Réseaux de neurones

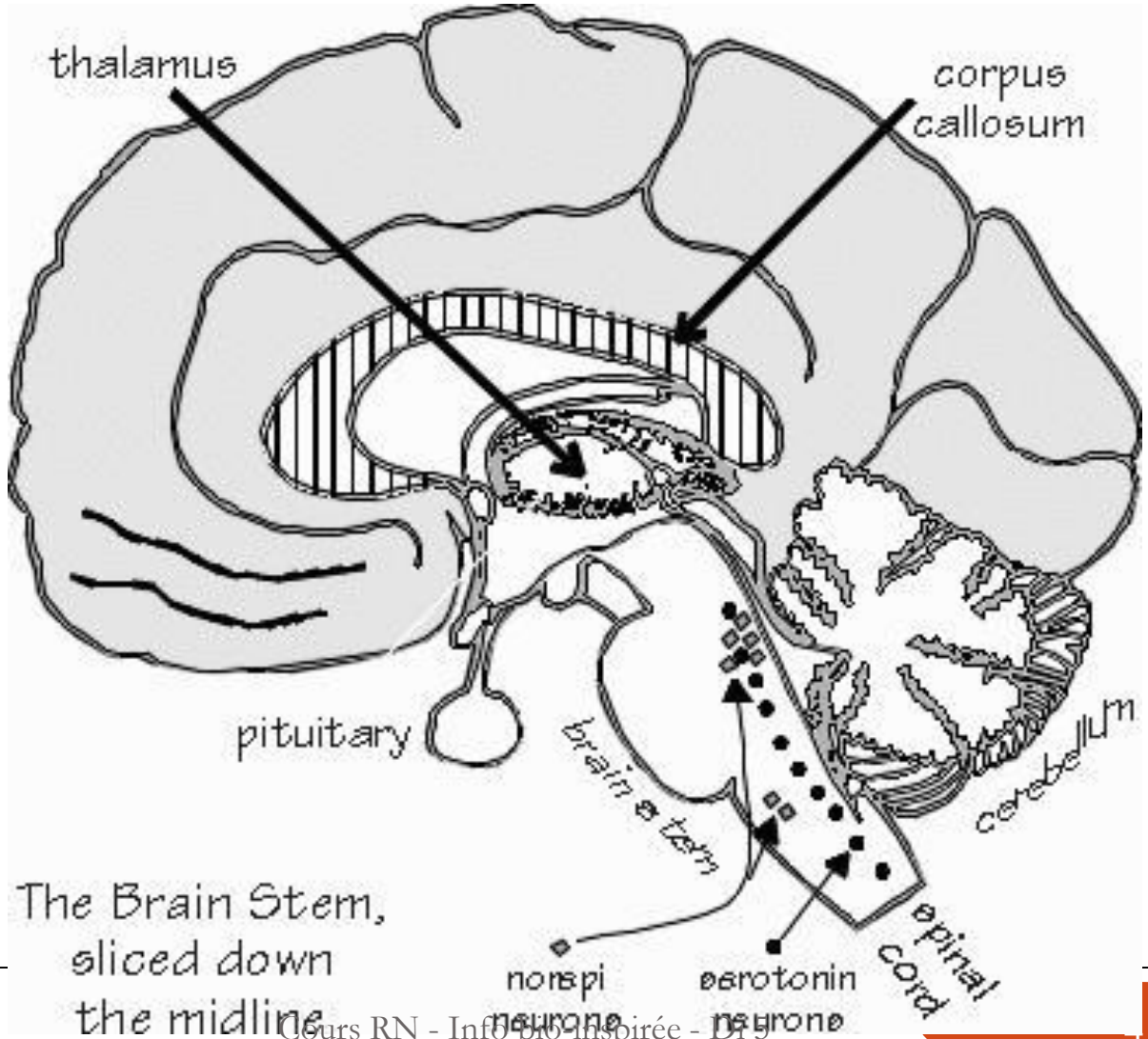
Histoire

- 1940 : La machine de Turing
- 1943 : Le neurone formel (McCulloch & Pitts)
- 1948 : Les réseaux d'automates (Von Neuman)
- 1949 : Première règle d'apprentissage (Hebb)
- 1958-62 : Le perceptron (Rosenblatt)
- 1960 : L'adaline (Widrow & Hoff)
- 1969 : *Perceptrons* (Minsky & Papert)
  ➔ les limites du Perceptron
  ➔ besoin d'architectures + complexes,
  Comment effectuer l'apprentissage ? On ne sait pas !

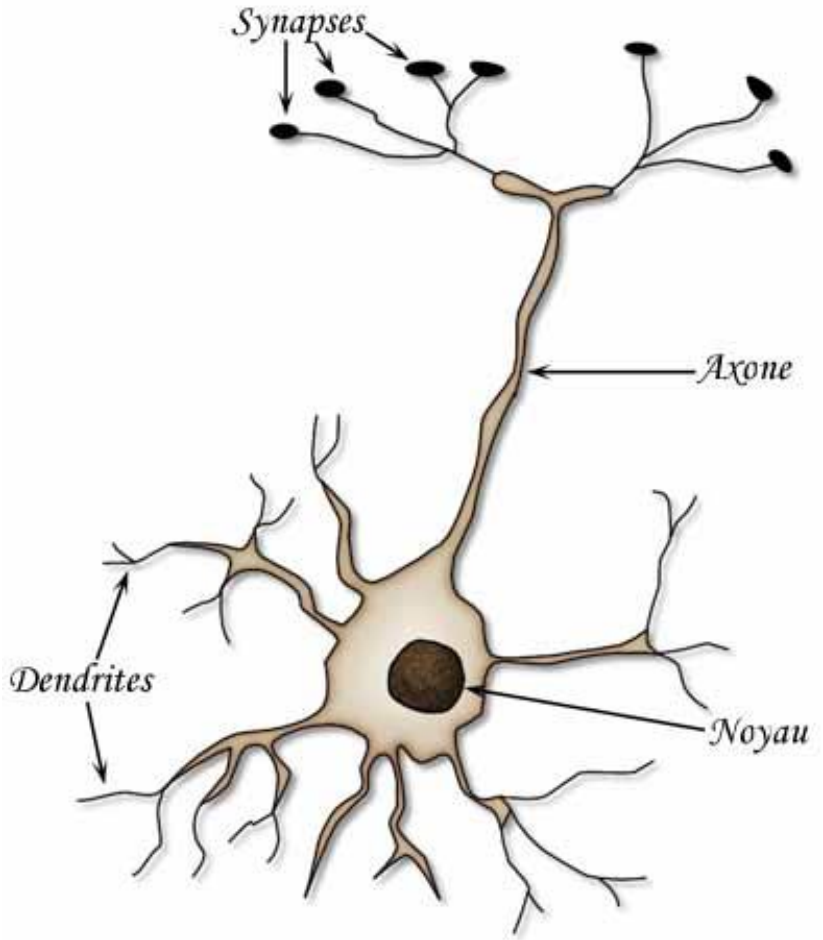
Histoire

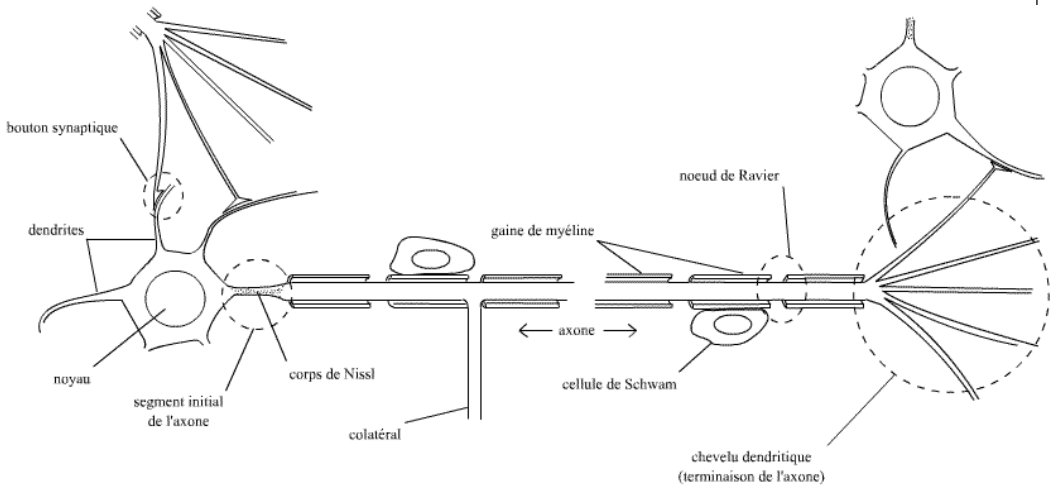
- 1974 : Rétropropagation (Werbos)
  ➔ pas de succès !?!?
- 1986 : Rétropropagation (Rumelhart & McClelland)
  ➔ nouvelles architectures de Réseaux de Neurones
  ➔ applications :
  - reconnaissance de l'écriture
  - reconnaissance/synthèse de la parole
  - vision (traitement d'images)
- 2006 : Deep Neural Network
  - Nouvelles architectures neuronales : DBN, CNN…

Le cerveau

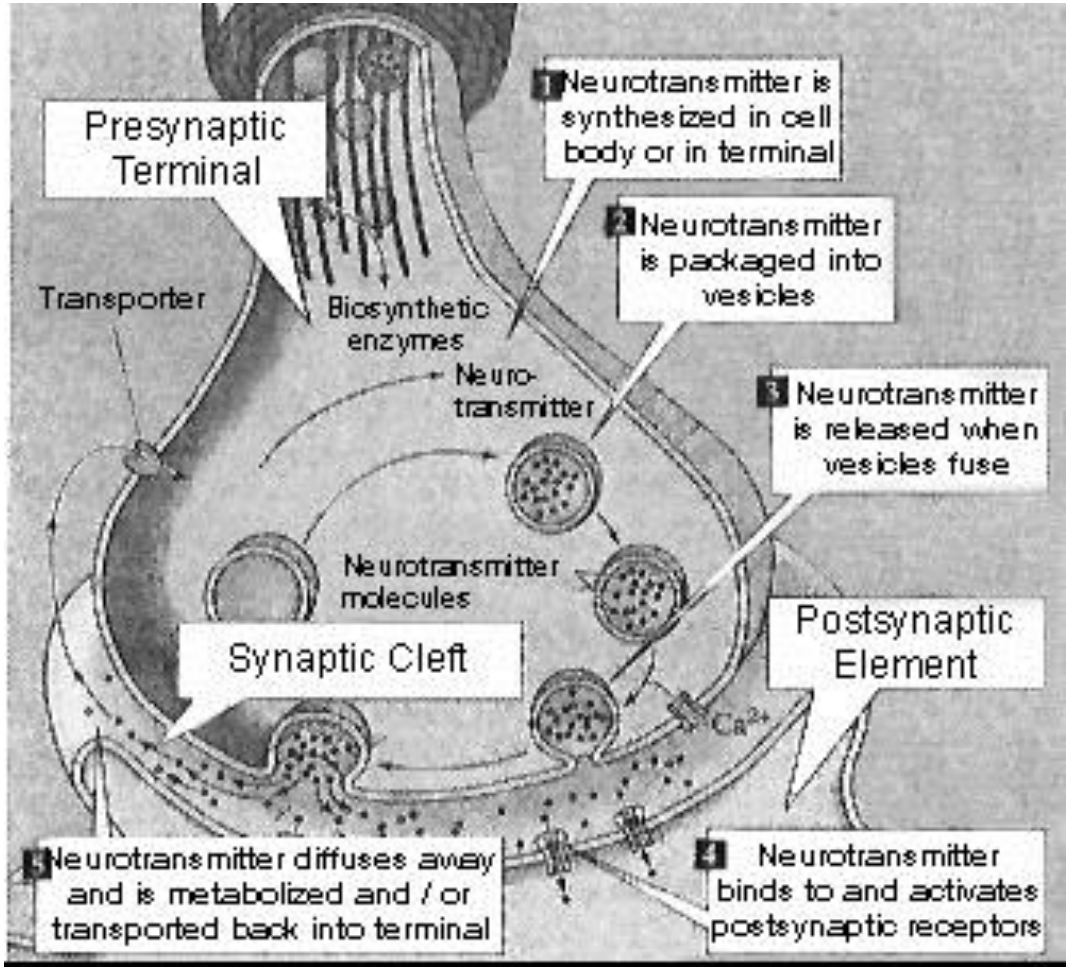

thalamus
corpus callosum
pituitary
brain stem
cerebellum
spinal cord
noradrenaline neurone
serotonin neurone
The Brain Stem, sliced down the midline

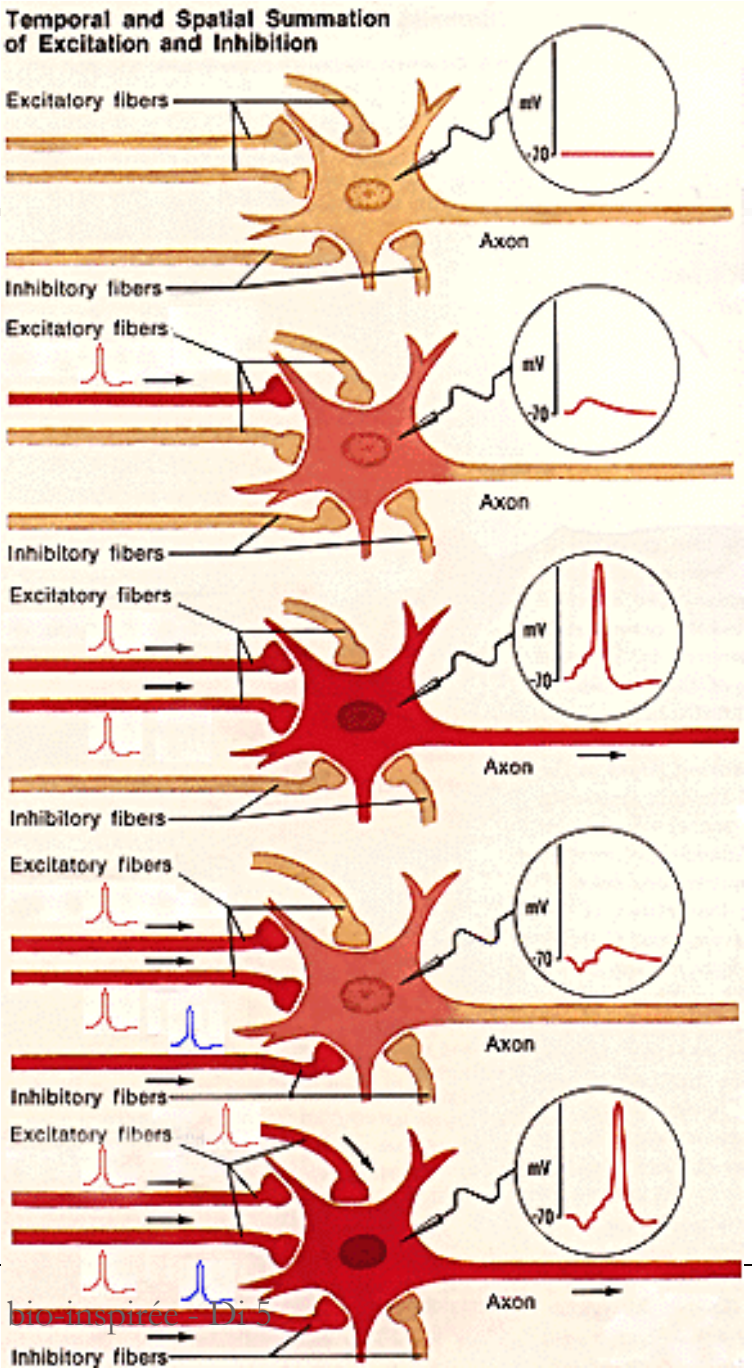
Le neurone biologique



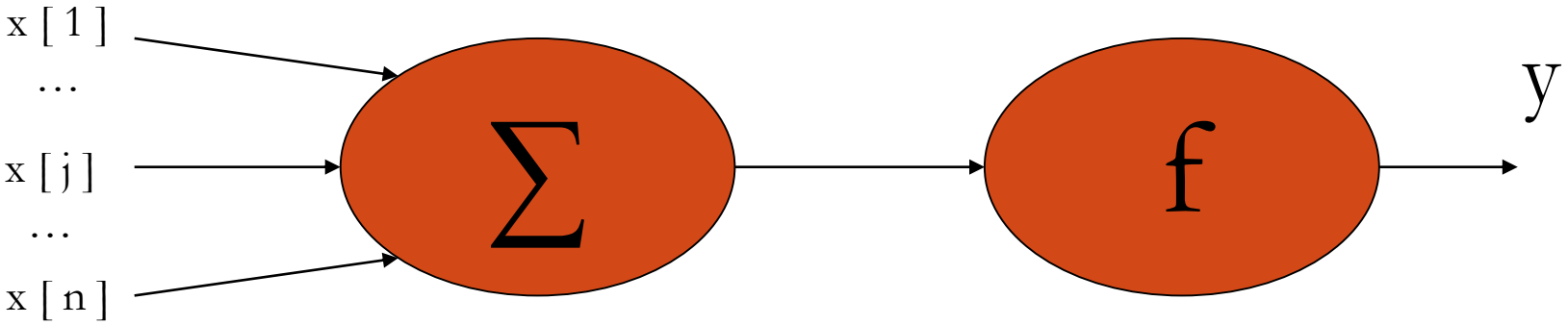


La synapse


Presynaptic Terminal
1 Neurotransmitter is synthesized in cell body or in terminal
2 Neurotransmitter is packaged into vesicles
Transporter
Biosynthetic enzymes
Neuro-transmitter
3 Neurotransmitter is released when vesicles fuse
Neurotransmitter molecules
Synaptic Cleft
Postsynaptic Element
Ca^{2+}
5 Neurotransmitter diffuses away and is metabolized and / or transported back into terminal
4 Neurotransmitter binds to and activates postsynaptic receptors
Components of a Synapse


Temporal and Spatial Summation
of Excitation and Inhibition
Excitatory fibers
Inhibitory fibers
Axon
mV
-70

Le neurone formel

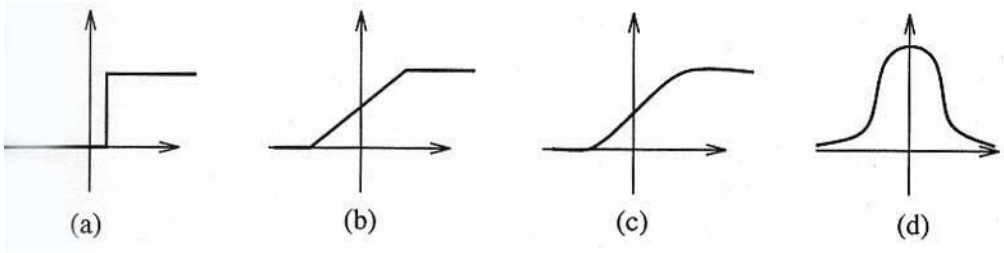


$$y = f\left(\sum_{j=1}^{n} x[j].w[j] + \theta\right)$$

$$y = f\left(\sum_{j=0}^{n} x[j].w[j]\right) \quad avec \quad \begin{cases} x[0] = 1 \\ w[0] = \theta \end{cases}$$

Fonction d'activation du neurone

- Exemples de f ( )



- Sigmoïde : monotone, différentiable, lisse…
- 1/(1+exp(-bx)), tanh(x), 2/pi . Arctan(bx)

- Nous en verrons d'autres plus tard…

Architecture

Type d'architectures

Dynamique de fonctionnement

Apprentissage

Types d'architectures

- RN à couches
- RN récurrents

Dynamique de fonctionnement

- initialisation des neurones d'entrée
- fonctionnement du RN
- état des neurones de sortie

Si RN à couches : propagation de la couche d'entrée vers la couche de sortie

Si RN récurrent : les états des neurones évoluent au cours du temps jusqu'à convergence vers un état stable s'il existe ($S_i(t+1) \approx S_i(t)$)

M-à-j : synchrone, asynchrone (aléatoire, prédéfini)

Apprentissage

- En général, on présente des exemples et on modifie les poids en fonction des sorties obtenues
  - Supervisé : minimise écart entre sortie obtenue et sortie désirée
  - Renforcé : pénalité / récompense
  - Non supervisé : regroupement des exemples en fonction de ressemblance que le RN doit extraire

- http://www.asimovinstitute.org/neural-network-zoo/

Perceptron

- RN à une seule couche de poids (pas de couche cachée)
  fonction d'activation échelon : sign ( )

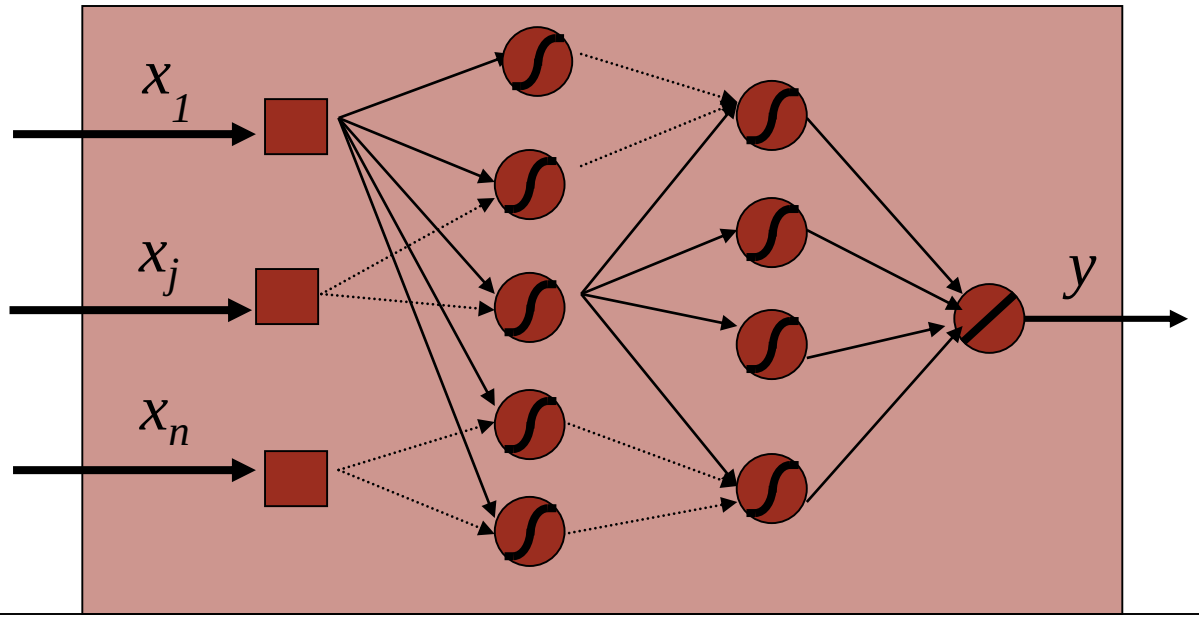
$$y_i = f\left(\sum_{j=0}^{n} w_{ij} x_j\right)$$

$$\Delta w_{ij} = \begin{cases} \eta d_i^k x_j^k & \text{si } d_i^k \neq y_i^k \\ 0 & \text{sinon} \end{cases}$$

- Mais XOR

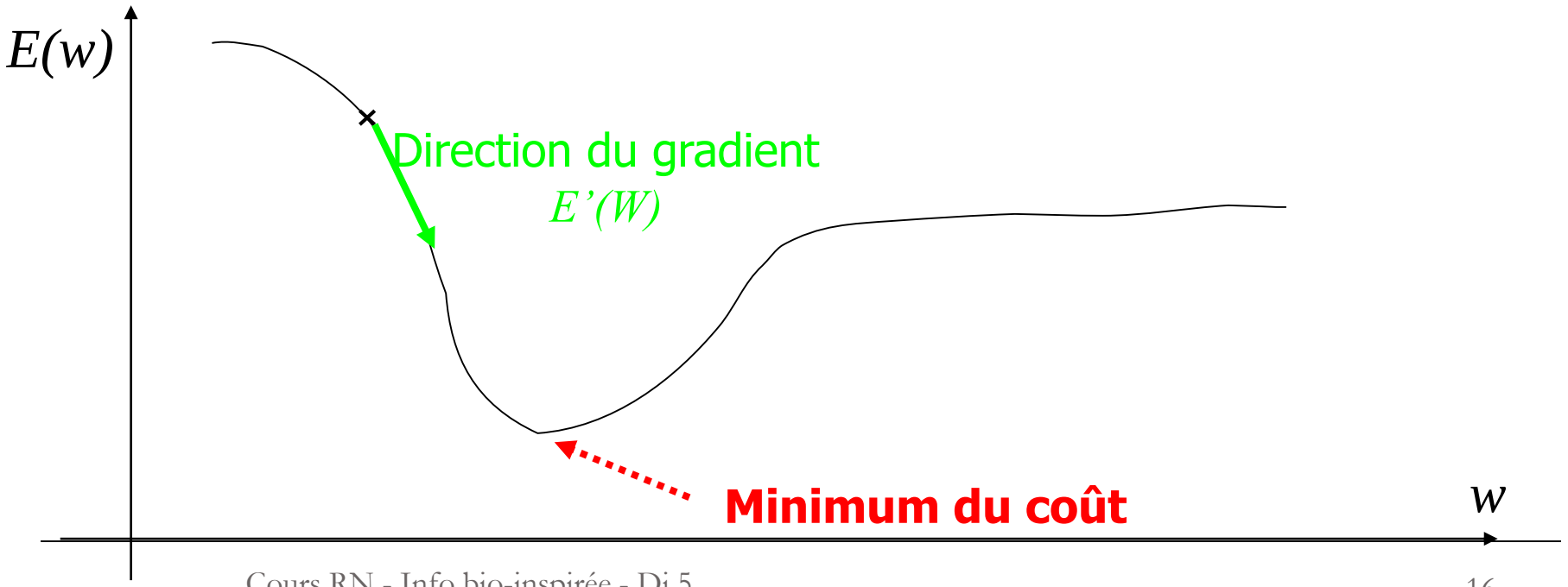
Perceptron multicouche (MLP)

- RN à plusieurs couches de poids (une ou deux couches cachées)
- fonction d'activation : souvent sigmoïde
  Connexions complètes entre couches successives

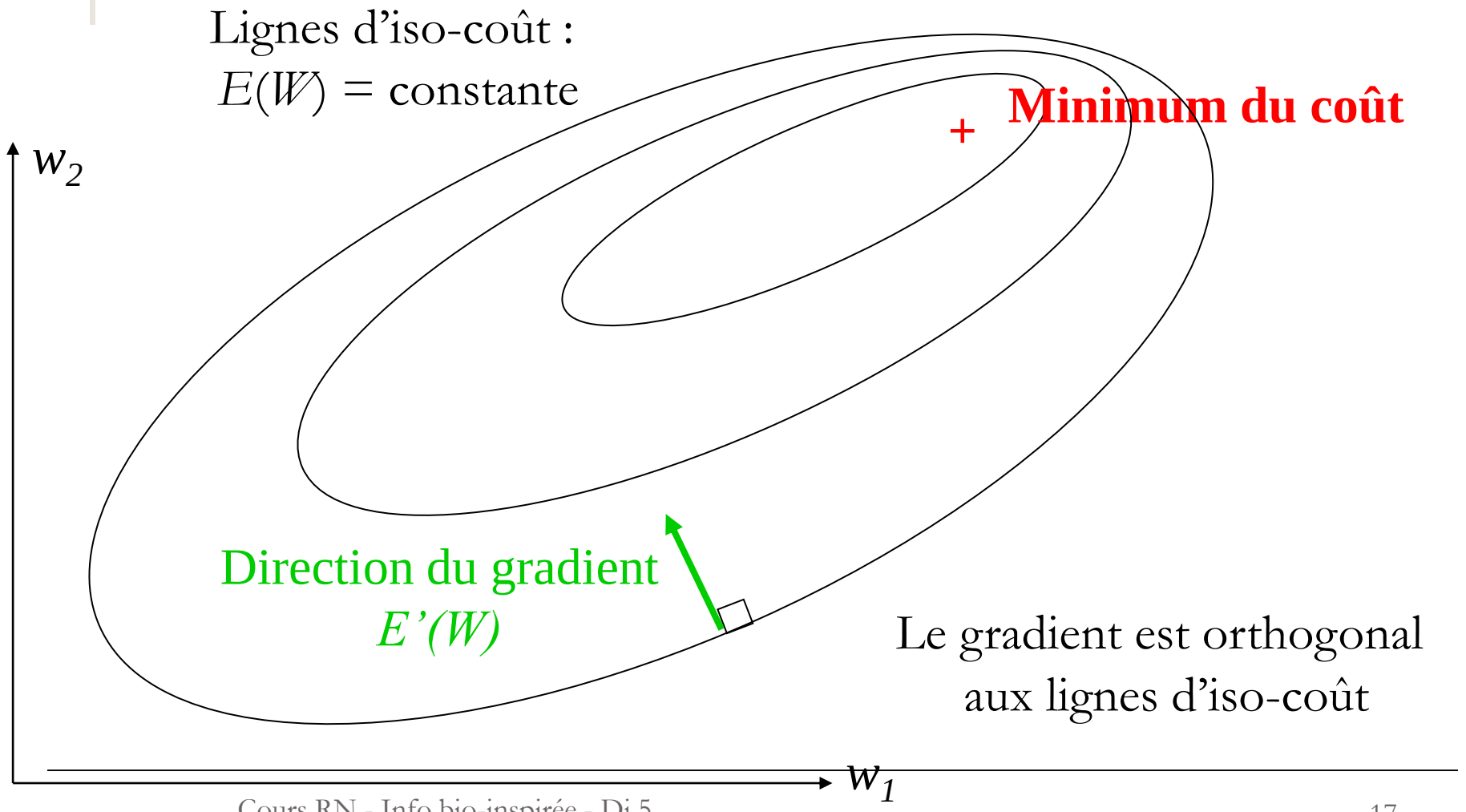


Descente du gradient en 1D

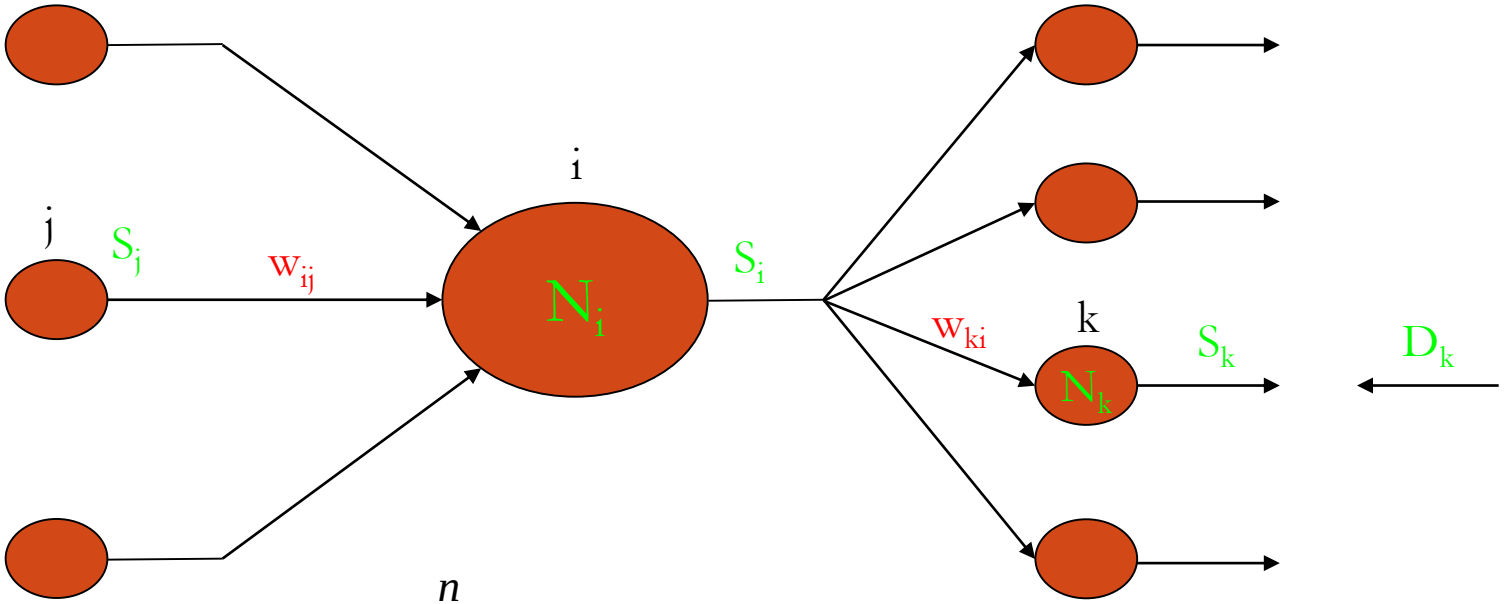
- Apprentissage par rétro-propagation du gradient de l'erreur



Descente du gradient en 2D



Notation



$$N_i = \sum_{j=0}^{n} w_{ij} S_j$$

$$S_i = f(N_i)$$

Vecteur sortie désirée : D

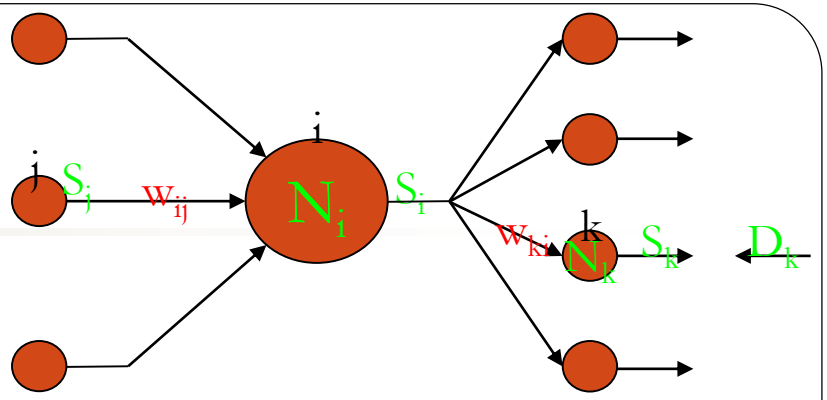
Apprentissage

- Erreur quadratique : $E = \frac{1}{2}\sum_k (S_k - D_k)^2$
  - Minimisation de l'erreur
  - Dérivée de l'erreur par rapport à un poids nous donne l'influence de ce poids sur l'erreur

- Pour la couche de sortie :

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial N_k} \cdot \frac{\partial N_k}{\partial w_{ki}}$$

$$\text{posons } \delta_k = \frac{\partial E}{\partial N_k} = \frac{\partial [\frac{1}{2}\sum_K (f(N_K) - D_K)]^2}{\partial N_k} = (S_k - D_k).f'(N_k)$$

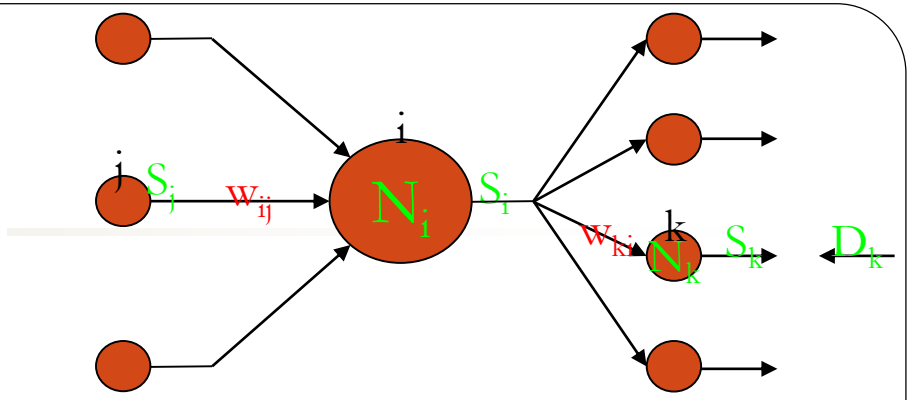


$$\text{Or} \quad \frac{\partial N_k}{\partial w_{ki}} = \frac{\partial[\sum_I (S_I w_{kI})]}{\partial w_{ki}} = S_i$$

$$\text{D'où} \quad \frac{\partial E}{\partial w_{ki}} = (S_k - D_k).f'(N_k).S_i$$

- Pour les autres couches

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial N_i} \cdot \frac{\partial N_i}{\partial w_{ij}}$$



posons $\delta_i = \frac{\partial E}{\partial N_i} = \frac{\partial E}{\partial S_i} . \frac{\partial S_i}{\partial N_i}$

$$\frac{\partial E}{\partial S_i} = \sum_k \left(\frac{\partial E}{\partial N_k} . \frac{\partial N_k}{\partial S_i} \right)$$ car S_i influence tous les N_k

$$= \sum_k (\delta_k w_{ki})$$

D'où $$\delta_i = \sum_k (\delta_k w_{ki}) . f'(N_i)$$

Donc $$\frac{\partial E}{\partial w_{ij}} = \sum_k (\delta_k w_{ki}) . f'(N_i) . S_j$$

Règles d'apprentissage

- Pour faire décroître la valeur de E, modification proportionnelle à la dérivée de E :

$$\Delta w_{ij} = -\eta . \delta_i . S_j$$

avec
$$\begin{cases} \delta_i = (S_i - D_i).f'(N_i) & \text{pour la couche de sortie} \\ \delta_i = \sum_k (\delta_k w_{ki}).f'(N_i) & \text{pour les autres couches} \end{cases}$$

et η : coefficient d'apprentissage $\in]0,1[$

Performances

- Les MLP à une seule couche cachée sont des approximateurs universels (1989) et donc des classificateurs universels.

- L'échec de représenter une fonction ne peut venir que du mauvais choix des paramètres (w_{ij}, b, θ) ou du nombre insuffisant de neurones cachés.

Problèmes

- Minima locaux
- Structure
- Vitesse de convergence
- Sur-apprentissage
  - nombre itérations
  - nombre de neurones

Nombre de couches cachées

- Problème du gradient vanishing ou exploding
- Shallow MLP -> Deep MLP
  > 2 couches cachées
- Extraction de caractéristiques automatique par les premières couches puis classif par les dernières couches
- Etat de l'art en termes de performances sur un grand nombre d'applications (par ex. ImageNet Challenge)

Nouvelles fonctions d'activation

- Rectifier or rectified linear unit (ReLU)
  relu(a) = max(0, a)
- Softmax (a)= $\frac{e^{a_i}}{\sum_j e^{a_j}}$

  Compris entre 0 et 1 : correspond à une probabilité

- Softplus (a) = log (1 + e^a)
- Maxout…

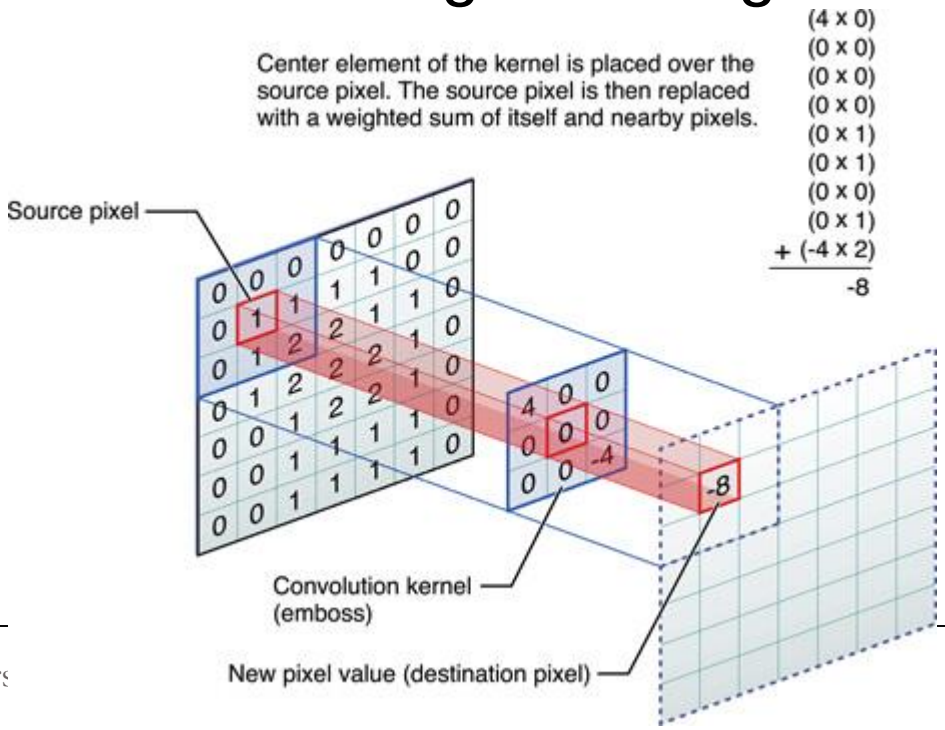
Dropout

- La structure des neurones est souvent fixe et quand on apprend les poids, il peut y avoir du sur-apprentissage : une solution est le dropout

- On enlève (désactive) certains neurones avec une probabilité p puis on apprend sur un mini-batch les poids des neurones restants. On recommence en tirant aléatoirement une nouvelle structure

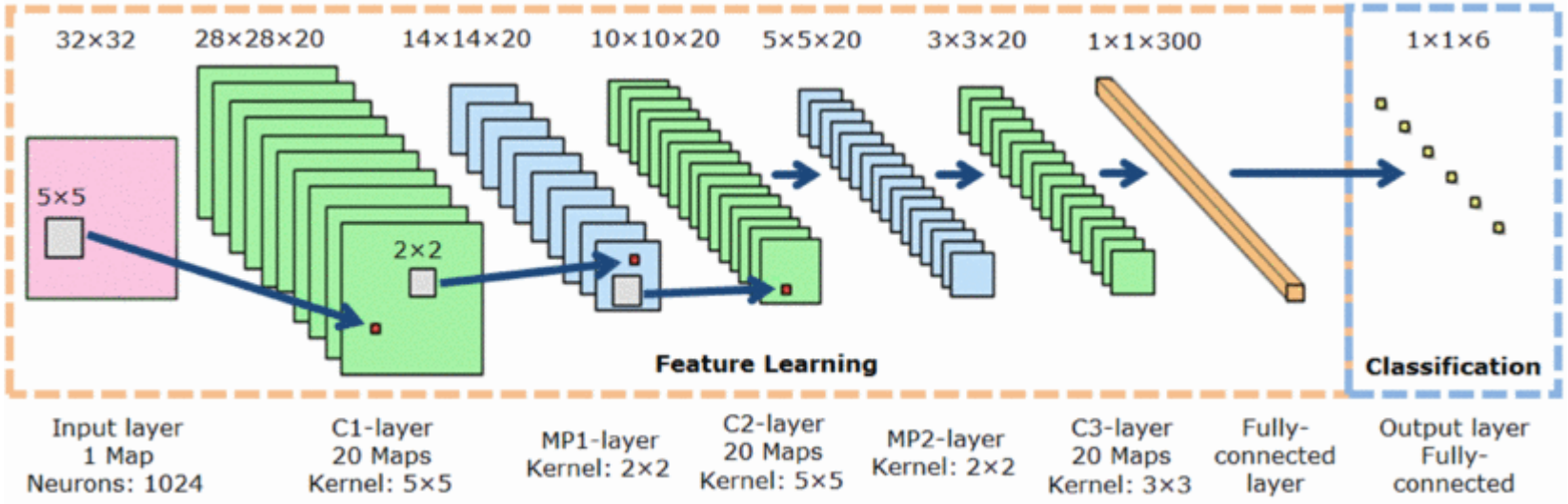
Convolutional Neural Networks

- CNN
- Connexions locales et poids partagés
- Important pour les images dans lesquelles les relations de voisinage sont significatives



Max-pooling

- On garde la valeur max du masque
- Diminue la taille des maps
- Procure une certaine invariance à la translation



Autres RN et conclusion

- SOM (Kohonen Self Organizing Map)
- LSTM
- DBN
- AlexNet…

- Domaine de recherche très actif
- On se rapproche encore des performances humaines. Va-t-on les dépasser et si oui dans combien de temps ?
- Coopération humain / machine