Réseaux neuromimétiques

Chloé Clavel

Adaptation du cours d'Alain Grumbach

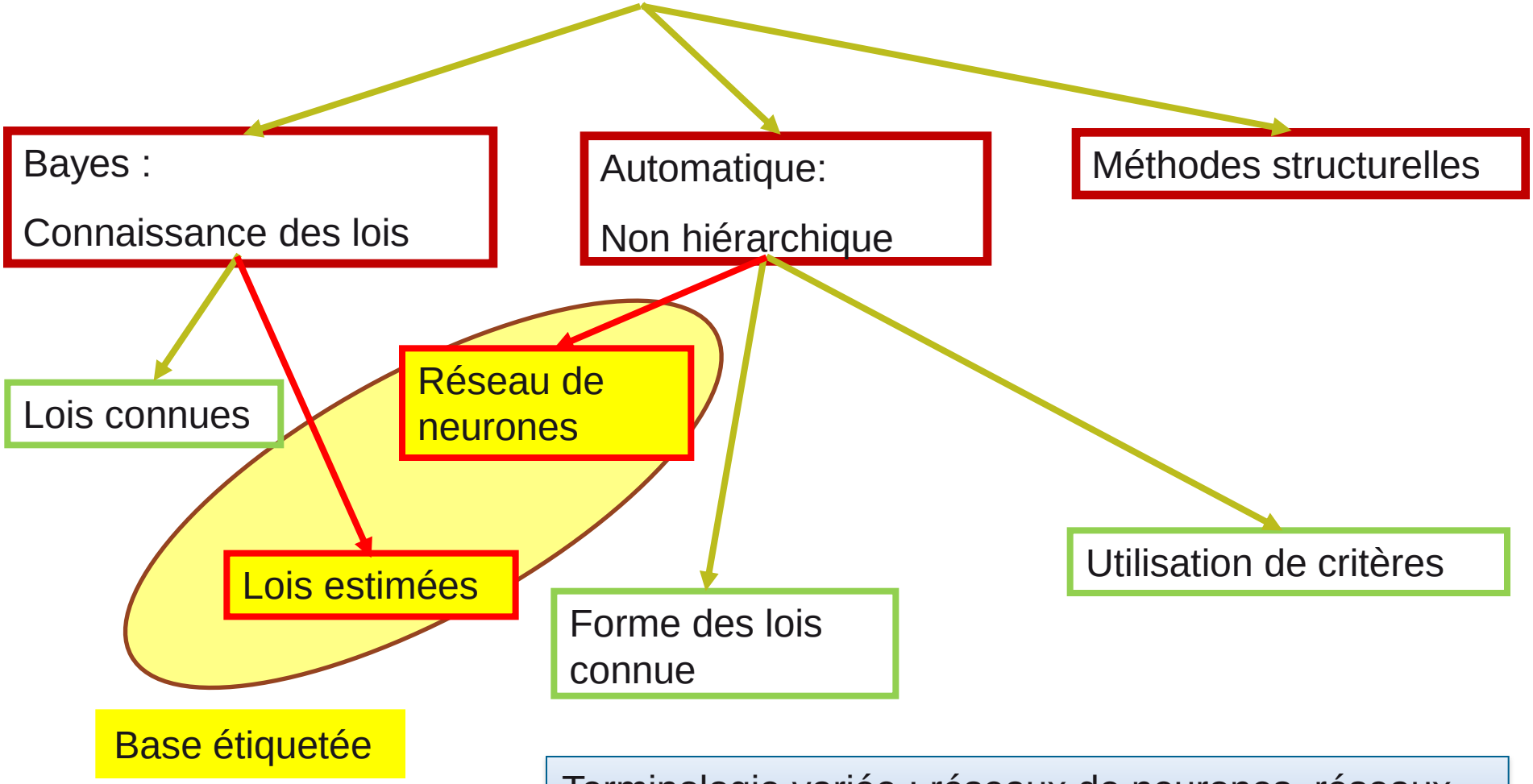
& Jean Marie Nicolas

Introduction

Terminologie variée : réseaux de neurones, réseaux neuromimétiques, réseaux connexionnistes

- **C'est une méthode de classification supervisée**
  - Apprentissage sur la base d'apprentissage :
    - Objectif : trouver la bonne architecture du système de classification
  - Développement (ou généralisation) :
    - Objectif : juger des performances de l'architecture sur des individus n'ayant pas servi à définir l'architecture
  - Test sur données non utilisées en apprentissage et en développpement!!

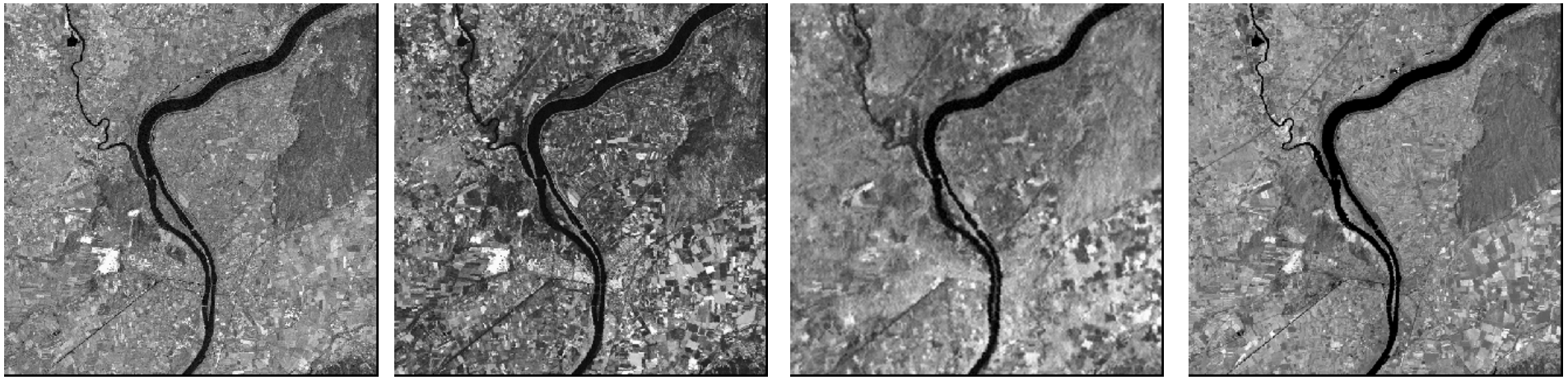
Méthodes de classification

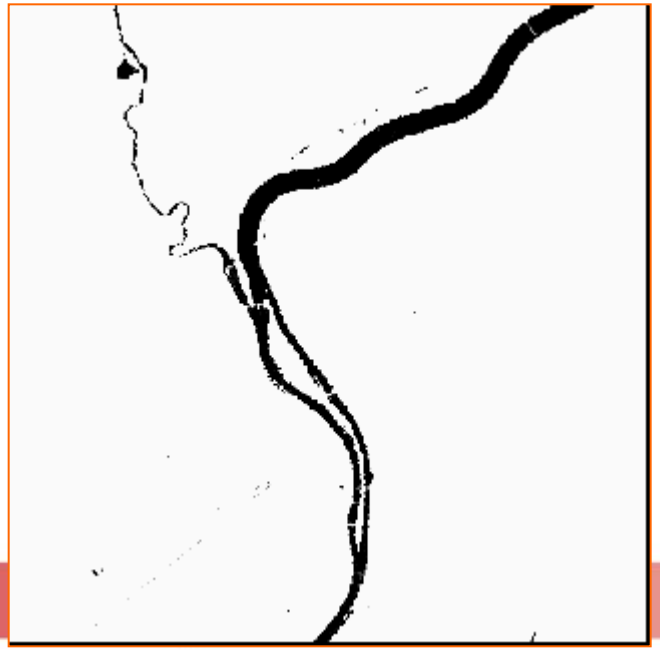


Terminologie variée : réseaux de neurones, réseaux neuromimétiques, réseaux connexionnistes

Exemple 1 : Le TP « Réseaux de neurones »

- Une image multispectrale Landsat sur Tarascon





Exemple 2 :

Reconnaissance de caractères

Y. Le Cun et al.

AT & T Bell Labs

1990

Le problème

Données: 9298 chiffres extraits de codes postaux manuscrits

3249 chiffres en caractères d'imprimerie (35 polices)

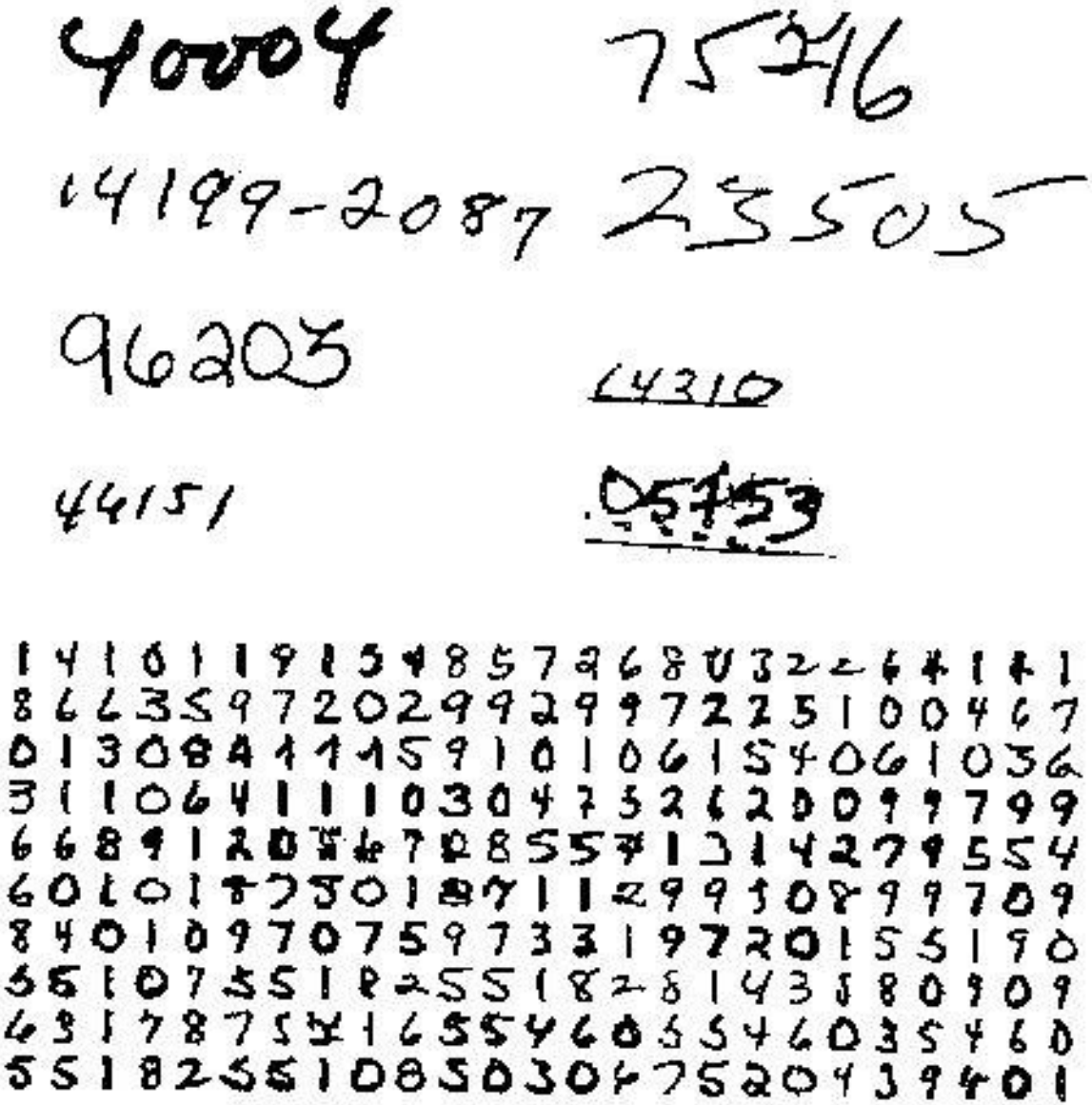
Base d'exemples : 7291 chiffres manuscrits

2549 chiffres en caractères d'imprimerie

Base de test : 2007 chiffres manuscrits

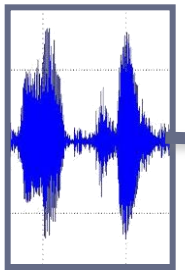
700 chiffres en caractères d'imprimerie

(les deux bases contiennent des exemples ambigus (naturels))



Reconnaissance de la parole

- Le problème : passer du signal à la transcription



Transcription automatique de la parole

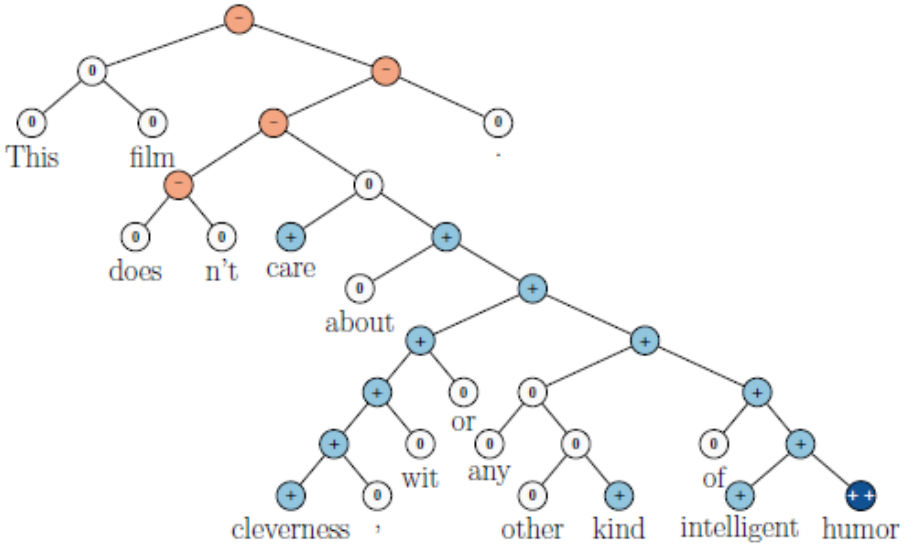
00:02:Et bienvenue dans votre édition de la nuit, l'actualité de ce mardi en 3 titres, le coup d'envoi du pacte de responsabilité lors des voeux de François Hollande force vive le président exige des contreparties en termes d'emplois aux allégements de charges annoncées, syndicats et patronat en ordre dispersé, une nouvelle étape dans la lutte anti-tabac, la Haute autorité de la santé demande instamment aux médecins qui retrouve impatient de le dissuader de fumer J moins 1 pour la la conférence de Genève sur l'avenir de la Suisse Syrie: les Iraniens sont absents, les rues sont mécontents, quelles sont les chances de réussite de cette conférence, c'est ce que nous verrons et puis le football avec les seizièmes de finale de la Coupe de France, Marseille reçoit Nice depuis 21 heures, mais combien de terre. Eh bien, c'est un régal absolu, cette huitième de finale de Coupe de France 4 buts à 3 pour l'OGC Nice face à l'Olympique de Marseille, c'est la 68E. Ce jeu à tout à l'heure, Yann Théroux et puis une question, quelles seraient les conséquences pour l'économie

Tiré de Voxalead :
http://voxalead.labs.exalead.com/

Lippmann, Richard P. "Review of neural networks for speech recognition."*Neural computation* 1.1 (1989): 1-38.

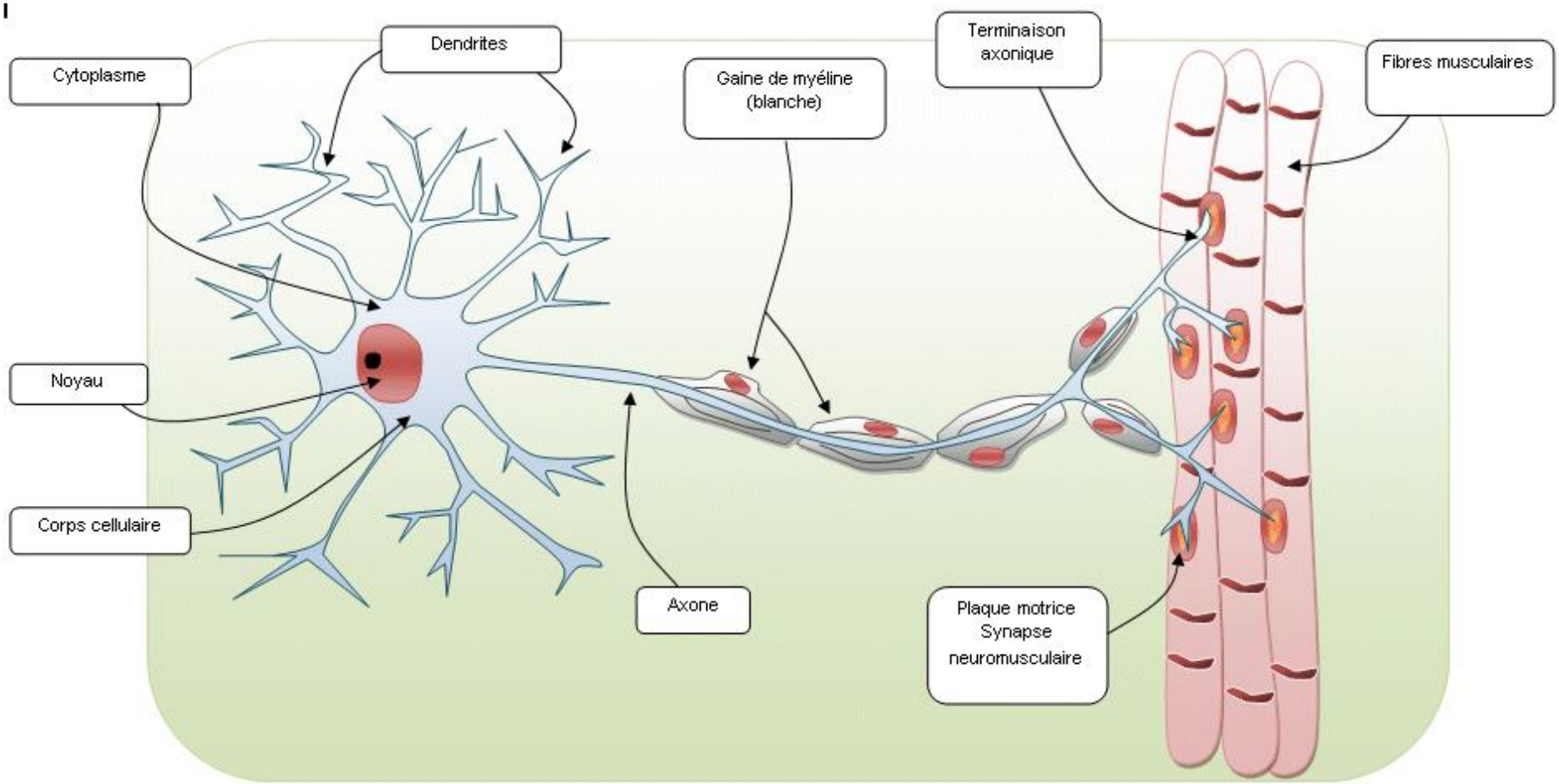
Opinion mining et deep learning

- **Remise au goût du jour des réseaux de neurones avec l'émergence du deep learning**
  - Utilisation des réseaux récursifs tensoriels
    - permettent de prendre en compte la structure d'une phrase.



- exemple d'utilisation des réseaux récurrents
  - REF : R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Stroudsburg, PA : Association for Computational Linguistics, October 2013, pp. 1631? 1642.

Un neurone



Organisation d'un neurone et jonction neurone fibres musculaires

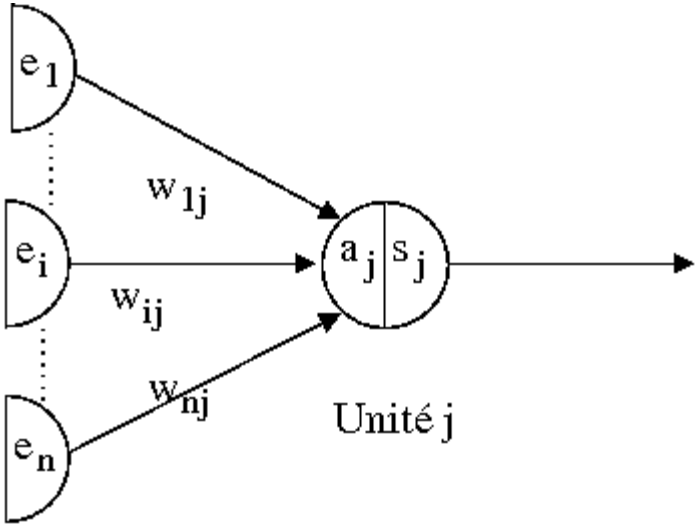
http:svt.ac-dijon.fr

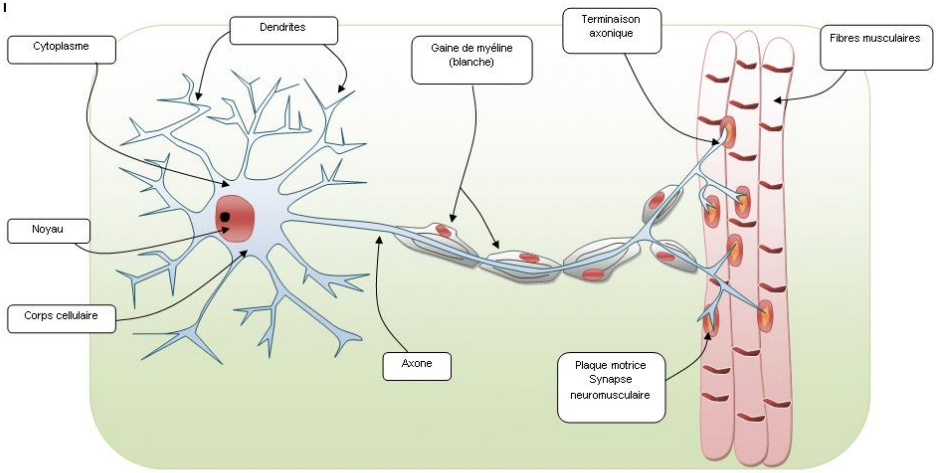
Les neurones en réseau….

On estime que le cerveau humain contient entre un et cent milliards de neurones. Chaque neurone est connecté en moyenne à 10.000 autres par le biais de synapses. Les stimulations synaptiques perçues par un neurone peuvent activer celui-ci qui transmet alors un signal électrique aux neurones suivants ou à d'autres cellules.

Neurone formel

McCulloch et Pitts (1943)





Neurone formel j
Entrées e_i $i \in [1,n]$
Activité a_i
Sortie s_i

$$a_j = \sum_{i=1}^{N} w_{ij} e_i$$

$$s_j = f(a_j)$$

Poids synaptique w_{ij}
Fonction d'activation f

Charge : + ou -

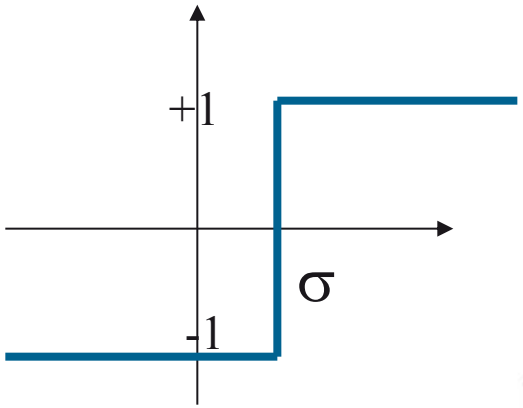
Neurone formel
La fonction d'activation

- **Fonction d'activation :**
  - Permet de calculer la sortie de l'unité en fonction de ses entrées
- **Exemple de fonction activation : fonction identité**
  - ALC : Adaptive Linear Combiner

$$s_j = \sum_{i=1}^{N} w_{ij} e_i$$

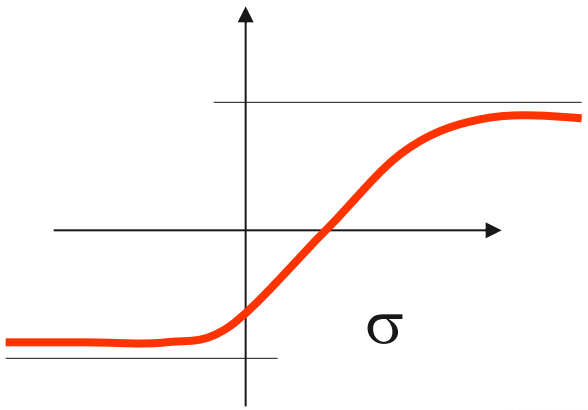
Neurone formel
La fonction d'activation

- **Exemple de fonction activation : la fonction seuil**
  - 0 ou 1
  - -1 ou 1 (cf charge +e ou –e : Charges + ou – dans les neurones biologiques)
  - Non dérivable



Neurone formel
La fonction d'activation

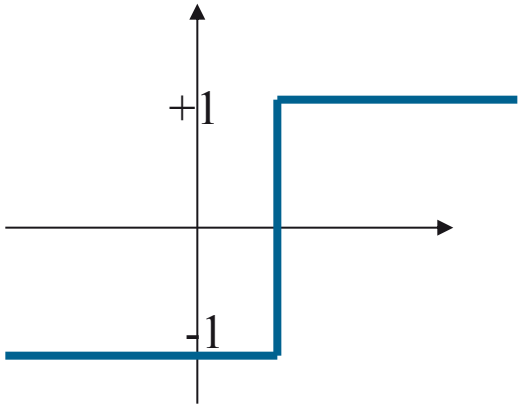
- **Exemple de fonction activation : fonction « sigmoïde »**
  - f(x) ∈]-1;1[
  - Réglage : pente en y=0
  - Cas limite : fonction seuil
  - dérivable

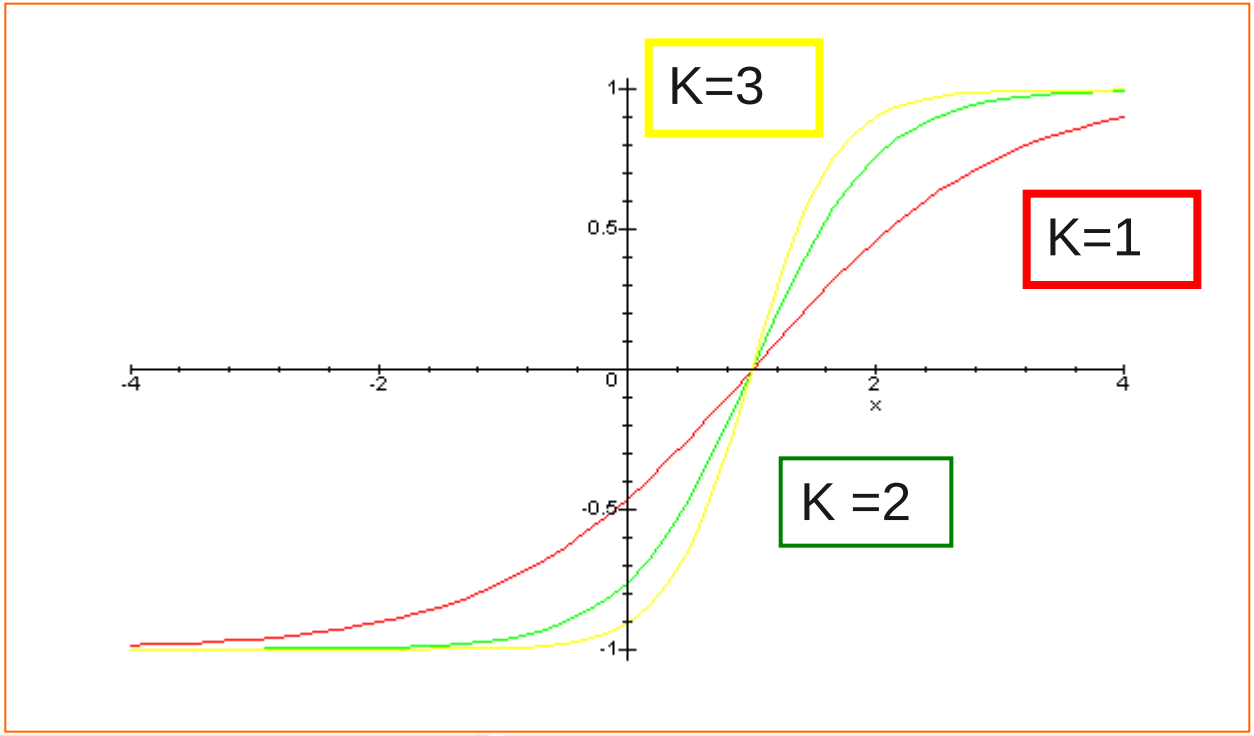


Fonction sigmoïde

$$f[K,\sigma](x) = \frac{e^{K(x-\sigma)} - 1}{e^{K(x-\sigma)} + 1}$$

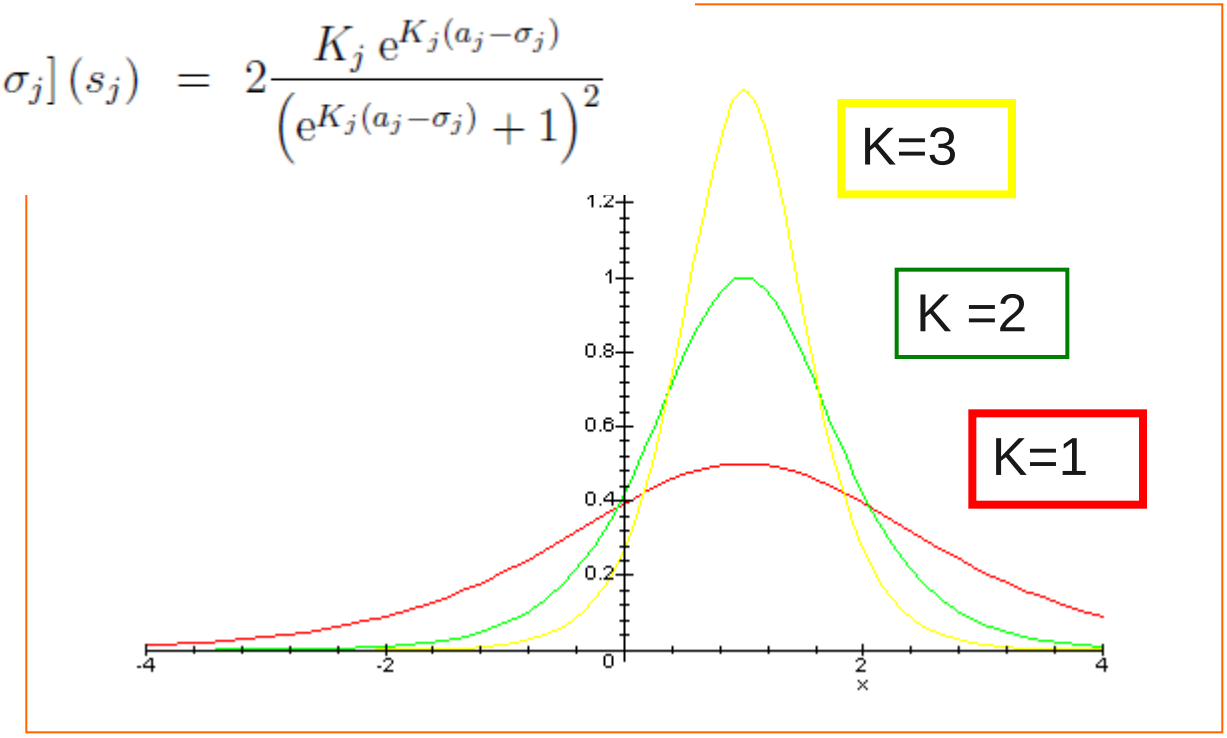
Limite :





Fonction sigmoïde : dérivable

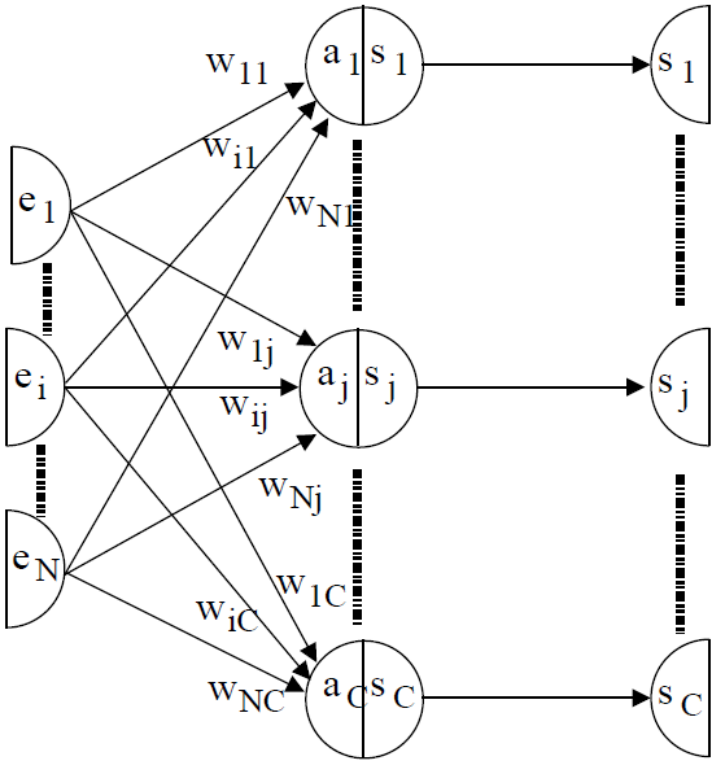
$$f'_{sigmoide}[K_j, \sigma_j](s_j) = 2\frac{K_j\, e^{K_j(a_j-\sigma_j)}}{\left(e^{K_j(a_j-\sigma_j)}+1\right)^2}$$



PERCEPTRON MONOCOUCHE

Perceptrons simples ou monocouches

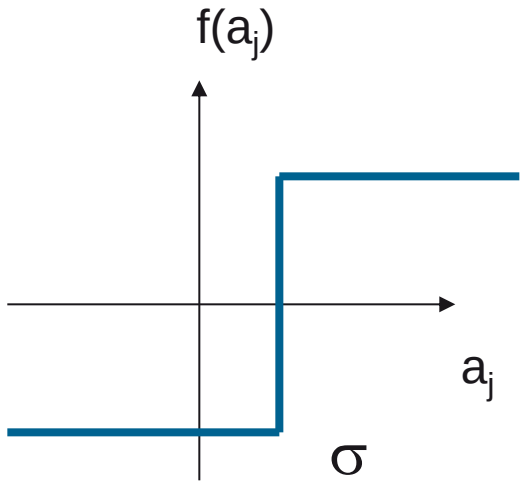
- Réseaux caractérisés par des connexions directes entre entrées et sorties
- Plusieurs unités/neurones formels organisés sur une seule couche



Cas de la fonction seuil : perceptron « à la Rosenblatt »

- **(x_k) ou (e_k) un vecteur d'entrée ($k \in [1,N]$)**
- **(s_j) sorties « binaires » (+1 ou -1), $j \in [1,R]$**
- **(d_j) sorties désirées(+1 ou -1), $j \in [1,R]$**
- **Poids (w_{kj})=W_j $k \in [1,N]$**
- **Fonction d'activation : fonction seuil f**

$$a_j = \sum_{k=1}^{N} w_{kj} x_k$$
$$s_j = f(a_j)$$



Apprentissage: le perceptron à la Rosenblatt

- **Problème à deux classes :**
  - Classe C^+ et classe C^-
  - Propriété souhaitée

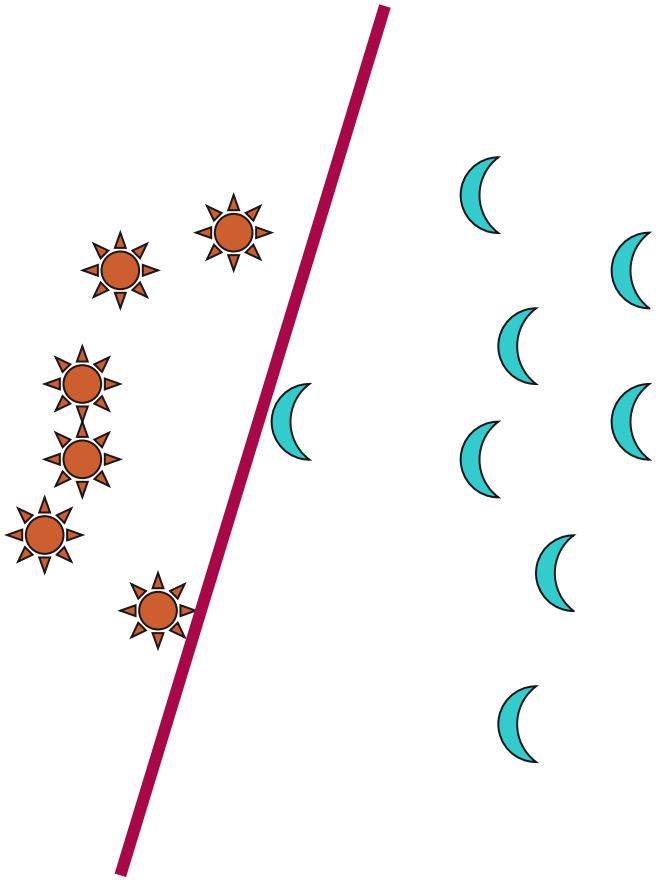
$$\forall x \in C^+ \; W^t x + b > 0$$

$$\forall x \in C^- \; W^t x + b < 0$$

$$b = -\sigma$$

  - Apprentissage :
    - Séparation par l'hyperplan : $W^t x + b = 0$

Apprentissage: le perceptron vu comme un séparateur linéaire



- **Equation de l'hyperplan**

$$W^t x + b = 0$$

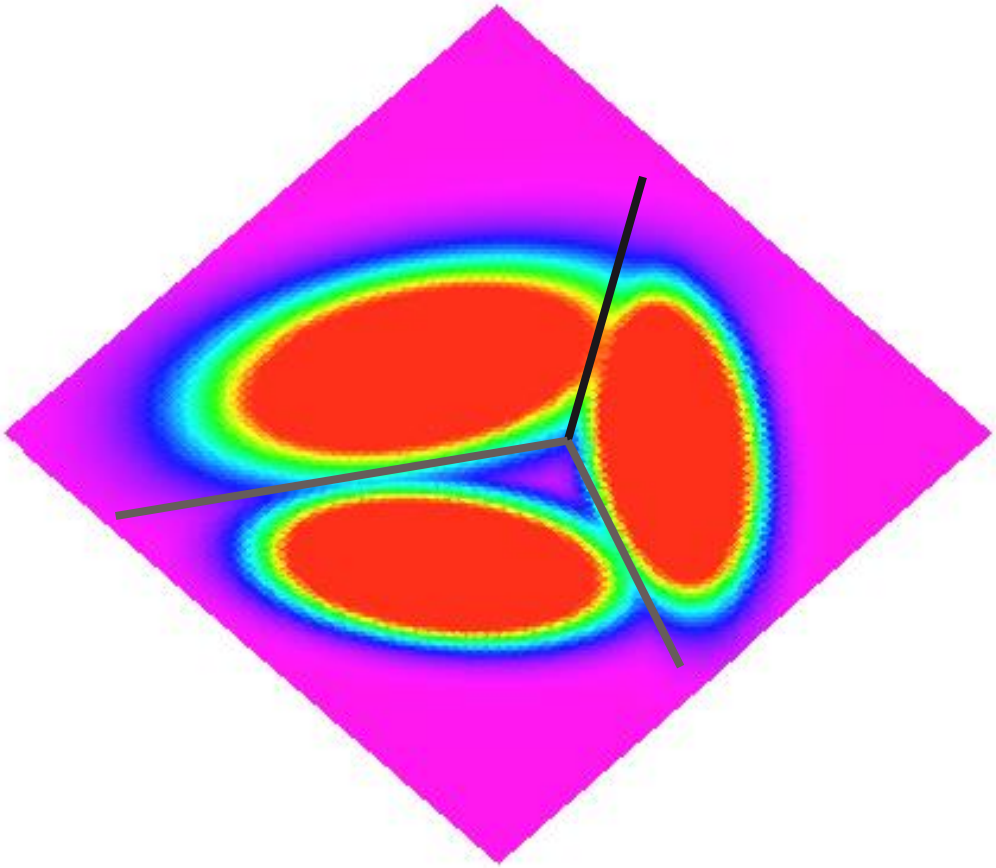
- **Si X appartient à la classe C^+**

$$W^t X + b > 0$$

- **Si X appartient à la classe C^-**

$$W^t X + b < 0$$

Cas multiclasses



- **Apprentissage :** Définir les séparatrices linéaires

Méthode - Un artifice de réalisation : rajouter une dimension

$$b = -\sigma$$

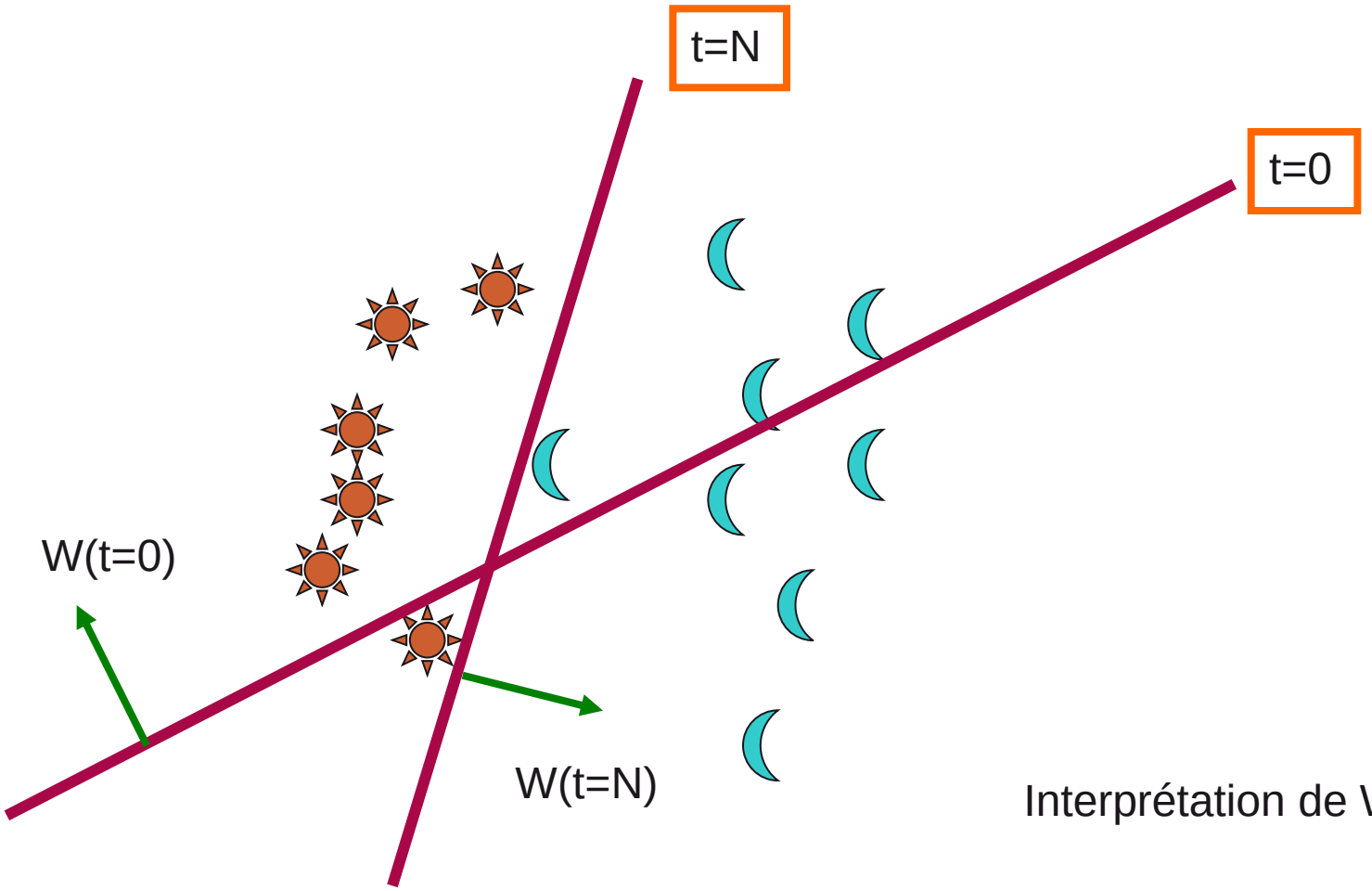
$$W^t x + b = 0$$

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \rightarrow \vec{\tilde{x}} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} w_1 & w_2 & \dots & w_n & -\sigma \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ 1 \end{pmatrix} = 0$$

$$\tilde{W}^t \tilde{X} = 0$$

Exemple linéairement séparable



Interprétation de W

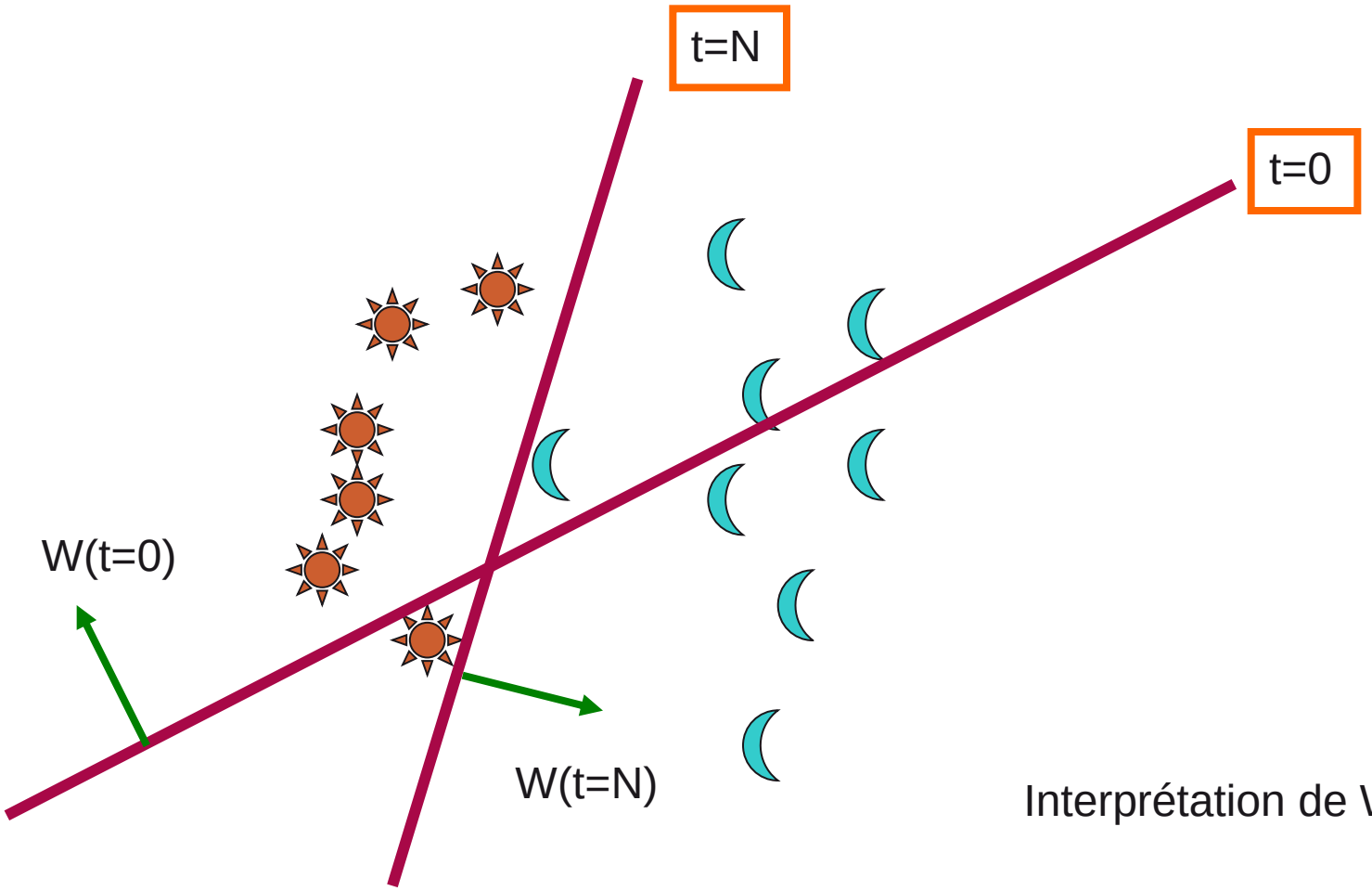
Apprentissage: la règle du perceptron

- **Règle du Perceptron (Rosenblatt, 1957) :**
  - Initialiser les poids des neurones
  - Modifier itérativement les poids si la sortie n'est pas égale à la sortie désirée par la règle

$$\tilde{W}(t+1) = \tilde{W}(t) \pm \eta \tilde{X}$$

- **W définit un hyperplan séparateur si le problème est linéairement séparable**

Exemple linéairement séparable



Interprétation de W

Test d'arrêt pour l'apprentissage

- **Au bout d'un certain nombre d'itérations (« époques »).**
- **Cas linéairement séparable**
  - Lorsque tous les vecteurs sont bien classés
- **Si non linéairement séparable**
  - Lorsque tous les vecteurs sont « à peu près » bien classés
    - Ceci requiert la définition d'une métrique (ex: définition de l'erreur)

Définition d'une métrique : l'erreur quadratique

APPRENTISSAGE DANS LE CAS NON LINEAIREMENT SEPARABLE

Définition d'une métrique : erreur de classification

- **Nombre de bien classés :**
  - Taux de bonne classification : τ_A
  - R_A individus :

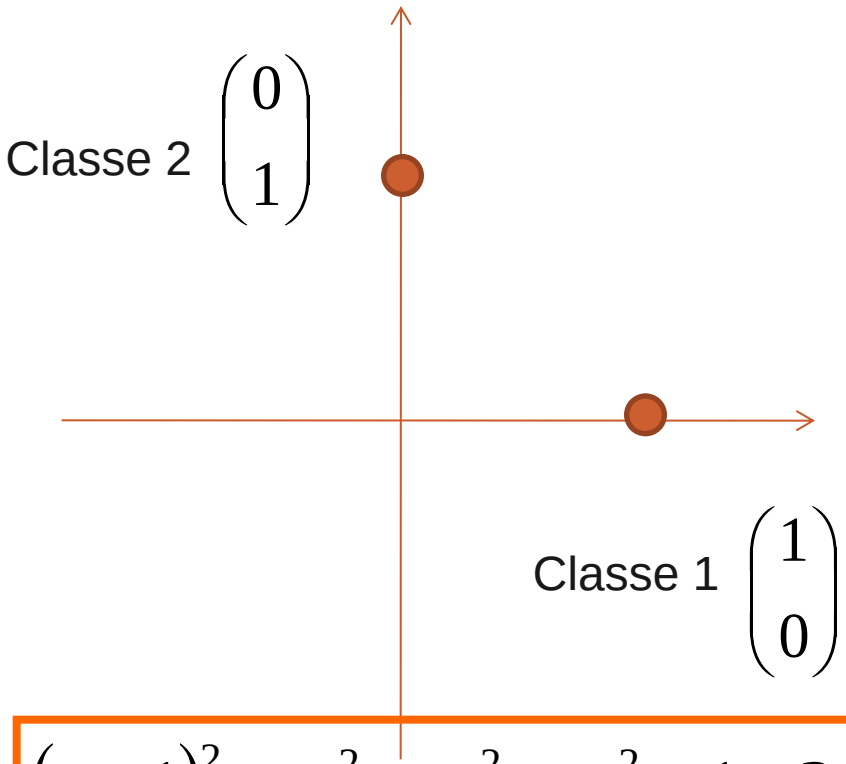
$$\tau_A = \frac{\text{Nombre d'individus bien classés}}{R_A}$$

Erreur de classification : Rappel : matrice de confusion

		Classification		
		Classe 1	**Classe j**	**Classe c**
Référence	**Classe 1**	X_{11}	X_{1j}	X_{1c}
	Classe i	X_{i1}	X_{ij}	X_{ic}
	Classe c	X_{c1}	X_{nj}	X_{cc}

Le taux de bonne classification revient à ne s'intéresser qu'à la diagonale

Exemple : deux classes espace de sortie en dimension 2



$$(x-1)^2 + y^2 = x^2 + y^2 + 1 - 2x$$

$$x^2 + (y-1)^2 = x^2 + y^2 + 1 - 2y$$

- Deux classes
- Deux étiquettes « objectif »
- A un point donné on cherche la classe la plus proche
- Si le point est mal classé, on peut définir une erreur : l'erreur quadratique

Définition d'une métrique : Erreur quadratique

- Expression bien connue
- Expression dérivable

R exemples d'apprentissage

$$J(W) = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k} \right)^2$$

cas où f est la fonction identité

$$f\left(W^t X_k\right) = W^t X_k$$

- **Ensemble des mal classés : Y(W)**

$$\tilde{J}(W) = \sum_{\substack{k=1 \\ k \in Y(W)}}^{R} \left(W^t X_k - d_{X_k} \right)^2$$

- Non dérivable (fonction d'appartenance)

Définition d'une métrique pour l'apprentissage

- **Erreur quadratique :**
  - Diminue au début de l'apprentissage
  - Attention au sur-apprentissage

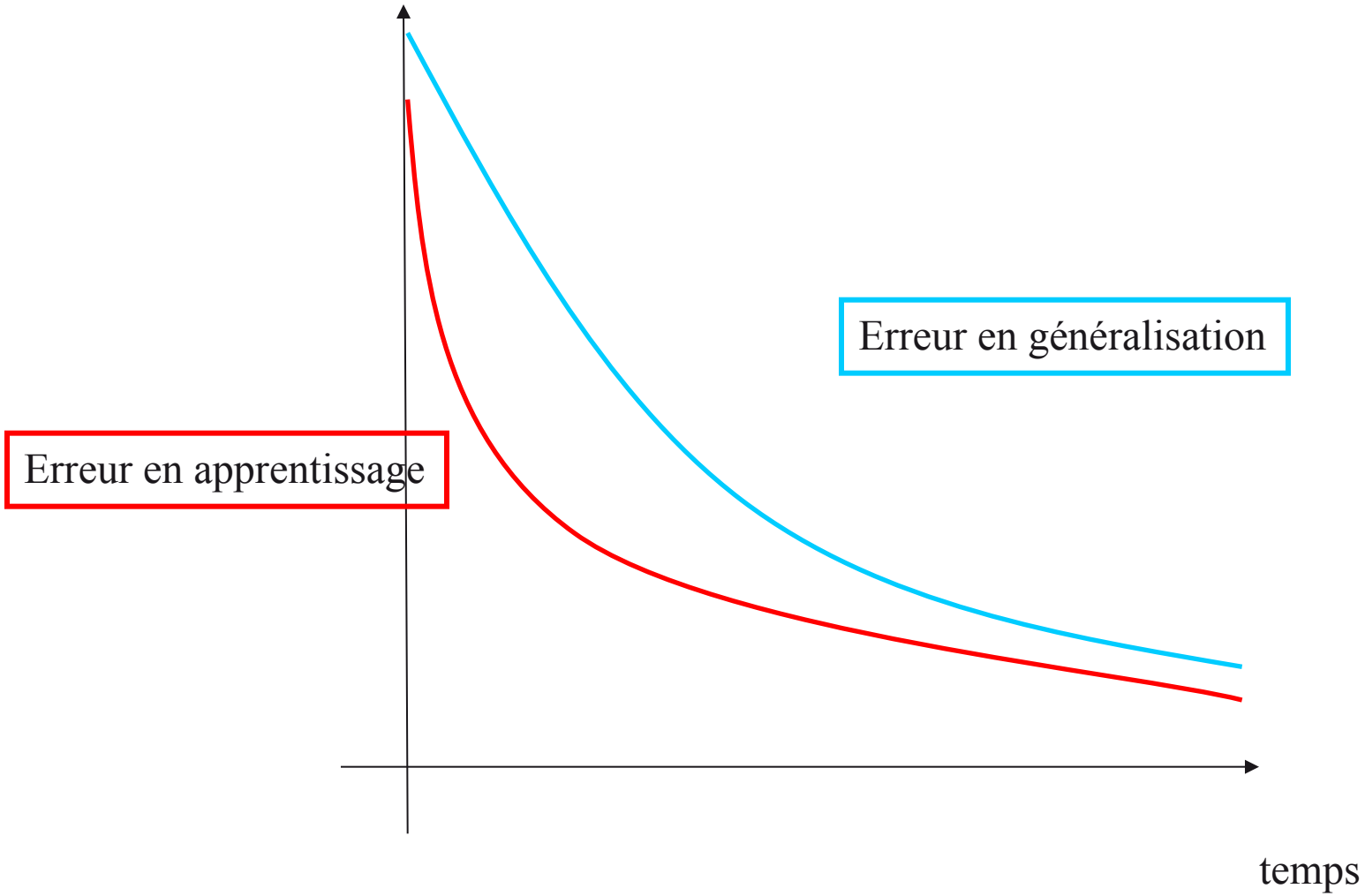
Critères sur la qualité de la classification : base de reconnaissance (généralisation)

- **Utiliser l'architecture définie par l'utilisation de la base d'apprentissage**
- **Nombre de bien classés :**
  - Taux de bonne classification : τ_G
  - R_G individus :

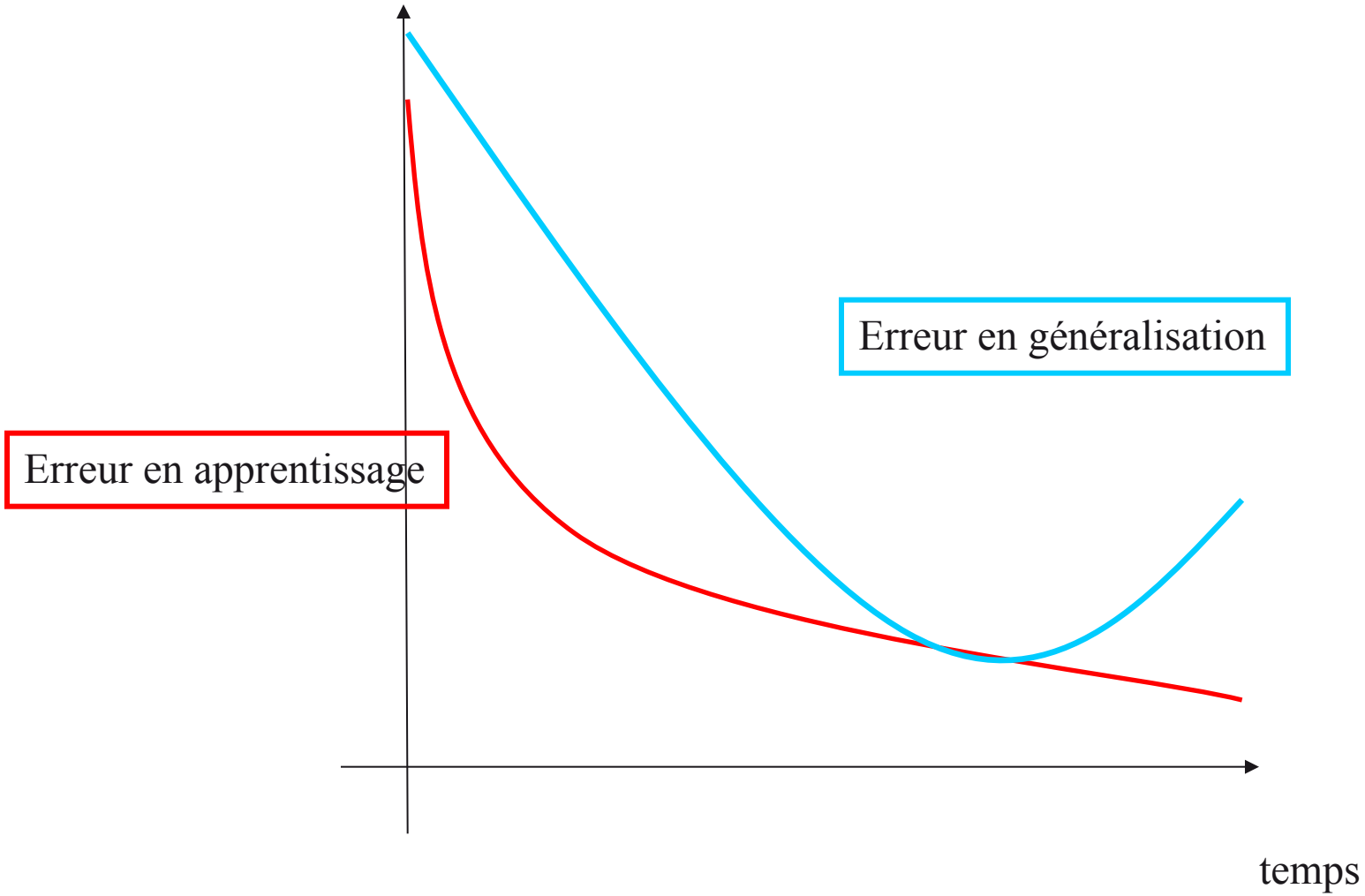
$$\tau_G = \frac{\text{Nombre d'individus bien classés}}{R_G}$$

- **Erreur quadratique**

Erreur quadratique en fonction du temps (époque)



Erreur quadratique : surapprentissage



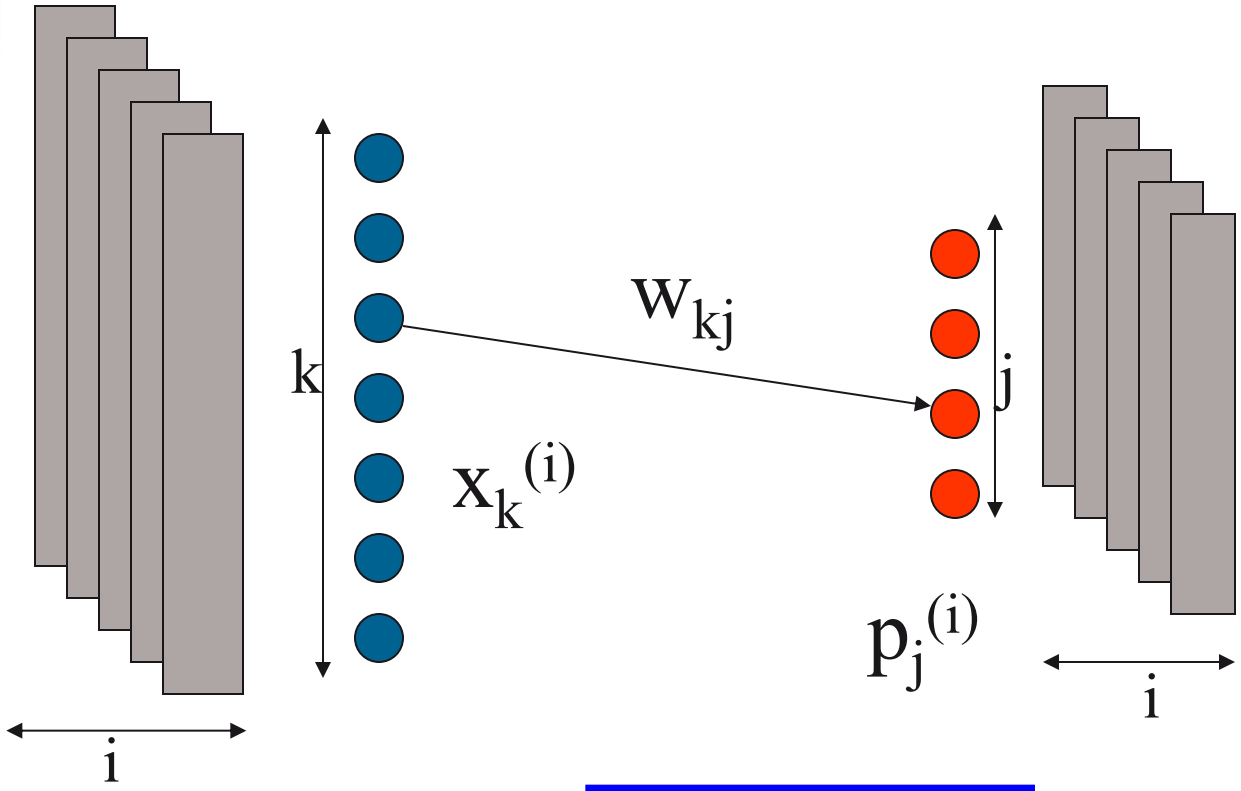
Algorithmes d'apprentissage

Cas monocouche

Apprentissage

- **Principe de Hebb**
  - Principe sur lequel repose les algorithmes d'apprentissage des réseaux de neurones
  - La modification du réseau pendant l'apprentissage consiste uniquement en la modification des poids de connexion
- Perceptrons (simples) : **règle du delta ou règle de Widrow-Hoff**
  - Principe d'optimisation: trouver les poids de connexion qui minimisent l'erreur quadratique de classification sur la base d'apprentissage

Règle Delta – perceptron monocouche



$$a_j = \sum_{k=1}^{N} w_{kj} x_k$$

$$s_j = f(a_j)$$

Règle delta – problématique

- **Coût aux moindres carrés : on prend tous les échantillons**

$$J = \sum_{j=1}^{c} \left(f\left(\sum w_{kj} x_k \right) - d_j \right)^2$$

Cas général : la fonction d'activation f doit être dérivable
Pour le perceptron (fonction seuil non dérivable) => utiliser la fonction sigmoïde

- **Fonction quadratique dérivable : on sait dériver et exprimer le gradient**

Calcul de la dérivée de l'erreur quadratique

$$J = \sum_{j=1}^{c} \left(f\left(\sum w_{kj} x_k \right) - d_j \right)^2$$

$$a_j = \sum_{k=1}^{N} w_{kj} x_k$$

$$s_j = f\left(a_j\right)$$

$$\frac{\partial J}{\partial w_{kj}} = \sum_{j'=1}^{c} \frac{\partial}{\partial w_{kj}} \left(s_{j'} - d_{j'} \right)^2$$

$$= \sum_{j'=1}^{c} \frac{\partial}{\partial s_j} \left(s_{j'} - d_{j'} \right)^2 \frac{\partial s_{j'}}{\partial w_{kj}} = \frac{\partial}{\partial s_j} \left(s_j - d_j \right)^2 \frac{\partial s_j}{\partial w_{kj}}$$

$$= 2\left(s_j - d_j \right) \frac{\partial f\left(a_j\right)}{\partial w_{kj}} = 2\left(s_j - d_j \right) f'\left(a_j\right) \frac{\partial a_j}{\partial w_{kj}}$$

$$a_j = \sum_{k=1}^{N} w_{kj} x_k$$

$$s_j = f(a_j)$$

$$\frac{\partial a_j}{\partial w_{kj}} = x_k$$

$$\frac{\partial J}{\partial w_{kj}} = 2(s_j - d_j) f'(a_j) \frac{\partial a_j}{\partial w_{kj}}$$

$$= 2(s_j - d_j) f'(a_j) x_k$$

- **Cas où f est l'identité**

$$\frac{\partial J}{\partial w_{kj}} = 2(s_j - d_j) x_k$$

Minimiser en inversant le système ???

$$J(W) = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k} \right)^2$$

$$\frac{\partial J(W)}{\partial w_{ij}} = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k} \right) w_{ij} X_{k,j}$$

Cas où la fonction d'activation est la fonction identité

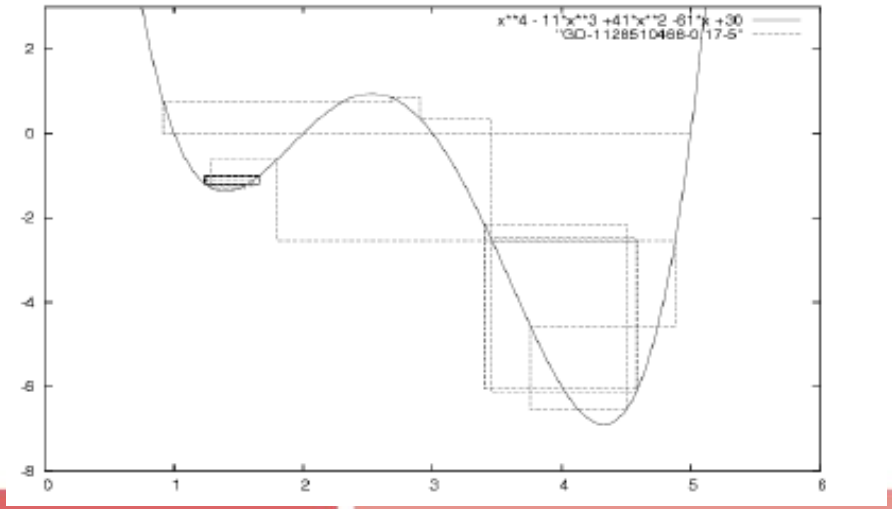
- **Trouver les wij qui annulent la dérivée**
- **Si vecteur d'état de dimension N, alors NxN coefficients à rechercher**
- **Généralement mal conditionné**
- **Résultats parfois discutables**

Règle delta – descente du gradient

- **Algo de descente du gradient**
- **Correction des poids w_k en fonction du gradient**

$$w^{(i+1)}{}_{kj} = w^{(i)}{}_{kj} + \Delta w_{kj}$$

$$\Delta w_{kj} = -\eta \frac{\partial J}{\partial w_{kj}}$$



Règle delta

$$\frac{\partial J(W)}{\partial w_{ij}} = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k}\right) w_{ij} X_{k,j}$$

- Initialiser les poids
- Modifier itérativement les poids si la sortie n'est pas égale à la sortie désirée

Comment modifier les poids, à chaque boucle => algo de descente du gradient

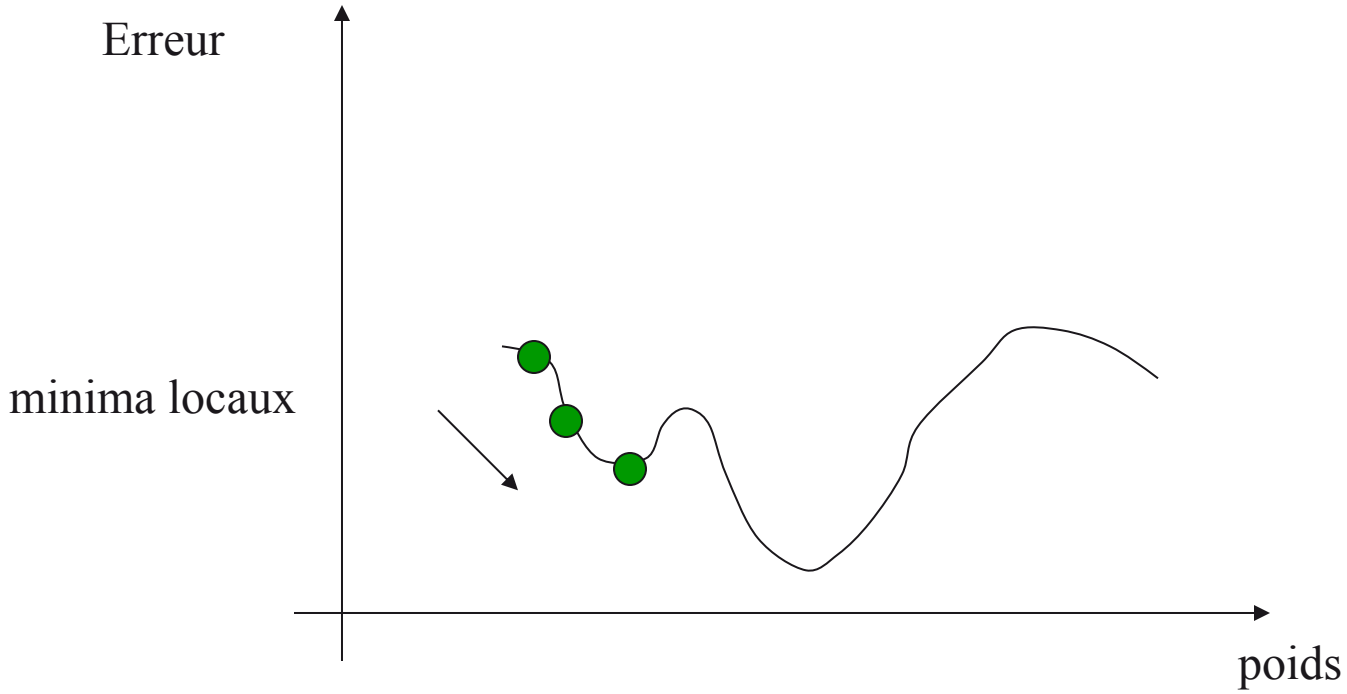
$$W(t+1) = W(t) + \eta\left(d_X - W^t(t) X\right) X$$

Modalités opératoires pour l'apprentissage

- **2 possibilités:**
  - 1e possibilité : Apprentissage global:
    - on calcule l'erreur quadratique moyenne sur l'ensemble d'apprentissage, on calcule le gradient et on corrige les poids avec la règle delta
    - On parcourt à nouveau l'ensemble d'apprentissage
    - => inconvénient: débouche sur des minima locaux

$$J = \sum_{j=1}^{c} \left(f\left(\sum w_{kj} x_k \right) - d_j \right)^2$$

Problèmes



Modalités opératoires pour l'apprentissage

- **2 possibilités:**

  - 2e possibilité : Apprentissage échantillon par échantillon:
    - On choisit un échantillon p dans la base d'apprentissage
    - On évalue son erreur quadratique et on calcule le gradient

$$J = (s_p - d_p)^2$$

$$\frac{\partial J}{\partial w_{kj}} = 2(s_j - d_j) f'(a_j) x_k$$

    - On modifie les poids selon la règle delta
    - On passe à l'échantillon suivant

$$\Delta w_{kj} = -\eta \frac{\partial J}{\partial w_{kj}}$$

Algorithme associé

- **Elaboration de la base d'exemples/apprentissage**
- **Définition de la structure**
- **Initialisation des poids (ex: valeurs aléatoires)**

1. **Prendre la base d'exemple (« époque » n=0)**
   - Sélection d'un exemple et calcul de la sortie
   - Modification des poids
   - Boucler sur les exemples
2. **Calcul de l'erreur J sur la base d'exemple**
3. **J > seuil :**
   - Incrémenter l'époque : n = n+1
   - Si n<Nmax : revenir en 1

$$\Delta w_{kj} = -\eta \frac{\partial J}{\partial w_{kj}}$$

$$\frac{\partial J}{\partial w_{kj}} = 2\left(s_j - d_j\right) f'\left(a_j\right) x_k$$

Règle delta – Cas particuliers

- **Fonction identité** $\Delta w_{kj} = -\eta\left(s_j - d_j\right)x_k$

- **Remarque : Si on prend la fonction seuil dans le cas bi-classes (+1 et -1) on a**

$\left(s_j - d_j\right) =$

– 0 si l'élément est bien classé. Dans ce cas, on ne modifie pas les poids.
– +2 ou -2 si l'élément est mal classé. On a alors :
– +2 si $s_p = 1$ et $d_p = -1$. On a alors

$$w_i' = w_i - 2e_{i,p}$$

– -2 si $s_p = -1$ et $d_p = +1$. On a alors

$$w_i' = w_i + 2e_{i,p}$$

On retrouve la règle du perceptron vue précédemment :

$$\tilde{W}(t+1) = \tilde{W}(t) \pm \eta \tilde{X}$$

Cas particulier de la règle delta pour la fonction seuil – règle du perceptron

- **Elaboration de la base d'exemple**
- **Définition de la structure**
- **Initialisation des poids (aléatoire)**

1. **Prendre la base d'exemple (« époque » n=0)**
   - Sélection d'un exemple et calcul de la sortie
   - Modification des poids
   - Boucler sur les exemples
2. **Calcul de l'erreur E sur la base d'exemple**
3. **E > seuil :**
   - Incrémenter l'époque : n = n+1
   - Si n<Nmax : revenir en 1

$$\Delta w_{kj} = -\eta\left(s_j - d_j\right)x_k$$

Exercice 1 : Apprentissage du perceptron monocouche

Soit un perceptron monocouche ayant comme fonction de transition f, la fonction seuil suivante : si $a > seuil$, $f(a) = 1$, sinon $f(a) = -1$. Nous fixons ici le seuil à 0.2.

Soit la base composée de 4 exemples d'apprentissage :

	e1	e2	d
(1)	1	1	1
(2)	-1	1	-1
(3)	-1	-1	-1
(4)	1	-1	-1

e1 et e2 sont les entrées du réseaux de neurones et d la sortie correspondante, telle qu'étiquetée dans l'ensemble d'apprentissage.

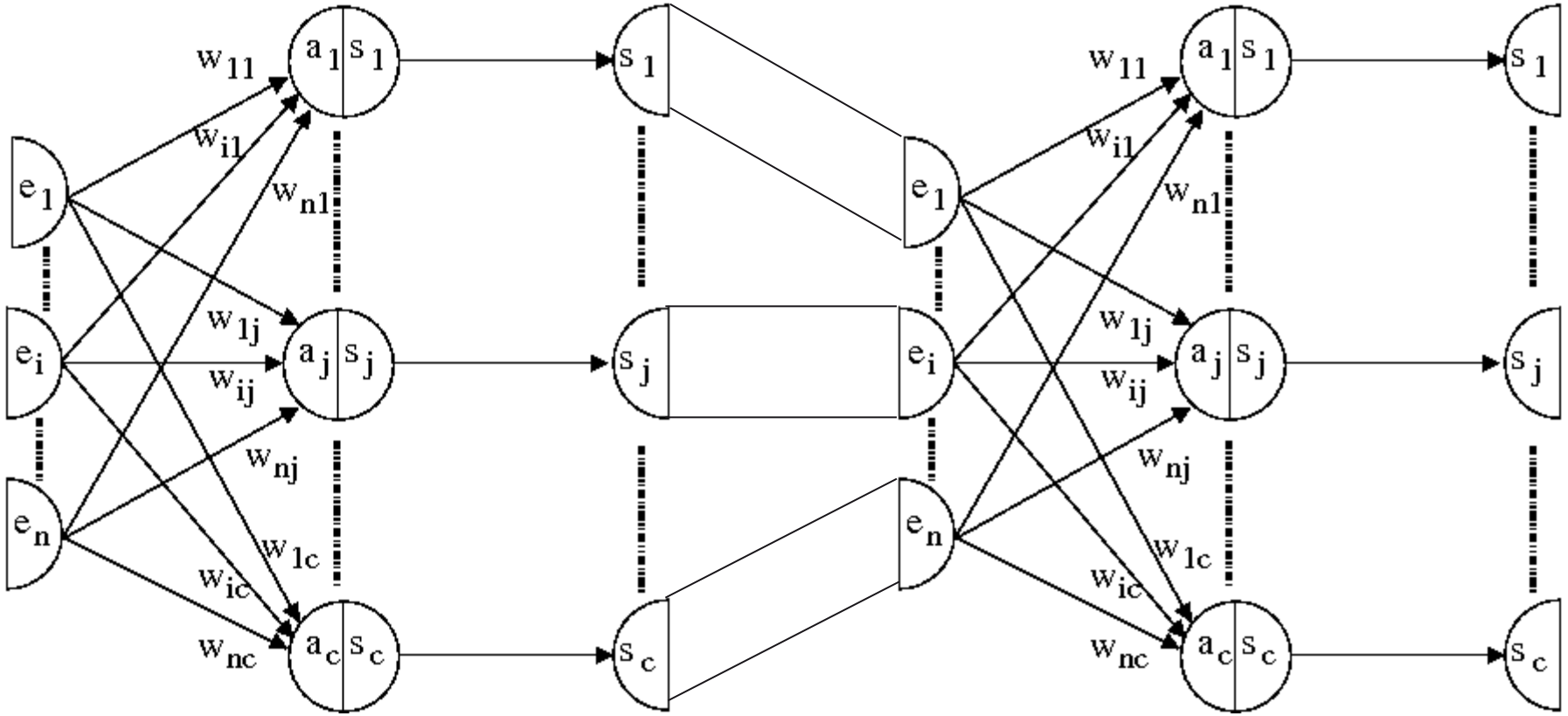
Q. 1 En appliquant la règle delta avec un pas de modification des poids η égal à 0.1, déterminez les poids du perceptron. Vous initialiserez l'algorithme avec les valeurs suivantes , $w1 = -0.2$, $w2 = 0.1$.

Q. 2 Représentez les données d'apprentissage dans un espace à 2 dimensions et tracez la droite séparatrice des 2 classes. Donnez son équation.

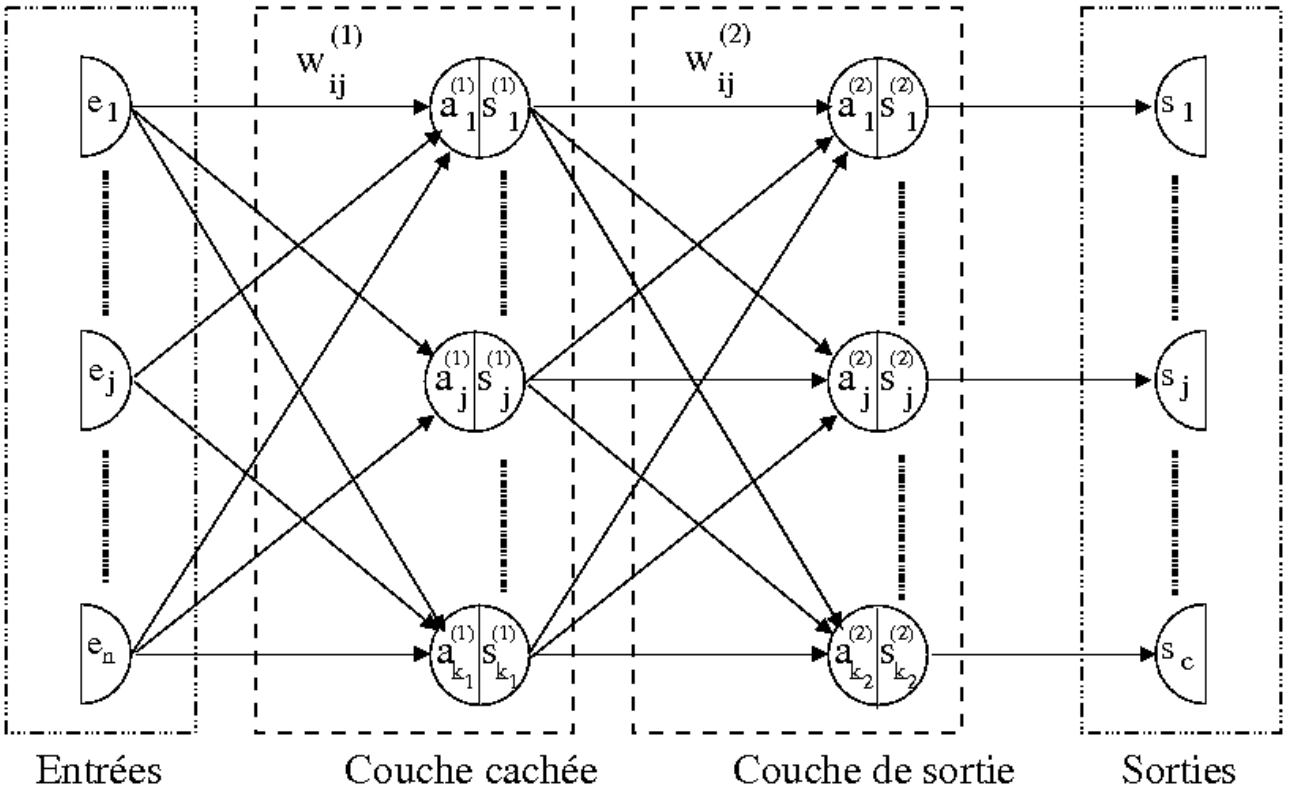
NOUVELLE ARCHITECTURE : LE PERCEPTRON MULTICOUCHE

Perceptron multicouches

- **Objectif : traiter le cas non linéairement séparable**
- **Principe : regrouper les neurones en couches**

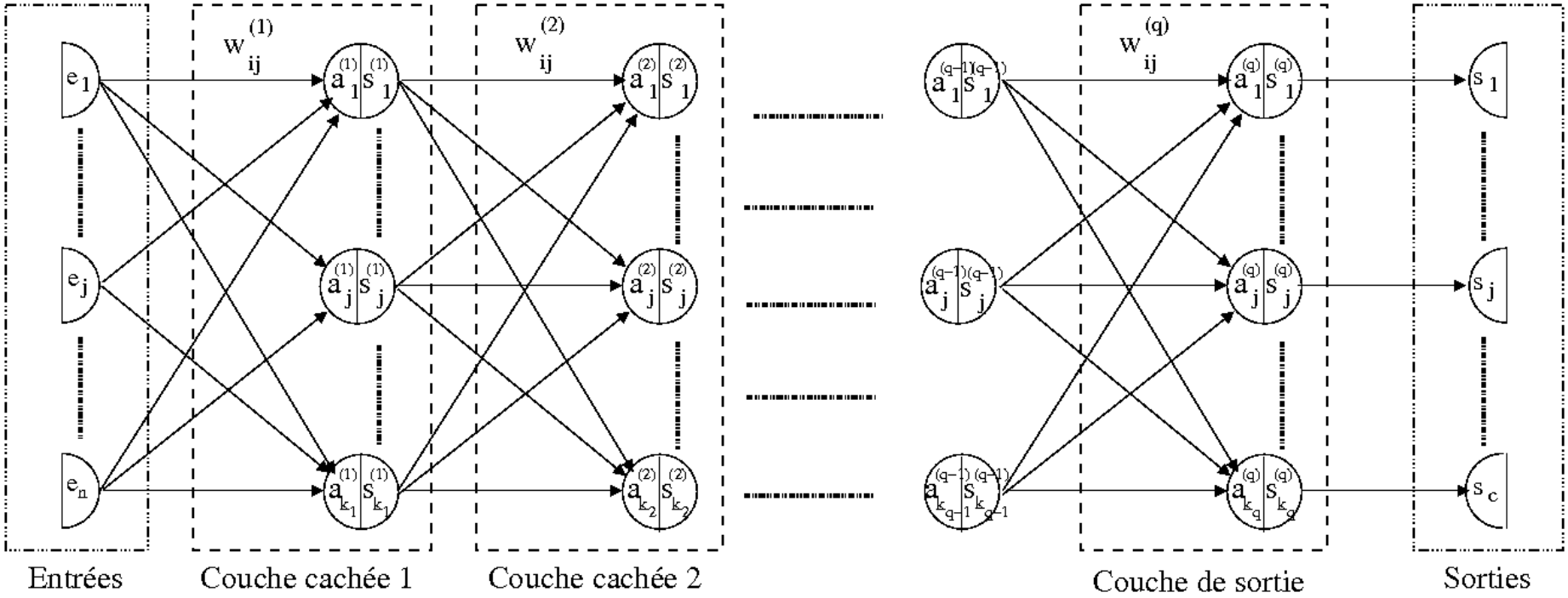


Exemple à deux couches



- **Notion de « couche cachée » :**
  - entre l'entrée et la sortie, couches dont les sorties ne sont pas connues

Réseaux à q couches



Notion de ***propagation*** : il faut calculer toutes les sorties de la couche m-1 pour calculer les sorties de la couche m

Classification « non linéaire » possible

- **Le cas du « et »:**
  - Linéairement séparable

0	1
0	0

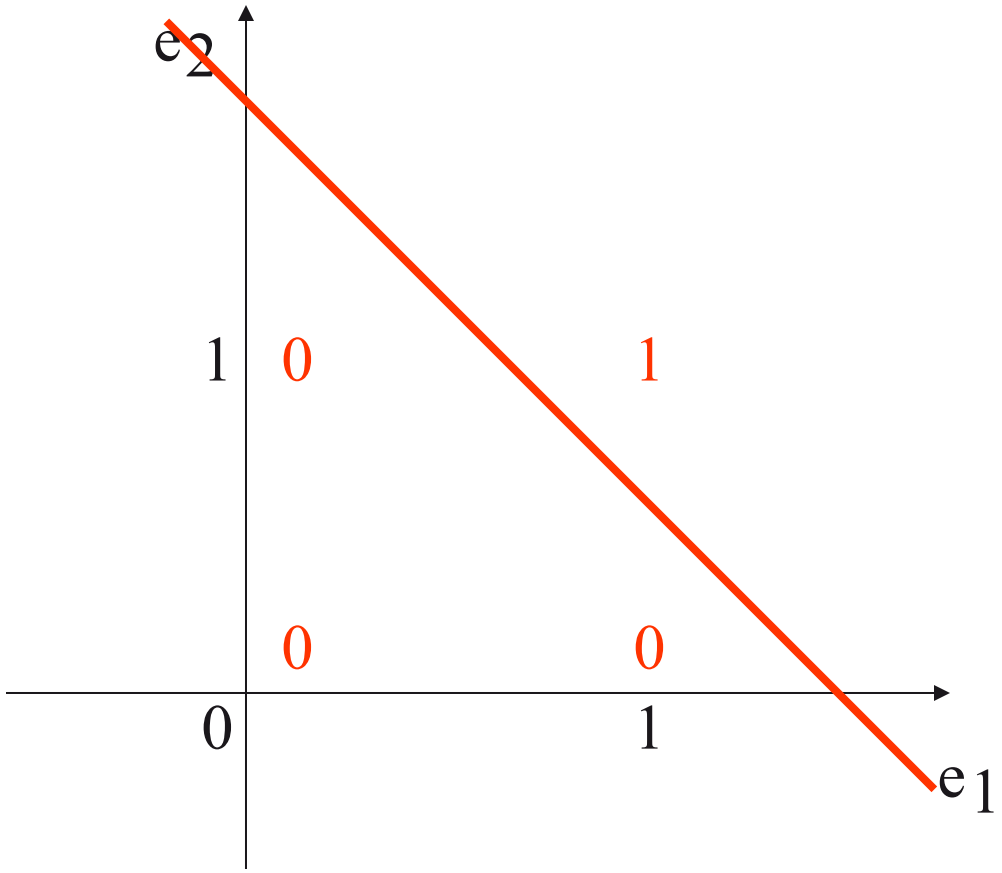
- **Le cas du « ou exclusif »**
  - Pas de solution par le perceptron
  - Rôle du « multicouche »

1	0
0	1

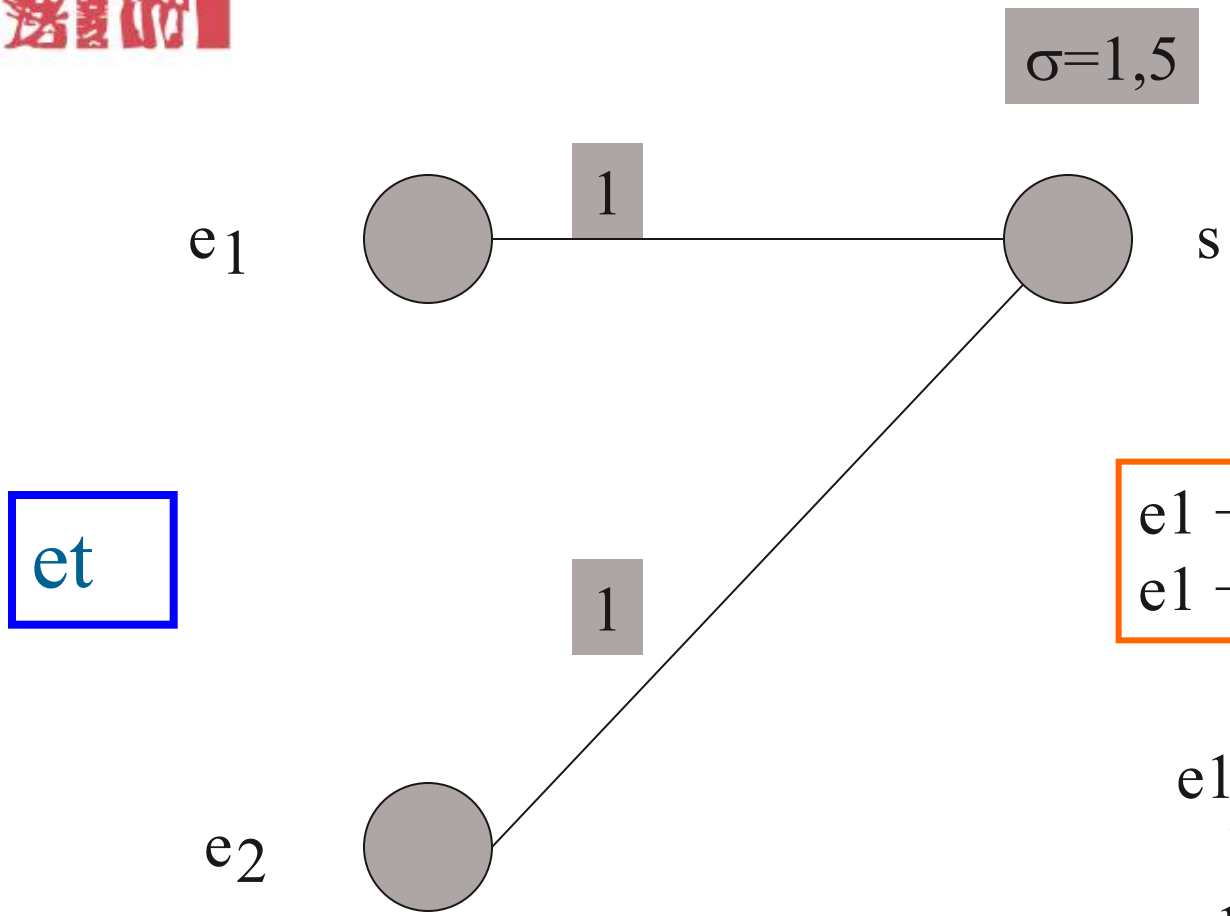
Cas d'école : le « et »

0	1
0	0

et

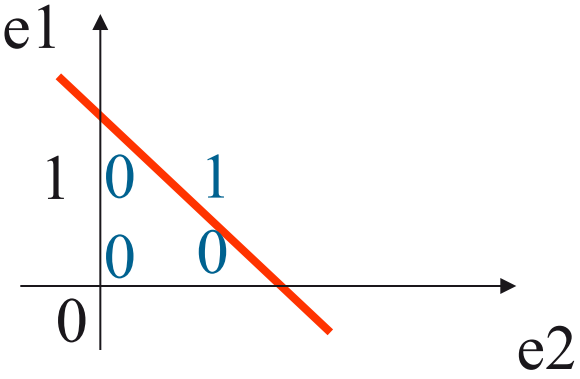


Séparatrice : pente=-1



e1 + e2 – 1,5 > 0 → 1
e1 + e2 – 1,5 < 0 → 0

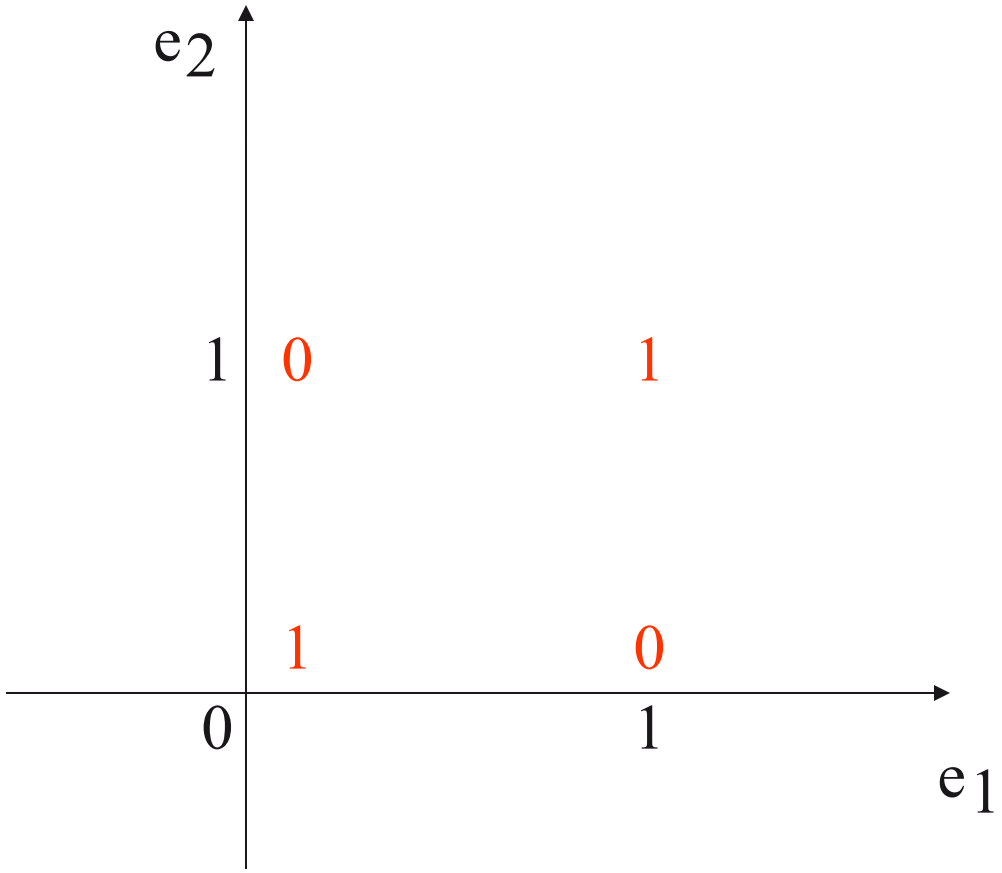
f : fonction à seuil
sur [0 … 1]



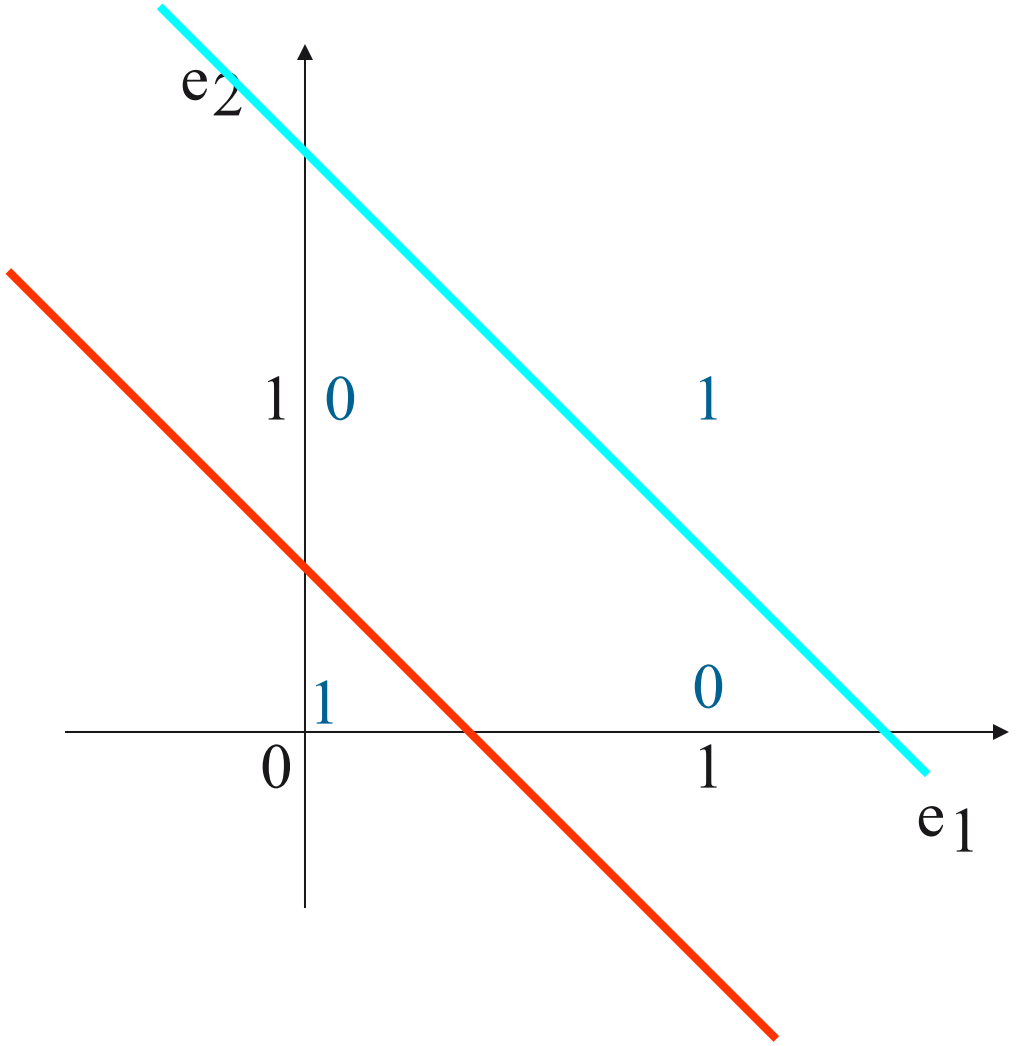
Cas d'école : le « ou exclusif »

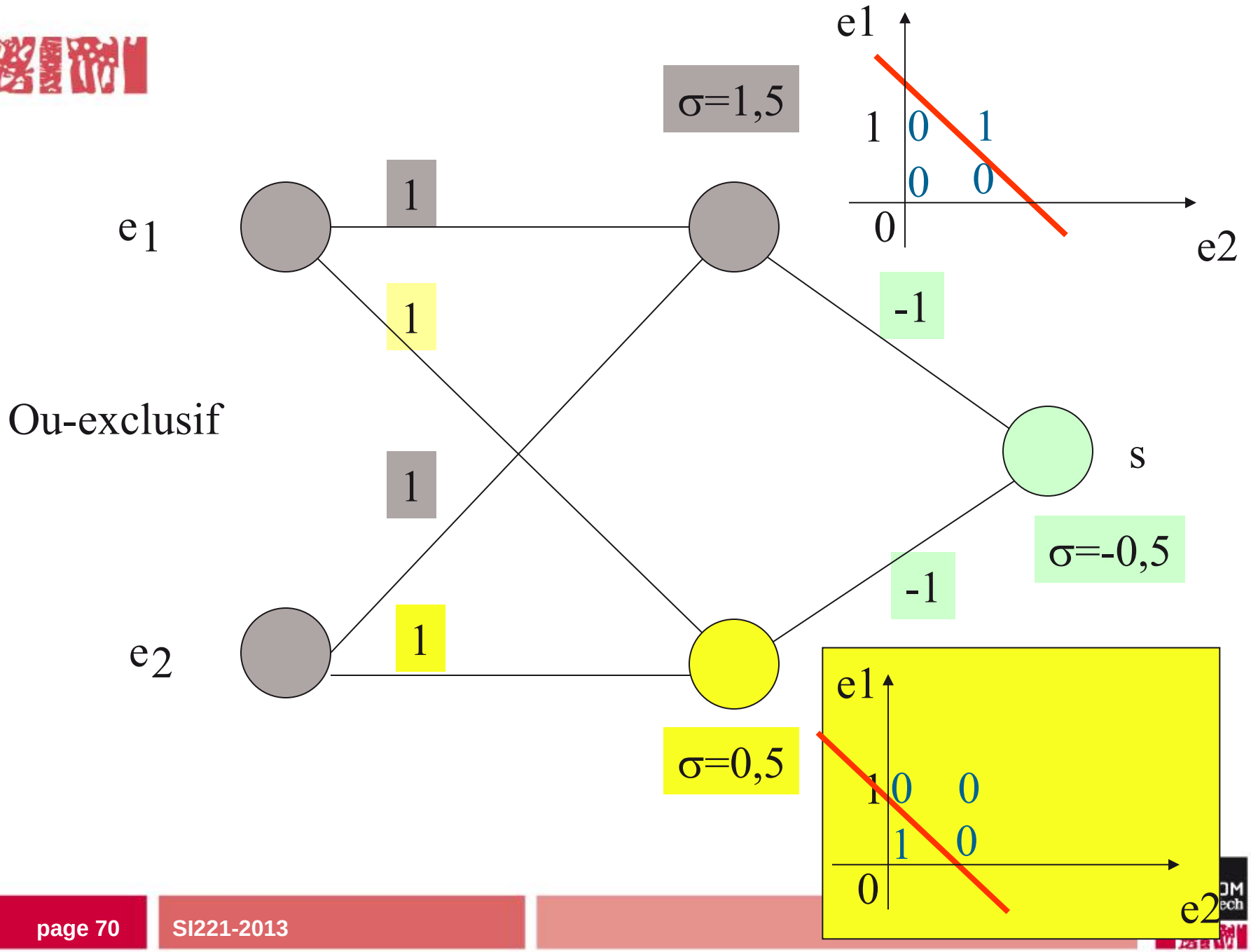
1	0
0	1

ou



Ou-exclusif




Ou-exclusif
e1
e2
1
1
1
1
σ=1,5
-1
-1
s
σ=-0,5
σ=0,5
e1
e2
1
0
1
0
0
0
e1
e2
1
0
0
1
0
0

ALGORITHME D'APPRENTISSAGE - CAS DU PERCEPTRON MULTICOUCHE

Algorithme de rétropopagation du gradient

- **Principe :**
  - L'erreur se propage de droite à gauche dans le réseau

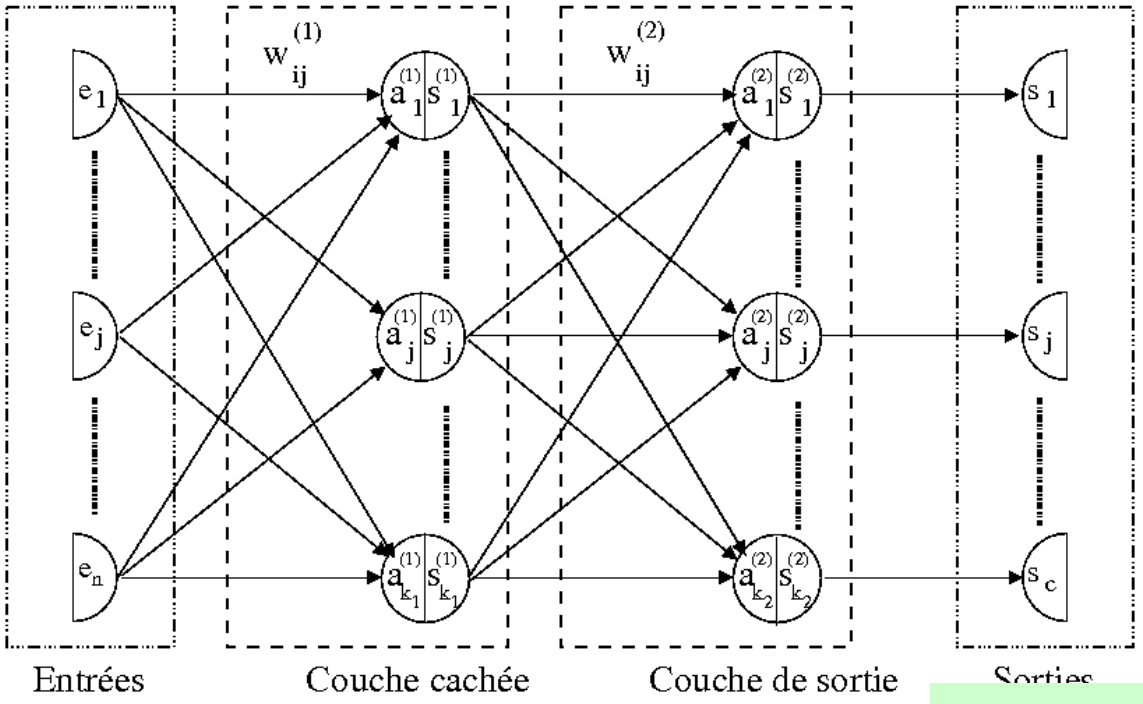
rétropropagation du gradient (cas 2 couches)

$$a_l^{(2)} = ?$$

$$s_l = ?$$

$$a_j^{(1)} = ?$$

$$s_j^{(1)} = ?$$



n

k_1 unités de la couche 1

$k_2 = c$

l=1:c, nombre de classes

Même fonction d'activation pour tous les neurones de toutes les couches

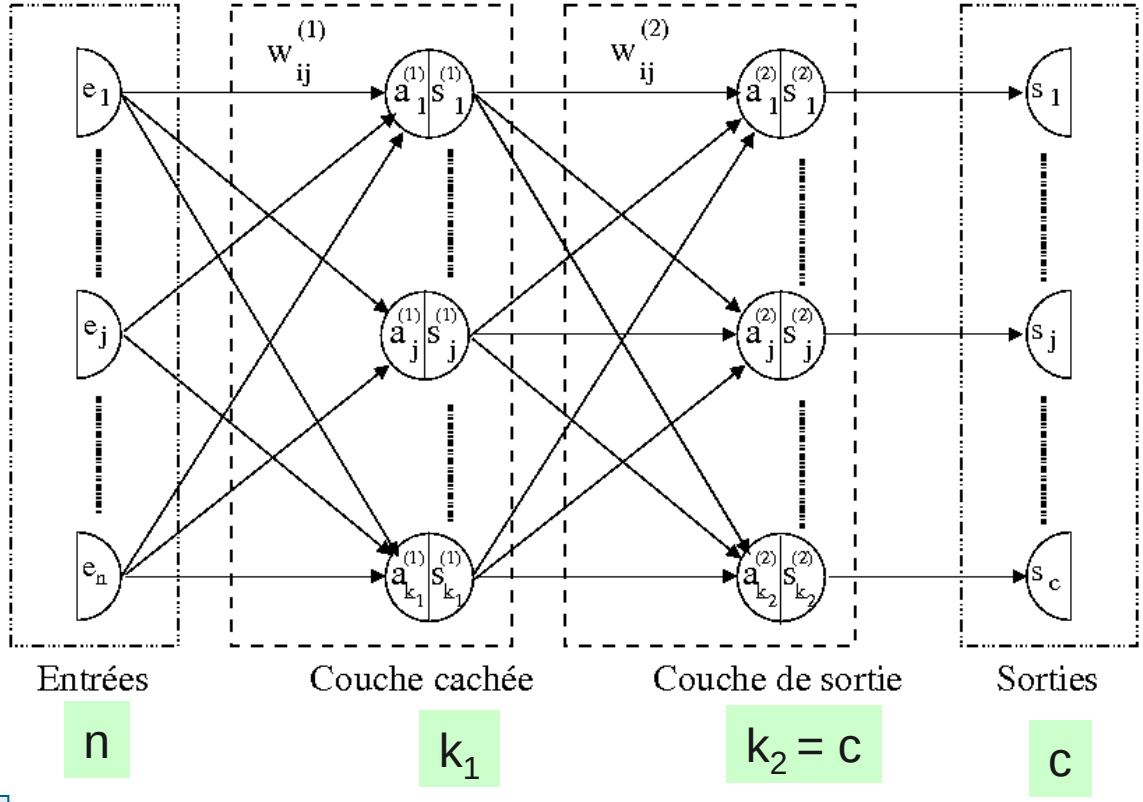
rétropropagation du gradient (cas 2 couches)

$$a_l^{(2)} = \sum_{j=1}^{k_1} w_{jl}^{(2)} s_j^{(1)}$$

$$s_l = f\left(a_l^{(2)}\right)$$

$$a_j^{(1)} = \sum_{k=1}^{n} w_{kj}^{(1)} e_k$$

$$s_j^{(1)} = f\left(a_j^{(1)}\right)$$



Même fonction d'activation pour tous les neurones de toutes les couches

Pour la dernière couche (2e couche)

- **Dériver en fonction du poids $w_{jl}^{(2)}$**

$$\frac{\partial E_i}{\partial w_{jl}^{(2)}} = 2\left(s_l - d_l\right) f'\left(a_l^{(2)}\right) s_j^{(1)}$$

l =1:c sorties (nombre de classes)
j=1:k1 unités de la 1e couche
$w_{jl}^{(2)}$ les poids de la 2e couche

- **Modification des poids $w_{jl}^{(2)}$ prenant en compte la direction du gradient pour minimiser l'erreur quadratique**
- **Calcul de l'erreur quadratique avec ces nouveaux poids et dériver en fonction de $w_{ij}^{(1)}$**
- **Modification des poids $w_{ij}^{(1)}$ prenant en compte la direction du gradient pour minimiser l'erreur quadratique**

Pour la couche cachée

$$s_l = f\left(\sum_{j=1}^{k_1} w_{jl}^{(2)} f\left(a_j^{(1)}\right)\right) = f\left(\sum_{j=1}^{k_1} w_{jl}^{(2)} f\left(\sum_{k=1}^{n} w_{kj}^{(1)} e_k\right)\right)$$

- **Dérivation des fonctions composées**
- **Possible puisque la sigmoïde est dérivable !!**

$$\frac{\partial E_i}{\partial w_{kj}^{(1)}} = 2e_k f'\left(a_j^{(1)}\right)\sum_{l=1}^{c}\left(w_{jl}^{(2)}\left(s_l - d_l\right) f'\left(a_l^{(2)}\right)\right)$$

- **On rétropropage le gradient !!!**

Base d'apprentissage

- **Sert à modifier les poids du réseau**
- **On présente les échantillons dans un ordre aléatoire**
  - Calcul du gradient de la dernière couche
  - Rétropropagation pour les couches cachées
  - Modification des poids
- **Critère d'arrêt**
  - Nombre d'époques

Caractérisation de l'architecture

- **Utilisation de la base de généralisation**
- **Deux sortes d'erreur**
  - Erreur de classification
  - Erreur quadratique

Conclusion,

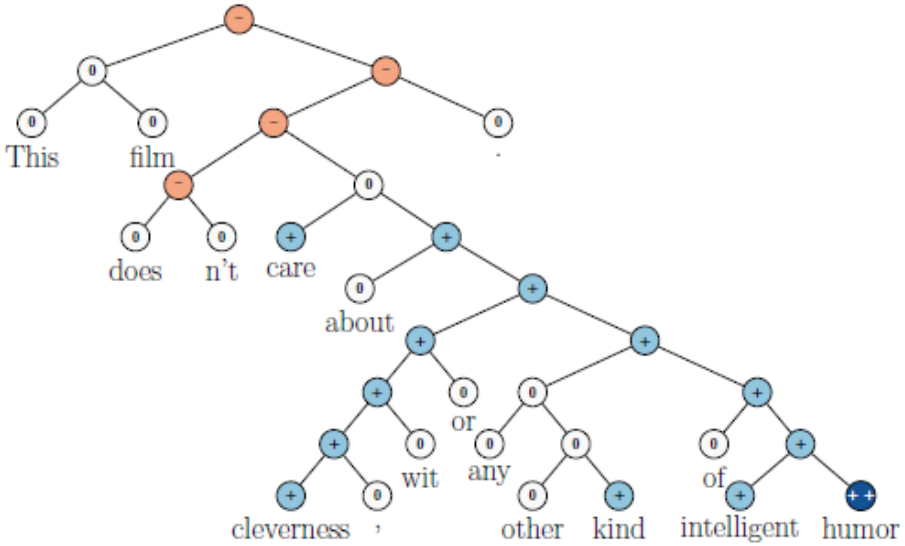
- **Domaine « mature »**
- **Des plateformes accessibles :**
  - Outils commerciaux :
    - Matlab : « neural networks » toolbox http://www.mathworks.com/products/neuralnet/
  - Outils open-source / gratuits :
    - JOONE : bibilothèque JAVA open source (licence LGPL) http://www.jooneworld.com/
    - Scilab : ANN toolbox http://www.scilab.org/

Pour aller plus loin

- **Les réseaux récurrents (Hopfield, 1982)**
  - Idéal pour la reconnaissance de séquences (reconnaissance de la parole , traitement du langage naturel)
  - La sortie à l'instant n-1 est utilisée pour calculer l'activation à l'instant n
- **L'apprentissage par renforcement (Sutton & Barto, 1998)**
  - Apprentissage dit "qualitatif" par pénalité/récompense.
  - Utilisé par exemple pour l'apprentissage de stratégies de dialogue human-machine
- ***Deep learning voir***
  http://neuralnetworksanddeeplearning.com/index.html

Réseaux de neurones et deep learning

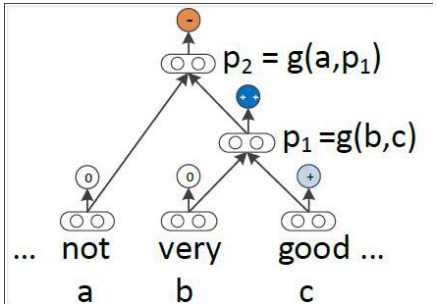
- **Remise au goût du jour des réseaux de neurones avec l'émergence du deep learning**
  - Utilisation des réseaux récursifs tensoriels
    - permettent de prendre en compte la structure d'une phrase.



- exemple d'utilisation des réseaux récurrents
  - REF : R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Stroudsburg, PA : Association for Computational Linguistics, October 2013, pp. 1631? 1642.

Réseaux de neurones et deep learning

- Utilisation des réseaux récursifs tensoriels
  - Représentation de la phrase par un arbre (utilisation du parseur de Stanford)
  - On applique les récursivement les fonctions d'activation:



  - Apprentissage : apprentissage de la fonction g du passage au parent dans l'arbre binaire représentation la phrase

Annexes