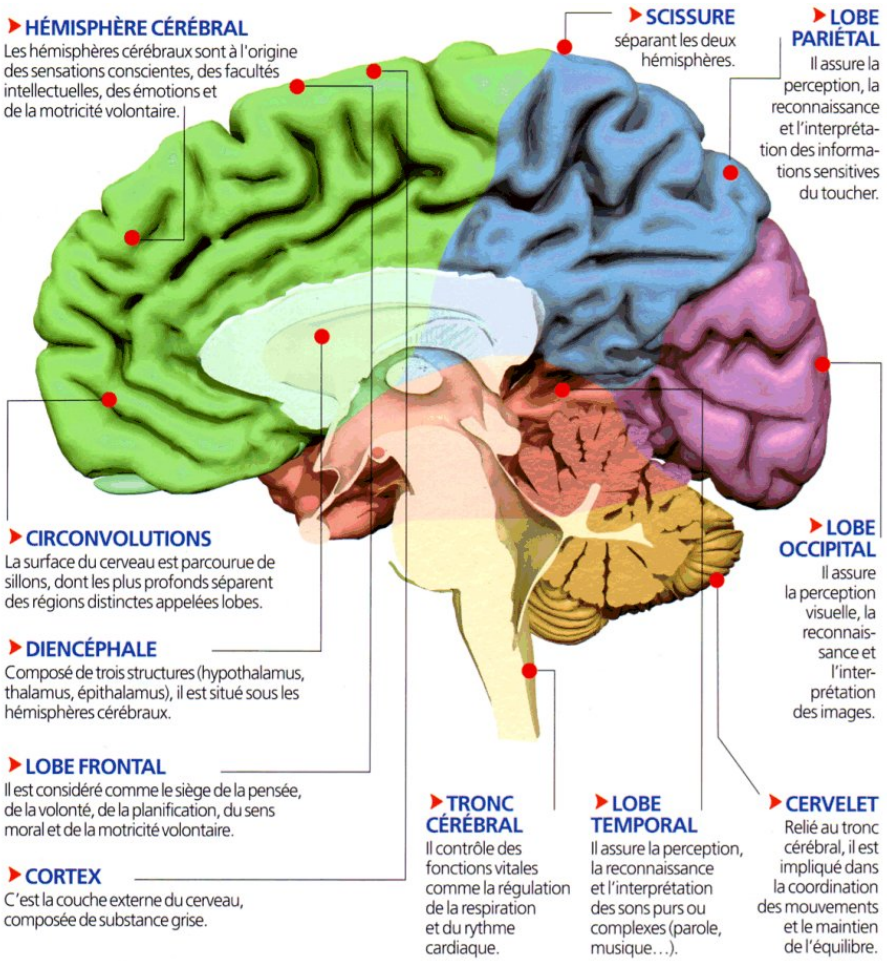
Techniques de l'Intelligence Artificielle
Réseaux de neurones

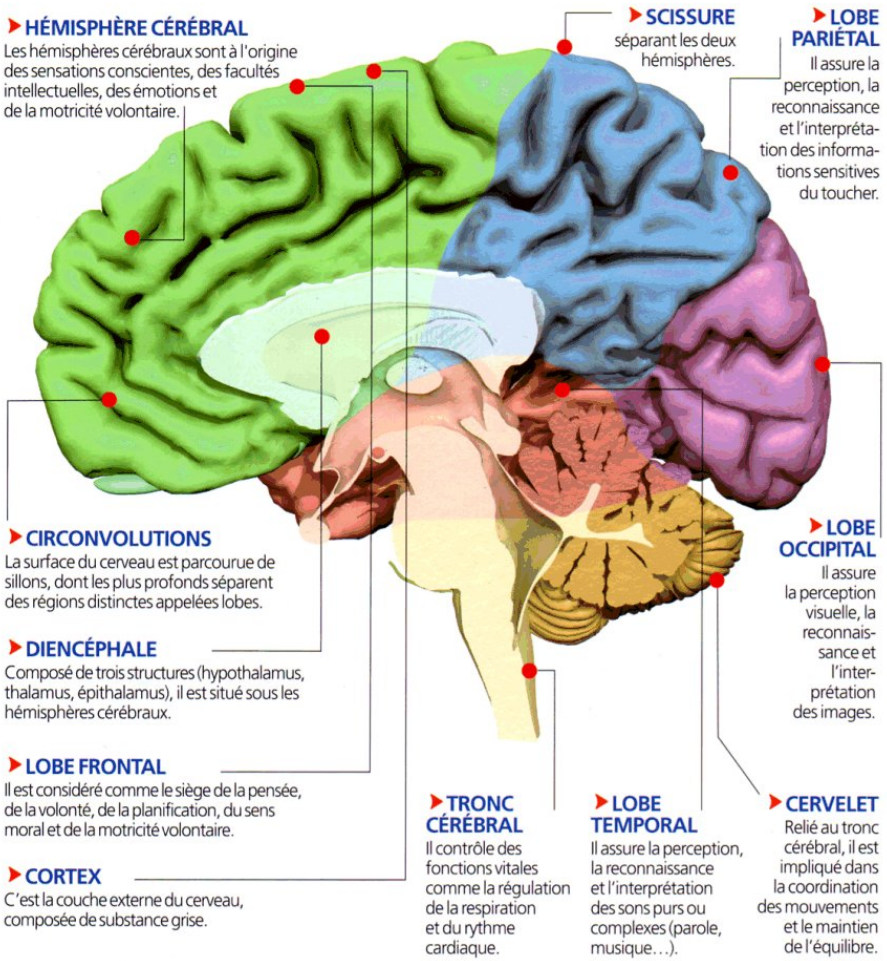
Maxime Guériau
Mathieu Lefort

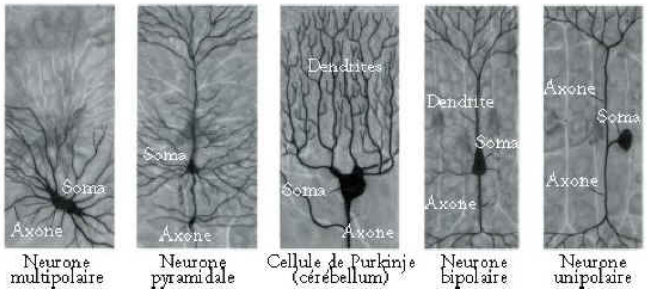
9 mars 2017

Un peu de biologie ...

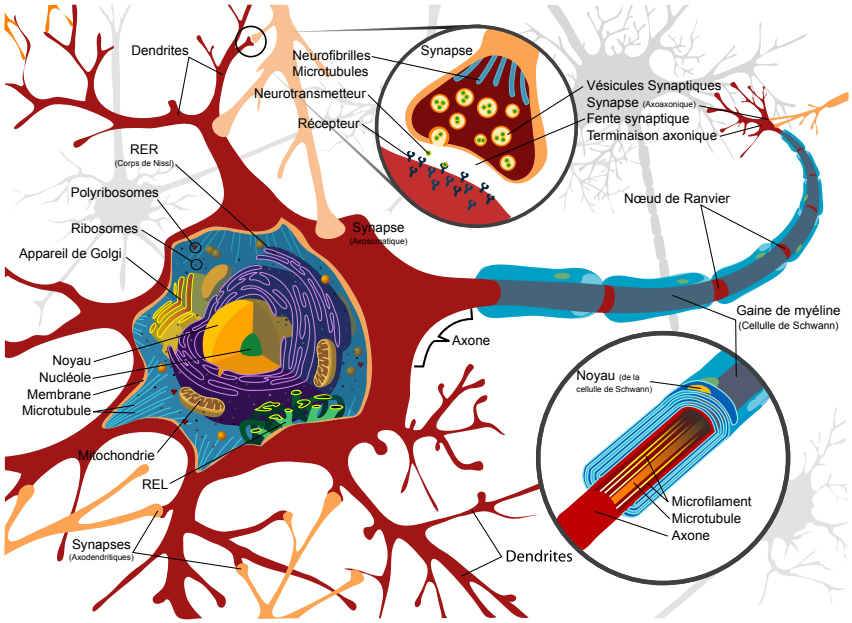


Un peu de biologie ...

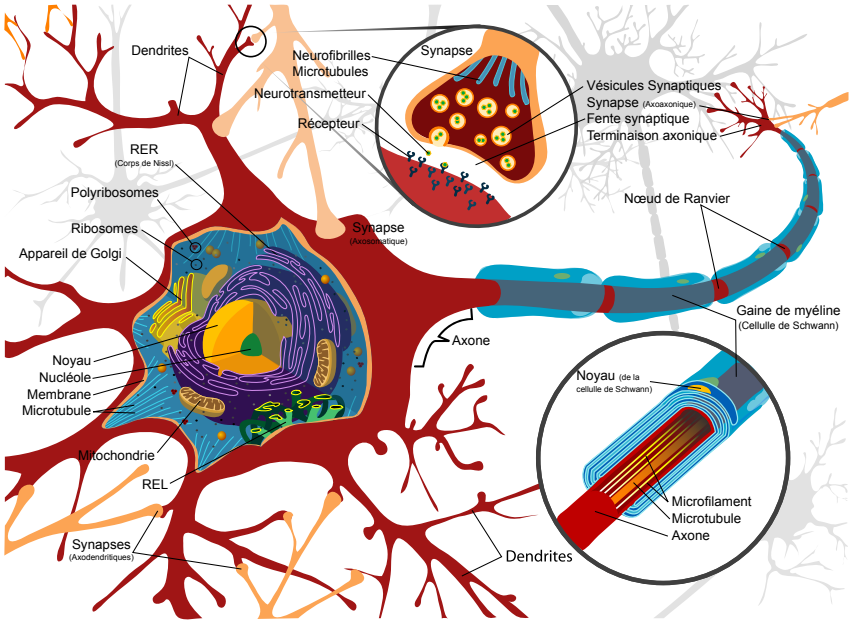


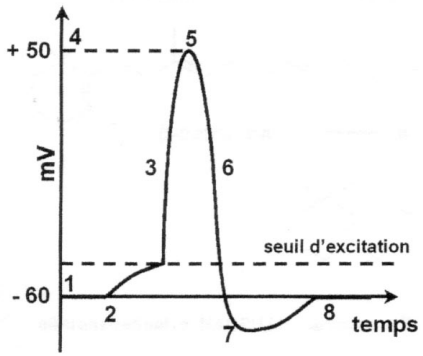


Un peu de biologie ...



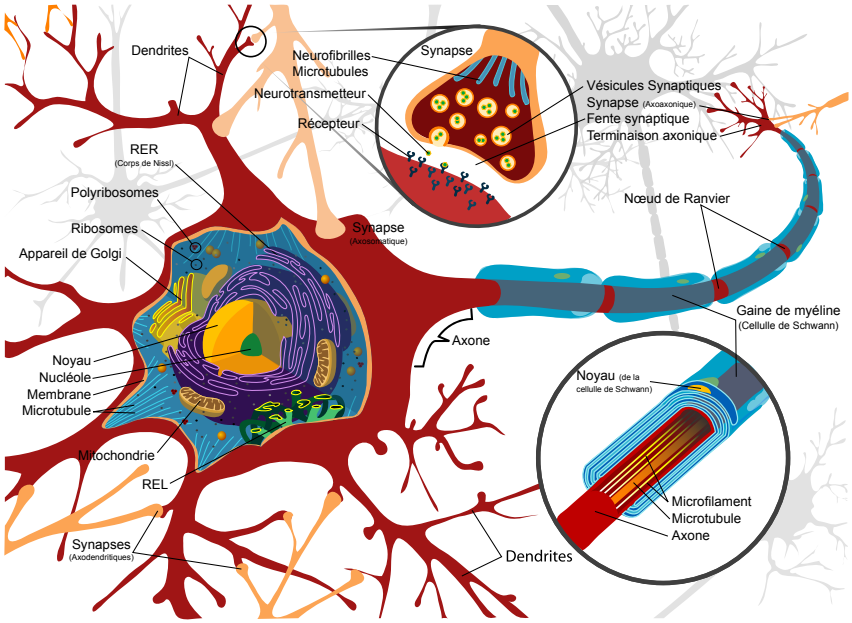
Un peu de biologie ...

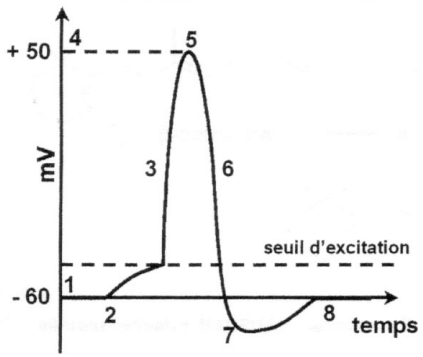




1. potentiel de repos ou potentiel de membrane
2. stimulation
3. activation des canaux sodium : dépolarisation
4. potentiel d 'équilibre du sodium
5. inactivation des canaux sodium
6. activation des canaux potassium : repolarisation
7. hyperpolarisation
8. retour au potentiel de membrane

Un peu de biologie ...



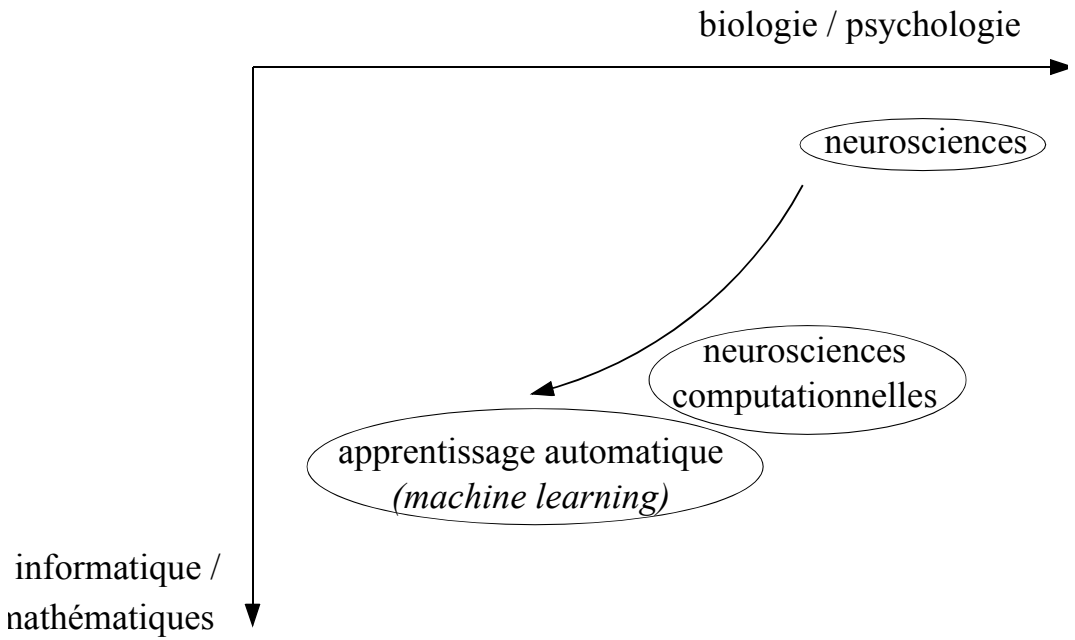


1. potentiel de repos ou potentiel de membrane
2. stimulation
3. activation des canaux sodium : dépolarisation
4. potentiel d 'équilibre du sodium
5. inactivation des canaux sodium
6. activation des canaux potassium : repolarisation
7. hyperpolarisation
8. retour au potentiel de membrane

Question

Pourquoi je vous parle de biologie ?

Plan du cours



Contenu du cours

Ce dont je vais vous parler

- Électricité
- Biologie
- Mathématiques
- IA
- Apprentissage
- Réseaux de neurones

Ce que vous devez retenir

- Qu'est ce qu'un neurone ?
- Qu'est ce qu'un réseau de neurones ?
- Quand et pourquoi les utiliser (avantages/inconvénients) ?
- Quels liens avec le reste de l'IA ?

Remarque

Ce n'est pas un cours d'apprentissage automatique mais une introduction/panorama aux réseaux de neurones en lien avec les problématiques de l'intelligence artificielle.

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 : $y_{n+1} = y_n + \delta t \phi(t, y_n, h)$. 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 : $y_{n+1} = y_n + \delta t \phi(t, y_n, h)$. 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

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

Problème

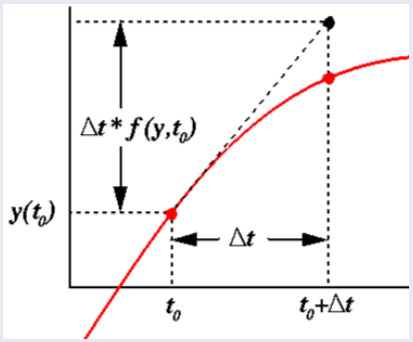
- Soit l'équation différentielle $\frac{dy(t)}{dt} = f(t, y(t))$ décrivant l'évolution de la grandeur $y(t)$ au cours du temps (qui vaut y_0 à t_0)
- On approxime $y(t_n) = y(t_0 + n\delta t)$ de manière itérative : $y_{n+1} = y_n + \delta t \phi(t, y_n, h)$. Cette mise à jour sera dite d'ordre p si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

Développement de Taylor

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

Méthode d'Euler

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

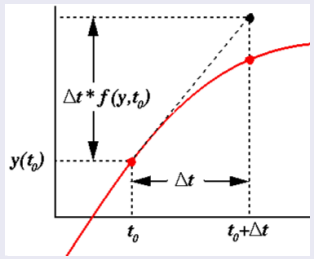


Problème

- 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 : $y_{n+1} = y_n + \delta t \phi(t, y_n, h)$. Cette mise à jour sera dite d'ordre p si $y(t_{n+1}) - y_{n+1} = O(\delta t^{p+1})$

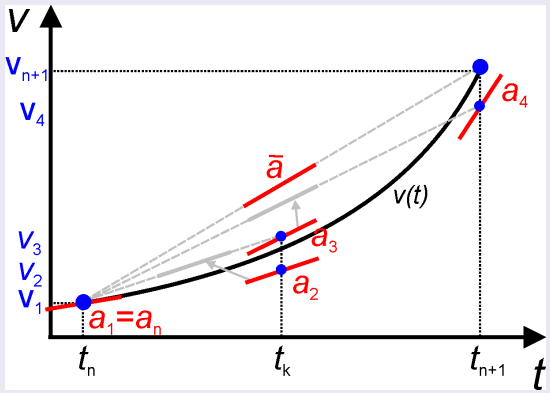
Méthode d'Euler

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

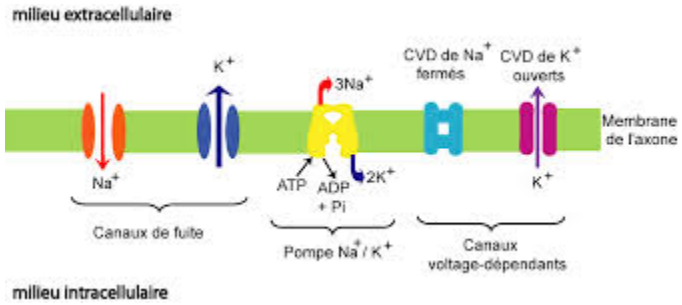


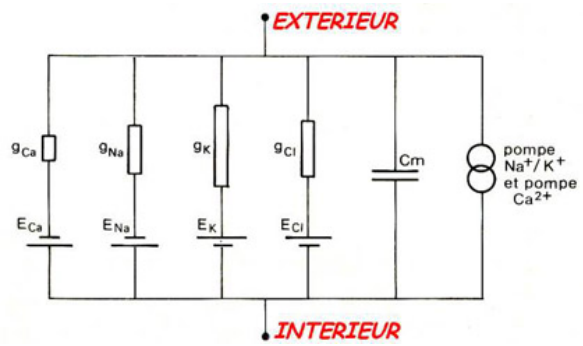
Méthode de Runge-Kutta (d'ordre 4)

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



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





$$\overbrace{I}^{\text{courant injecté}} = \overbrace{C_m}^{\text{capacité de membrane}} \frac{d\overbrace{V_m}^{\text{potentiel de membrane}}}{dt} + g_K n^4 (V_m - V_K) + g_{Na} m^3 h (V_m - V_{Na}) + g_l (V_m - V_l)$$

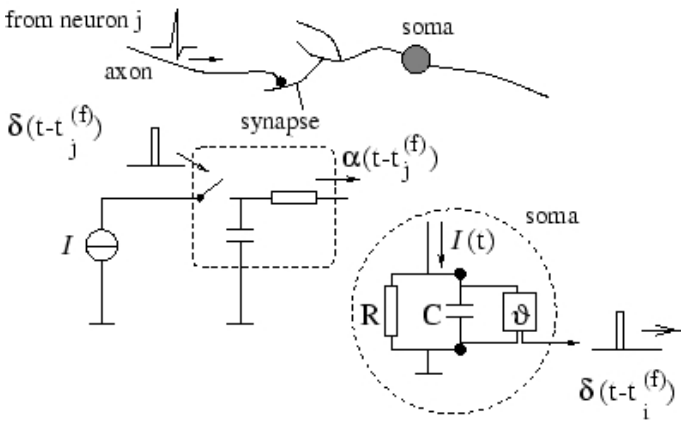
$$\frac{d\star}{dt} = \alpha_\star(V_m)(1-\star) - \beta_\star(Vm)\star \quad \text{avec } \star \in n, m, h$$

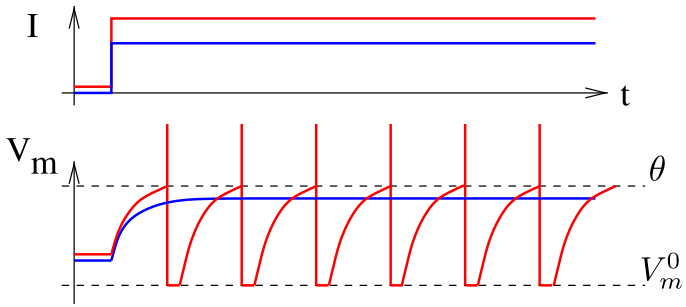
$$\alpha_n(V_m) = \frac{0.01(V_m - 10)}{e^{\frac{V_m - 10}{10}} - 1} \qquad \beta_n(V_m) = 0.125 e^{\frac{V_m}{80}}$$

$$\alpha_m(V_m) = \frac{0.1(V_m - 25)}{e^{\frac{V_m - 25}{10}} - 1} \qquad \beta_m(V_m) = 4 e^{\frac{V_m}{18}}$$

$$\alpha_h(V_m) = 0.07 e^{\frac{V_m}{20}} \qquad \beta_h(V_m) = \frac{1}{e^{\frac{V_m - 30}{10}} + 1}$$

Neurones à *spikes* - Modèle intègre et tire à fuite





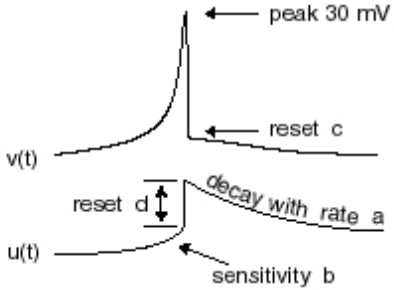
$$I = \frac{V_m}{R} + C\frac{dV_m}{dt} \text{ ou écrit autrement } \tau_m \frac{dV_m}{dt} = \overbrace{-V_m}^{\text{fuite}} + \overbrace{RI}^{\text{signal}}$$

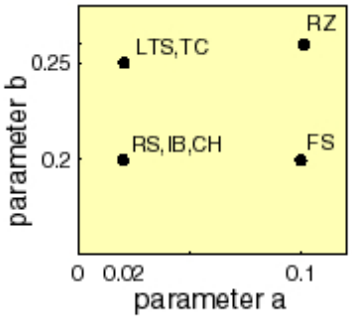
Un spike est émis à t^f si $V_m(t^f) = \theta$ et le potentiel de membrane est alors remis au potentiel de repos $V_m = V_m^0$ (éventuellement pendant toute la période réfractaire).

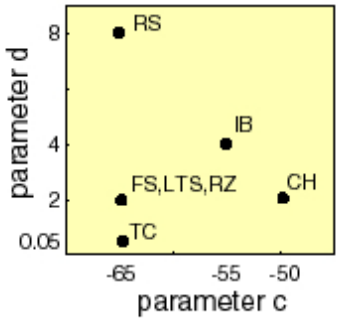
Neurones à *spikes* - Modèle d'Izhikevich (2003)

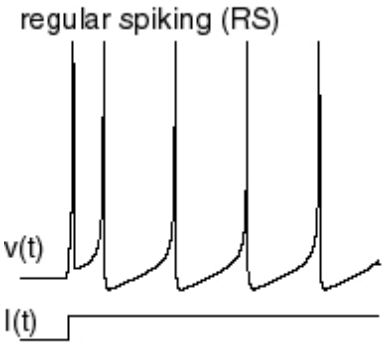
$v' = 0.04v^2 + 5v + 140 - u + I$
$u' = a(bv - u)$

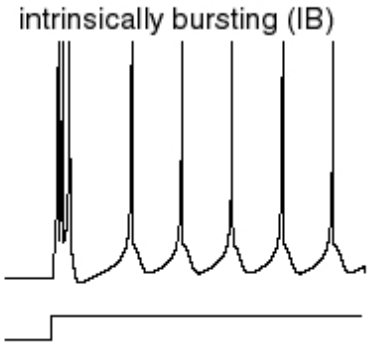
if $v = 30$ mV,
then $v \leftarrow c$, $u \leftarrow u + d$

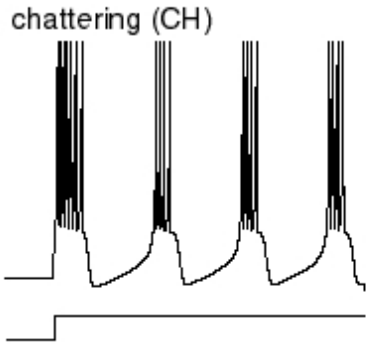


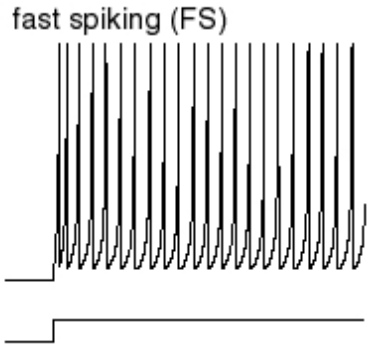


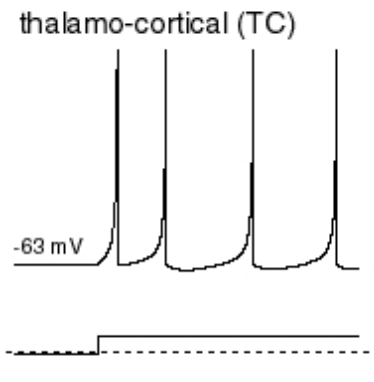


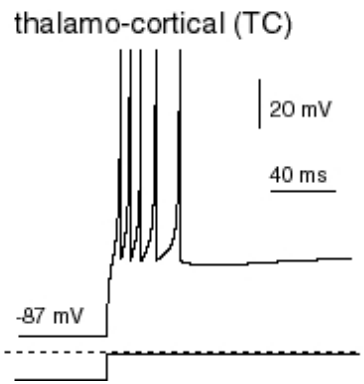


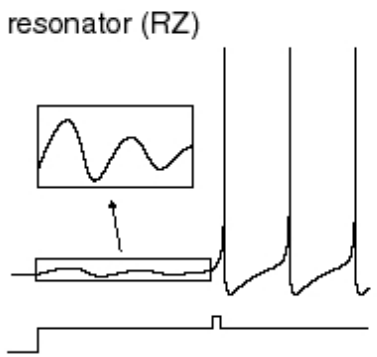


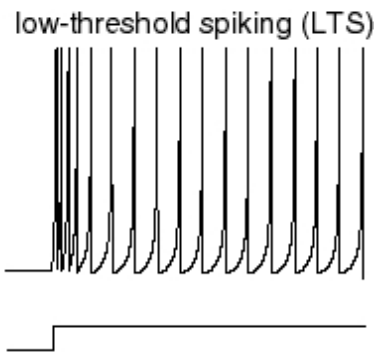




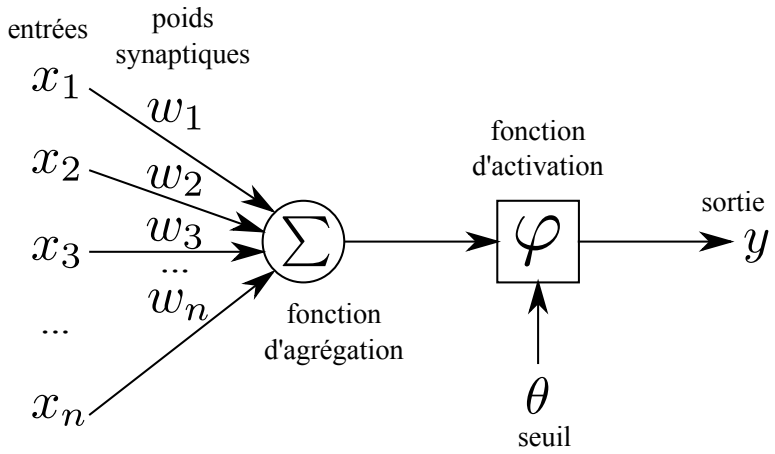








Neurones à fréquence de décharge - McCulloch et Pitts (1943)



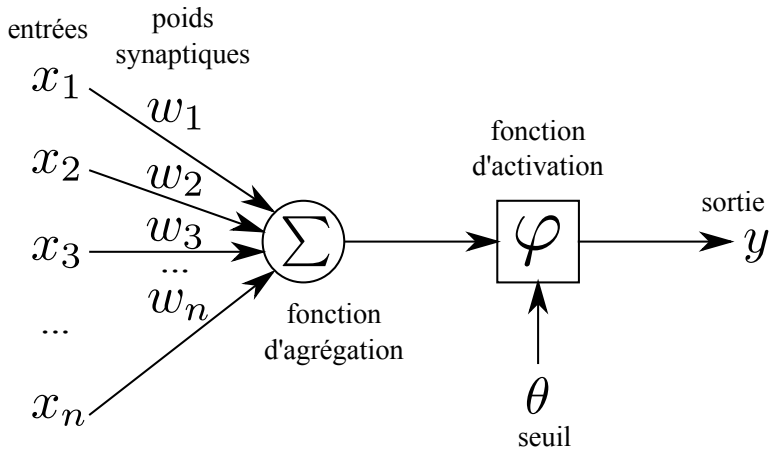
$y = \varphi(\sum_i w_i x_i - \theta)$ ou écrit autrement

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

Remarques

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

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



$y = \varphi(\sum_i w_i x_i - \theta)$ ou écrit autrement

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

Remarques

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

Question

Où sont passés les spikes ?

Neurones à *spikes* vs neurones à fréquence de décharge

Modèle neurone	*Spikes*	Fréquence de décharge
Transmission information		
Représentation information		
Évolution		
Parallèle biologique		
Simulation informatique		

Neurones à *spikes* vs neurones à fréquence de décharge

Modèle neurone	*Spikes*	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	*Spikes*	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	*Spikes*	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	*Spikes*	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	*Spikes*	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	*Spikes*	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, ... ?

Neurones à *spikes* vs neurones à fréquence de décharge

Modèle neurone	*Spikes*	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

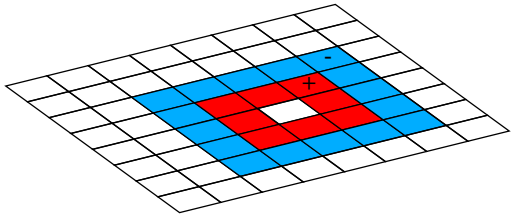
Question

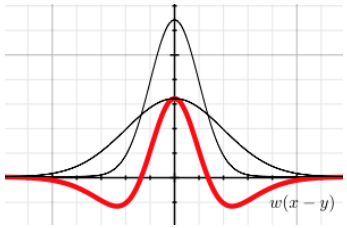
Quelles hypothèses sur la conscience, l'intelligence, ... ?

Question

Et maintenant comment on fait un réseau ?

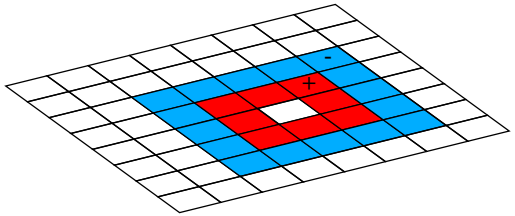
Dynamic Neural Field - Wilson et Cowan (1973)

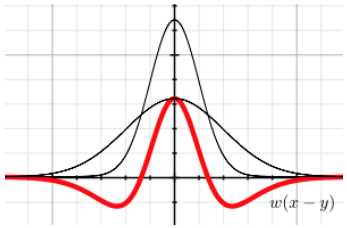




$$\tau \frac{d \overbrace{V_m(x,t)}^{\text{potentiel de membrane}}}{dt} = \overbrace{-V_m(x,t)}^{\text{terme de fuite}} + \overbrace{\int_{-\infty}^{+\infty} w(x-y) f(V_m(y, t - \frac{|x-y|}{v}))dy}^{\text{influence latéral}} + \overbrace{I(x,t)}^{\text{excitation}} + \overbrace{h}^{\text{potentiel de repos}}$$

Dynamic Neural Field - Wilson et Cowan (1973)



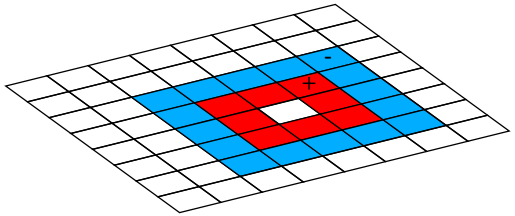


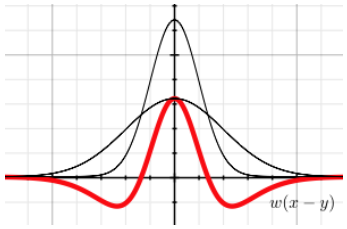
$$\tau \frac{d \overbrace{V_m(x,t)}^{\text{potentiel de membrane}}}{dt} = \overbrace{-V_m(x,t)}^{\text{terme de fuite}} + \overbrace{\int_{-\infty}^{+\infty} w(x-y) f(V_m(y, t - \frac{|x-y|}{v}))dy}^{\text{influence latéral}} + \overbrace{I(x,t)}^{\text{excitation}} + \overbrace{h}^{\text{potentiel de repos}}$$

Question

Une idée de ce que cela donne ?

Dynamic Neural Field - Wilson et Cowan (1973)

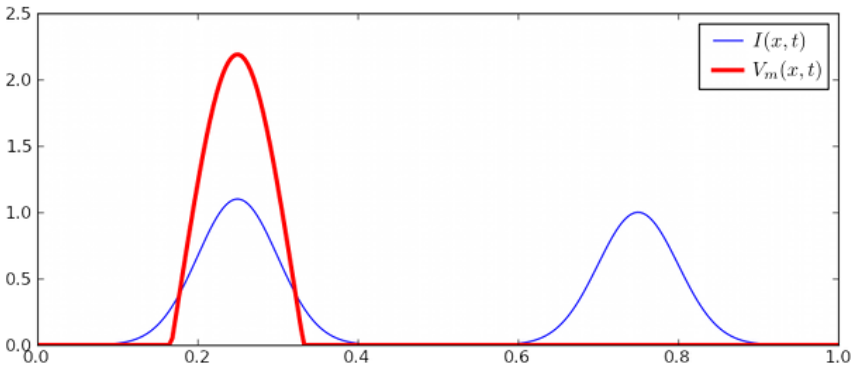


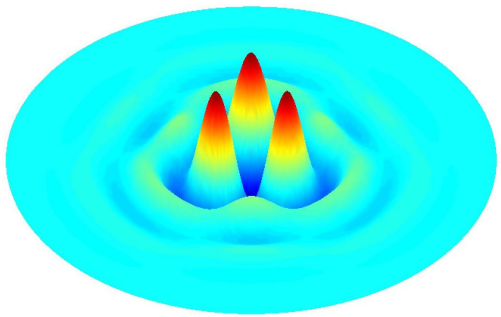


$$\tau \frac{d \overbrace{V_m(x,t)}^{\text{potentiel de membrane}}}{dt} = \overbrace{-V_m(x,t)}^{\text{terme de fuite}} + \overbrace{\int_{-\infty}^{+\infty} w(x-y) f(V_m(y, t - \frac{|x-y|}{v}))dy}^{\text{influence latéral}} + \overbrace{I(x,t)}^{\text{excitation}} + \overbrace{h}^{\text{potentiel de repos}}$$

Question

Une idée de ce que cela donne ?





Démo

Apprentissage

Question

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

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, ...

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, ...

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 vidéos

Question

Comment ?

Question

Comment ?

Moyens

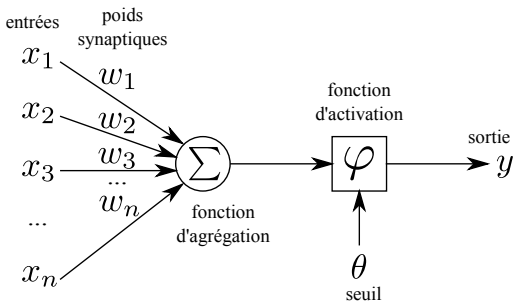
- plasticité synaptique : modification des poids du réseau
- neuro-genèse : rajout d'unités dans le réseau
- dendrito-genèse : rajout de connexions dans le réseau
- homéostasie : régulation de l'activité dans le neurone/réseau

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"



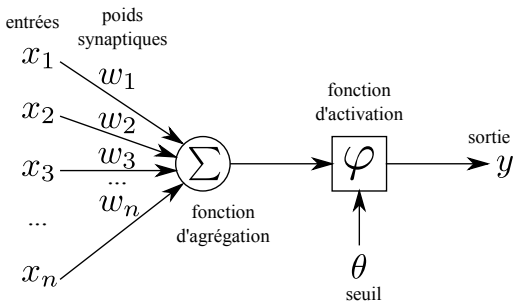
$\overbrace{w_i(t+1) - w_i(t)}^{\Delta w_i} = \eta x_i y, \forall i$ ou écrit autrement $\Delta \mathbf{W} = \eta \mathbf{X} y$

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

η est appelé le taux d'apprentissage

Apprentissage non supervisé - Règle de Hebb (1949)

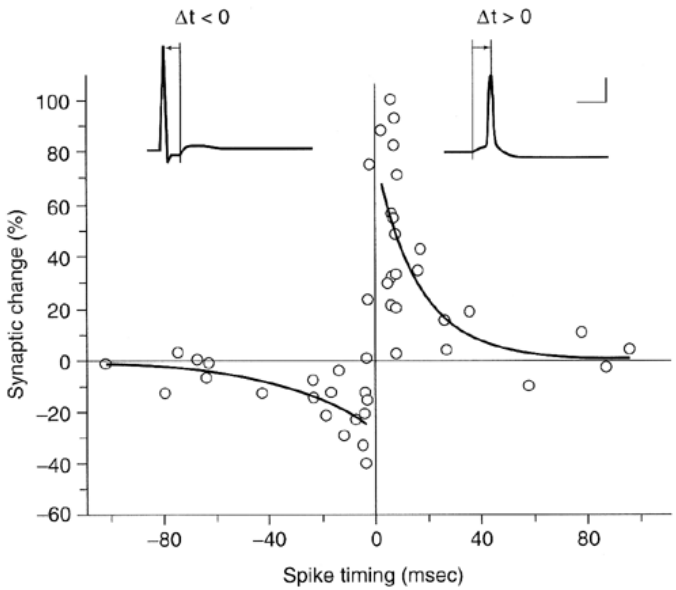
"Cells that fire together, wire together"



$$\overbrace{w_i(t+1) - w_i(t)}^{\Delta w_i} = \eta x_i y, \forall i$$ ou écrit autrement $\Delta \mathbf{W} = \eta \mathbf{X} y$

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

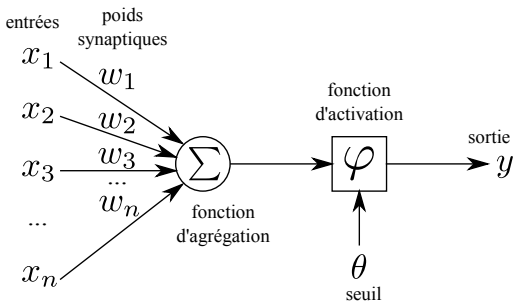
η est appelé le taux d'apprentissage



Spike Timing Dependent Plasticity

Apprentissage non supervisé - Règle de Hebb (1949)

"Cells that fire together, wire together"



$w_i(t+1) - w_i(t)$

$\overbrace{\Delta w_i} = \eta x_i y, \forall i$ 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

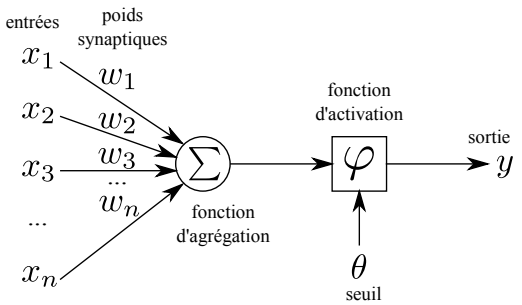
Algorithme (en ligne)

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"



$\overbrace{w_i(t+1) - w_i(t)}^{\Delta w_i} = \eta x_i y, \forall i$ 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 ?

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 :

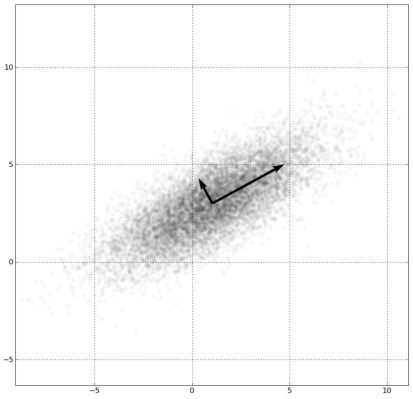
$$
\begin{aligned}
w_i(t+1) &= \frac{w_i + \eta x_i y}{\sqrt{\sum_j (w_j + \eta x_j y)^2}} \\
&= \frac{w_i}{\sqrt{\sum_j w_j^2}} + \eta \left(\frac{x_i y}{\sqrt{\sum_j w_j^2}} - \frac{w_i \sum_j w_j x_j y}{\sqrt{\sum_j w_j^2}^3} \right) + O(\eta^2) \qquad \text{(Taylor)} \\
&\approx w_i + \eta y (x_i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

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

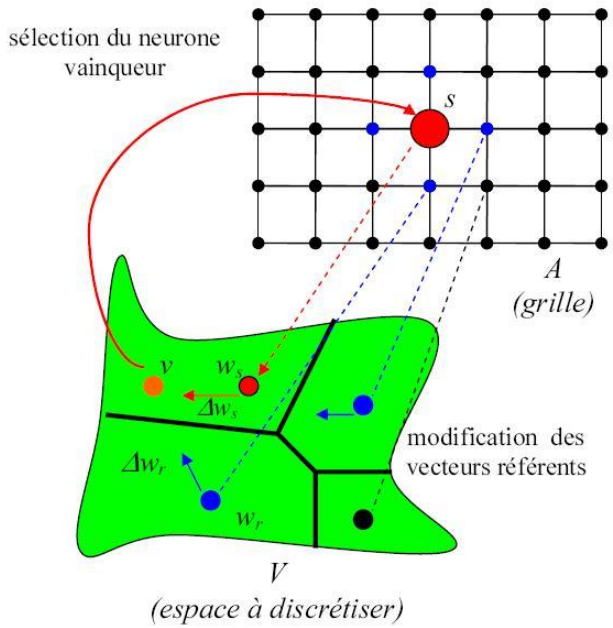
Bilan apprentissage de réseaux de neurones à fréquence de décharge

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

$$\underline{\textbf{Sortie du neurone } j} : y_j = \begin{cases} 1 & \text{si } ||\mathbf{W_j} - \mathbf{X}|| = \min\limits_{j'} ||\mathbf{W_{j'}} - \mathbf{X}|| \\ 0 & \text{sinon} \end{cases}$$

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)

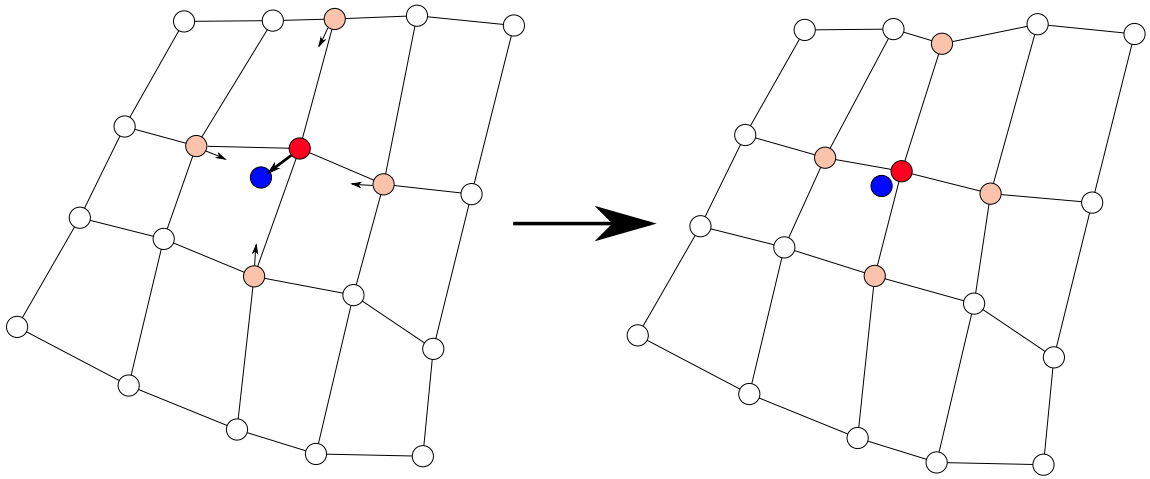


Apprentissage non supervisé - Carte auto-organisatrice de Kohonen (1984)

Principe :

Deux espaces à différencier :

- espace des neurones (positions des neurones = poids des neurones dans l'espace)
- espace d'entrée



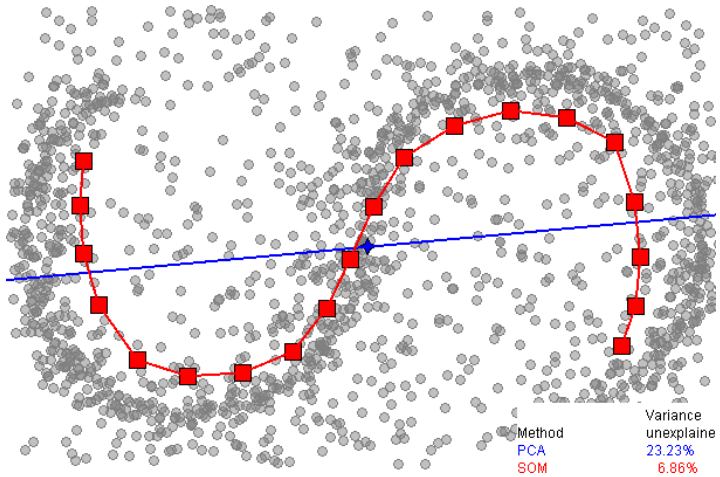
Exemple :

- nouvelle entrée (en bleu)
- modification des poids du neurone "gagnant" (le plus proche, en rouge)
- modification des poids de ses voisins (définis par les liens)

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } ||\mathbf{W_j} - \mathbf{X}|| = \min\limits_{j'} ||\mathbf{W_{j'}} - \mathbf{X}|| \\ 0 & \text{sinon} \end{cases}$

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 $E = \sum\limits_{\mathbf{X}} ||\mathbf{W_{j^*}} - \mathbf{X}||^2$ (+ terme topologique)
- Convergence garantie si $\eta \underset{t}{\to} 0$ et $V(j, j^*) \underset{t}{\to} 0$

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
Kohonen	X	X	X	✓	Grille		Projection topologique

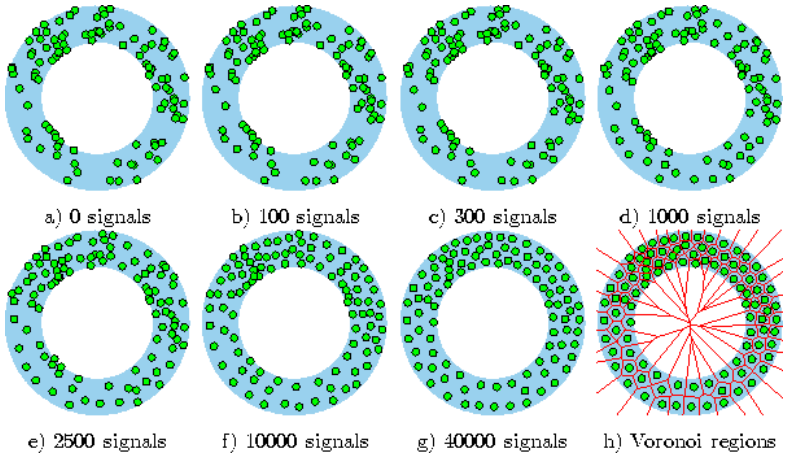
Apprentissage non supervisé - Gaz neuronal - Martinetz et Schulten (1991)

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } ||\mathbf{W_j} - \mathbf{X}|| = \min\limits_{j'} ||\mathbf{W_{j'}} - \mathbf{X}|| \\ 0 & \text{sinon} \end{cases}$

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 $E = \sum\limits_{\mathbf{X}} ||\mathbf{W_{j^*}} - \mathbf{X}||^2$
- Convergence garantie si $\eta \underset{t}{\to} 0$ et $\lambda \underset{t}{\to} 0$
- 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

Apprentissage non supervisé - Gaz neuronal croissant - Fritzke (1995)

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } ||\mathbf{W_j} - \mathbf{X}|| = \min\limits_{j'} ||\mathbf{W_{j'}} - \mathbf{X}|| \\ 0 & \text{sinon} \end{cases}$

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 $\Delta\mathbf{W_{j^{*\prime}}} = \eta'(\mathbf{X} - \mathbf{W_{j^{*\prime}}})$ 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'

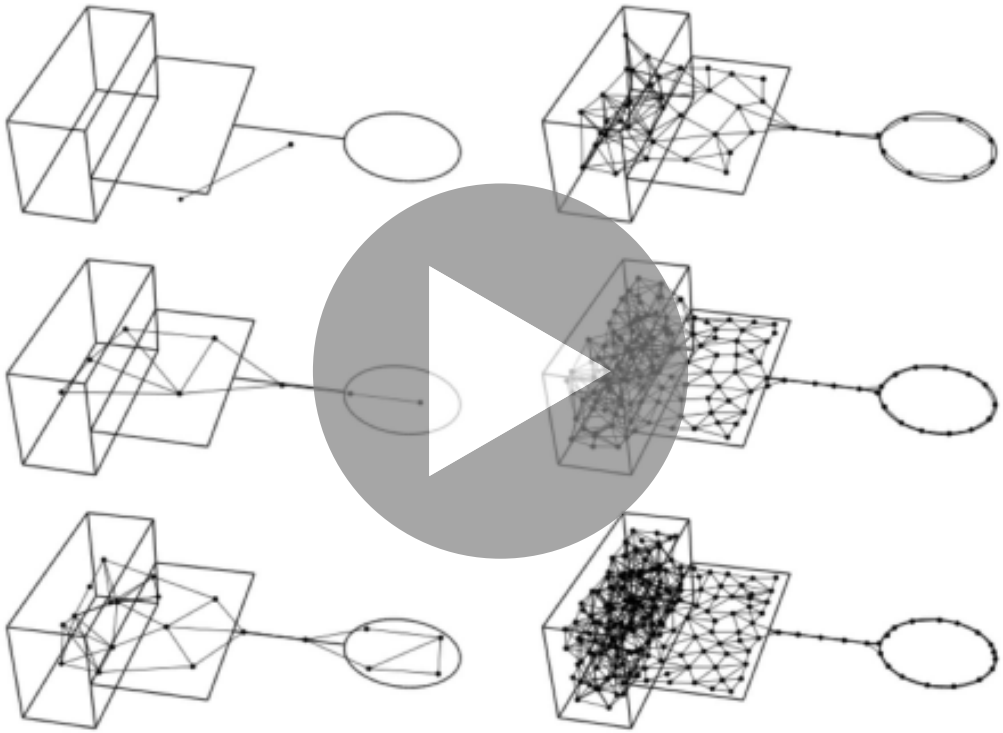
Propriétés

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

Apprentissage non supervisé - Gaz neuronal croissant - Fritzke (1995)

Sortie du neurone j : $y_j = \begin{cases} 1 & \text{si } ||\mathbf{W_j} - \mathbf{X}|| = \min\limits_{j'} ||\mathbf{W_{j'}} - \mathbf{X}|| \\ 0 & \text{sinon} \end{cases}$

Réseau : Initialement deux neurones

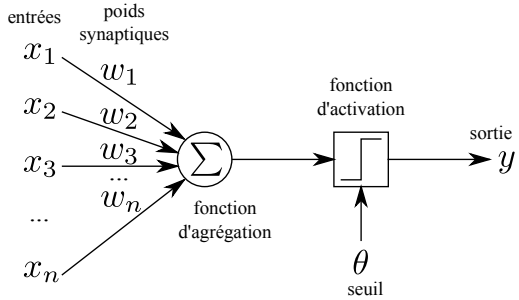


Démonstration : `http://www.demogng.de/js/demogng.html`

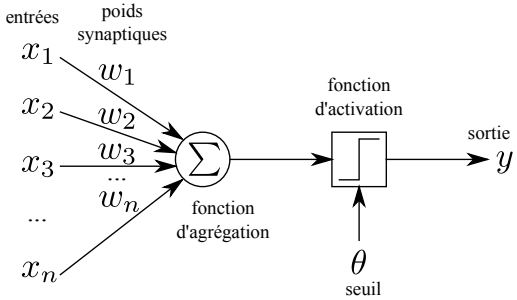
Bilan apprentissage de réseaux de neurones à fréquence de décharge

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

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



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



Question

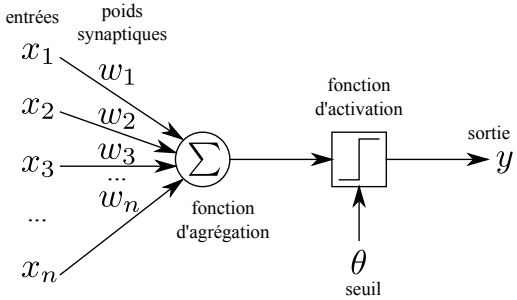
Comment faire un OU ?

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

un ET ?

x_1	x_2	ET
0	0	0
0	1	0
1	0	0
1	1	1

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



Question

Comment faire un OU ?

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

un ET ?

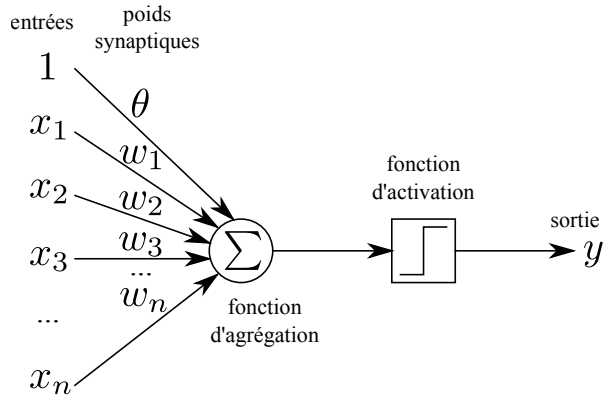
x_1	x_2	ET
0	0	0
0	1	0
1	0	0
1	1	1

Question

Une idée de la règle d'apprentissage ?

Apprentissage supervisé - Perceptron - Rosenblatt (1957)

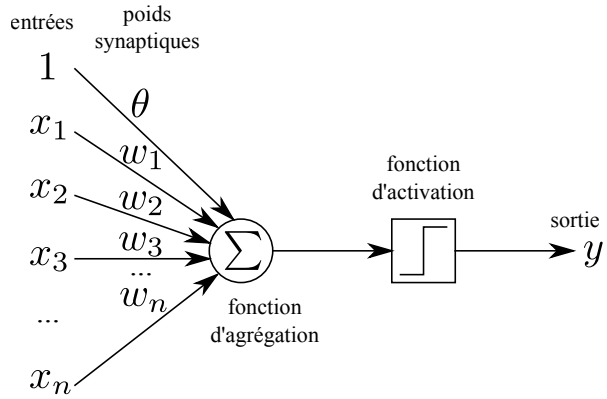
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$



Apprentissage supervisé - Perceptron - Rosenblatt (1957)

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

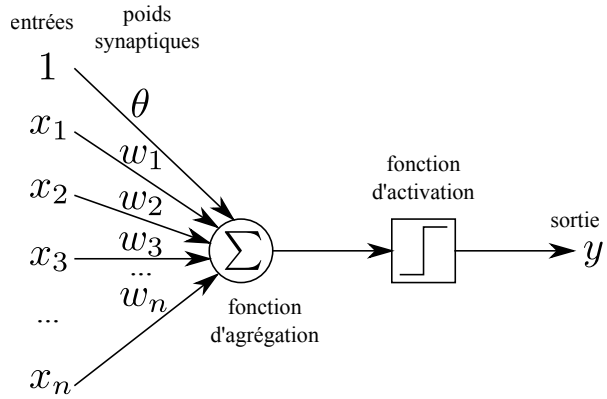
Apprentissage : $\Delta \mathbf{W} = \eta \mathbf{X}(t - y)$



Apprentissage supervisé - Perceptron - Rosenblatt (1957)

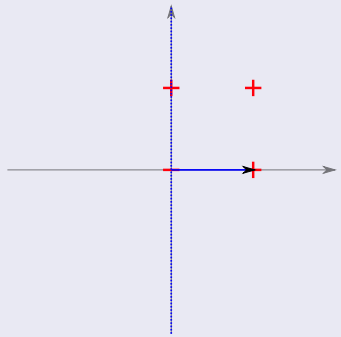
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



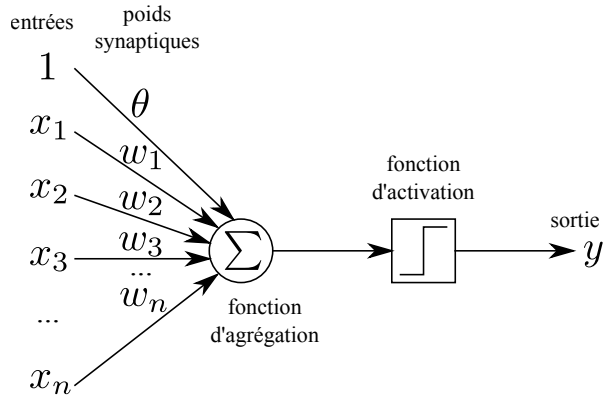
Un exemple : le OU

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



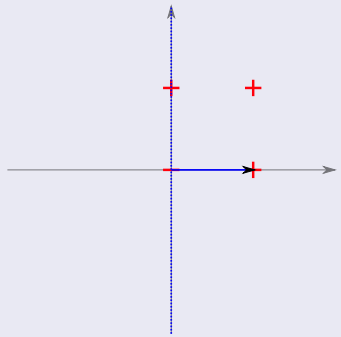
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



Un exemple : le OU

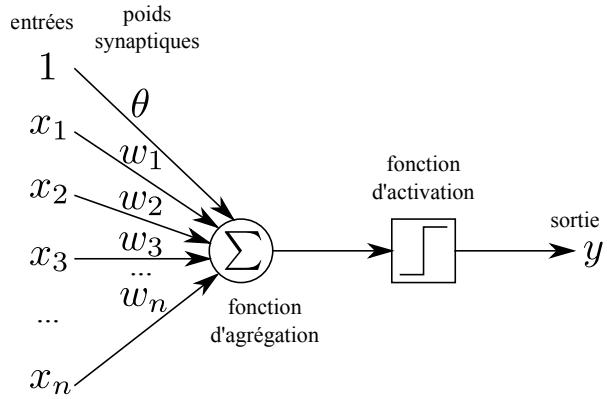
Entrée **X**			Poids **W**			Sortie	Sortie voulue
x_0	x_1	x_2	w_0	w_1	w_2	y	t
1	0	1	0	1	0	0	1



Apprentissage supervisé - Perceptron - Rosenblatt (1957)

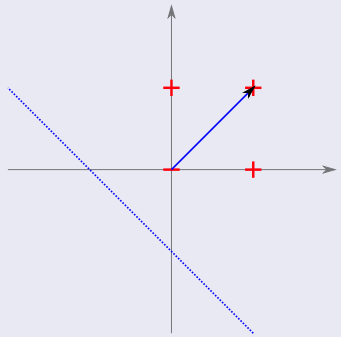
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



Un exemple : le OU

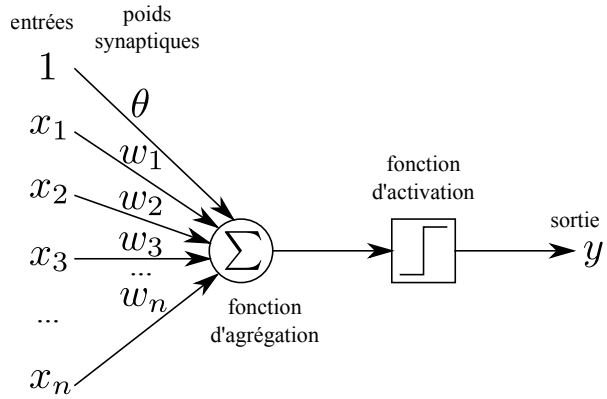
Entrée **X**			Poids **W**			Sortie	Sortie voulue
x_0	x_1	x_2	w_0	w_1	w_2	y	t
1	0	1	1	1	1	0	1



Apprentissage supervisé - Perceptron - Rosenblatt (1957)

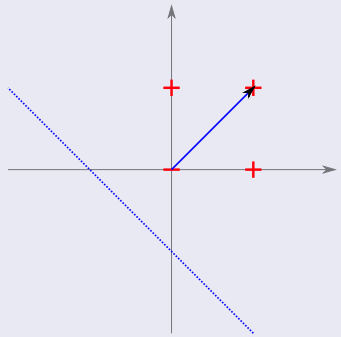
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



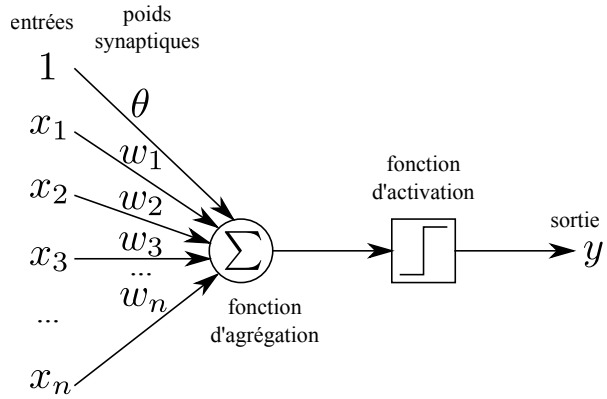
Un exemple : le OU

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



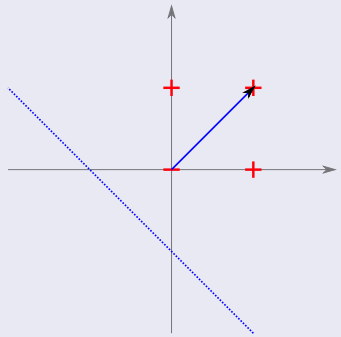
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



Un exemple : le OU

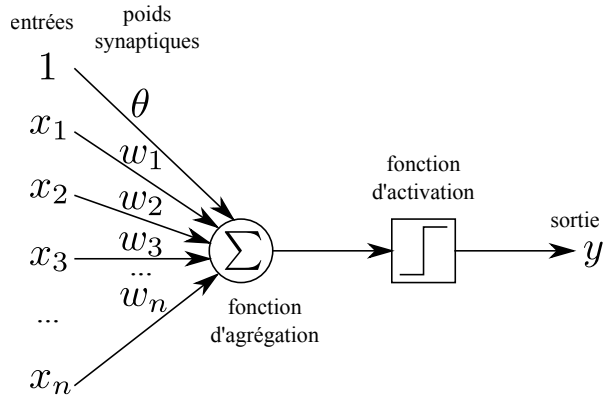
Entrée **X**			Poids **W**			Sortie	Sortie voulue
x_0	x_1	x_2	w_0	w_1	w_2	y	t
1	1	1	1	1	1	1	1



Apprentissage supervisé - Perceptron - Rosenblatt (1957)

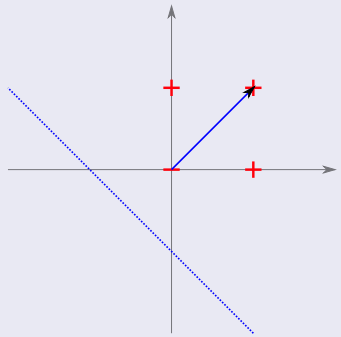
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



Un exemple : le OU

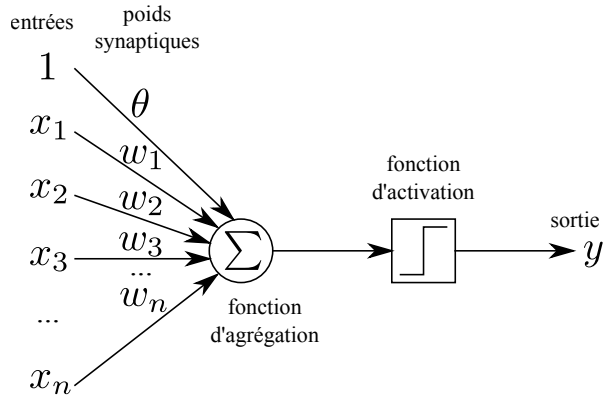
Entrée **X**			Poids **W**			Sortie	Sortie voulue
x_0	x_1	x_2	w_0	w_1	w_2	y	t
1	0	0	1	1	1	1	0



Apprentissage supervisé - Perceptron - Rosenblatt (1957)

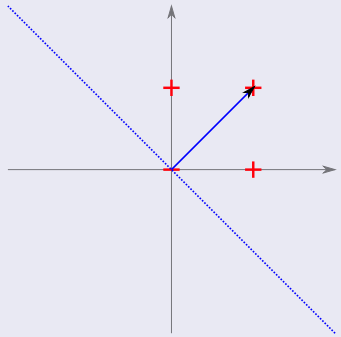
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



Un exemple : le OU

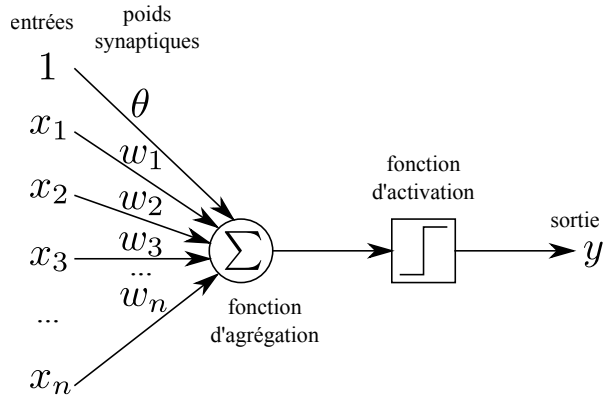
Entrée **X**			Poids **W**			Sortie	Sortie voulue
x_0	x_1	x_2	w_0	w_1	w_2	y	t
1	0	0	0	1	1	1	0



Apprentissage supervisé - Perceptron - Rosenblatt (1957)

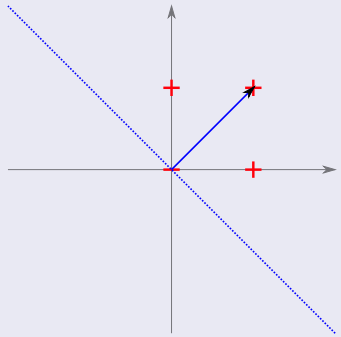
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



Un exemple : le OU

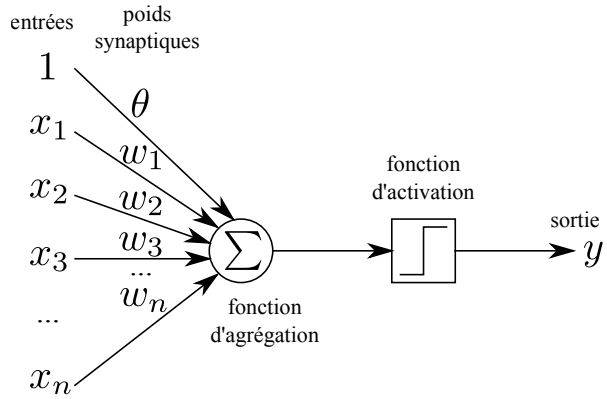
Entrée **X**			Poids **W**			Sortie	Sortie voulue
x_0	x_1	x_2	w_0	w_1	w_2	y	t
1	0	1	0	1	1	1	1



Apprentissage supervisé - Perceptron - Rosenblatt (1957)

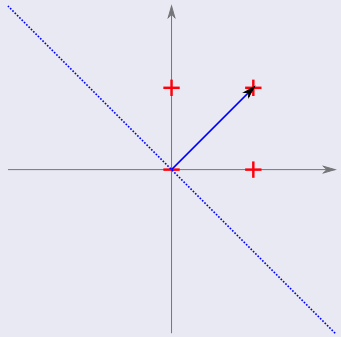
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



Un exemple : le OU

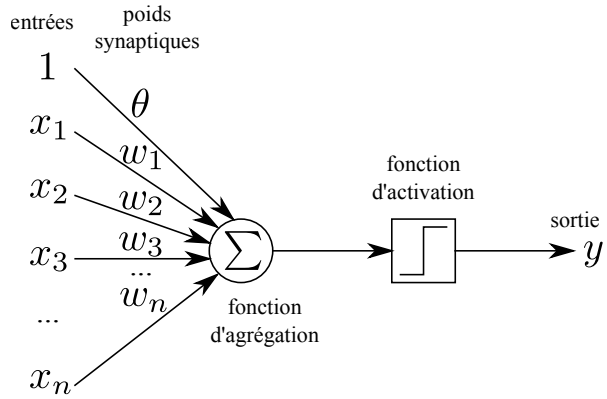
Entrée **X**			Poids **W**			Sortie	Sortie voulue
x_0	x_1	x_2	w_0	w_1	w_2	y	t
1	1	0	0	1	1	1	1



Apprentissage supervisé - Perceptron - Rosenblatt (1957)

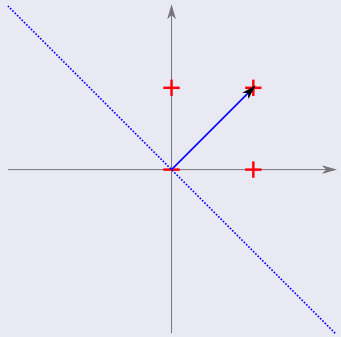
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



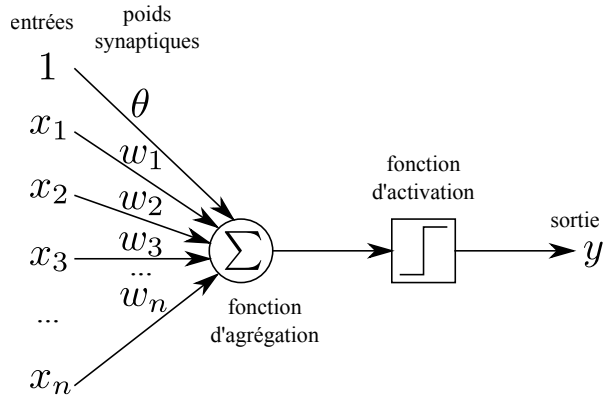
Un exemple : le OU

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



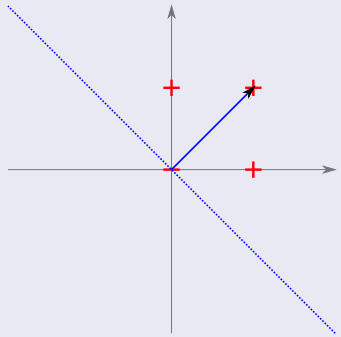
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



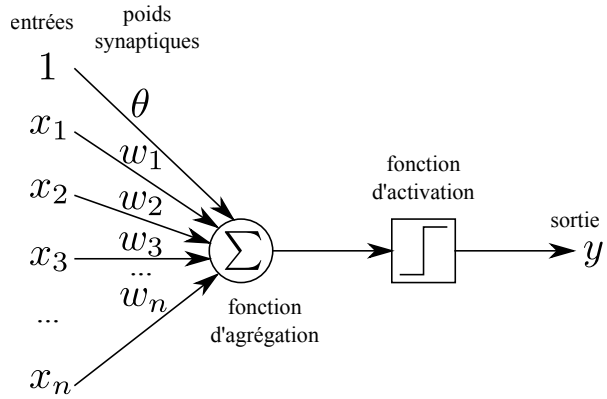
Un exemple : le OU

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



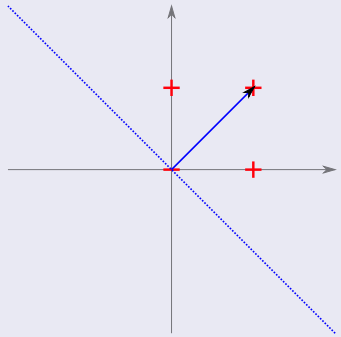
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



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
1	0	0	0	1	1	0	0

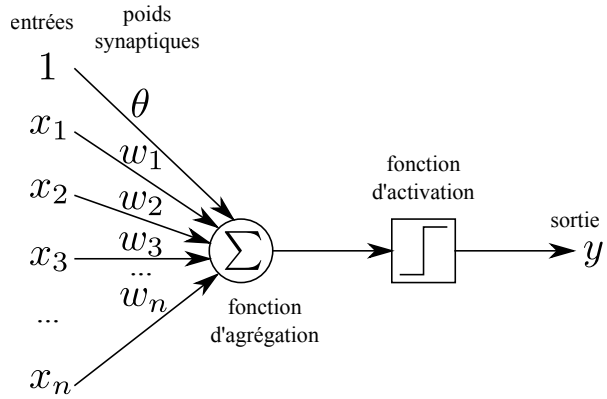


Propriétés

- Classifieur linéaire (découpage de l'espace avec un hyperplan)
- Poids = vecteur normal à l'hyperplan, Seuil (biais) = ordonnée à l'origine
- Convergence si η est (infinitésimalement) petit et si les données sont linéairement séparables

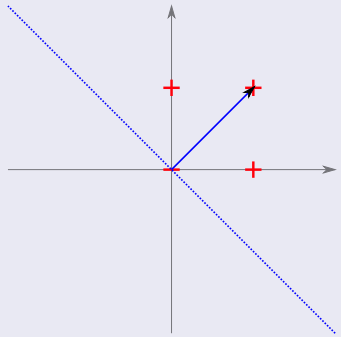
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



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
1	0	0	0	1	1	0	0



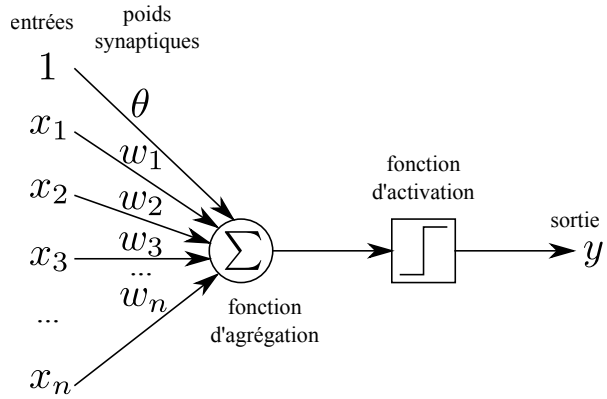
Question

Et si l'on a plusieurs sorties ?

Apprentissage supervisé - Perceptron - Rosenblatt (1957)

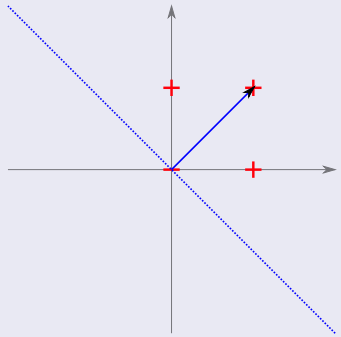
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



Un exemple : le OU

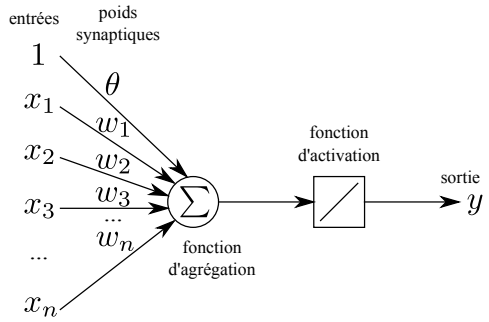
Entrée **X**			Poids **W**			Sortie	Sortie voulue
x_0	x_1	x_2	w_0	w_1	w_2	y	t
1	0	0	0	1	1	0	0



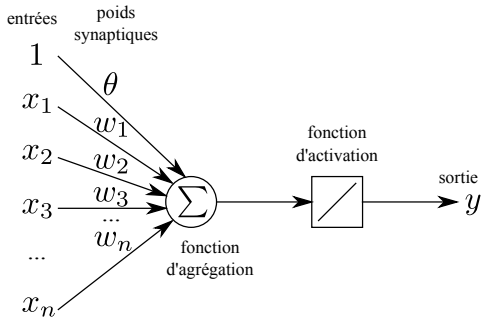
Question

Que se passe-t-il en cas de bruit / point aberrant (i.e. les données ne sont pas linéairement séparables) ?

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

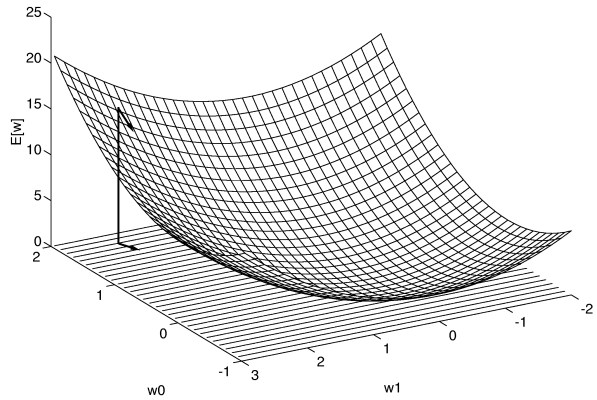


Sortie du neurone : $y = \sum_{i=0}^{n} w_i x_i$

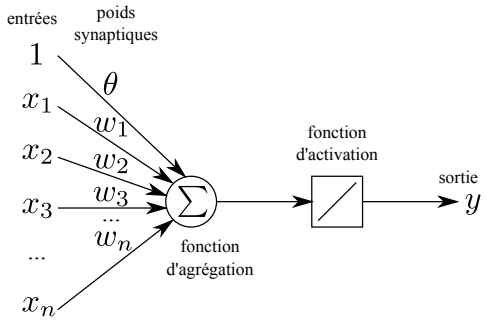


Apprentissage : On veut minimiser l'erreur

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



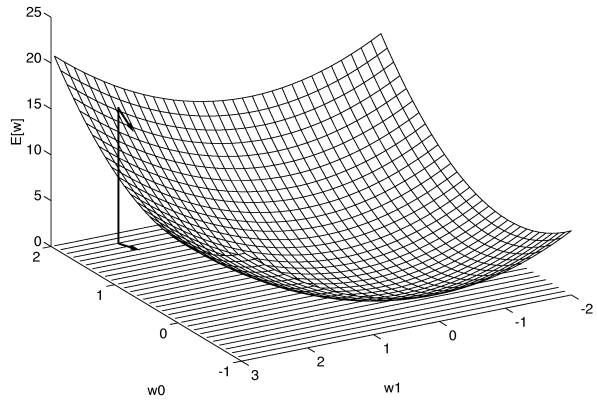
Sortie du neurone : $y = \sum_{i=0}^{n} w_i x_i$



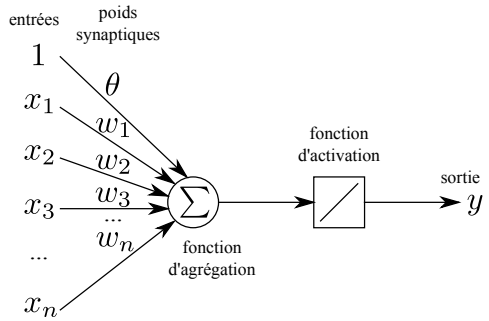
Apprentissage : On veut minimiser l'erreur

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

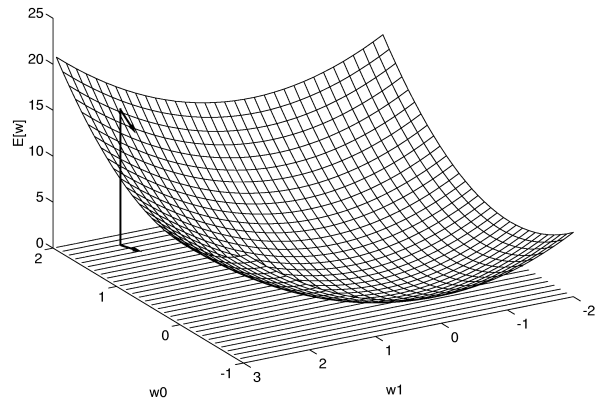
$$\begin{aligned} \Delta w_i &= -\eta \frac{\partial E}{\partial w_i} \\ &= -\frac{\eta}{2} \sum_{\{\mathbf{X}\}} \frac{\partial (t-y)^2}{\partial w_i} \\ &= -\frac{\eta}{2} \sum_{\{\mathbf{X}\}} -2 \frac{\partial y}{\partial w_i} (t - y) \\ &= \sum_{\{\mathbf{X}\}} \eta x_i (t - y) \end{aligned}$$



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



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

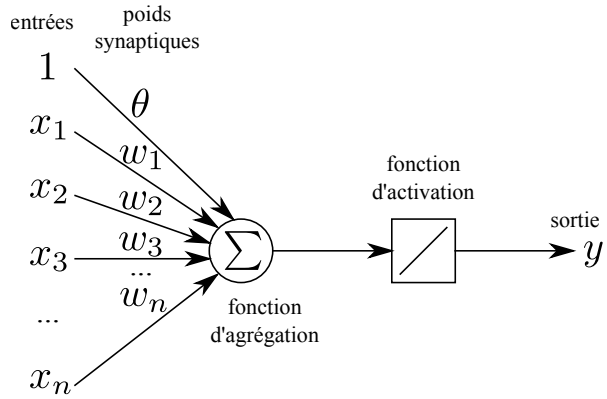
- Classifieur linéaire
- Convergence garantie si η est faible (même si les données ne sont pas linéairement séparables)
- Convergence efficace (car la fonction à optimiser est quadratique)
- Convergence souvent plus rapide en mode en ligne mais pas garantie comme en mode *batch*

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

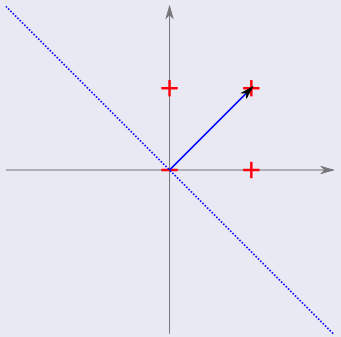
Sortie du neurone : $y = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 & \text{sinon} \end{cases}$

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



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
1	0	0	0	1	1	0	0

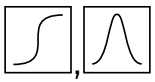


Question

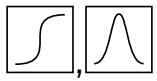
Comment faire un OU EXCLUSIF ?

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

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

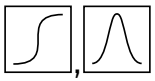
Bilan apprentissage de réseaux de neurones à fréquence de décharge

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.

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

Question

A-t-on résolu les problématiques de l'IA ?