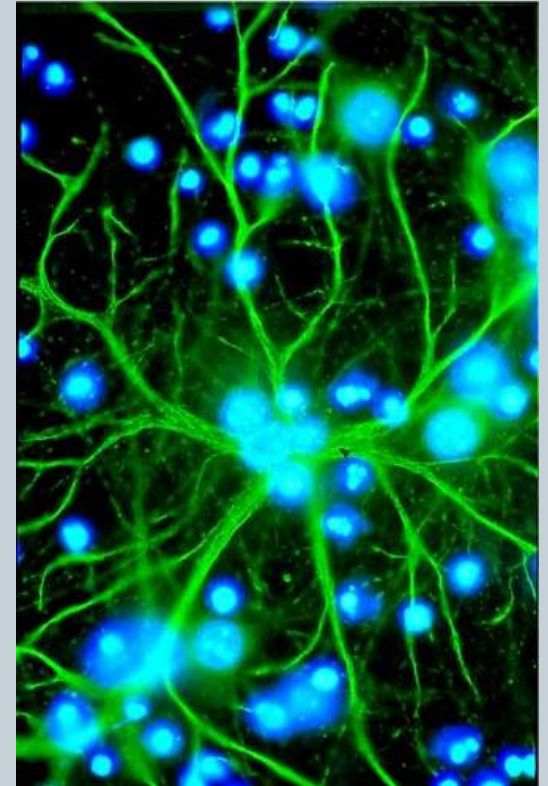
Réseaux multicouches à rétro-propagation de l'erreur

Plan de la partie

42

Voici les parties que nous allons aborder:

- Réseaux de neurones formels
- Apprentissage d'un réseau multicouche
- L'algorithme d'apprentissage

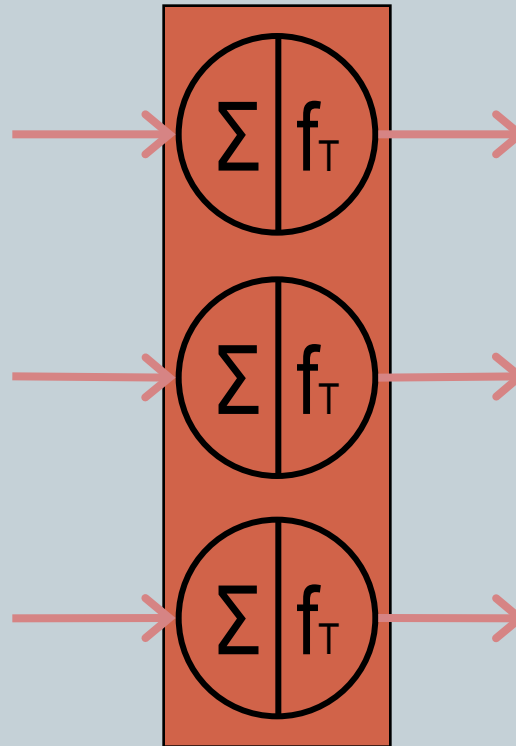


Réseaux de neurones formels

43

Réseau à une couche de neurones

COUCHE UNIQUE*

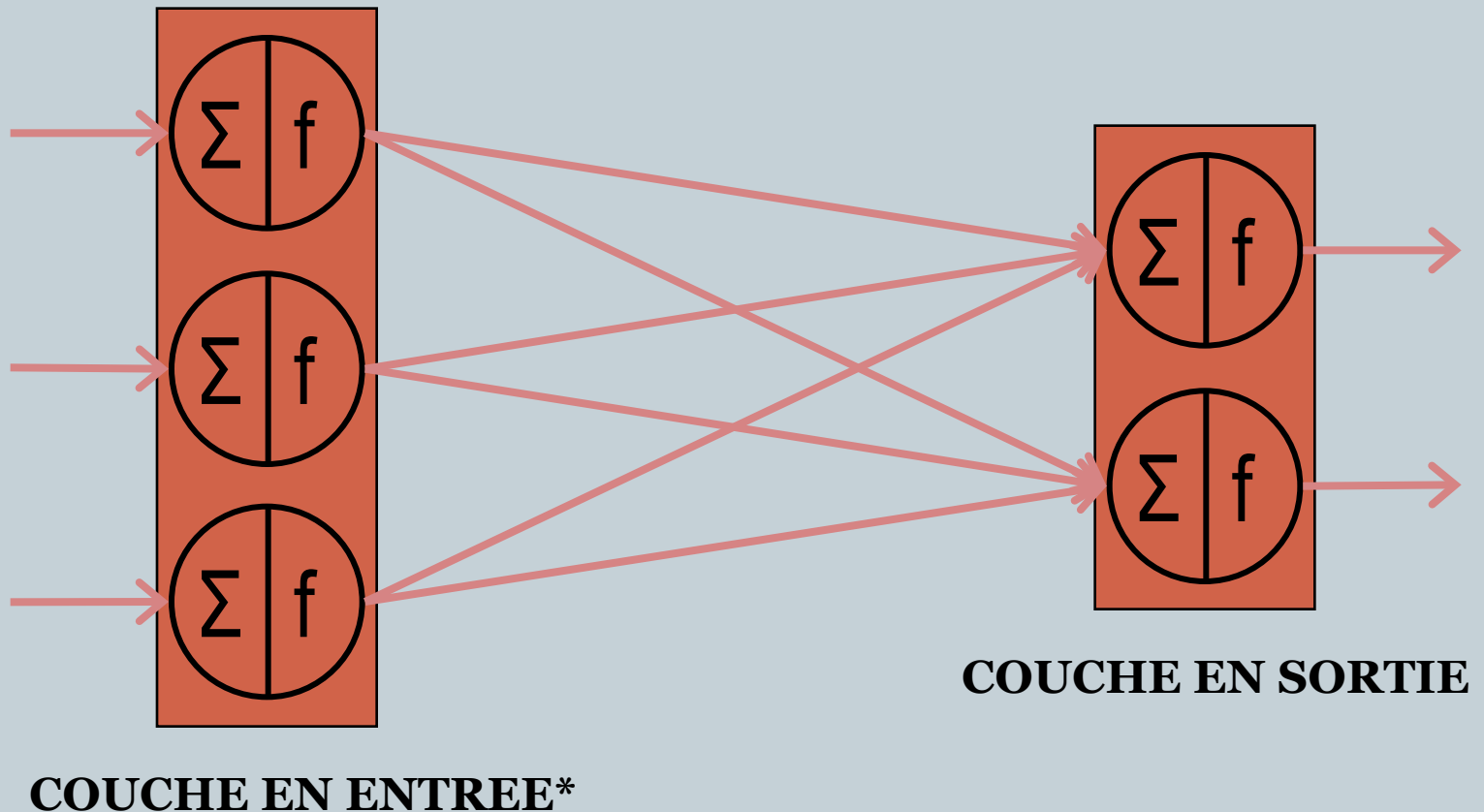


*Les entrées des réseaux sont soit des sorties d'autres neurones, soit des entrées directes dans le réseau (par exemple des pixels).

Réseaux de neurones formels

44

Réseau à deux couches de neurones

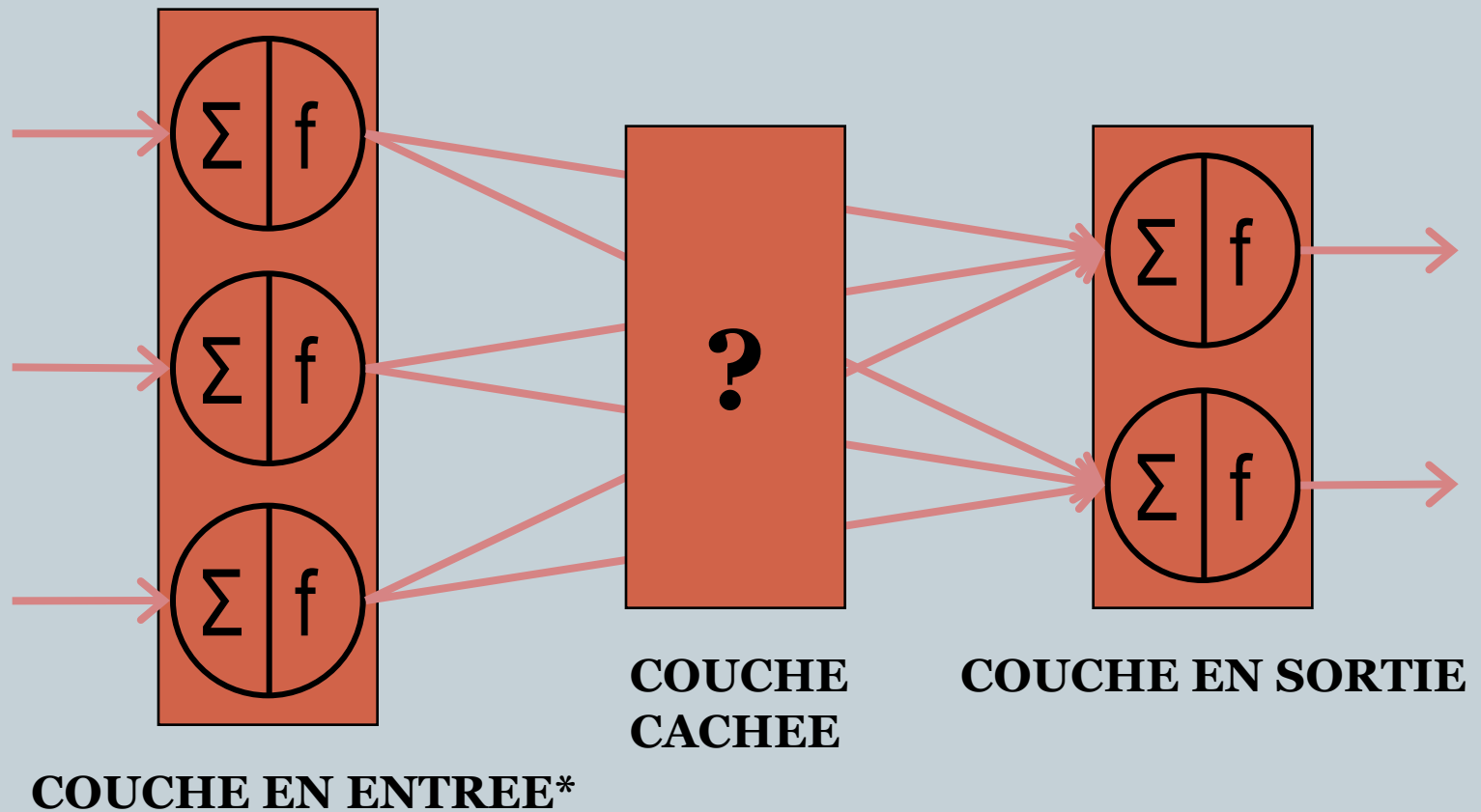


*Les entrées des réseaux sont soit des sorties d'autres neurones, soit des entrées directes dans le réseau (par exemple des pixels).

Réseaux de neurones formels

45

Réseau avec une couche cachée (3 couches de neurones).



*Les entrées des réseaux sont soit des sorties d'autres neurones, soit des entrées directes dans le réseau (par exemple des pixels).

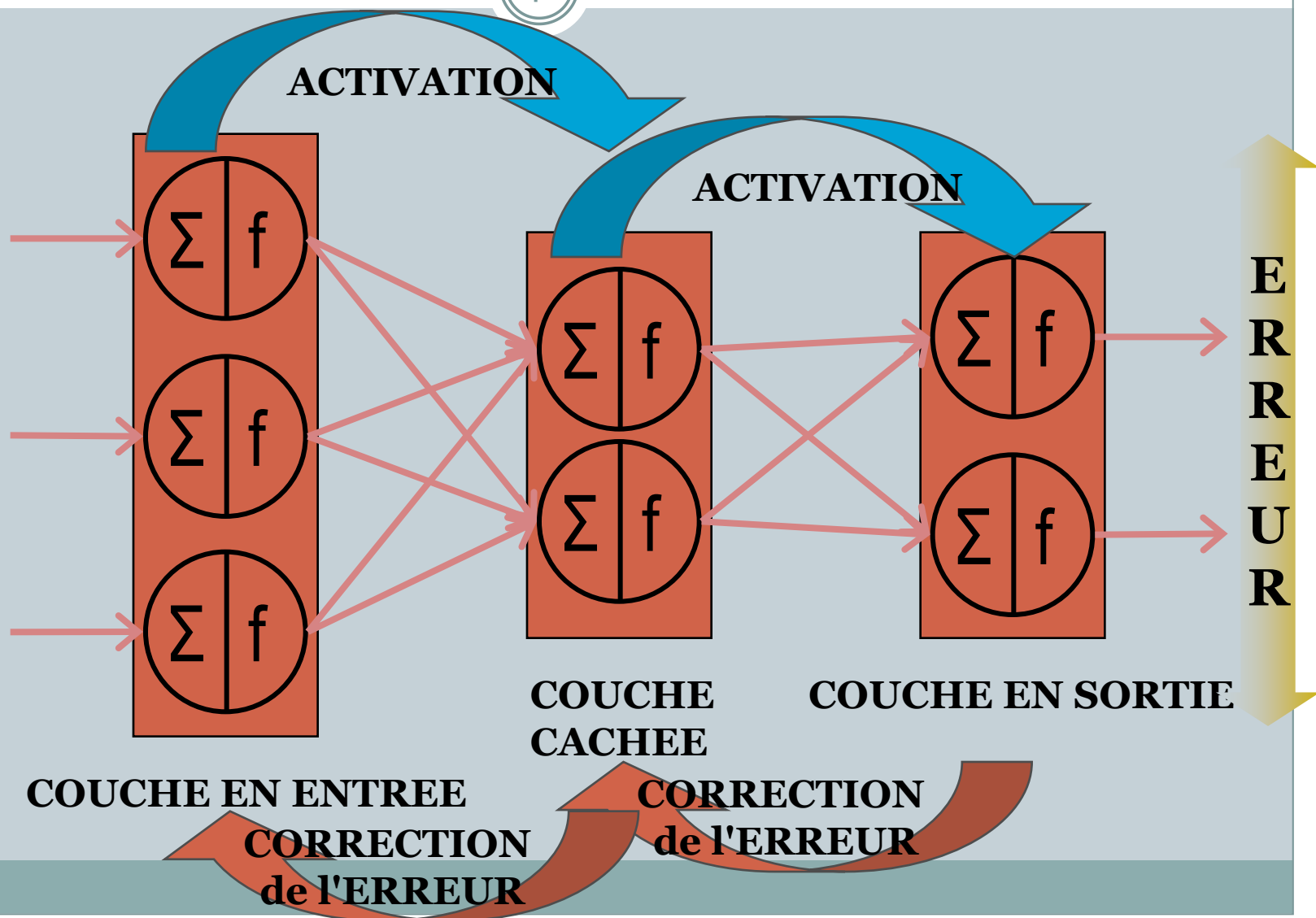


Réseaux multicouches à rétro-propagation de l'erreur

Apprentissage d'un réseau multicouche

Principe de fonctionnement général.

46



Apprentissage d'un réseau multicouche

Notations :

47

- $x_1, x_2, x_3, \dots, x_k$: les formes présentées en entrée.
- x_k : vecteur à I éléments.
- X : matrice $I \times K$ des K formes à apprendre.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1I} \\ x_{21} & x_{22} & \cdots & x_{2I} \\ \vdots & \vdots & \ddots & \vdots \\ x_{K1} & x_{K2} & \cdots & x_{KI} \end{bmatrix}$$

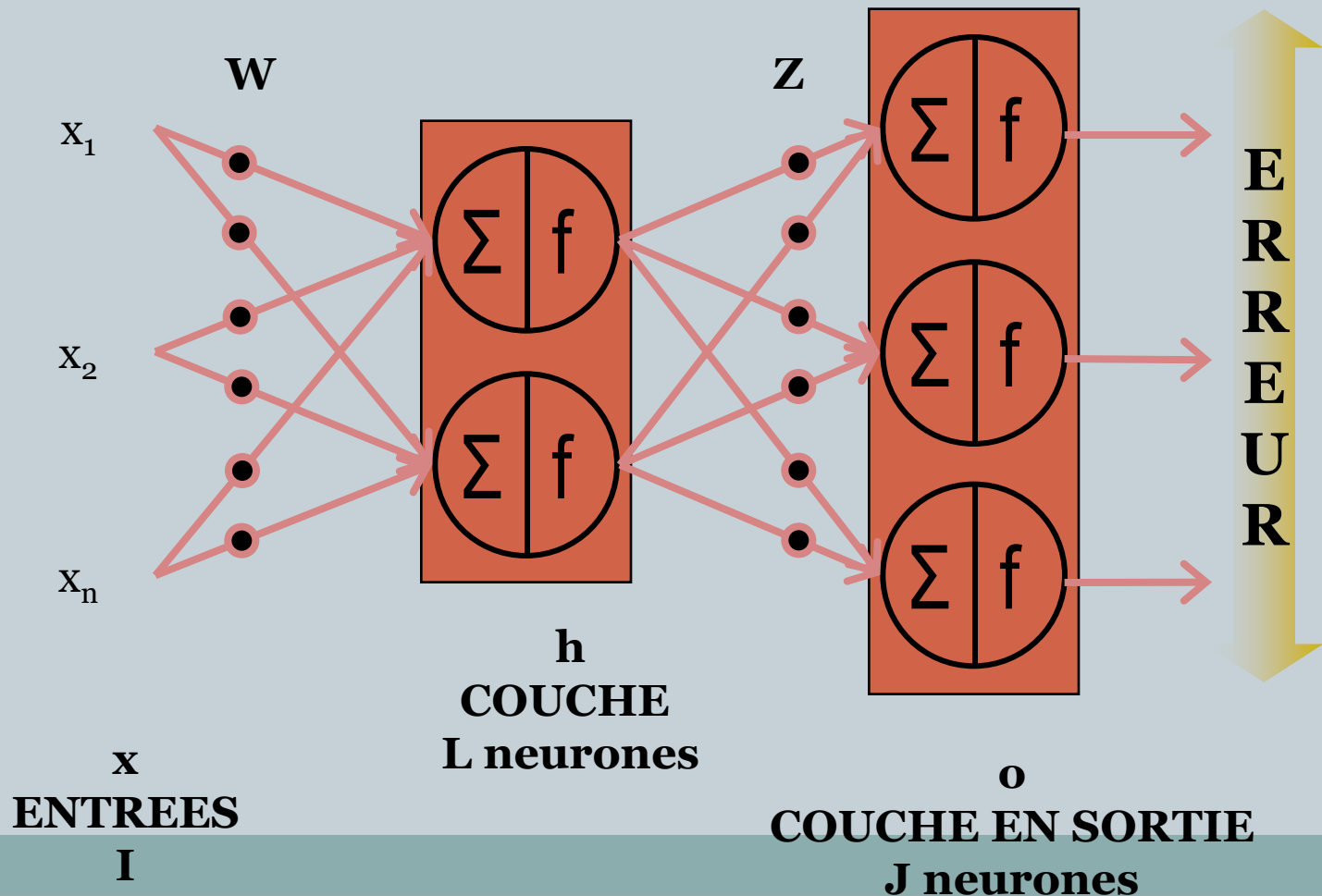
Chaque couche fournit un « vecteur réponse » :

- h_k : vecteur à L éléments, réponse de la couche cachée à la k ième forme.
- o_k : vecteur à J éléments, réponse de la couche de sortie à la k ième forme.
- t_k : vecteur à J éléments, réponse désirée (théorique) de la couche de sortie à la k ième forme.
- T : matrice $J \times K$ des réponses désirées (théoriques).
- W : matrice $L \times I$ des poids de connexions entre les entrées et la couche cachée ($w_{l,i}$: connexion entrée i - neurone l caché).
- Z : matrice $J \times L$ des poids de connexions entre la couche cachée et la couche de sortie ($z_{j,l}$: connexion neurone l caché - neurone j sortie).

Apprentissage d'un réseau multicouche

Les synapses modifiables (et leur matrice W et Z).

48



Apprentissage d'un réseau multicouche

Principe d'activation non linéaire.

49

- Soit un neurone n (d'une couche cachée ou de sortie) et son potentiel nommé a ; sa sortie o sera de la forme :

$$o_n = f(a_n)$$

- avec f : sa fonction de transfert (non linéaire, dérivable).

Exemples courants de fonction de transfert :

- La fonction logistique (ou sigmoïde) :

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

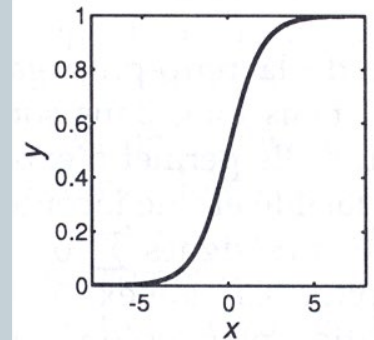
Apprentissage d'un réseau multicouche

Représentation graphique des fonctions de transfert.

50

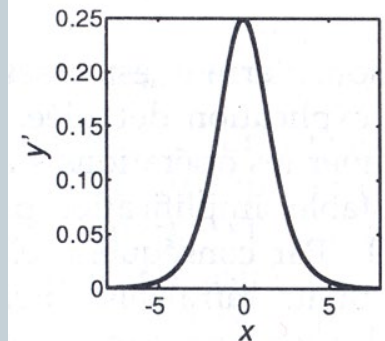
- La fonction logistique (ou sigmoïde) :

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



- et sa dérivée :

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = \frac{1}{1 + e^{-x}} \cdot \frac{e^{-x}}{1 + e^{-x}} = f(x)[1 - f(x)]$$



L'algorithme d'apprentissage

51

Algorithme de rétro-propagation de l'erreur,

Etape 1 : Transmission du signal entre l'entrée et la sortie via la couche cachée

- Soit le vecteur x_k à l'entrée (forme k).
- La réponse de la cellule cachée est le vecteur :
- La réponse des cellules de la couche de sortie est le vecteur :
- Les matrices des poids, W et Z , déterminent le comportement du réseau.

$$h_k = f(Wx_k)$$

$$o_k = f(Zh_k)$$

L'algorithme d'apprentissage

52

Algorithme de rétro-propagation de l'erreur,

Etape 2 : Calcul de l'erreur en sortie

- On compare la réponse donnée (vecteur o_k) à la réponse théorique (vecteur t_k).
- Erreur pour la k ème forme : $e_k = (t_k - o_k)$
- Le signal d'erreur résulte en pondérant l'erreur e_k par l'état d'activation de chaque cellule (une activation forte est plus nocive qu'une activation faible).

$$\delta_{\text{sortie},k} = f'(Zh_k) * e_k = o_k * (1 - o_k) * (t_k - o_k)$$

$*$: produit (de Hadamar) de deux matrices

$f'(Zh_k)$: intensité de l'activation des cellules de sortie

L'algorithme d'apprentissage

53

Algorithme de rétro-propagation de l'erreur,

Etape 3 : Correction des poids des connexions "cachée/sortie"

- La matrice des connexions Z est corrigée par des itérations successives.

$$Z_{t+1} = Z_t + \eta \cdot \delta_{sortie,k} \cdot h_k^T = Z_t + \Delta_t Z$$

η : nombre réel positif (le pas d'apprentissage)

L'algorithme d'apprentissage

54

Algorithme de rétro-propagation de l'erreur,

Etape 4 : Calcul de l'erreur en sortie des couches cachées

- Le problème est d'estimer l'erreur de l'activité (inconnue) des cellules cachées (pour une réponse attendue et connue en sortie uniquement !)
- Il n'y a pas de réponse idéale disponible.
- On l'estime à partir : $\delta_{sortie,k}$
 - du signal d'erreur :
 - de l'activation des cellules cachées.

L'algorithme d'apprentissage

55

Algorithme de rétro-propagation de l'erreur,

Etape 4 (suite) : Calcul de l'erreur en sortie des couches cachées

- L'erreur $\delta_{sortie,k}$ se propage en sens inverse (back-propagation) à travers les connexions Z .
- Elle est pondérée par l'activation $f'(Wx_k)$ des cellules cachées.
- Le signal d'erreur :

$$\delta_{cachée,k} = f'(Wx_k) * \left(Z_t^T \delta_{sortie,k} \right) = h_k * (1 - h_k) * \left(Z_t^T \delta_{sortie,k} \right)$$

L'algorithme d'apprentissage

56

Algorithme de rétro-propagation de l'erreur,

Etape 5 : Correction des poids des connexions "entrée/cachée"

- Calcul des poids des neurones d'entrée :

$$W_{t+1} = W_t + \eta \cdot \delta_{cachée,k} \cdot x_k^T = W_t + \Delta_t W$$

Résumé

Perceptron :
principe de
séparabilité
linéaire

NEURONE
=
Combinaison
+
Transfert

**Rétro-
propagation**
de l'erreur :
estimation de
l'erreur en
couches
cachées

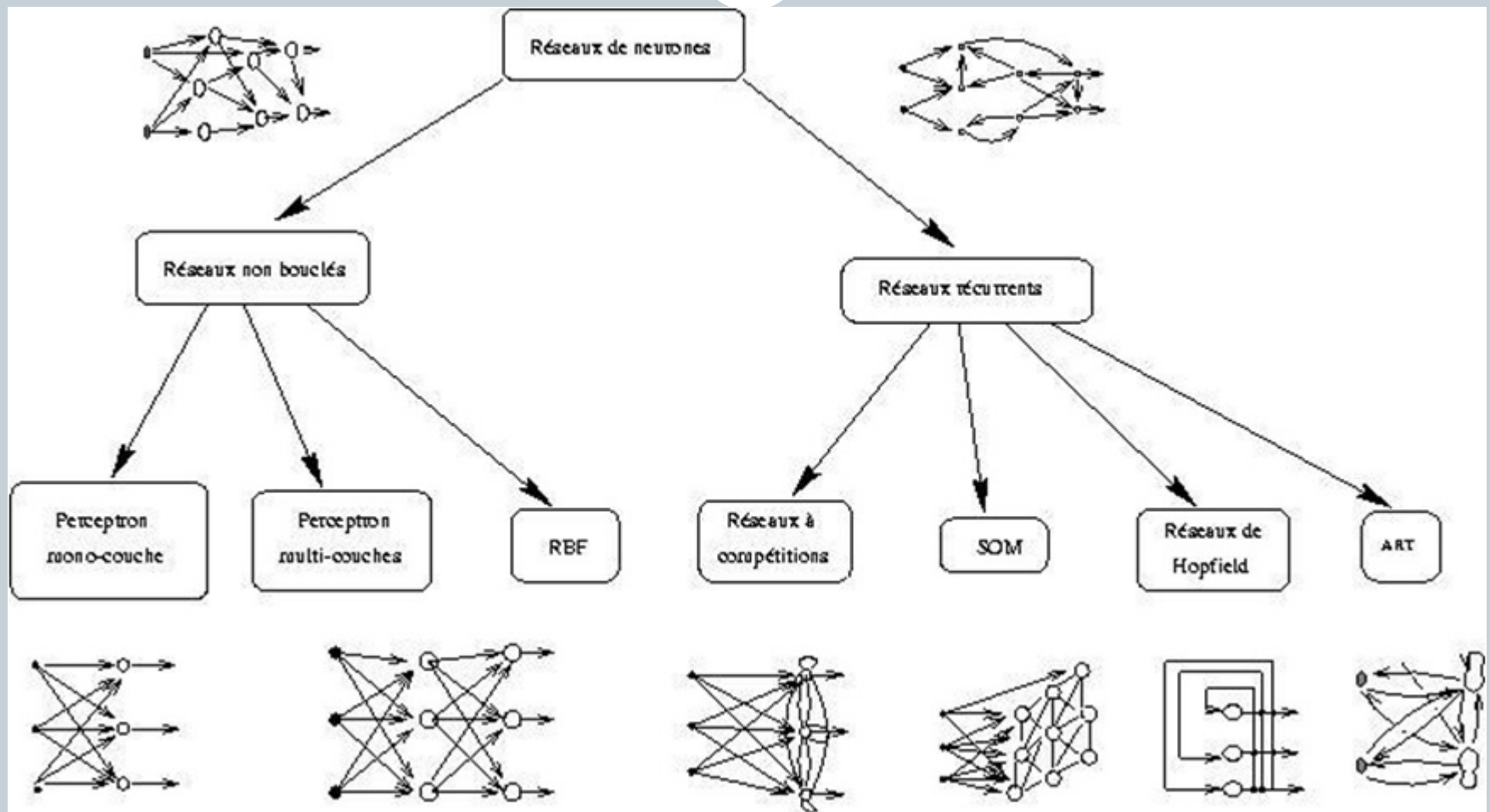
Fonction
Combinaison
:
 $\sum W_i E_i$

Fonction Transfert :
en échelon, ou
linéaire par
morceaux, ou
dérivable

Résumé

58

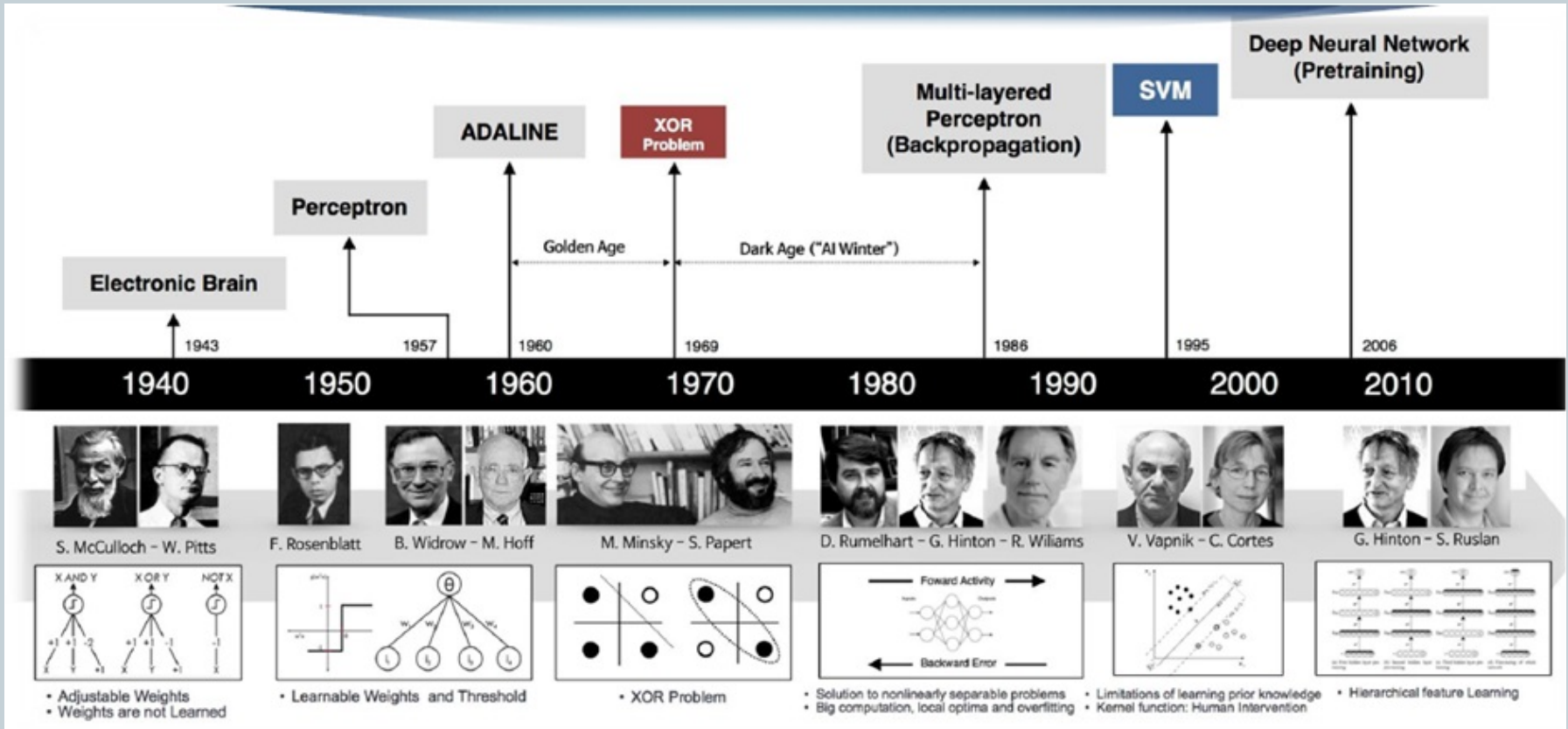
- Nous avons présenté la théorie des réseaux de neurones artificiels *feed-forward* (c'est à dire ne comportant pas de connexions vers des couches précédentes).
- Il existe des structures de réseaux de neurones beaucoup plus complexes. Si cela vous intéresse, renseignez-vous sur les **réseaux récurrents** et les **cartes de Kohonen**, par exemple.



- Introduction au Deep Learning

Historiquement

61



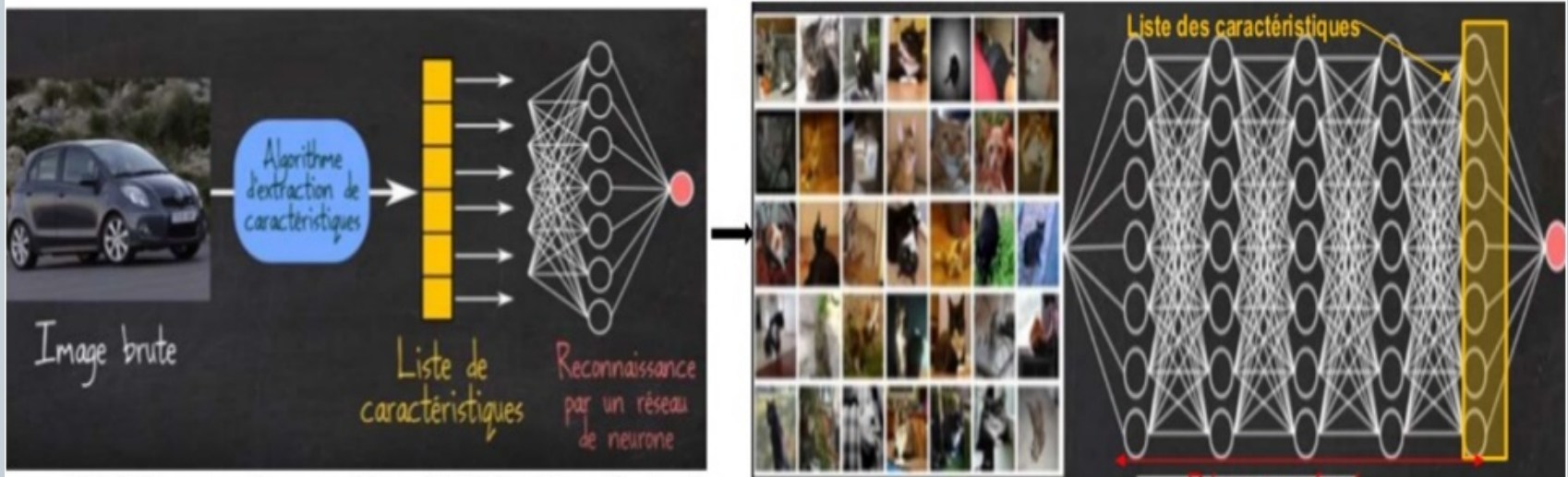
- L'apprentissage profond est l'une des branches de l'apprentissage automatique, il est extrait des réseaux de neurones en 2006, c'est tout simplement des réseaux de neurone avec un nombre de couches massive, plus volumineux et plus profonds.

- C'est un type particulier d'apprentissage machine qui atteint une grande puissance et flexibilité en apprenant à représenter le monde comme une hiérarchie imbriquée de concepts, chaque concept étant défini par rapport à des concepts plus simples et des représentations plus abstraites calculées en termes moins abstraits.

- **Définition : L'apprentissage profond est un ensemble de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d'abstraction des données grâce à des architectures articulées autour de différentes transformations non linéaires.**
- Elles fonctionnent avec un apprentissage à plusieurs niveaux de détails ou de représentations des données. A travers les différentes couches on passe de paramètres de bas niveau à des paramètres de plus haut niveau.
- Ces différents niveaux correspondent à différents niveaux d'abstraction des données

Principe général

65



La qualité de la reconnaissance dépend de l'algorithme d'extraction de caractéristique

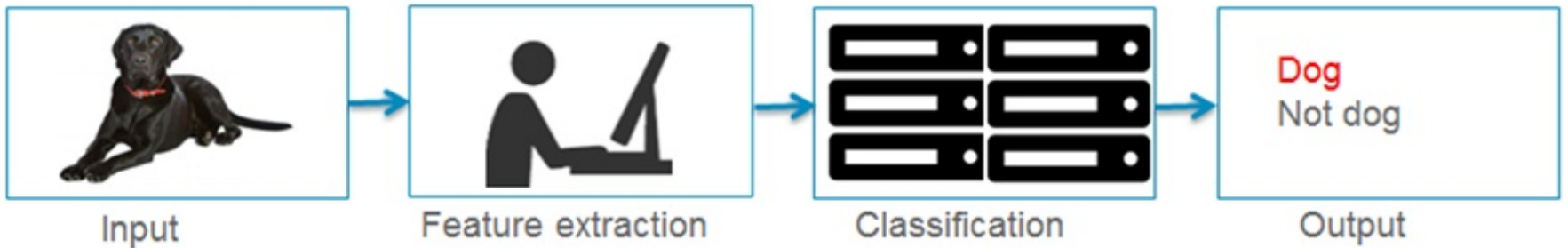
Les techniques d'apprentissage profond "apprentissage dans les réseaux de neurones profonds" constituent une classe d'algorithmes d'apprentissage automatique qui :

- utilisent différentes couches d'unité de traitement non linéaire pour l'extraction et la transformation des caractéristiques ; chaque couche prend en entrée la sortie de la précédente.
- fonctionnent avec un apprentissage à plusieurs niveaux de détail ou de représentation des données ; à travers les différentes couches.

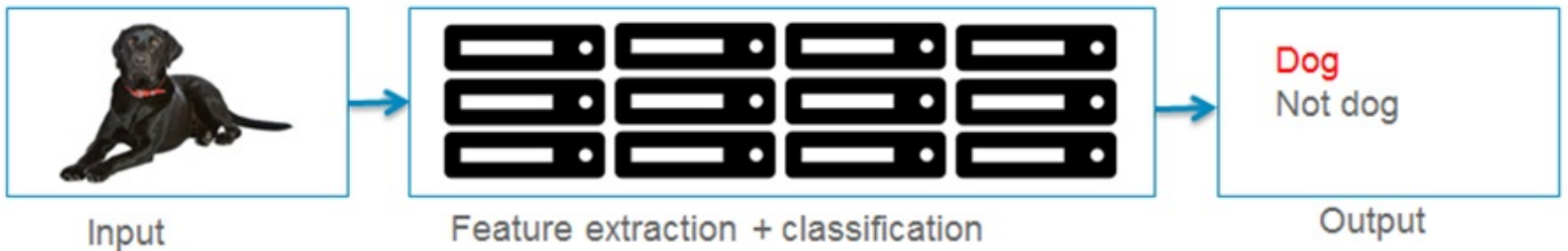
[Yoshua Bengio]

Deep
learning

Traditional machine learning



Deep learning

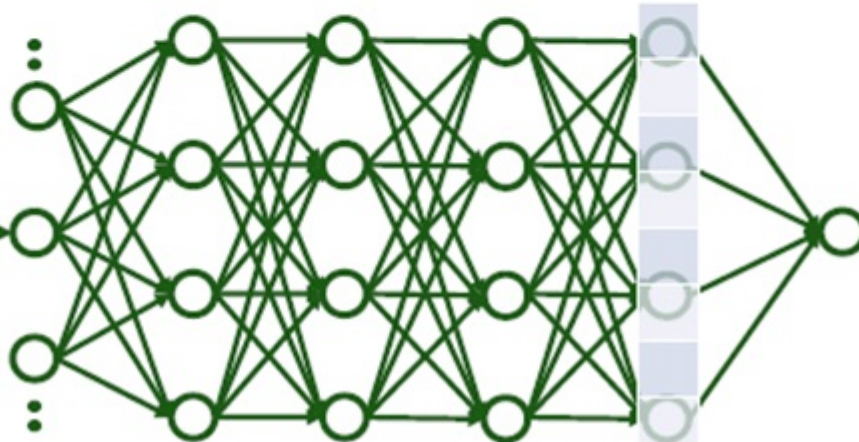


DeepLearning

67



Extraction de
caractéristiques



Résumé

68

- Le Deep Learning a de nombreuses utilités.
- C'est cette technologie qui est **utilisée pour la reconnaissance faciale de Facebook par exemple**, afin d'identifier automatiquement vos amis sur les photos.
- C'est également cette technologie qui permet à la reconnaissance faciale Face ID de l'iPhone X d'Apple de s'améliorer au fil du temps. Comme expliqué précédemment, l'apprentissage automatique est également la technologie centrale de la reconnaissance d'images.
- Les possibilités offertes par cette technologie augmenteront à mesure que nous découvrons les secrets de notre propre organe.
- Offre des résultats intéressant en apprentissage machine , futur prometteur.