

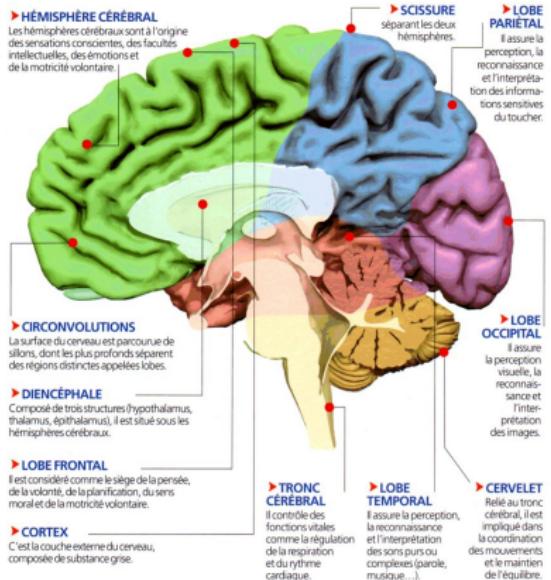
Intelligence Artificielle

Réseaux de neurones

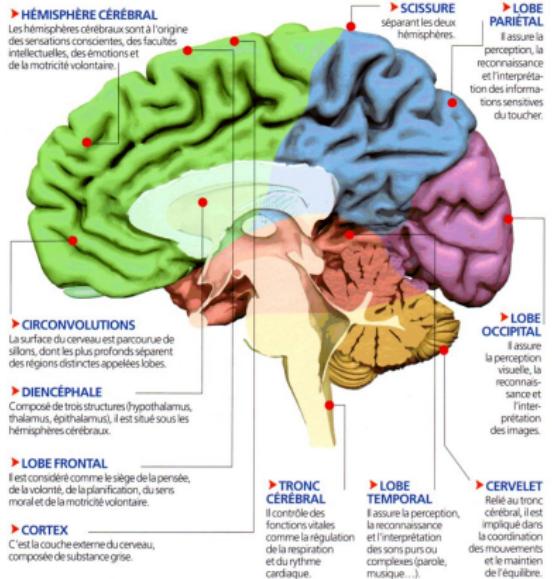
Mathieu Lefort

5 mars 2018

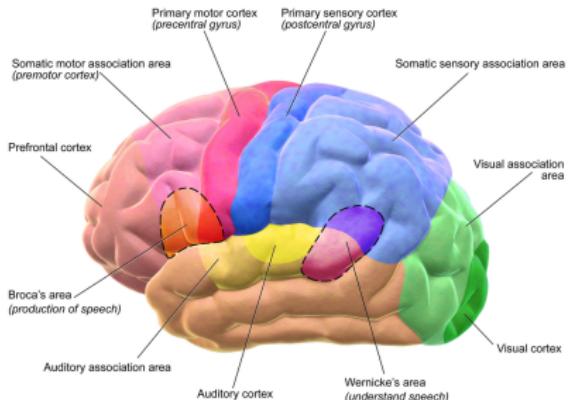
Un peu de biologie ...



Un peu de biologie ...

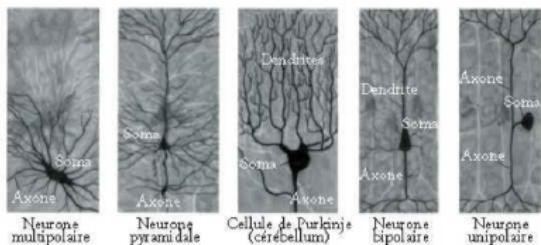
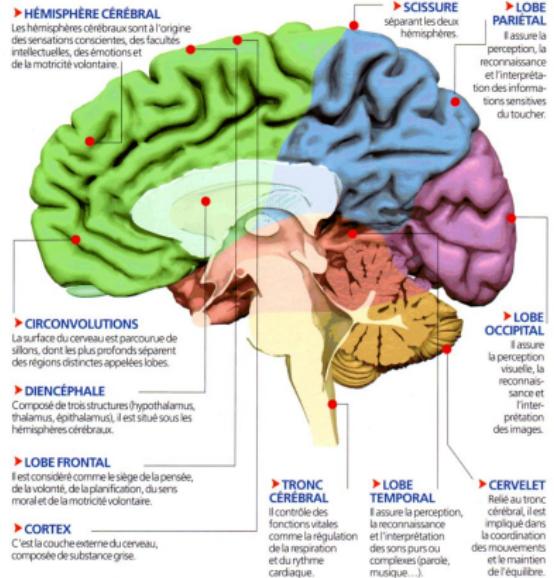


Motor and Sensory Regions of the Cerebral Cortex

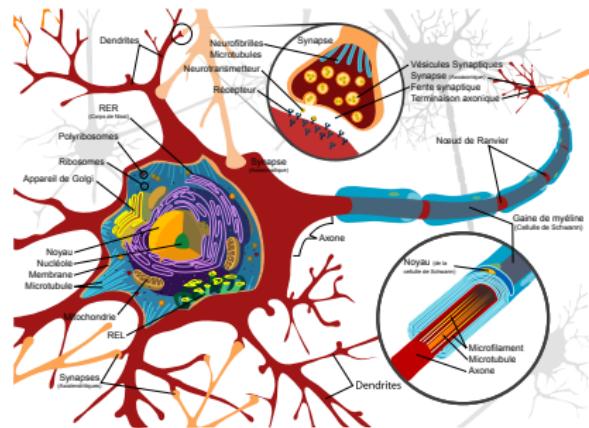


Medical gallery of Blausen Medical 2014

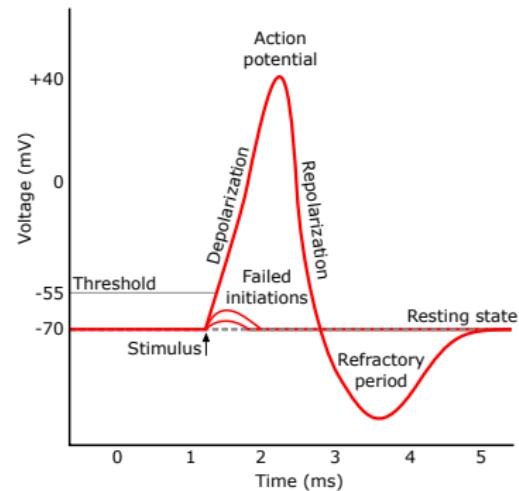
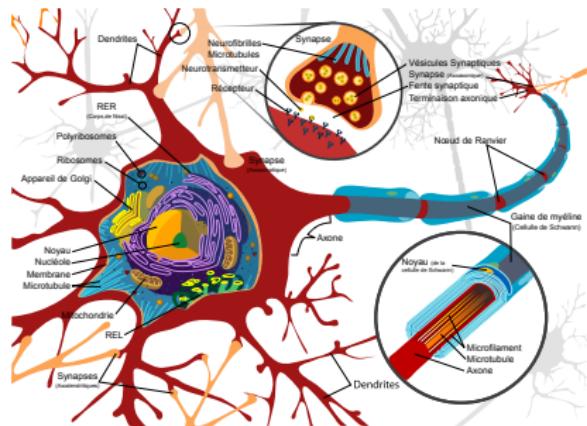
Un peu de biologie ...



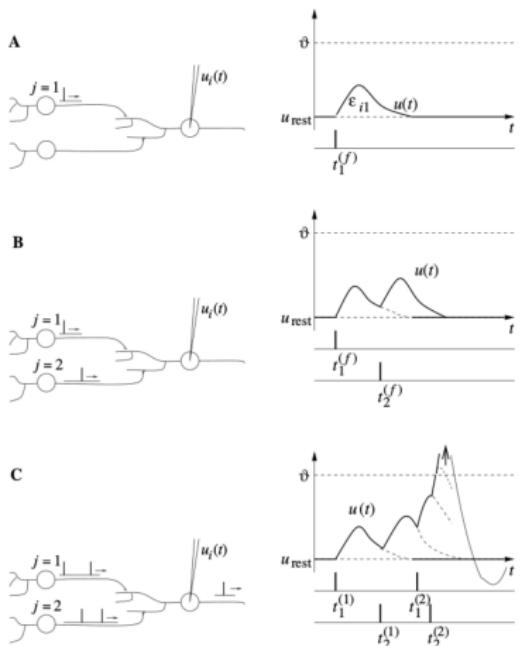
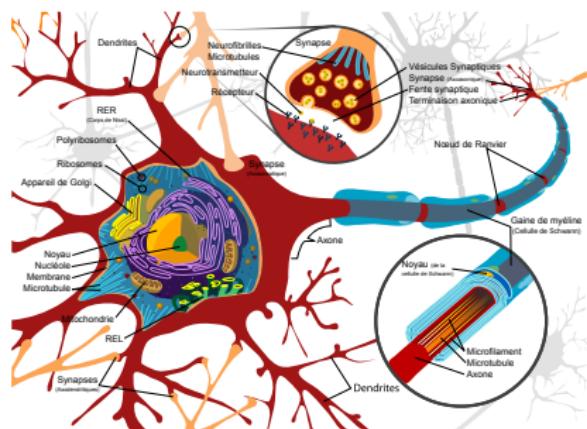
Un peu de biologie ...



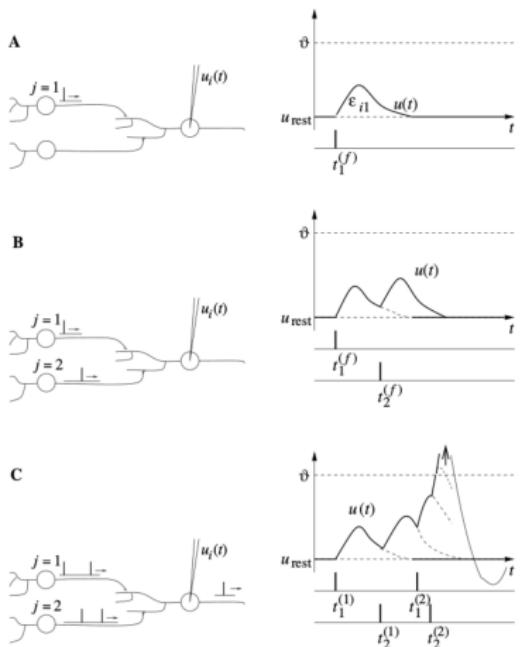
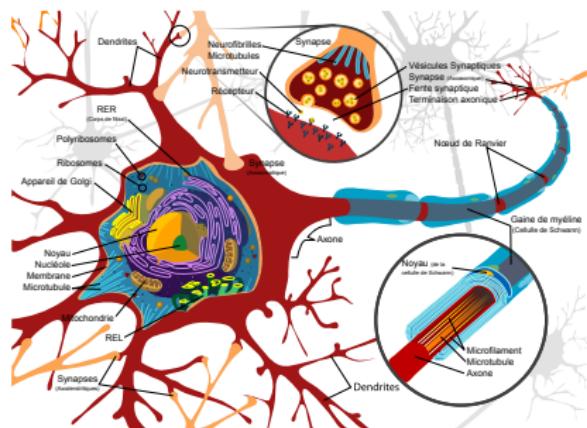
Un peu de biologie ...



Un peu de biologie ...



Un peu de biologie ...



Question

Et les cellules gliales ?

Question

Comment les informations sont représentées ?

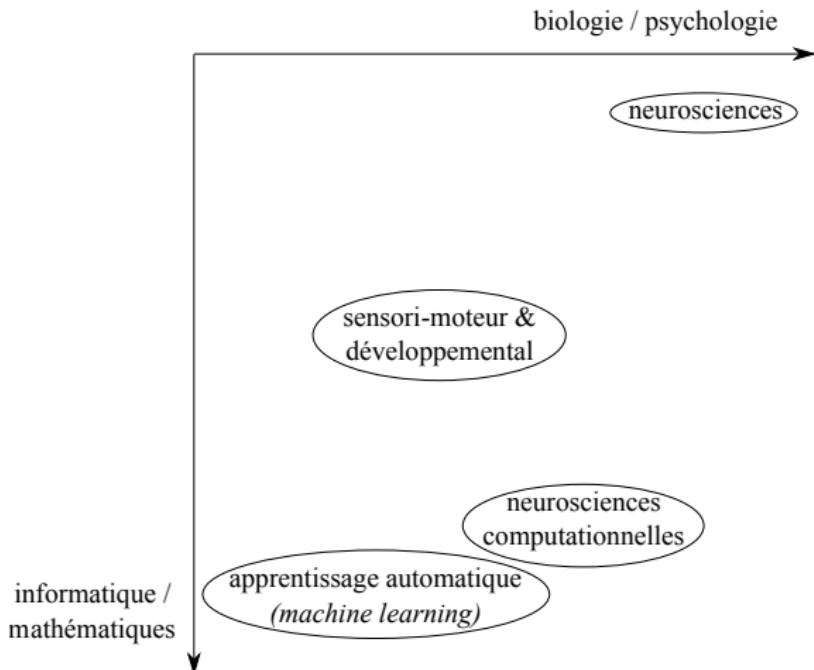
Question

Comment les informations sont représentées ?

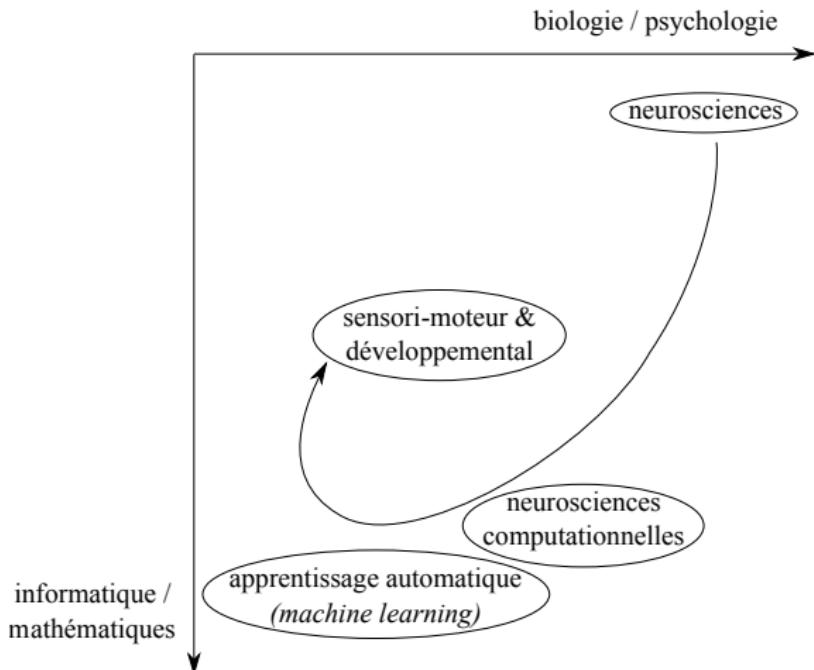
Plusieurs codages

- Codage fréquentiel
- Codage temporel
- Codage probabiliste
- Codage par ordre
- Codage par population

Plan du cours



Plan du cours



Contenu du cours

Ce dont je vais vous parler

- Électricité
- Biologie
- Mathématiques
- IA
- Apprentissage
- Remarque : Ce n'est pas un cours d'apprentissage automatique mais une introduction aux réseaux de neurones en lien avec les problématiques de l'IA.

Ce que vous devez retenir

- Qu'est ce qu'un neurone (biologique et artificiel) ?
- Comment faire un réseau de neurones ?
- Pourquoi, quand et comment utiliser les différents modèles ?
- Quels liens avec le reste de l'IA ?

Organisation

- Volume horaire : 4h CM + 4h TP
- Notation : Examen + TP noté
- Questionnaire d'évaluation



Problème

- Soit l'équation différentielle $\frac{dy(t)}{dt} = f(t, y(t))$ décrivant l'évolution de la grandeur $y(t)$ au cours du temps (qui vaut y_0 à t_0)
- On approxime $y(t_n) = y(t_0 + n\delta t)$ de manière itérative : $y_{n+1} = y_n + \delta t \phi(t, y_n, f)$. Cette mise à jour sera dite d'ordre p si $y(t_{n+1}) - y_{n+1} = O(\delta t^{p+1})$

Problème

- Soit l'équation différentielle $\frac{dy(t)}{dt} = f(t, y(t))$ décrivant l'évolution de la grandeur $y(t)$ au cours du temps (qui vaut y_0 à t_0)
- On approxime $y(t_n) = y(t_0 + n\delta t)$ de manière itérative : $y_{n+1} = y_n + \delta t \phi(t, y_n, f)$. Cette mise à jour sera dite d'ordre p si $y(t_{n+1}) - y_{n+1} = O(\delta t^{p+1})$

Développement de Taylor

- $g(x + h) = \sum_{i=0}^p h^i \frac{g^{(i)}(x)}{i!} + O(h^{p+1})$
- $y_{n+1} = y_n + \delta t \sum_{i=0}^p \delta t^i \frac{f(\cdot, y(\cdot))^{(i)}(t)}{(i+1)!} + O(\delta t^{p+1}) \quad (g \rightarrow y, x \rightarrow t_n, h \rightarrow \delta t)$

Problème

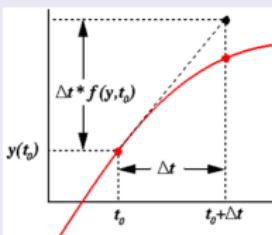
- Soit l'équation différentielle $\frac{dy(t)}{dt} = f(t, y(t))$ décrivant l'évolution de la grandeur $y(t)$ au cours du temps (qui vaut y_0 à t_0)
- On approxime $y(t_n) = y(t_0 + n\delta t)$ de manière itérative : $y_{n+1} = y_n + \delta t \phi(t, y_n, f)$. Cette mise à jour sera dite d'ordre p si $y(t_{n+1}) - y_{n+1} = O(\delta t^{p+1})$

Développement de Taylor

- $g(x + h) = \sum_{i=0}^p h^i \frac{g^{(i)}(x)}{i!} + O(h^{p+1})$
- $y_{n+1} = y_n + \delta t \sum_{i=0}^p \delta t^i \frac{f(\cdot, y(\cdot))^{(i)}(t)}{(i+1)!} + O(\delta t^{p+1}) \quad (g \rightarrow y, x \rightarrow t_n, h \rightarrow \delta t)$

Méthode d'Euler

$$y_{n+1} = y_n + \delta t f(t_n, y_n)$$

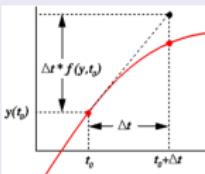


Problème

- Soit l'équation différentielle $\frac{dy(t)}{dt} = f(t, y(t))$ décrivant l'évolution de la grandeur $y(t)$ au cours du temps (qui vaut y_0 à t_0)
- On approxime $y(t_n) = y(t_0 + n\delta t)$ de manière itérative : $y_{n+1} = y_n + \delta t \phi(t, y_n, f)$. Cette mise à jour sera dite d'ordre p si $y(t_{n+1}) - y_{n+1} = O(\delta t^{p+1})$

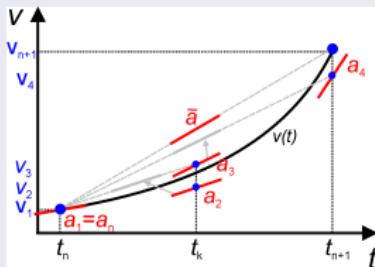
Méthode d'Euler

$$y_{n+1} = y_n + \delta t f(t_n, y_n)$$

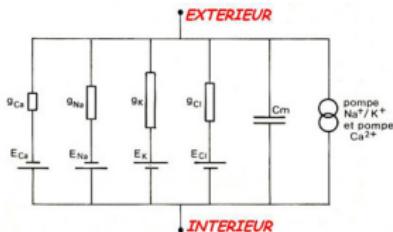
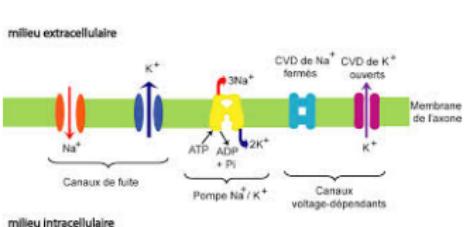


Méthode de Runge-Kutta (d'ordre 4)

- $a_1 = f(t_n, y_n)$
- $a_2 = f(t_n + \frac{\delta t}{2}, y_n + \frac{\delta t}{2} a_1)$
- $a_3 = f(t_n + \frac{\delta t}{2}, y_n + \frac{\delta t}{2} a_2)$
- $a_4 = f(t_n + \delta t, y_n + \delta t a_3)$
- $y_{n+1} = y_n + \frac{\delta t}{6} (a_1 + 2a_2 + 2a_3 + a_4)$



Neurones à spikes - Modèle d'Hodgkin-Huxley (1952)



$$\begin{aligned}
 \overbrace{I}^{\text{courant injecté}} &= \overbrace{C_m}^{\text{capacité de membrane}} \frac{d}{dt} \overbrace{V_m}^{\text{potentiel de membrane}} + g_K n^4 (V_m - V_K) + g_{\text{Na}} m^3 h (V_m - V_{\text{Na}}) + g_I (V_m - V_I) \\
 \frac{d\star}{dt} &= \alpha_\star(V_m)(1 - \star) - \beta_\star(V_m)\star \quad \text{avec } \star \in n, m, h
 \end{aligned}$$

$$\alpha_n(V_m) = \frac{0.01(V_m - 10)}{e^{\frac{V_m - 10}{10}} - 1}$$

$$\beta_n(V_m) = 0.125 e^{\frac{V_m}{80}}$$

$$\alpha_m(V_m) = \frac{0.1(V_m - 25)}{e^{\frac{V_m - 25}{10}} - 1}$$

$$\beta_m(V_m) = 4 e^{\frac{V_m}{18}}$$

$$\alpha_h(V_m) = 0.07 e^{\frac{V_m}{20}}$$

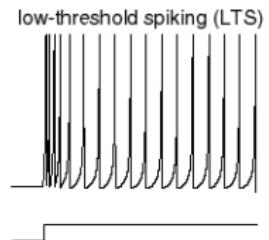
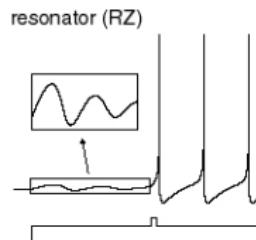
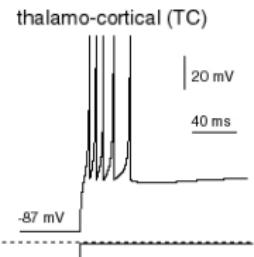
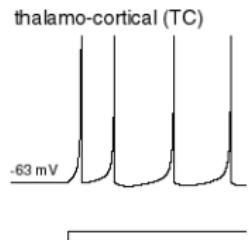
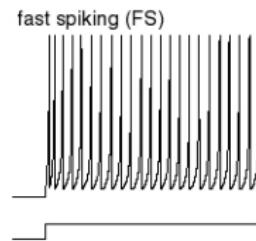
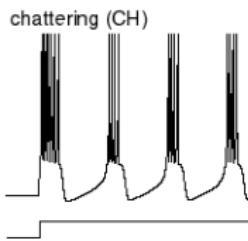
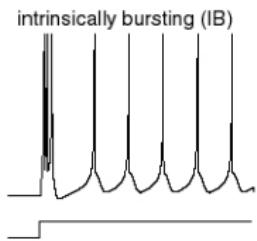
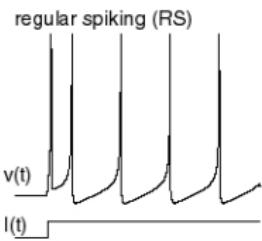
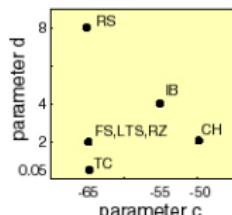
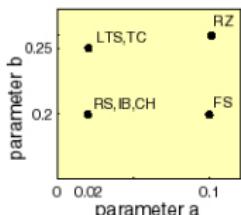
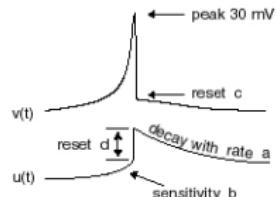
$$\beta_h(V_m) = \frac{1}{e^{\frac{V_m - 30}{10}} + 1}$$

Neurones à spikes - Modèle d'Izhikevich (2003)

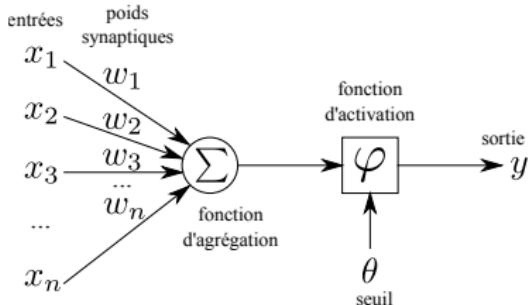
$$v' = 0.04v^2 + 5v + 140 - u + I$$

$$u' = a(bv - u)$$

if $v = 30 \text{ mV}$,
then $v \leftarrow c$, $u \leftarrow u + d$



Neurones à fréquence de décharge - McCulloch et Pitts (1943)



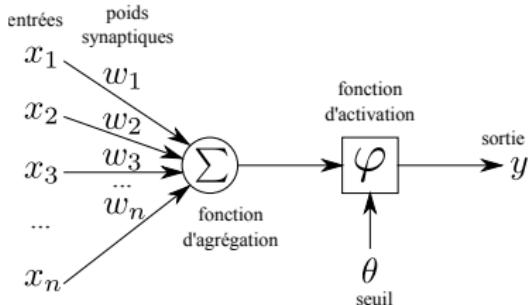
$$y = \varphi\left(\sum_i w_i x_i - \theta\right) \text{ ou écrit autrement}$$

$$y = \varphi(\mathbf{W}^T \mathbf{X} - \theta) \text{ avec } \mathbf{W} = [w_1 \ w_2 \ \dots \ w_n]^T \text{ et } \mathbf{X} = [x_1 \ x_2 \ \dots \ x_n]^T$$

Remarques

- La fonction d'agrégation est souvent une somme pondérée $\mathbf{W} \cdot \mathbf{X}^T$ (codage par intensité) ou une distance $\|\mathbf{W} - \mathbf{X}\|$ (codage tabulaire)
- La fonction d'activation est souvent une sigmoïde $\varphi(y) = \frac{1}{1+e^{-y}}$, une tangente hyperbolique $\varphi(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}}$ ou une gaussienne $\varphi(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{y^2}{2\sigma^2}}$

Neurones à fréquence de décharge - McCulloch et Pitts (1943)



$$y = \varphi\left(\sum_i w_i x_i - \theta\right) \text{ ou écrit autrement}$$

$$y = \varphi(\mathbf{W}^T \mathbf{X} - \theta) \text{ avec } \mathbf{W} = [w_1 \ w_2 \ \dots \ w_n]^T \text{ et } \mathbf{X} = [x_1 \ x_2 \ \dots \ x_n]^T$$

Remarques

- La fonction d'agrégation est souvent une somme pondérée $\mathbf{W} \cdot \mathbf{X}^T$ (codage par intensité) ou une distance $\|\mathbf{W} - \mathbf{X}\|$ (codage tabulaire)
- La fonction d'activation est souvent une sigmoïde $\varphi(y) = \frac{1}{1+e^{-y}}$, une tangente hyperbolique $\varphi(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}}$ ou une gaussienne $\varphi(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{y^2}{2\sigma^2}}$

Question

Où sont passés les spikes ?

Neurones à *spikes* vs neurones à fréquence de décharge

Modèle neurone	<i>Spikes</i>	Fréquence de décharge
Transmission information		
Représentation information		
Évolution		
Parallèle biologique		
Simulation informatique		

Neurones à *spikes* vs neurones à fréquence de décharge

Modèle neurone	<i>Spikes</i>	Fréquence de décharge
Transmission information	Potentiel d'action	Valeur réelle
Représentation information		
Évolution		
Parallèle biologique		
Simulation informatique		

Neurones à *spikes* vs neurones à fréquence de décharge

Modèle neurone	<i>Spikes</i>	Fréquence de décharge
Transmission information	Potentiel d'action	Valeur réelle
Représentation information	Temps (synchronie, séquence)	Motif d'activation
Évolution		
Parallèle biologique		
Simulation informatique		

Neurones à *spikes* vs neurones à fréquence de décharge

Modèle neurone	<i>Spikes</i>	Fréquence de décharge
Transmission information	Potentiel d'action	Valeur réelle
Représentation information	Temps (synchronie, séquence)	Motif d'activation
Évolution	Asynchrone	Souvent synchrone
Parallèle biologique		
Simulation informatique		

Neurones à *spikes* vs neurones à fréquence de décharge

Modèle neurone	<i>Spikes</i>	Fréquence de décharge
Transmission information	Potentiel d'action	Valeur réelle
Représentation information	Temps (synchronie, séquence)	Motif d'activation
Évolution	Asynchrone	Souvent synchrone
Parallèle biologique	Potentiel d'action Cortices sensoriels	Fréquence de décharge Cortices moteurs
Simulation informatique		

Neurones à *spikes* vs neurones à fréquence de décharge

Modèle neurone	<i>Spikes</i>	Fréquence de décharge
Transmission information	Potentiel d'action	Valeur réelle
Représentation information	Temps (synchronie, séquence)	Motif d'activation
Évolution	Asynchrone	Souvent synchrone
Parallèle biologique	Potentiel d'action Cortices sensoriels	Fréquence de décharge Cortices moteurs
Simulation informatique	Événementiel Architecture analogique (HBP)	Algèbre matricielle GP-GPU

Neurones à *spikes* vs neurones à fréquence de décharge

Modèle neurone	<i>Spikes</i>	Fréquence de décharge
Transmission information	Potentiel d'action	Valeur réelle
Représentation information	Temps (synchronie, séquence)	Motif d'activation
Évolution	Asynchrone	Souvent synchrone
Parallèle biologique	Potentiel d'action Cortices sensoriels	Fréquence de décharge Cortices moteurs
Simulation informatique	Événementiel Architecture analogique (HBP)	Algèbre matricielle GP-GPU

Question

Quelles hypothèses sur la conscience, l'intelligence, ... ?

Question

Et maintenant comment on fait un réseau ?

Question

Et maintenant comment on fait un réseau ?

Paramètres

- Quel(s) modèle(s) de neurone ?
- Combien ?
- Qui est connecté à qui ?
- Comment ?
- Quelles entrées/sorties ?
- Apprentissage ?

Question

Qu'attendre d'un système apprenant ? Pourquoi ? Comment ?

Objectifs

- mémorisation : retrouver la réponse à une entrée déjà vue
- généralisation : trouver la réponse à une entrée inconnue (\neq sur-apprentissage)
- hypothèses : fonction localement continue, rasoir d'Occam
- techniques : ensemble apprentissage/validation/test, régularisation, ...

Objectifs

- mémorisation : retrouver la réponse à une entrée déjà vue
- généralisation : trouver la réponse à une entrée inconnue (\neq sur-apprentissage)
- hypothèses : fonction localement continue, rasoir d'Occam
- techniques : ensemble apprentissage/validation/test, régularisation, ...

Types

- supervisé : $\mathbf{Y} = f(\mathbf{X})$, avec $\{\mathbf{X}\}$ les entrées et $\{\mathbf{Y}\}$ les sorties correspondantes (régression (\mathbf{Y} continu) ou classification (\mathbf{Y} discret) / discriminatif)
- non supervisé : $f(\mathbf{X})$, avec $\{\mathbf{X}\}$ les entrées (*clustering* ou génératif)
- par renforcement, par imitation, ...

Apprentissage

Objectifs

- mémorisation : retrouver la réponse à une entrée déjà vue
- généralisation : trouver la réponse à une entrée inconnue (\neq sur-apprentissage)
- hypothèses : fonction localement continue, rasoir d'Occam
- techniques : ensemble apprentissage/validation/test, régularisation, ...

Types

- supervisé : $\mathbf{Y} = f(\mathbf{X})$, avec $\{\mathbf{X}\}$ les entrées et $\{\mathbf{Y}\}$ les sorties correspondantes (régression (\mathbf{Y} continu) ou classification (\mathbf{Y} discret) / discriminatif)
- non supervisé : $f(\mathbf{X})$, avec $\{\mathbf{X}\}$ les entrées (*clustering* ou génératif)
- par renforcement, par imitation, ...

Applications

- reconnaissance d'images, de texte, de parole, ...
- contrôle qualité
- moteurs de recherche
- jeux

Question

Comment ?

Question

Comment ?

Moyens

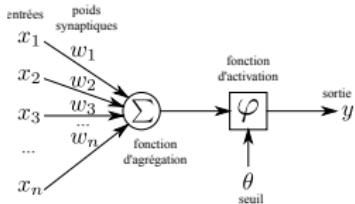
- plasticité synaptique : modification des poids du réseau
- neuro-genèse : rajout d'unités dans le réseau
- dendrite-genèse : rajout de connexions dans le réseau
- homéostasie : régulation de l'activité dans le neurone/réseau

Propriétés (potentielles)

- décentralisé : chaque neurone/synapse se met à jour indépendamment
- local : chaque neurone/synapse se met à jour à partir des informations à sa disposition (pré et post synaptiques)
- en ligne : présentation séquentielle des entrées et évaluation/apprentissage simultané (\neq batch)
- distribution/épars : l'entrée est représentée par un ensemble restreint de neurones actifs

Apprentissage non supervisé - Règle de Hebb (1949)

“Cells that fire together, wire together”



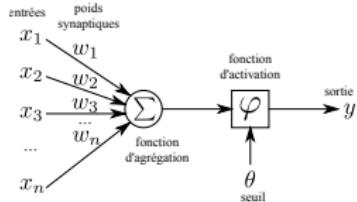
$$\Delta w_i = \eta x_i y, \forall i \text{ ou écrit autrement } \Delta \mathbf{W} = \eta \mathbf{X} y$$

avec $\mathbf{W} = [w_1 \ w_2 \ \dots \ w_n]^T$ et $\mathbf{X} = [x_1 \ x_2 \ \dots \ x_n]^T$

η est appelé le taux d'apprentissage

Apprentissage non supervisé - Règle de Hebb (1949)

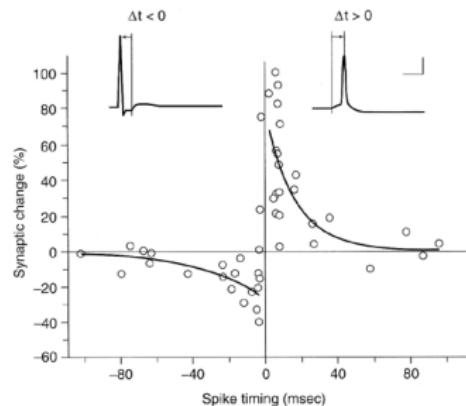
“Cells that fire together, wire together”



$$\Delta w_i = \eta x_i y, \forall i \text{ ou écrit autrement } \Delta \mathbf{W} = \eta \mathbf{X} y$$

avec $\mathbf{W} = [w_1 \ w_2 \ \dots \ w_n]^T$ et $\mathbf{X} = [x_1 \ x_2 \ \dots \ x_n]^T$

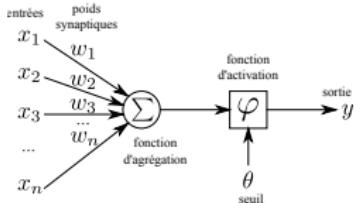
η est appelé le taux d'apprentissage



Spike Timing Dependent Plasticity

Apprentissage non supervisé - Règle de Hebb (1949)

“Cells that fire together, wire together”



$$\Delta w_i = \eta x_i y, \forall i \text{ ou écrit autrement } \Delta \mathbf{W} = \eta \mathbf{X} y$$

avec $\mathbf{W} = [w_1 \ w_2 \ \dots \ w_n]^T$ et $\mathbf{X} = [x_1 \ x_2 \ \dots \ x_n]^T$

η est appelé le taux d'apprentissage

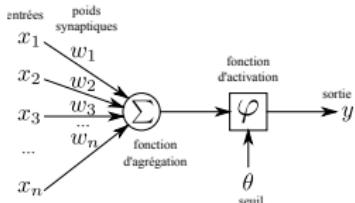
Algorithme (en ligne)

Pendant n pas de temps

- ① Présentation d'une nouvelle entrée \mathbf{X} (Actif)
- ② Calcul de la sortie du neurone y (Modèle)
- ③ Mise à jour des poids \mathbf{W} (Apprentissage)

Apprentissage non supervisé - Règle de Hebb (1949)

“Cells that fire together, wire together”



$$\Delta w_i = \eta x_i y, \forall i \text{ ou écrit autrement } \Delta \mathbf{W} = \eta \mathbf{X} y$$

avec $\mathbf{W} = [w_1 \ w_2 \ \dots \ w_n]^T$ et $\mathbf{X} = [x_1 \ x_2 \ \dots \ x_n]^T$

η est appelé le taux d'apprentissage

Algorithme (en ligne)

Pendant n pas de temps

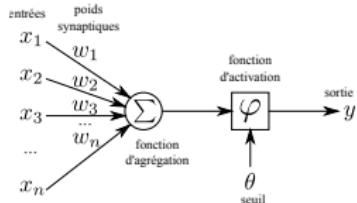
- ① Présentation d'une nouvelle entrée \mathbf{X} (Actif)
- ② Calcul de la sortie du neurone y (Modèle)
- ③ Mise à jour des poids \mathbf{W} (Apprentissage)

Question

Une idée de ce que cela donne ?

Apprentissage non supervisé - Règle de Hebb (1949)

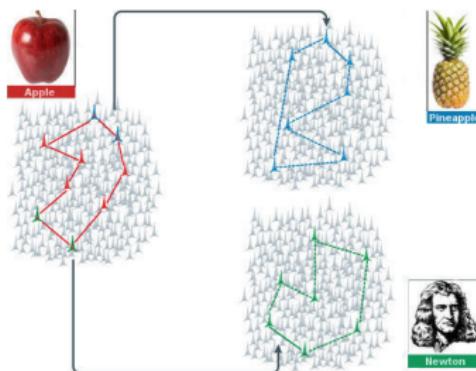
“Cells that fire together, wire together”



$$\Delta w_i = \eta x_i y, \forall i \text{ ou écrit autrement } \Delta \mathbf{W} = \eta \mathbf{X} y$$

avec $\mathbf{W} = [w_1 \ w_2 \ \dots \ w_n]^T$ et $\mathbf{X} = [x_1 \ x_2 \ \dots \ x_n]^T$

η est appelé le taux d'apprentissage



Bilan apprentissage de réseaux de neurones à fréquence de décharge

Règle	Supervisé	Local	Croissant	En ligne	Réseau	Sortie	Propriété
Hebb	X	✓	X	✓			Assemblée de neurones

Sortie du neurone : $y = \sum_i w_i x_i$, ($\varphi = Id$ et $\theta = 0$).

Apprentissage : On reprend la règle de Hebb et on normalise les poids à 1 :

$$\begin{aligned}w_i(t+1) &= \frac{w_i + \eta x_i y}{\sqrt{\sum_j (w_j + \eta x_j y)^2}} \\&= \frac{w_i}{\sqrt{\sum_j w_j^2}} + \eta \left(\frac{x_i y}{\sqrt{\sum_j w_j^2}} - \frac{w_i \sum_j w_j x_j y}{\sqrt{\sum_j w_j^2}^3} \right) + O(\eta^2) \quad (\text{Taylor}) \\&\approx w_i + \eta y(x_i - w_i y)\end{aligned}$$

ou écrit autrement $\Delta \mathbf{W} = \eta y(\mathbf{X} - \mathbf{W}y)$

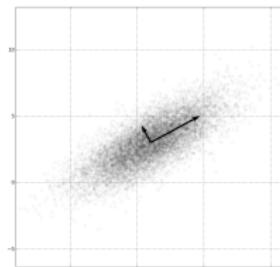
Sortie du neurone : $y = \mathbf{W}^T \mathbf{X} = \mathbf{X}^T \mathbf{W}$

Règle de Oja :

$$\begin{aligned}\Delta \mathbf{W} &= \eta y (\mathbf{X} - \mathbf{W}y) \\ &= \eta (\mathbf{X} \mathbf{X}^T \mathbf{W} - y^2 \mathbf{W})\end{aligned}$$

À convergence $\sum_{\{\mathbf{X}\}} \Delta W = 0$, donc \mathbf{W} est un vecteur

propre de la matrice de covariance contenant l'ensemble des \mathbf{X} et y^2 est la valeur propre associée.



Propriétés

- \mathbf{W} est une composante principale (et l'analyse de stabilité montre que c'est forcément la plus grande composante principale) des données reçues.
- On peut alors construire un réseau de neurones (inhibiteurs) extrayant les composantes principales des données reçues.

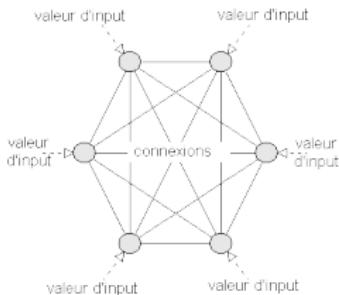
Bilan apprentissage de réseaux de neurones à fréquence de décharge

Règle	Supervisé	Local	Croissant	En ligne	Réseau	Sortie	Propriété
Hebb	X	✓	X	✓			Assemblée de neurones
Oja	X	✓	X	✓	Inhibiteur		ACP

Apprentissage non supervisé - Réseau de Hopfield (1982)

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } \mathbf{W}_j^T \mathbf{X} > \theta_j \\ -1 & \text{sinon} \end{cases}$

Réseau :



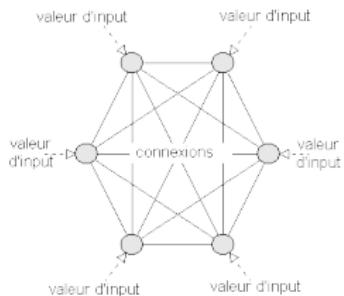
Connexions totales et symétriques ($w_{ij} = w_{ji}$)

Apprentissage : $w_{ij} = \frac{1}{\text{card}(\{\mathbf{X}\})} \sum_{\{\mathbf{X}\}} x_i x_j$

Apprentissage non supervisé - Réseau de Hopfield (1982)

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } \mathbf{W}_j^T \mathbf{X} > \theta_j \\ -1 & \text{sinon} \end{cases}$

Réseau :



Connexions totales et symétriques ($w_{ij} = w_{ji}$)

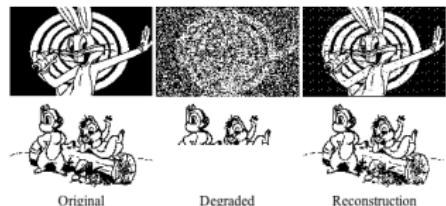
Apprentissage : $w_{ij} = \frac{1}{\text{card}(\{\mathbf{X}\})} \sum_{\{\mathbf{X}\}} x_i x_j$

Fonction d'énergie :



$$E = -\frac{1}{2} \sum_{i,j} w_{ij} x_i x_j + \sum_i \theta_i x_i$$

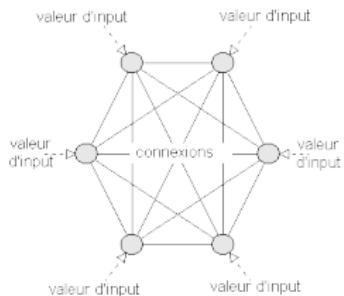
Exemple :



Apprentissage non supervisé - Réseau de Hopfield (1982)

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } \mathbf{W}_j^T \mathbf{X} > \theta_j \\ -1 & \text{sinon} \end{cases}$

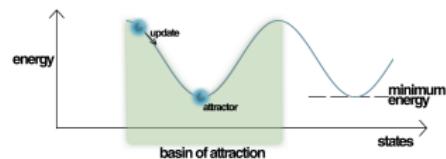
Réseau :



Connexions totales et symétriques ($w_{ij} = w_{ji}$)

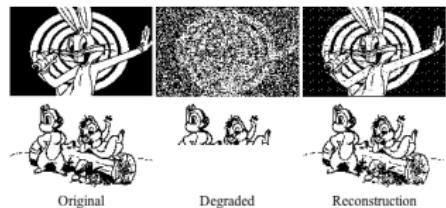
Apprentissage : $w_{ij} = \frac{1}{\text{card}(\{\mathbf{X}\})} \sum_{\{\mathbf{X}\}} x_i x_j$

Fonction d'énergie :



$$E = -\frac{1}{2} \sum_{i,j} w_{ij} x_i x_j + \sum_i \theta_i x_i$$

Exemple :



Propriétés

- Mémoire adressable par le contenu
- Peut retenir au maximum $0.14N$ entrées, avec N le nombre de neurones du réseau

Bilan apprentissage de réseaux de neurones à fréquence de décharge

Règle	Supervisé	Local	Croissant	En ligne	Réseau	Sortie	Propriété
Hebb	X	✓	X	✓			Assemblée de neurones
Oja	X	✓	X	✓	Inhibiteur		ACP
Hopfield	X	✓	X	✓ / X	Symétrique		Mémoire associative

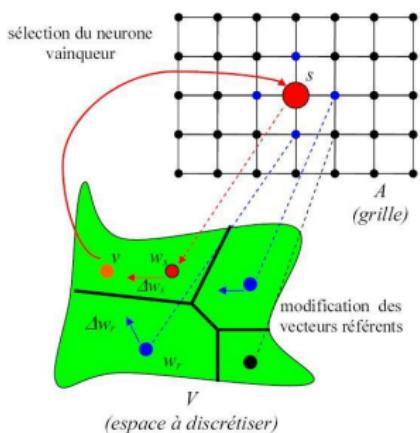
Apprentissage non supervisé - Carte auto-organisatrice de Kohonen (1984)

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } ||\mathbf{W}_j - \mathbf{X}|| = \min_{j'} ||\mathbf{W}_{j'} - \mathbf{X}|| \\ 0 & \text{sinon} \end{cases}$

Réseau : Organisation topologique régulière des neurones (souvent sous forme de grille)

Apprentissage : $\Delta \mathbf{W}_j = \eta V(j, j^*)(\mathbf{X} - \mathbf{W}_j)$
avec $V(j, j^*)$ une fonction de voisinage

(généralement gaussienne $e^{-\frac{||j-j^*||^2}{2\sigma^2}}$)
entre j et j^* le neurone ayant une sortie à 1
(i.e. le plus proche de l'entrée courante)



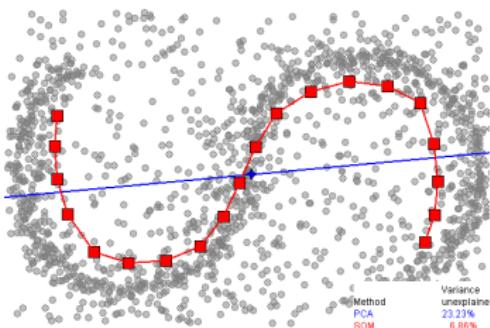
Apprentissage non supervisé - Carte auto-organisatrice de Kohonen (1984)

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } ||\mathbf{W}_j - \mathbf{X}|| = \min_{j'} ||\mathbf{W}_{j'} - \mathbf{X}|| \\ 0 & \text{sinon} \end{cases}$

Réseau : Organisation topologique régulière des neurones (souvent sous forme de grille)

Apprentissage : $\Delta \mathbf{W}_j = \eta V(j, j^*)(\mathbf{X} - \mathbf{W}_j)$
avec $V(j, j^*)$ une fonction de voisinage

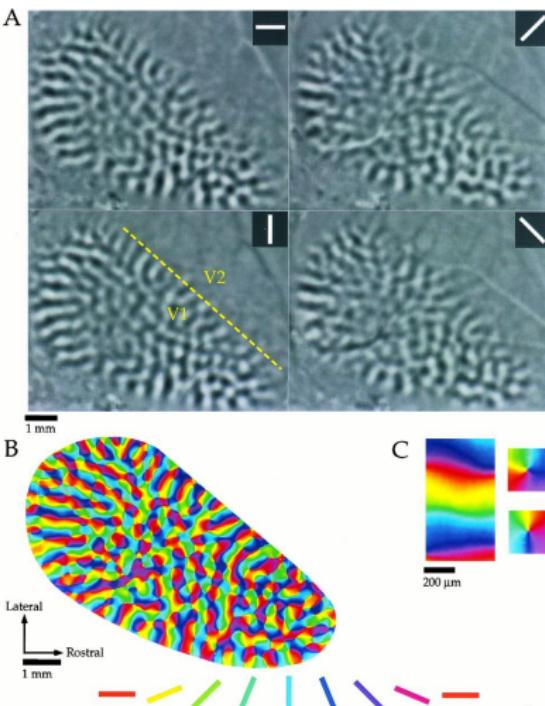
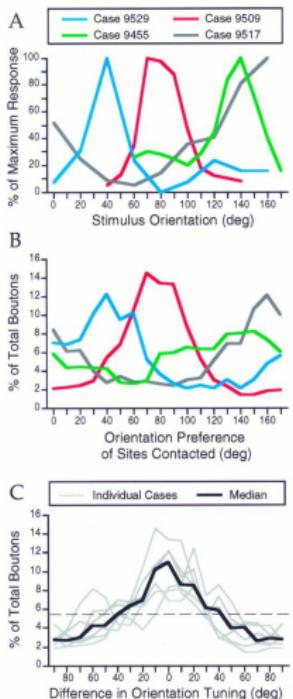
(généralement gaussienne $e^{-\frac{||j-j^*||^2}{2\sigma^2}}$)
entre j et j^* le neurone ayant une sortie à 1
(i.e. le plus proche de l'entrée courante)



Propriétés

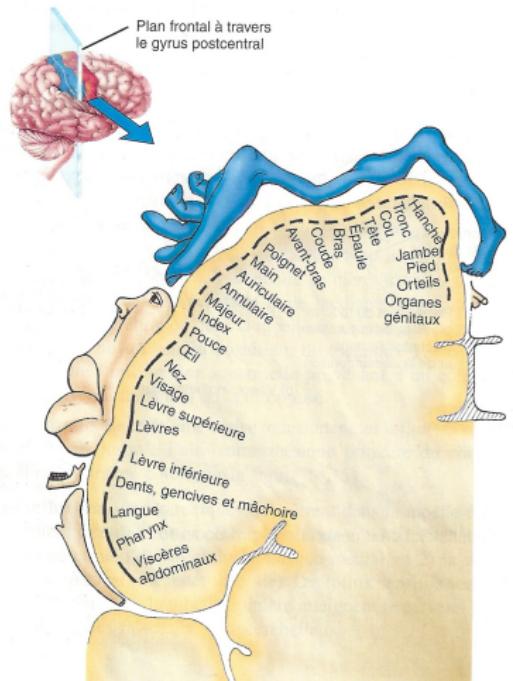
- Projection topologique auto-organisée de l'espace d'entrée
- Quantification vectorielle $E = \sum_{\mathbf{X}} ||\mathbf{W}_{j^*} - \mathbf{X}||^2$ (+ terme topologique)
- Convergence garantie si $\eta \xrightarrow[t]{} 0$ et $V(j, j^*) \xrightarrow[t]{} 0$

Apprentissage non supervisé - Carte auto-organisatrice de Kohonen (1984)

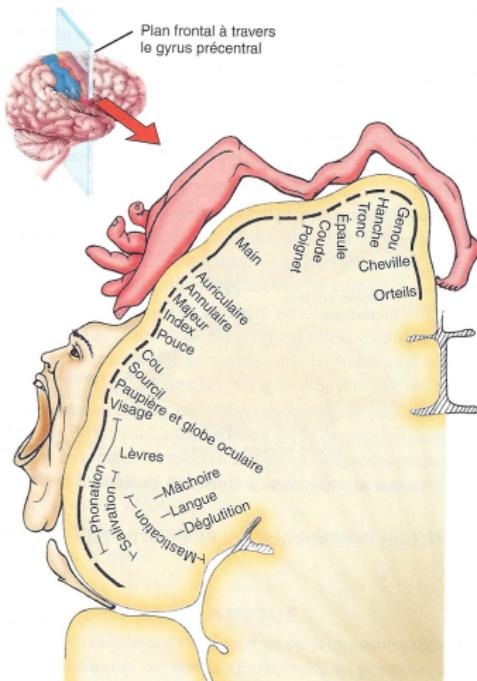


Bosking, W., Zhang, Y., Schofield, B., and Fitzpatrick, D. (1997). Orientation selectivity and the arrangement of horizontal connections in tree shrew striate cortex, *The Journal of Neuroscience*

Apprentissage non supervisé - Carte auto-organisatrice de Kohonen (1984)



(a) Coupe frontale de l'aire somesthésique primaire dans l'hémisphère cérébral droit



(b) Coupe frontale de l'aire motrice primaire dans l'hémisphère cérébral droit

Bilan apprentissage de réseaux de neurones à fréquence de décharge

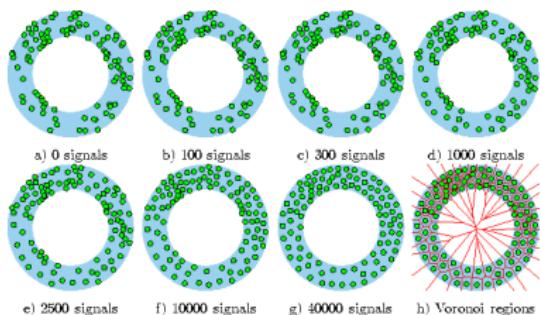
Règle	Supervisé	Local	Croissant	En ligne	Réseau	Sortie	Propriété
Hebb	X	✓	X	✓			Assemblée de neurones
Oja	X	✓	X	✓	Inhibiteur		ACP
Hopfield	X	✓	X	✓/X	Symétrique		Mémoire associative
Kohonen	X	X	X	✓	Grille		Projection topologique

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } ||\mathbf{W}_j - \mathbf{X}|| = \min_{j'} ||\mathbf{W}_{j'} - \mathbf{X}|| \\ 0 & \text{sinon} \end{cases}$

Réseau : Ensemble non connecté de neurones

Apprentissage :

$\Delta \mathbf{W}_j = \eta e^{\frac{-k}{\lambda}} (\mathbf{X} - \mathbf{W}_j)$ avec k le rang du neurone j dans l'ordre décroissant des $||\mathbf{W}_j - \mathbf{X}||$



Propriétés

- Partitionnement de l'espace d'entrée
- Quantification vectorielle $E = \sum_{\mathbf{X}} ||\mathbf{W}_{j^*} - \mathbf{X}||^2$
- Convergence garantie si $\eta \xrightarrow[t]{} 0$ et $\lambda \xrightarrow[t]{} 0$
- Version en ligne de l'algorithme des k-moyennes

Bilan apprentissage de réseaux de neurones à fréquence de décharge

Règle	Supervisé	Local	Croissant	En ligne	Réseau	Sortie	Propriété
Hebb	X	✓	X	✓			Assemblée de neurones
Oja	X	✓	X	✓	Inhibiteur		ACP
Hopfield	X	✓	X	✓ / X	Symétrique		Mémoire associative
Kohonen	X	X	X	✓	Grille		Projection topologique
NG	X	X	X	✓			Quantification vectorielle

$$\text{Sortie du neurone } j : y_j = \begin{cases} 1 & \text{si } \|\mathbf{W}_j - \mathbf{X}\| = \min_{j'} \|\mathbf{W}_{j'} - \mathbf{X}\| \\ 0 & \text{sinon} \end{cases}$$

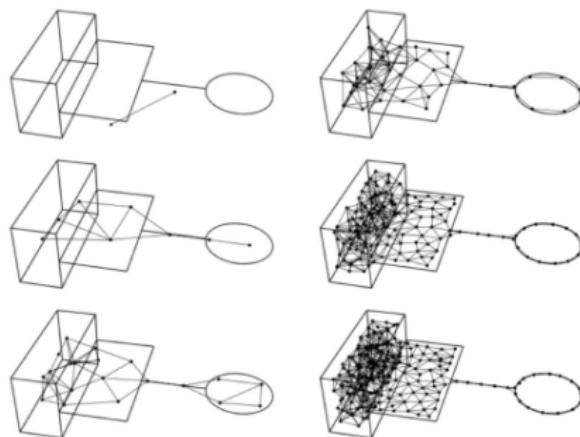
Réseau : Initialement deux neurones

Apprentissage : Tant qu'une performance ou un nombre de neurones n'est pas atteint

- ① soit j^* et j^{**} les 2 neurones les plus proches de \mathbf{X}
- ② $\Delta \mathbf{W}_{j^*} = \eta(\mathbf{X} - \mathbf{W}_{j^*})$ et $\Delta \mathbf{W}_{j^{**}} = \eta'(\mathbf{X} - \mathbf{W}_{j^{**}})$ pour tous les $j^{*'}'$ voisins de j^*
- ③ incrémenter l'âge de toutes les connexions de j^* , création d'une connexion entre j^* et j^{**} (ou remise à 0 de son âge)
- ④ suppression des connexions avec un âge $> a_{max}$ (et des neurones isolés)
- ⑤ incrémenter l'erreur de j^* de $\|\mathbf{W}_{j^*} - \mathbf{X}\|$
- ⑥ si le nombre d'itérations est un multiple de n
 - déterminer j le neurone ayant la plus forte erreur et j' son voisin avec la plus forte erreur
 - rajouter un neurone j'' entre j et j'
 - rajouter deux connexions entre j'' et j et j' et supprimer la connexion entre j et j'
 - décrémenter l'erreur des neurones j et j' d'un facteur α et initialiser l'erreur de j'' à celle de j
- ⑦ décrémenter l'erreur des neurones d'un facteur α'

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } \|\mathbf{W}_j - \mathbf{X}\| = \min_{j'} \|\mathbf{W}_{j'} - \mathbf{X}\| \\ 0 & \text{sinon} \end{cases}$

Réseau : Initialement deux neurones



Propriétés

- Partitionnement de l'espace d'entrée
- Quantification vectorielle $E = \sum_{\mathbf{X}} \|\mathbf{W}_{j^*} - \mathbf{X}\|$
- Apprentissage de la topologie

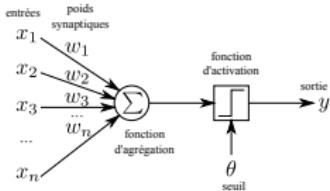
Bilan apprentissage de réseaux de neurones à fréquence de décharge

Règle	Supervisé	Local	Croissant	En ligne	Réseau	Sortie	Propriété
Hebb	X	✓	X	✓			Assemblée de neurones
Oja	X	✓	X	✓	Inhibiteur		ACP
Hopfield	X	✓	X	✓ / X	Symétrique		Mémoire associative
Kohonen	X	X	X	✓	Grille		Projection topologique
NG	X	X	X	✓			Quantification vectorielle
GNG	X	X	✓	✓			Quantification vectorielle Apprentissage topologie

<http://www.demogng.de/js/demogng.html>

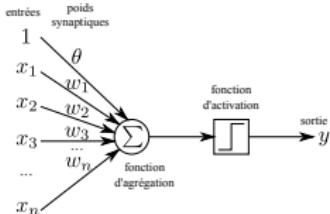
Apprentissage supervisé - Perceptron - Rosenblatt (1957)

Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=1}^n w_i x_i - \theta > 0 \\ 0 & \text{sinon} \end{cases}$



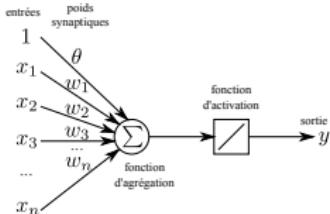
Apprentissage supervisé - Perceptron - Rosenblatt (1957)

Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^n w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$



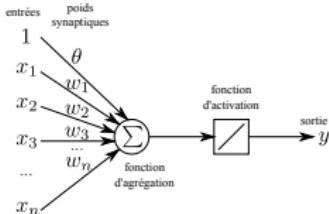
Apprentissage supervisé - Perceptron - Rosenblatt (1957)

Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^n w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$



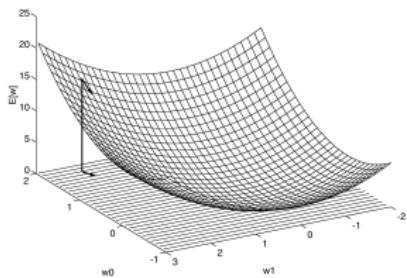
Apprentissage supervisé - Perceptron - Rosenblatt (1957)

Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^n w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$



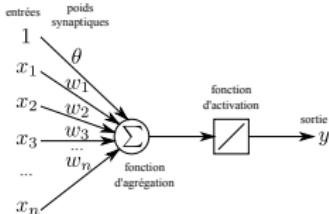
Apprentissage : On veut minimiser l'erreur

$$\text{quadratique } E = \frac{1}{2} \sum_{\{\mathbf{x}\}} (t - y)^2$$



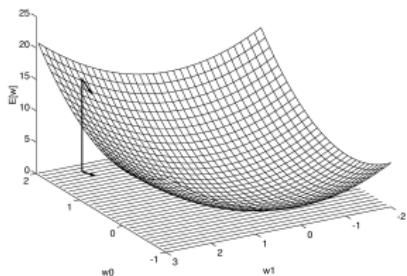
Apprentissage supervisé - Perceptron - Rosenblatt (1957)

Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^n w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$



Apprentissage : On veut minimiser l'erreur

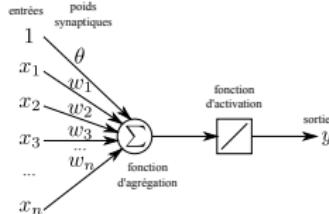
$$\text{quadratique } E = \frac{1}{2} \sum_{\{\mathbf{x}\}} (t - y)^2$$



$$\begin{aligned}\Delta w_i &= -\eta \frac{\partial E}{\partial w_i} \\ &= -\frac{\eta}{2} \sum_{\{\mathbf{x}\}} \frac{\partial(t - y)^2}{\partial w_i} \\ &= -\frac{\eta}{2} \sum_{\{\mathbf{x}\}} -2 \frac{\partial y}{\partial w_i} (t - y) \\ &= \sum_{\{\mathbf{x}\}} \eta x_i (t - y)\end{aligned}$$

Apprentissage supervisé - Perceptron - Rosenblatt (1957)

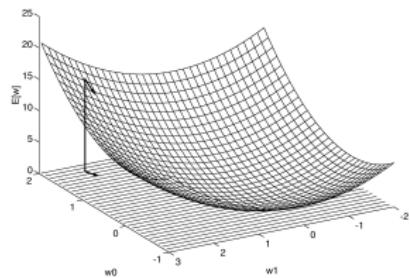
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^n w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$



Apprentissage : On veut minimiser l'erreur

quadratique $E = \frac{1}{2} \sum_{\{\mathbf{x}\}} (t - y)^2$ par descente de

gradient stochastique : $\Delta \mathbf{W} = \eta \mathbf{X}(t - y)$



Propriétés

- Classifieur linéaire
- Convergence garantie si η est faible (même si les données ne sont pas linéairement séparables)
- Convergence efficace (car la fonction à optimiser est quadratique)
- Convergence souvent plus rapide en mode en ligne mais pas garantie comme en mode *batch*

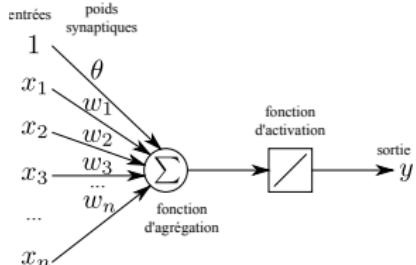
Bilan apprentissage de réseaux de neurones à fréquence de décharge

Règle	Supervisé	Local	Croissant	En ligne	Réseau	Sortie	Propriété
Hebb	X	✓	X	✓			Assemblée de neurones
Oja	X	✓	X	✓	Inhibiteur		ACP
Hopfield	X	✓	X	✓/X	Symétrique		Mémoire associative
Kohonen	X	X	X	✓	Grille		Projection topologique
NG	X	X	X	✓			Quantification vectorielle
GNG	X	X	✓	✓			Quantification vectorielle Apprentissage topologie
Perceptron	✓	✓	X	✓/X			Classifieur linéaire

Apprentissage supervisé - Perceptron - Rosenblatt (1957)

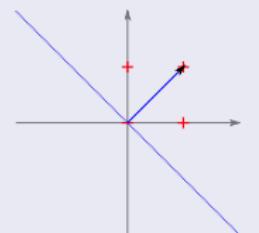
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^n w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

Apprentissage : $\Delta W = \eta X(t - y)$



Un exemple : le OU

Entrée X			Poids W			Sortie		Sortie voulue	
x_0	x_1	x_2	w_0	w_1	w_2	y		t	0
1	0	0	0	1	1	0			

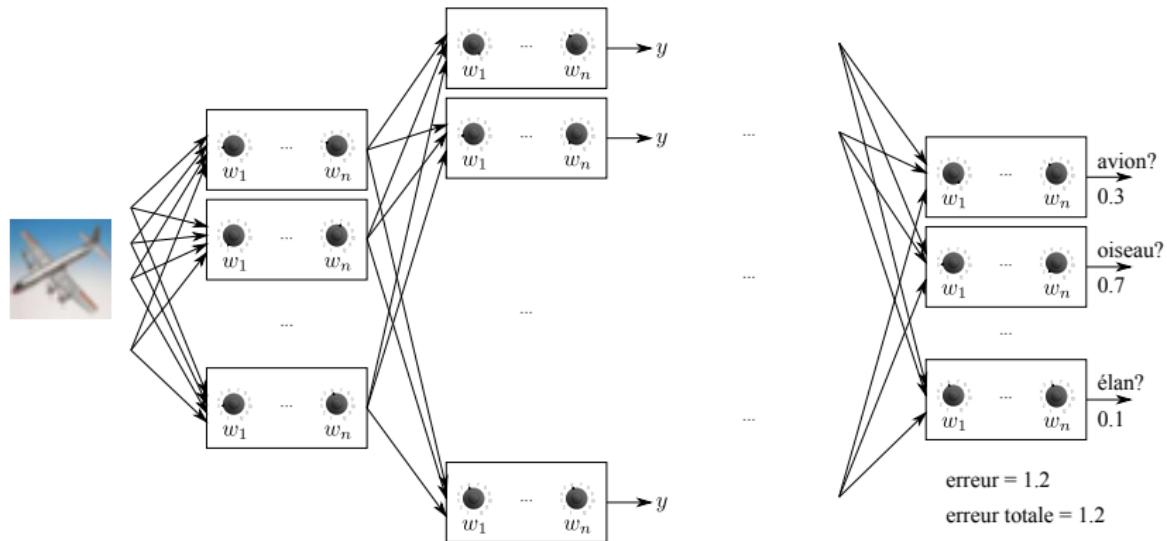


Question

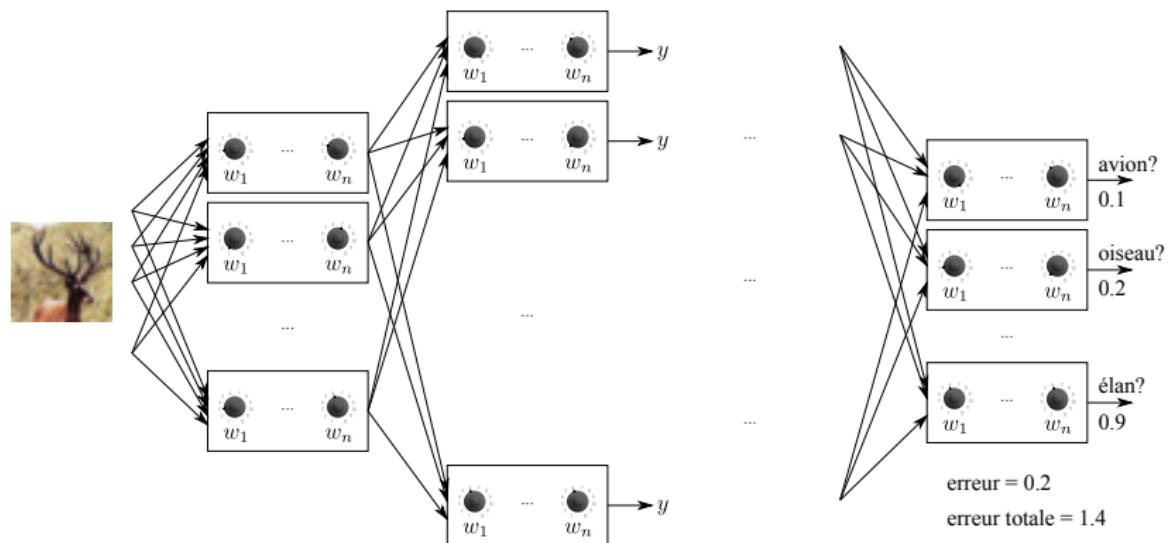
Comment faire un OU EXCLUSIF ?

x_1	x_2	OU EXCLUSIF
0	0	0
0	1	1
1	0	1
1	1	0

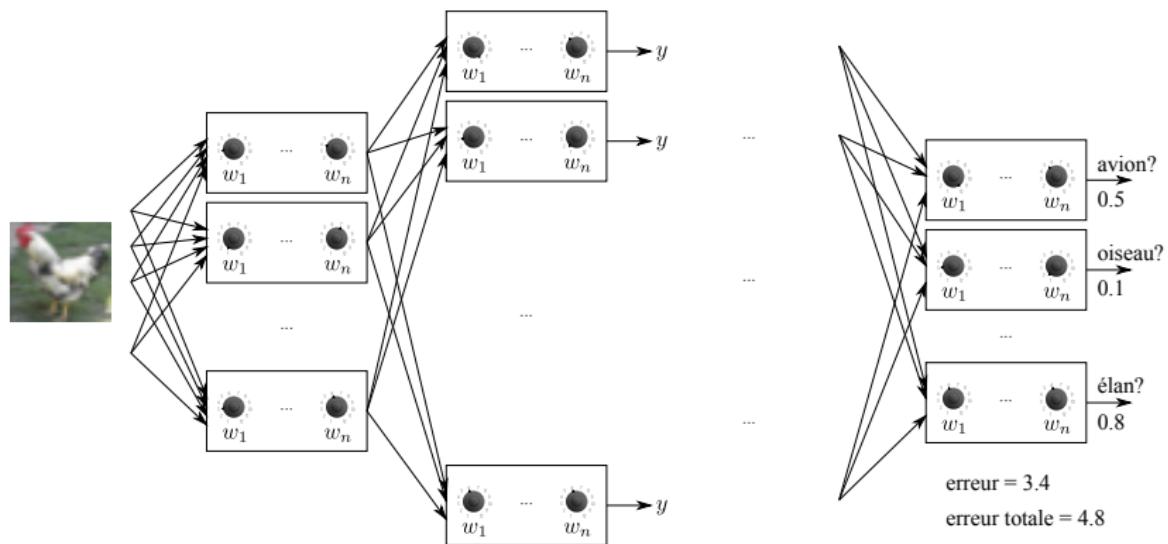
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



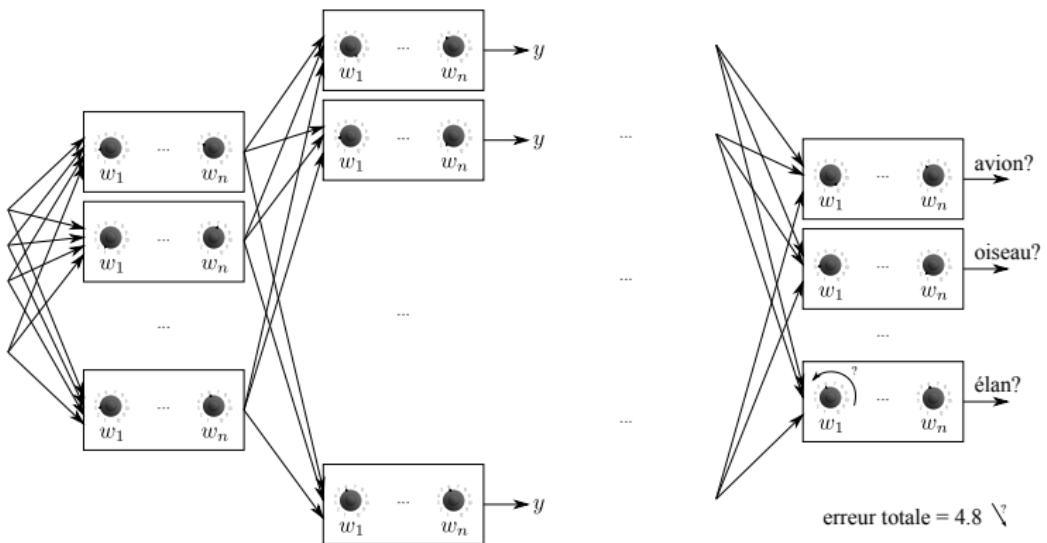
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



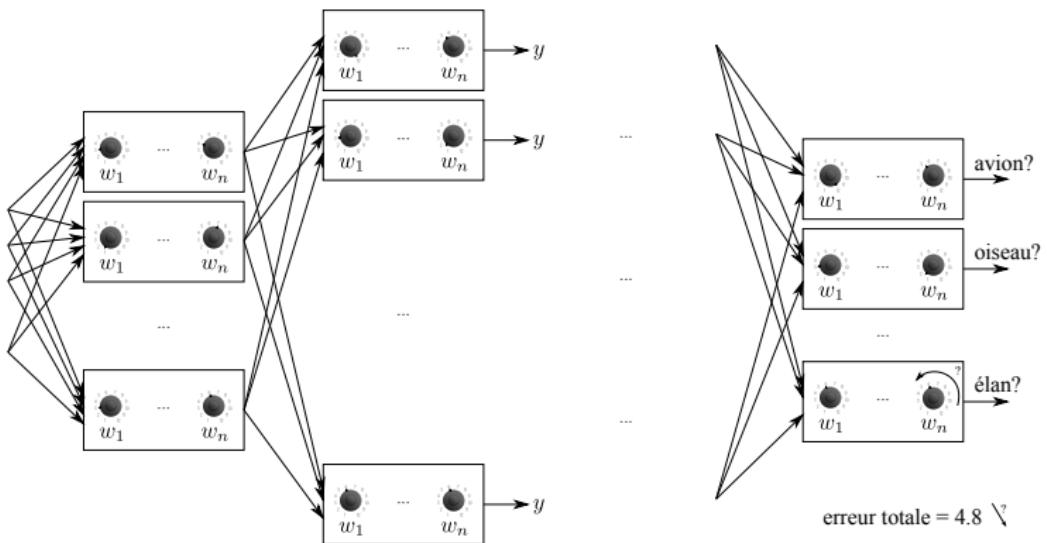
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



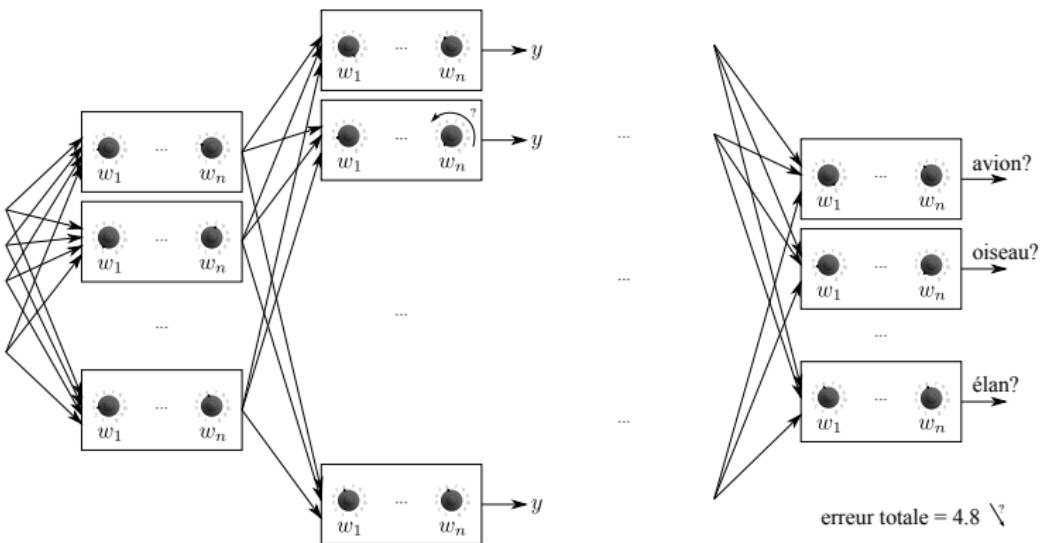
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



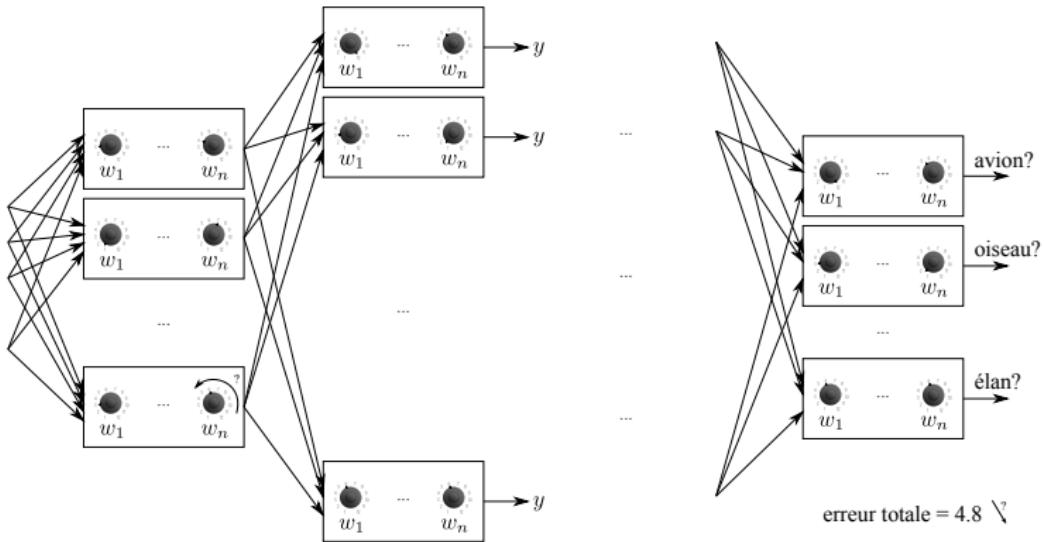
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



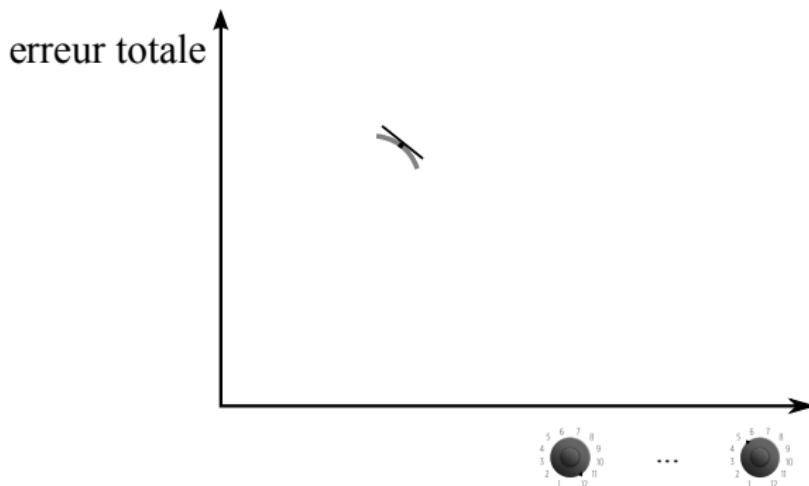
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



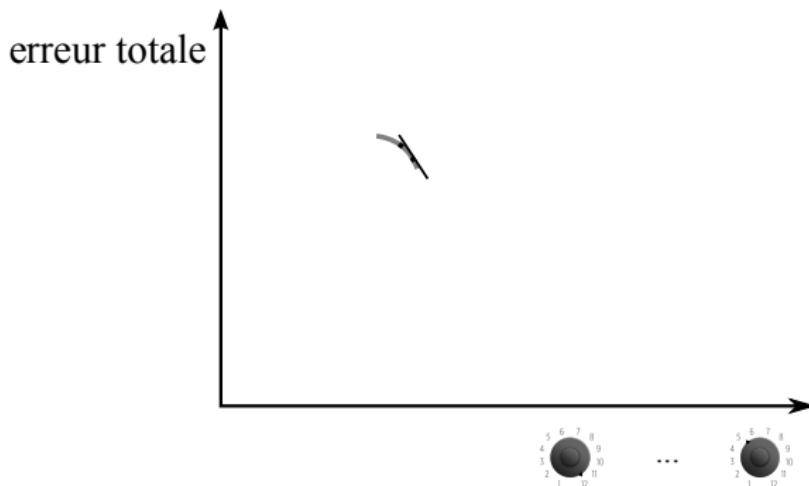
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



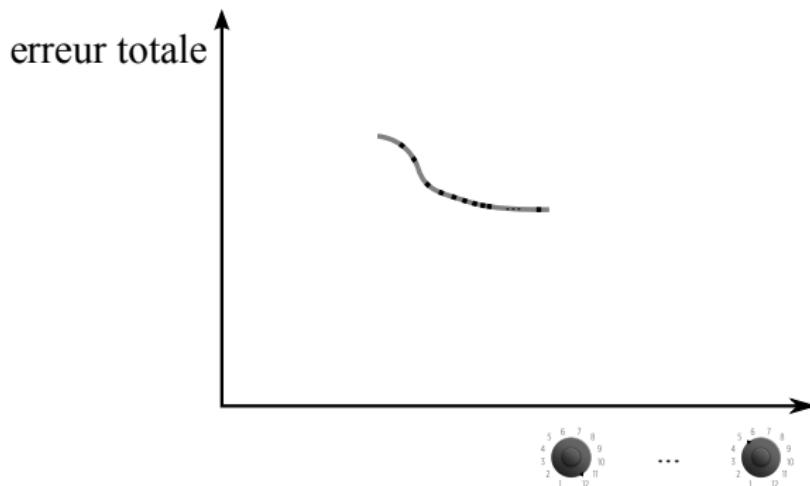
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



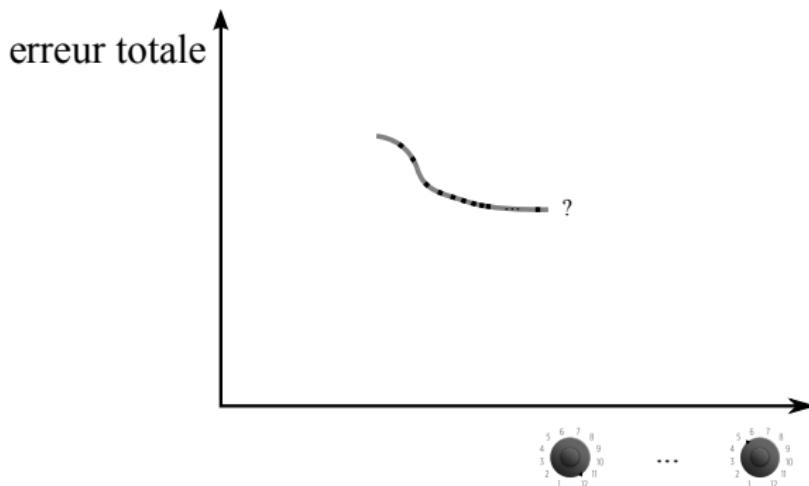
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



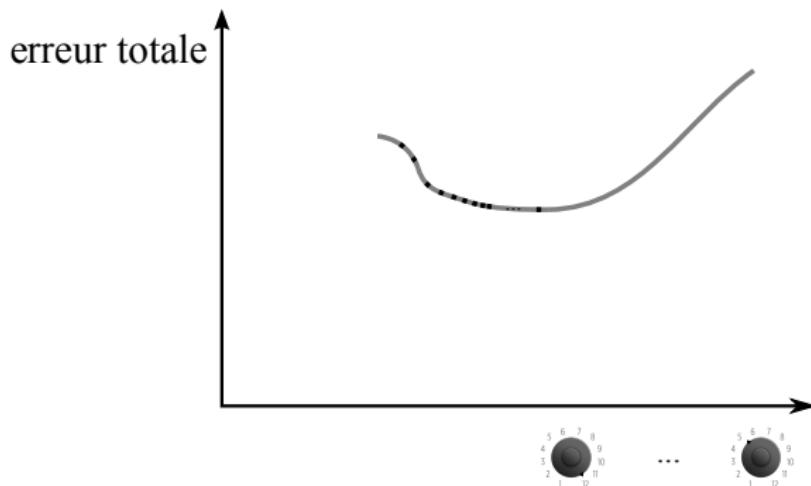
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



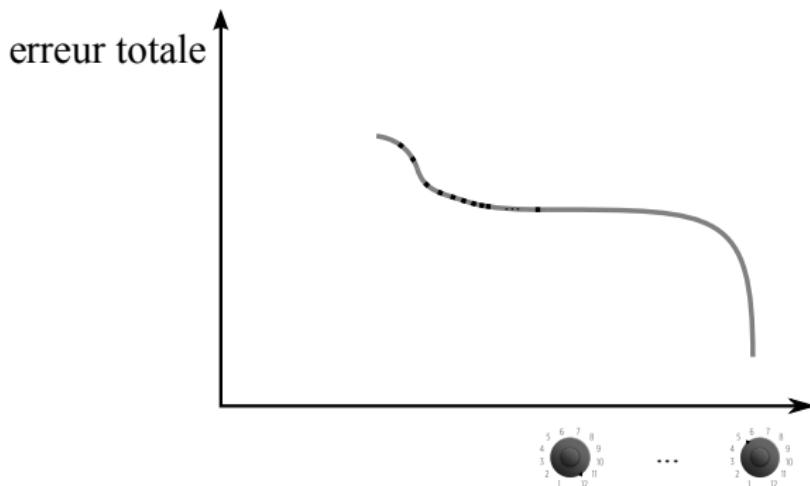
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



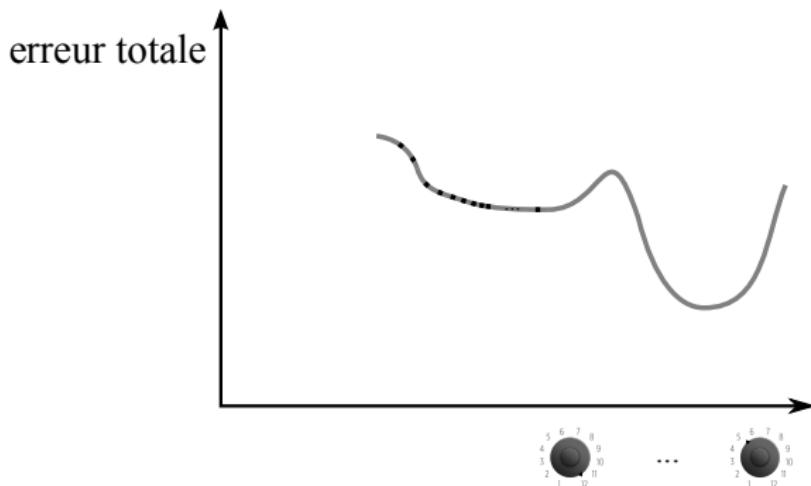
Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



Perceptron multi-couches - Werbos & Rumelhard (1984-1986)

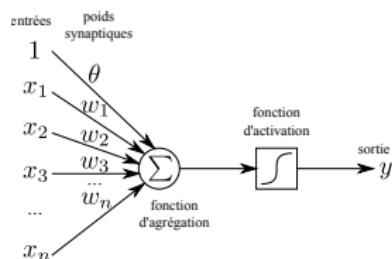


Perceptron multi-couches - Werbos & Rumelhard (1984-1986)



Sortie du neurone : $s = \sum_{i=0}^n w_i x_i$

$$y = \frac{1}{1 + e^{-s}}$$

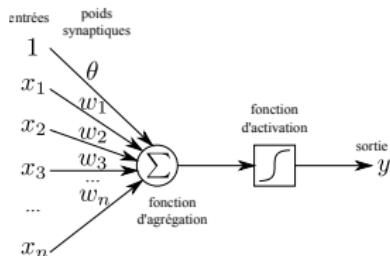


Sortie du neurone : $s = \sum_{i=0}^n w_i x_i$

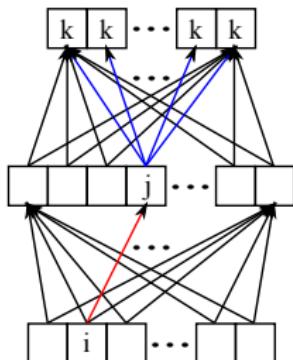
$$y = \frac{1}{1 + e^{-s}}$$

Apprentissage : On veut minimiser l'erreur quadratique $E = \frac{1}{2} \sum_{\{\mathbf{x}\}} (t - y)^2$ par descente de gradient stochastique : $\Delta \mathbf{W} = \eta \mathbf{X}(t - y)y(1 - y)$

$$\begin{aligned}\Delta w_i &= -\eta \frac{\partial E}{\partial w_i} \\ &= -\eta \frac{\partial E}{\partial s} \frac{\partial s}{\partial w_i} \\ &= \eta \sum_{\{\mathbf{x}\}} \frac{\partial y}{\partial s} (t - y) \frac{\partial s}{\partial w_i} \\ &= \sum_{\{\mathbf{x}\}} \eta y(1 - y)(t - y)x_i\end{aligned}$$



Réseau : connectivité complète entre la couche d'entrée, la(les) couche(s) cachée(s) et la couche de sortie

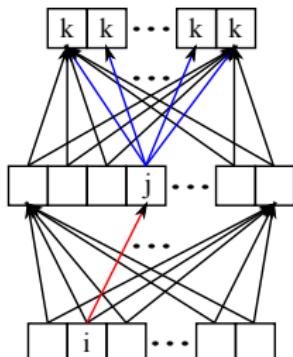


Réseau : connectivité complète entre la couche d'entrée, la(les) couche(s) cachée(s) et la couche de sortie

Apprentissage : On veut minimiser l'erreur quadratique

$$E = \frac{1}{2} \sum_{\{\mathbf{x}\}} (\mathbf{T} - \mathbf{Y})^2 \text{ par descente de gradient stochastique,}$$

on suppose connu les $\delta_k = -\frac{\partial E}{\partial s_k}$ de la couche supérieure



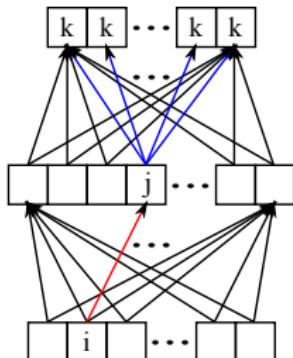
$$\begin{aligned} -\frac{\partial E}{\partial s_j} &= -\sum_k \frac{\partial E}{\partial s_k} \frac{\partial s_k}{\partial y_j} \frac{\partial y_j}{\partial s_j} \\ &= -\sum_k -\delta_k w_{jk} y_j (1 - y_j) \\ &= y_j (1 - y_j) \sum_k \delta_k w_{jk} \end{aligned}$$

Réseau : connectivité complète entre la couche d'entrée, la(les) couche(s) cachée(s) et la couche de sortie

Apprentissage : On veut minimiser l'erreur quadratique

$$E = \frac{1}{2} \sum_{\{\mathbf{x}\}} (\mathbf{T} - \mathbf{Y})^2 \text{ par descente de gradient stochastique,}$$

on suppose connu les $\delta_k = -\frac{\partial E}{\partial s_k}$ de la couche supérieure



$$\begin{aligned} -\frac{\partial E}{\partial s_j} &= -\sum_k \frac{\partial E}{\partial s_k} \frac{\partial s_k}{\partial y_j} \frac{\partial y_j}{\partial s_j} \\ &= -\sum_k -\delta_k w_{jk} y_j (1 - y_j) \\ &= y_j (1 - y_j) \sum_k \delta_k w_{jk} \end{aligned}$$

$$\begin{aligned} \Delta w_{ij} &= -\eta \frac{\partial E}{\partial w_{ij}} \\ &= -\eta \frac{\partial E}{\partial s_j} \frac{\partial s_j}{\partial w_{ij}} \\ &= \eta \delta_j x_i \end{aligned}$$

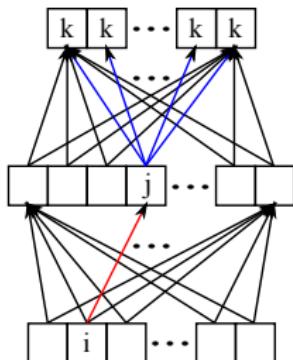
Réseau : connectivité complète entre la couche d'entrée, la(les) couche(s) cachée(s) et la couche de sortie

Apprentissage : Pour chaque entrée reçue :

- ① Calculer la sortie \mathbf{Y} du réseau par propagation (couche par couche) de l'activité
- ② Calculer l'erreur de la couche de sortie :

$$\delta_k = y_k(1 - y_k)(t_k - y_k)$$
- ③ Rétropopager l'erreur à travers chaque couche j du réseau

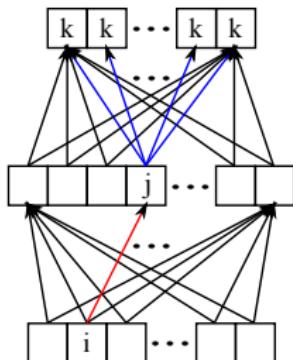
$$\delta_j = y_j(1 - y_j) \sum_k \delta_k w_{jk}$$
- ④ Modifier chaque poids $\Delta w_{ij} = \eta \delta_j x_i$



Réseau : connectivité complète entre la couche d'entrée, la(les) couche(s) cachée(s) et la couche de sortie

Apprentissage : Pour chaque entrée reçue :

- ① Calculer la sortie \mathbf{Y} du réseau par propagation (couche par couche) de l'activité
- ② Calculer l'erreur de la couche de sortie :
$$\delta_k = y_k(1 - y_k)(t_k - y_k)$$
- ③ Rétropopager l'erreur à travers chaque couche j du réseau
$$\delta_j = y_j(1 - y_j) \sum_k \delta_k w_{jk}$$
- ④ Modifier chaque poids $\Delta w_{ij} = \eta \delta_j x_i$



Propriétés

- Approximateur universel (avec fonction d'activation sigmoïde ou gaussienne)
- Convergence potentiellement peu efficace

Question

Qu'est ce que l'apprentissage profond ?

Question

Qu'est ce que l'apprentissage profond ?

L'apprentissage profond c'est

- un réseau avec au moins deux couches cachées
- qui permet une factorisation des représentations (plus grand pouvoir expressif),
voire un apprentissage de séquences
- qui pose la question de l'apprentissage : plateaux et problème du gradient
évanescence (termes de régularisation, gradient adaptatif, momentum, ...)

Question

Qu'est ce que l'apprentissage profond ?

L'apprentissage profond c'est

- un réseau avec au moins deux couches cachées
- qui permet une factorisation des représentations (plus grand pouvoir expressif),
voire un apprentissage de séquences
- qui pose la question de l'apprentissage : plateaux et problème du gradient
évanescence (termes de régularisation, gradient adaptatif, momentum, ...)

Question

Pourquoi maintenant ?

Question

Qu'est ce que l'apprentissage profond ?

L'apprentissage profond c'est

- un réseau avec au moins deux couches cachées
- qui permet une factorisation des représentations (plus grand pouvoir expressif),
voire un apprentissage de séquences
- qui pose la question de l'apprentissage : plateaux et problème du gradient
évanescence (termes de régularisation, gradient adaptatif, momentum, ...)

Question

Pourquoi maintenant ?

Il est de nouveau exploité car

- les données d'apprentissage sont disponibles
- les puissances de calcul sont disponibles
- on maîtrise (un peu) mieux l'apprentissage
- c'est à la mode

Bilan apprentissage de réseaux de neurones à fréquence de décharge

Règle	Supervisé	Local	Croissant	En ligne	Réseau	Sortie	Propriété
Hebb	X	✓	X	✓			Assemblée de neurones
Oja	X	✓	X	✓	Inhibiteur		ACP
Hopfield	X	✓	X	✓/X	Symétrique		Mémoire associative
Kohonen	X	X	X	✓	Grille		Projection topologique
NG	X	X	X	✓			Quantification vectorielle
GNG	X	X	✓	✓			Quantification vectorielle Apprentissage topologie
Perceptron	✓	✓	X	✓/X			Classifieur linéaire
MLP	✓	X	X	✓/X	Couches		Approximateur universel

Bilan apprentissage de réseaux de neurones à fréquence de décharge

Règle	Supervisé	Local	Croissant	En ligne	Réseau	Sortie	Propriété
Hebb	X	✓	X	✓			Assemblée de neurones
Oja	X	✓	X	✓	Inhibiteur		ACP
Hopfield	X	✓	X	✓/X	Symétrique		Mémoire associative
Kohonen	X	X	X	✓	Grille		Projection topologique
NG	X	X	X	✓			Quantification vectorielle
GNG	X	X	✓	✓			Quantification vectorielle Apprentissage topologie
Perceptron	✓	✓	X	✓/X			Classifieur linéaire
MLP	✓	X	X	✓/X	Couches		Approximateur universel

Remarque

Les réseaux de neurones sont une façon (parmi d'autres) de faire de l'apprentissage de combinaisons de fonction linéaires et non linéaires.

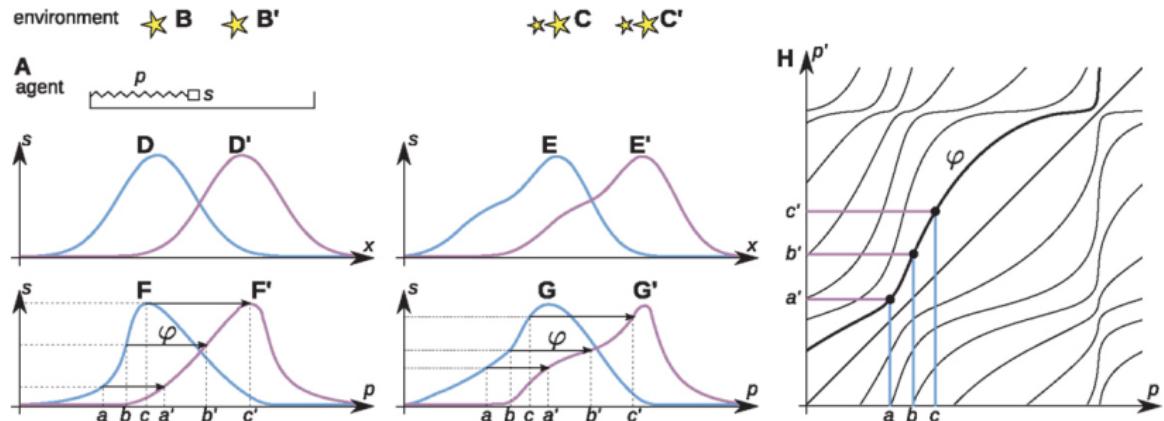
Question

Que faut-il apprendre pour rendre un agent intelligent ?

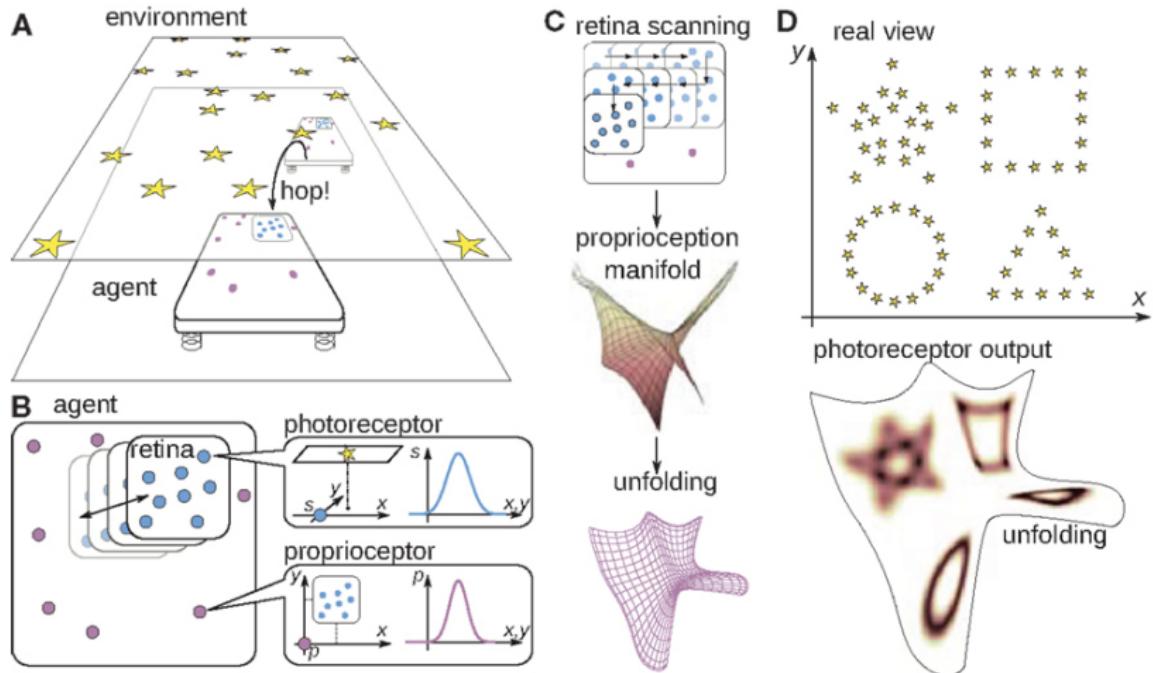
Question

L'IA se résume-t-elle à de l'apprentissage ?

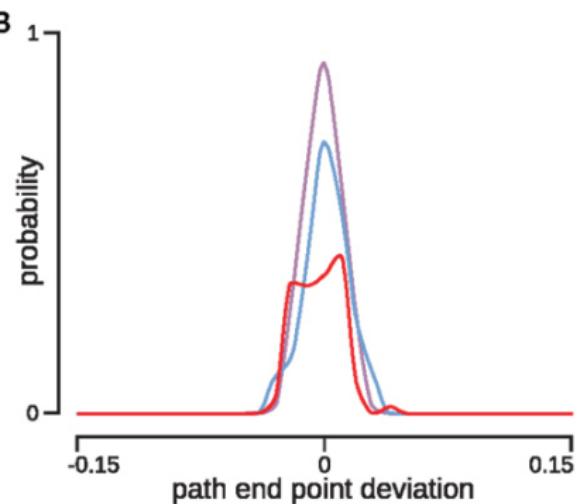
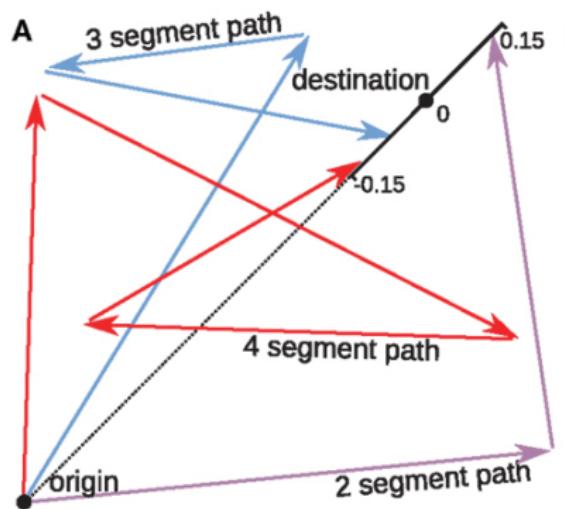
Théories sensori-motrices - Incorporation - Perception active



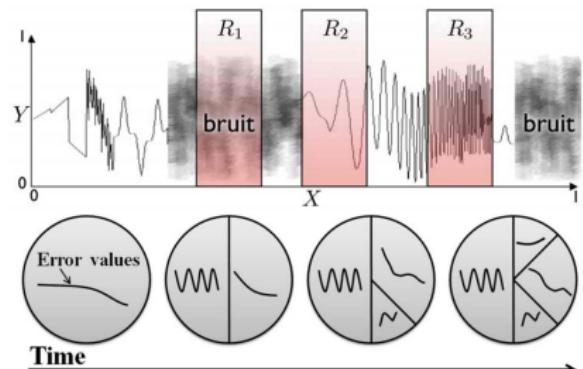
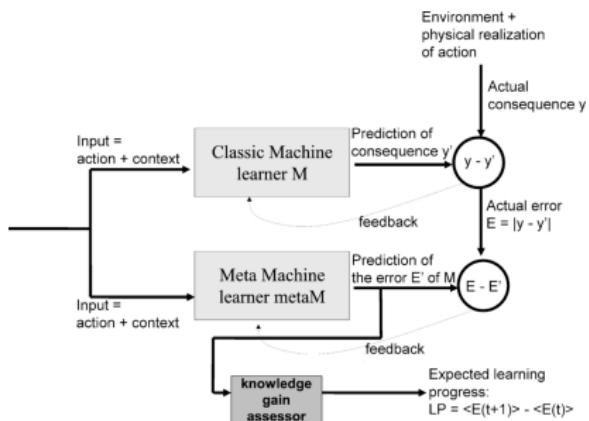
Théories sensori-motrices - Incorporation - Perception active



Théories sensori-motrices - Incorporation - Perception active



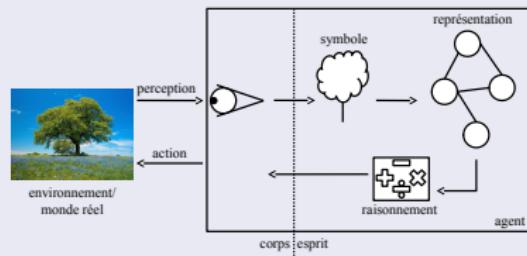
Motivation intrinsèque - Curiosité artificielle



P.-Y. Oudeyer et al (2007) Intrinsic Motivation Systems for Autonomous Mental Development

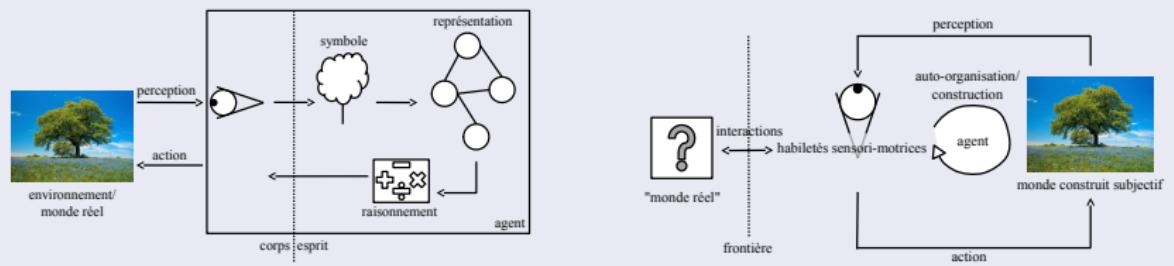
A. Baranès (2011) Motivations Intrinsèques et Contraintes Maturationnelles pour l'Apprentissage Sensorimoteur

Cognitivisme vs constructivisme



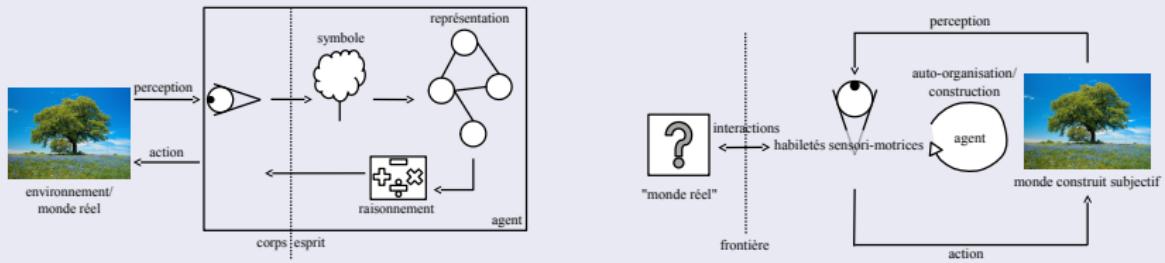
Constructivisme - Développemental

Cognitivisme vs constructivisme



Constructivisme - Développemental

Cognitivisme vs constructivisme



Apprentissage développemental

PHYSICAL DEVELOPMENT	Average age skills begin	3 months	6 months	9 months	1 year	2 years	3 years	5 years	What to do if a child is behind
Head and trunk control	lifts head part way up	holds head up briefly	holds head up high and well	turns head and shifts weight	NO holds head up well when lifted	YES moves and holds head easily in all directions			Activities to improve head and trunk control (see p. 397).
Rolling		rolls easily to back	rolls back to belly	rolls over and over easily in play					Activities to develop rolling and back rolling (see p. 394).
Sitting	sits only with full support	sits with full support	sits with some support	begins to sit without support	sits well without support	twists and moves easily while sitting			Work on sitting. Special seating if needed (p. 398).
Crawling and walking		begins to crawl	pulls to standing	takes steps	walks	walks easily backward			Activities to improve balance (see p. 396).
Arm and hand control	grasps finger put into hand	reaches and grasps with whole hand	passes object from one hand to other	grasps with thumb and forefinger	easily moves fingers back and forth and nose to moving object	throws and catches ball			Eye-hand activities. Use toys and games to improve hand and finger control (see p. 305).
Seeing	follows close object with eyes	recognizes different faces	eyes focus on far object	looks at small three-dimensional shapes	sees small shapes clearly (see p. 453 for test).				Have eyes checked (see p. 432); if poor, see Chapter 30.
Hearing	answers or cries at a loud noise	enjoys rhythmic music	understands simple words	hears clearly and understands most simple language					Have hearing checked. If poor, see Chapter 31.

