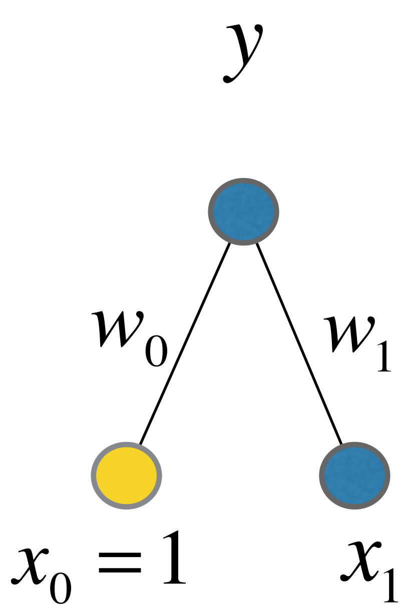
Modèles mathématiques et computationnels en neurosciences

Apprentissage non-supervisé. Réseaux récurrents

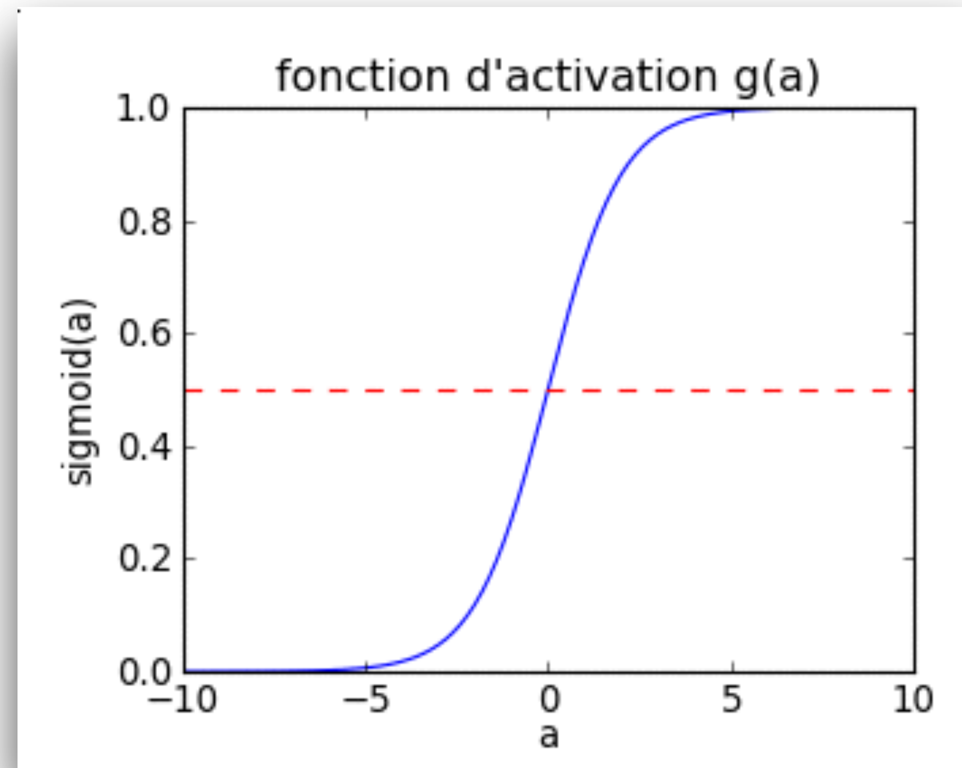
4I702 CM 8

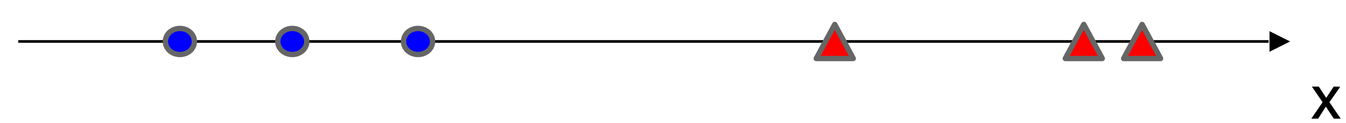
denis.sheynikhovich@upmc.fr

Rappel : classification supervisée



$$y = g(\vec{w} \cdot \vec{x}) = \frac{1}{1 + e^{-w_0 - w_1 x_1}}$$



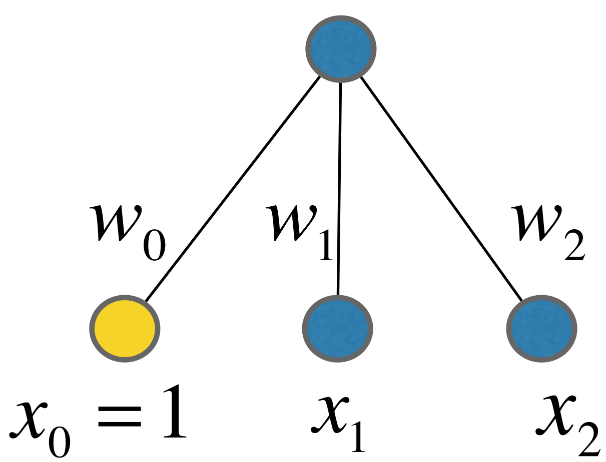


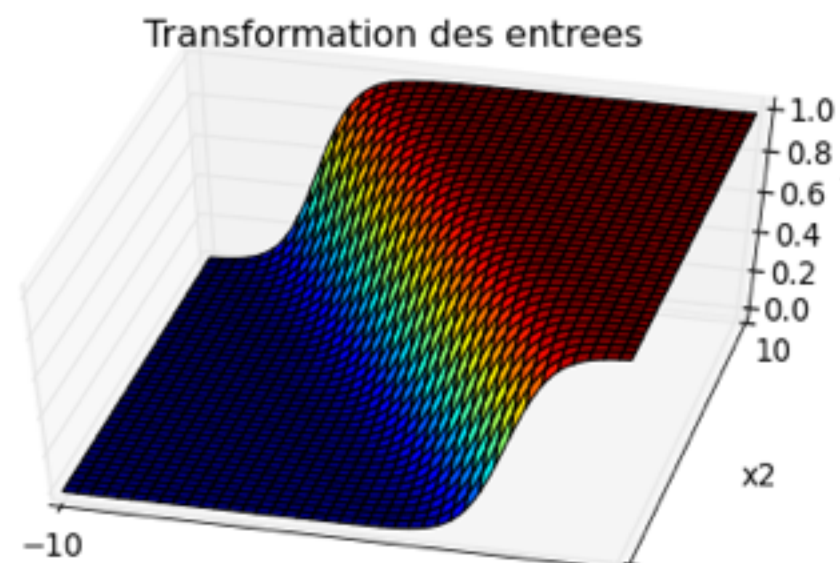
classe ‘bleu’ classe ‘rouge’

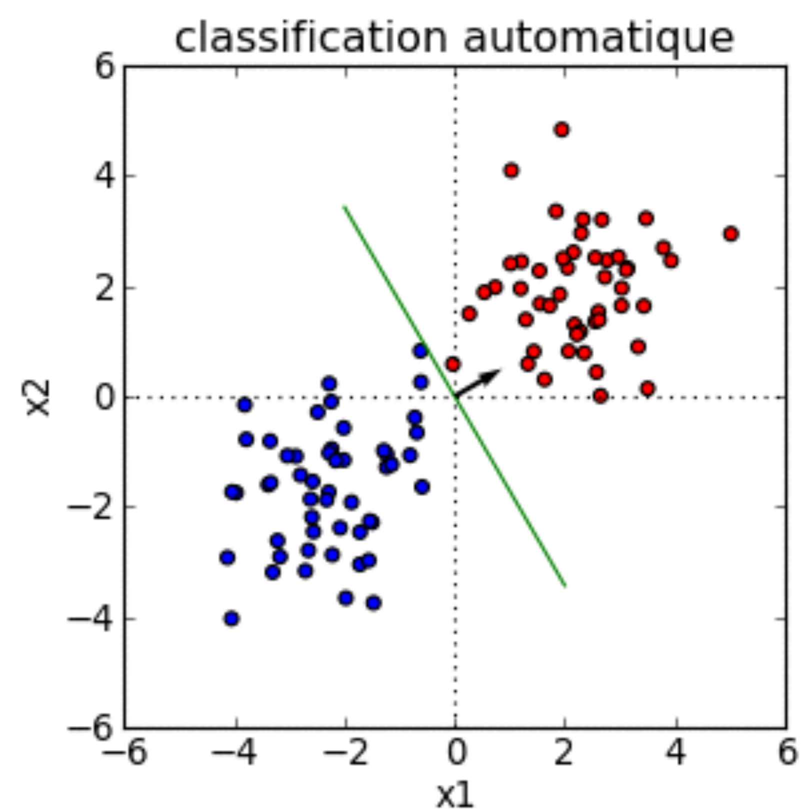
2 classes
linéairement séparables

Rappel : classification supervisée

$$y = g(\vec{w} \cdot \vec{x}) = \frac{1}{1 + e^{-w_0 - w_1 x_1 - w_2 x_2}}$$



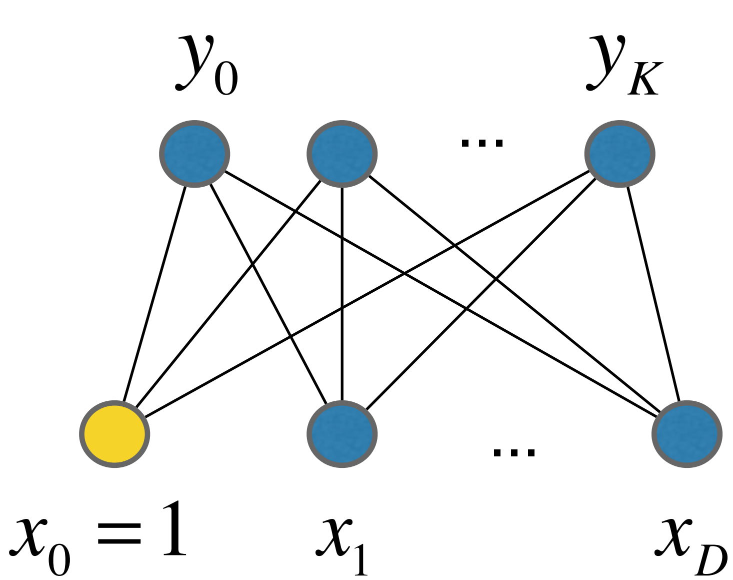




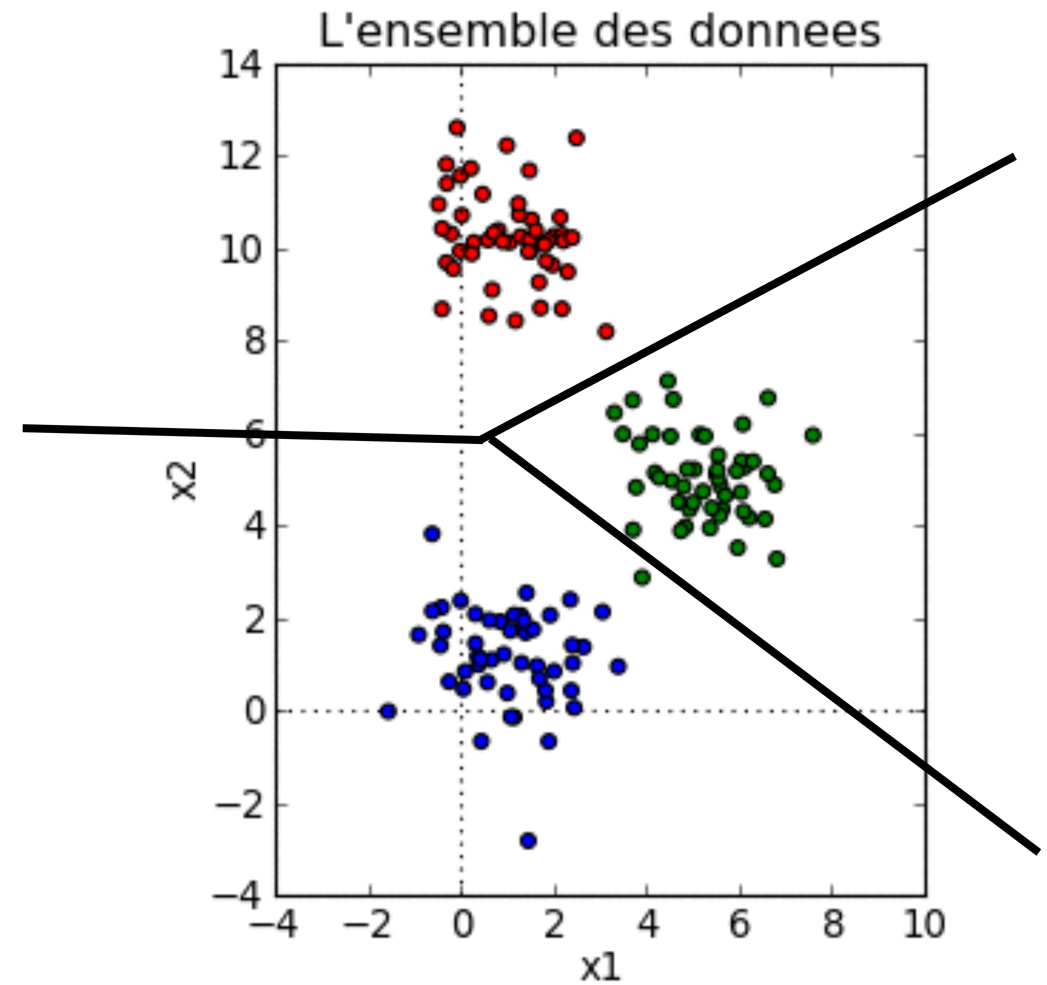
2 classes
linéairement séparables

Rappel : classification supervisée

$$y_k = g(\vec{w}_k \cdot \vec{x})$$

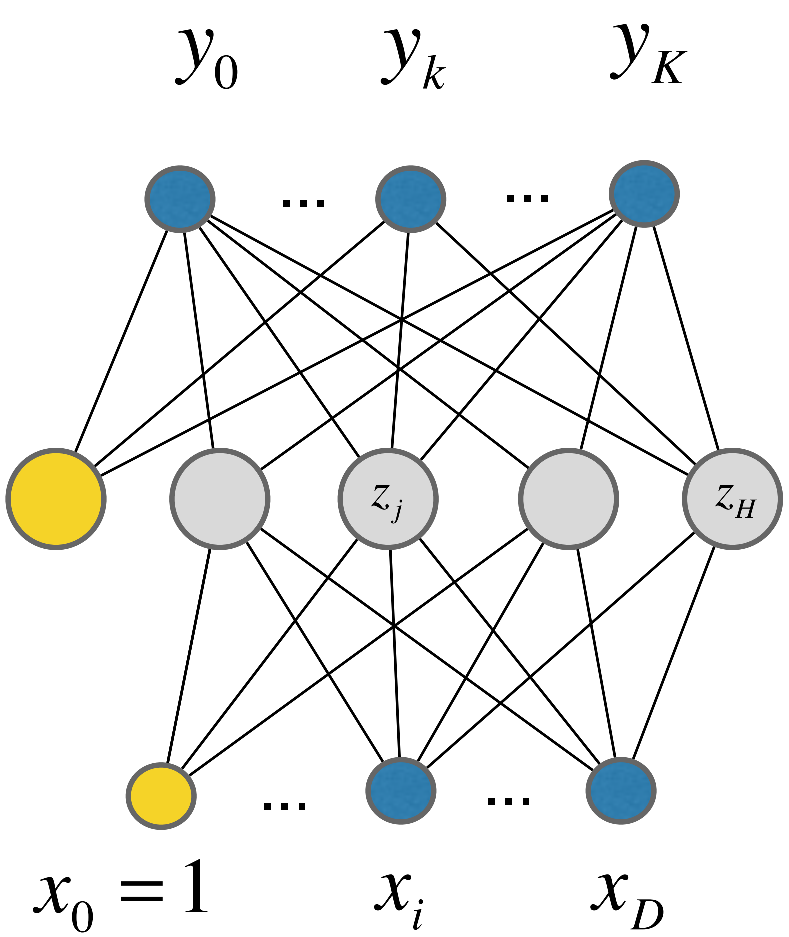


“perceptron"

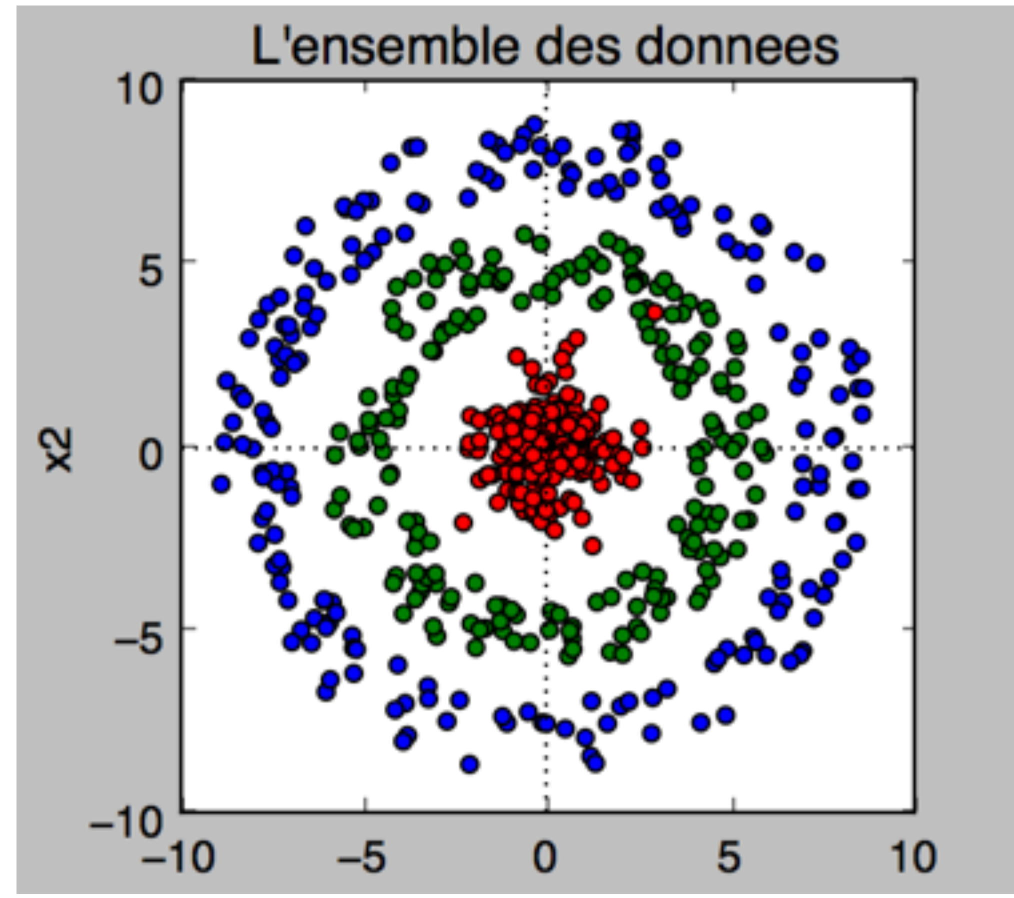


K classes linéairement séparables

Rappel : classification supervisé



perceptron multicouche (multilayer perceptron)



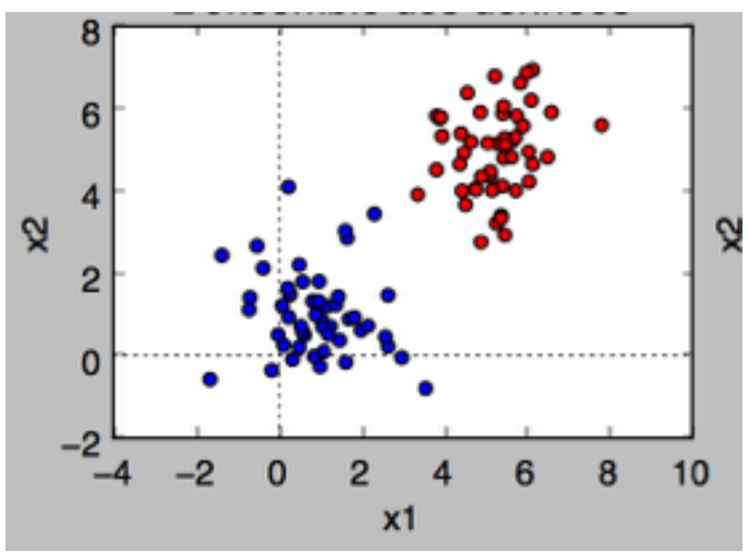
Classes en plusieurs dimensions avec des **surfaces de séparation arbitraires**

Rappel : apprentissage par la rétropropagation de l'erreur

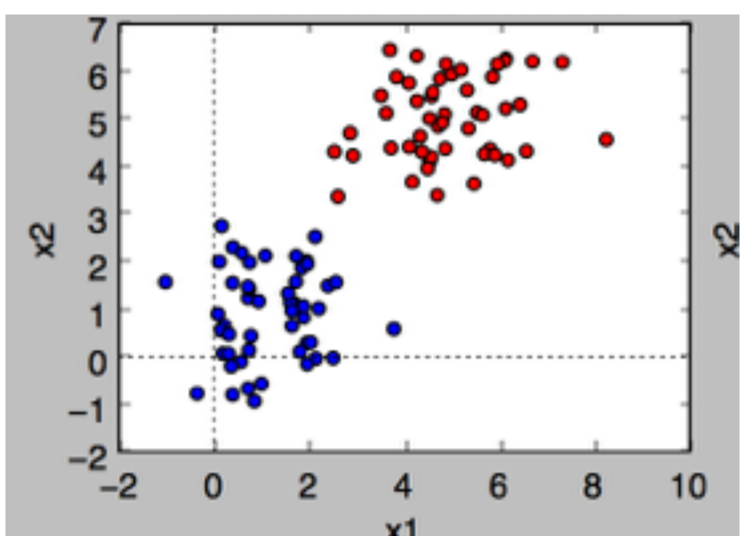
Ensemble des exemples
avec des étiquettes des classes

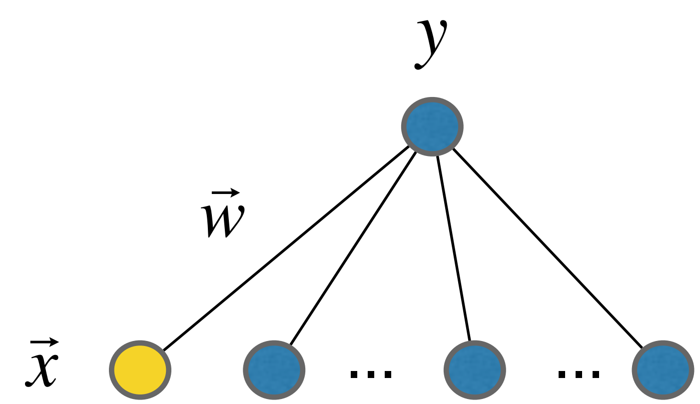
$t = 1$
rouge

$t = 0$
bleu



Classification des nouveaux
données (sans étiquettes)





Entrainement du réseau

1. Choisir un exemple $\vec{x}^{(p)}$ de la base des exemples
2. Calculer la réponse du réseau

$$y^{(p)} = g(\vec{w} \cdot \vec{x}^{(p)})$$

3. Calculer l'erreur de classification

$$\delta = g'(a)(y^{(p)} - t^{(p)})$$

5. Ajuster les poids synaptiques selon la règle d'apprentissage

$$\Delta\vec{w} = -\eta\delta\vec{x}$$

6. Répéter 1-5 jusqu'à la fin d'entrainement

Apprentissage

- Principe d'apprentissage **supervisé** :

?

Supervisé

0

Supervisé

1

Supervisé

0

Supervisé

1

Apprentissage

- Principe d'apprentissage **supervisé** :

Minimisation d'erreur
(erreur = réponse correcte - réponse donnée)

$$\Delta\vec{w} = -\eta\delta\vec{x}$$

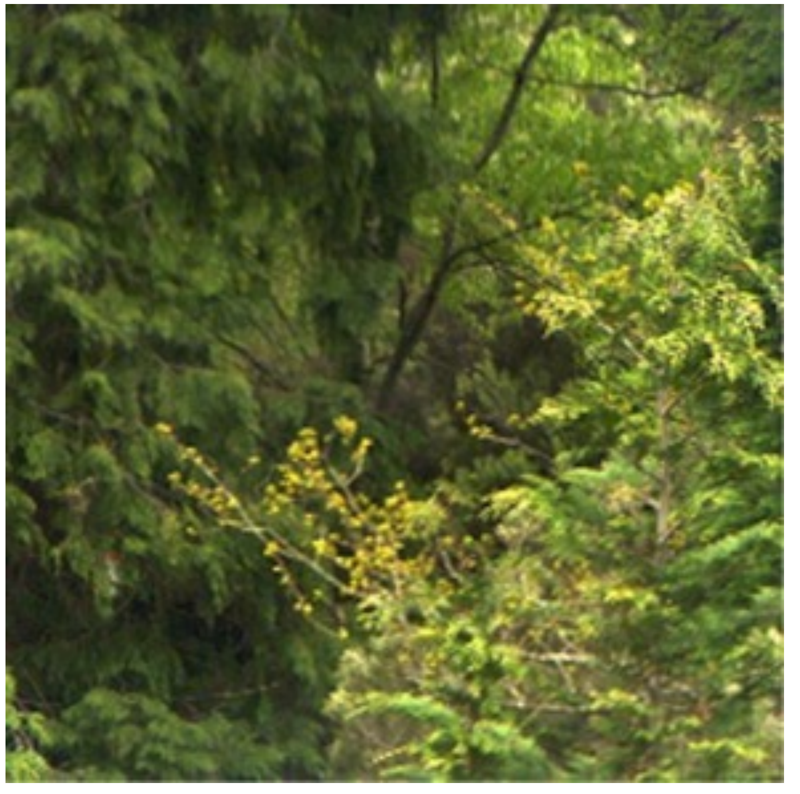
- Principe d'apprentissage **non-supervisé** :

?

Non-supervisé

Non-supervisé

Non-supervisé



Non-supervisé

Apprentissage

- Principe d'apprentissage **supervisé** :

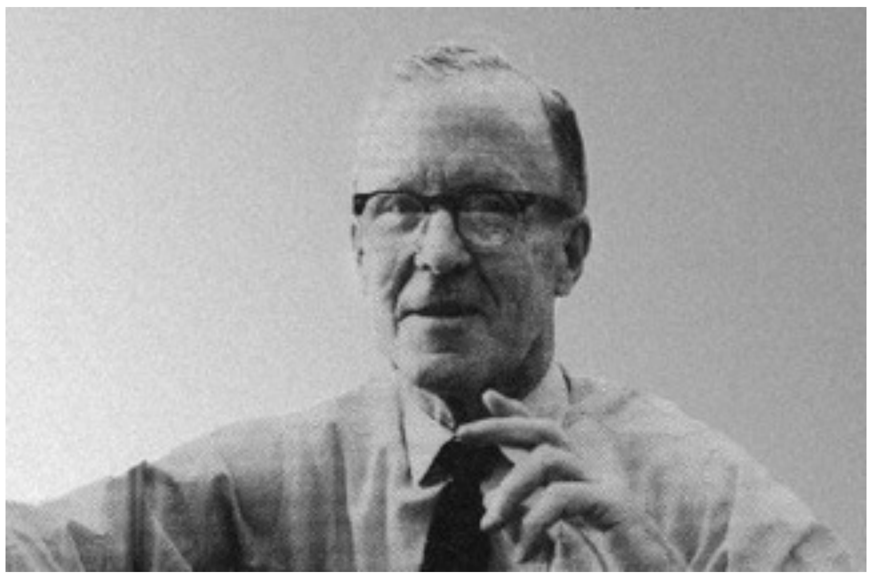
Minimisation d'erreur
(erreur = réponse correcte - réponse donnée)

$$\Delta\vec{w} = -\eta\delta\vec{x}$$

- Principe d'apprentissage **non-supervisé** :

Corrélation (similarité) des données

Théorie de Hebb

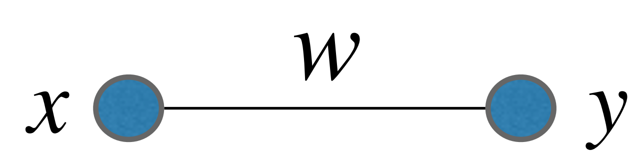


Donald Hebb

"The Organization of Behavior", 1949

- La base d'apprentissage :
    - similarité
    - associativité
    - corrélation

- Au niveau neuronal :

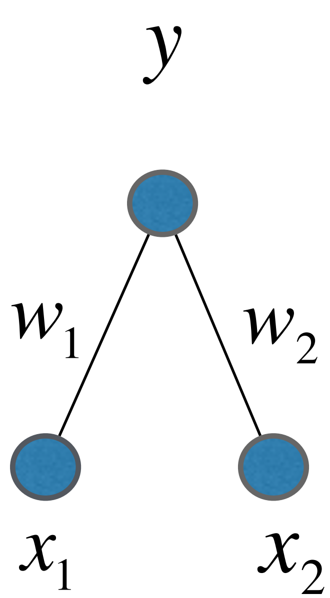


$$\Delta w = \eta x y$$

règle d'apprentissage dite "Hebbienne"

http://deeplearning.cs.cmu.edu/pdfs/Hebb.1949.pdf

Question principale



- Réseau feed-forward
- Il y a un ensemble d'entrainement (**sans étiquettes**)
- Apprentissage non-supervisé Hebbien

$$\Delta w_i = \eta y x_i$$

$$y = g(a) = \sum_{i=1}^{D} w_i x_i$$

Pour simplifier, on analyse d'abord
le cas de la fonction d'activation linéaire

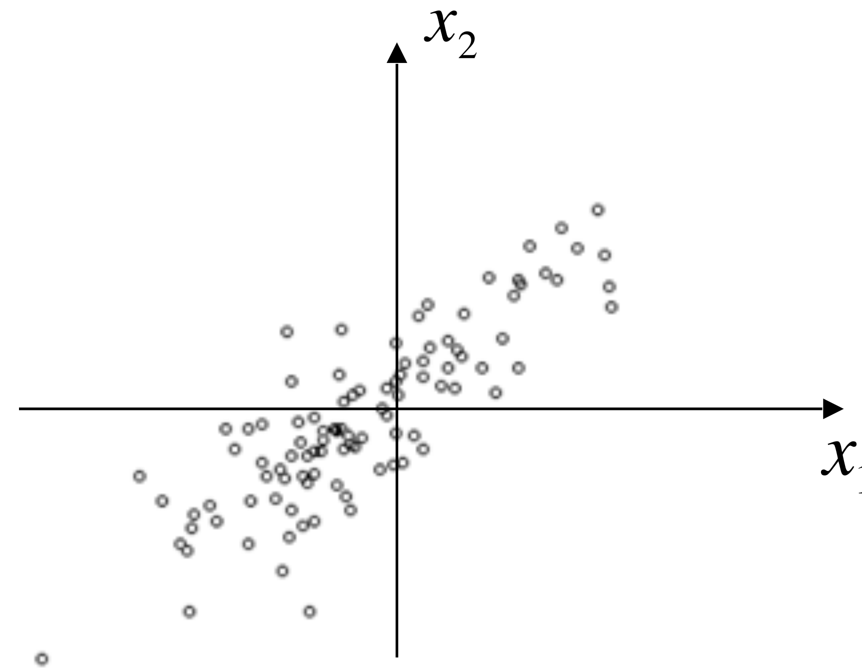
$$g(a) = a$$

Que fait ce réseau ?

Rappel : analyse en composantes principales (ACP)

Rappel : analyse en composantes principales (ACP)

- Ensemble de données en 2D **centrées** ($\bar{x}_1 = 0$, $\bar{x}_2 = 0$)
- Chaque point $\vec{x} = (x_1, x_2)$
- Statistiquement, on peut considérer l'ensemble comme réalisations de deux variables aléatoires X_1 et X_2



Ensemble de données

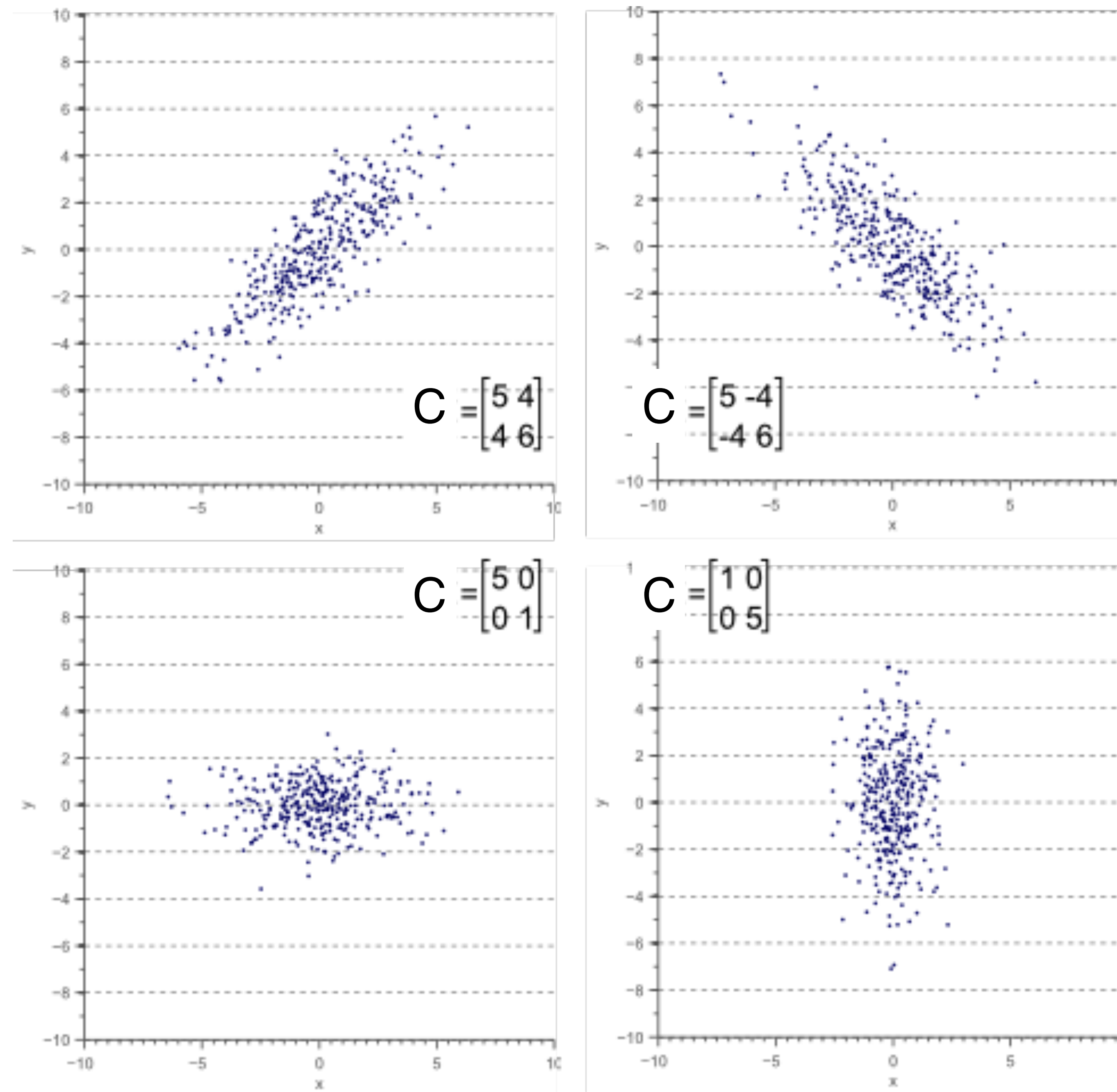
$$\begin{array}{c c} & \begin{array}{cc} X_1 & X_2 \end{array} \\ \begin{array}{c} \vec{x}^{(1)} \\ \vec{x}^{(2)} \\ \dots \\ \vec{x}^{(N)} \end{array} = & \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ \dots & \dots \\ x_1^{(N)} & x_2^{(N)} \end{bmatrix} \end{array}$$

Matrice de covariance

$$C = \begin{bmatrix} \mathrm{cov}(X_1, X_1) & \mathrm{cov}(X_1, X_2) \\ \mathrm{cov}(X_2, X_1) & \mathrm{cov}(X_2, X_2) \end{bmatrix} = \begin{bmatrix} \sigma_{X_1}^2 & \sigma_{X_1 X_2} \\ \sigma_{X_1 X_2} & \sigma_{X_2}^2 \end{bmatrix}$$

$$\mathrm{cov}(X_1, X_2) = \frac{\sum_{p=1}^{N} (x_1^{(p)} - \bar{x}_1)(x_2^{(p)} - \bar{x}_2)}{N}$$

Rappel : analyse en composantes principales (ACP)

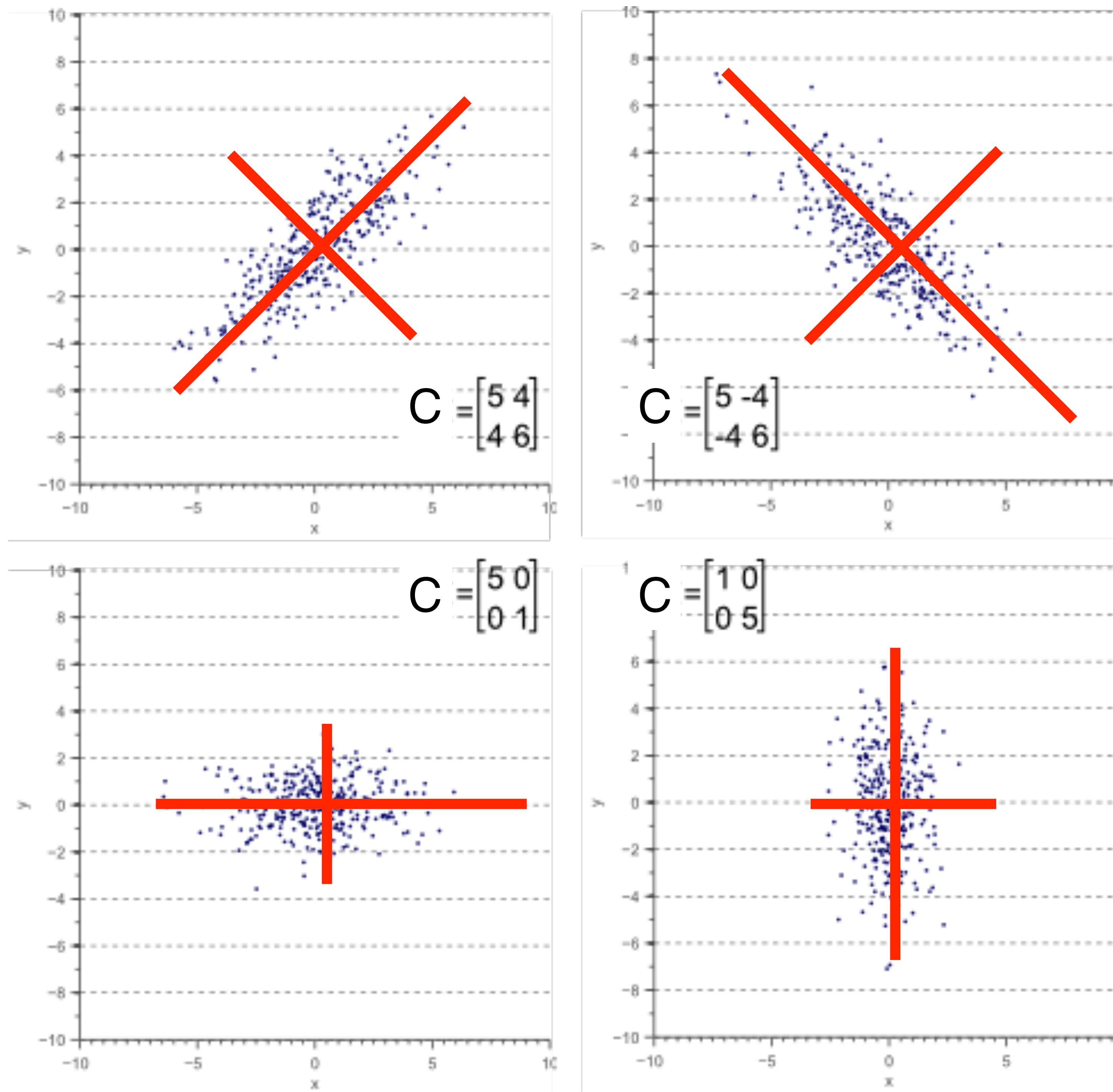


Matrice de covariance $$C = \begin{bmatrix} \mathrm{cov}(X_1, X_1) & \mathrm{cov}(X_1, X_2) \\ \mathrm{cov}(X_2, X_1) & \mathrm{cov}(X_2, X_2) \end{bmatrix} = \begin{bmatrix} \sigma_{X_1}^2 & \sigma_{X_1 X_2} \\ \sigma_{X_1 X_2} & \sigma_{X_1}^2 \end{bmatrix}$$

Rappel : analyse en composantes principales (ACP)

- Vecteurs propres de la matrice de covariance correspondent aux axes de variances maximales

- Valeurs propres correspondent aux variances le long de ces axes



Matrice de covariance $$C = \begin{bmatrix} \mathrm{cov}(X_1, X_1) & \mathrm{cov}(X_1, X_2) \\ \mathrm{cov}(X_2, X_1) & \mathrm{cov}(X_2, X_2) \end{bmatrix} = \begin{bmatrix} \sigma_{X_1}^2 & \sigma_{X_1 X_2} \\ \sigma_{X_1 X_2} & \sigma_{X_1}^2 \end{bmatrix}$$

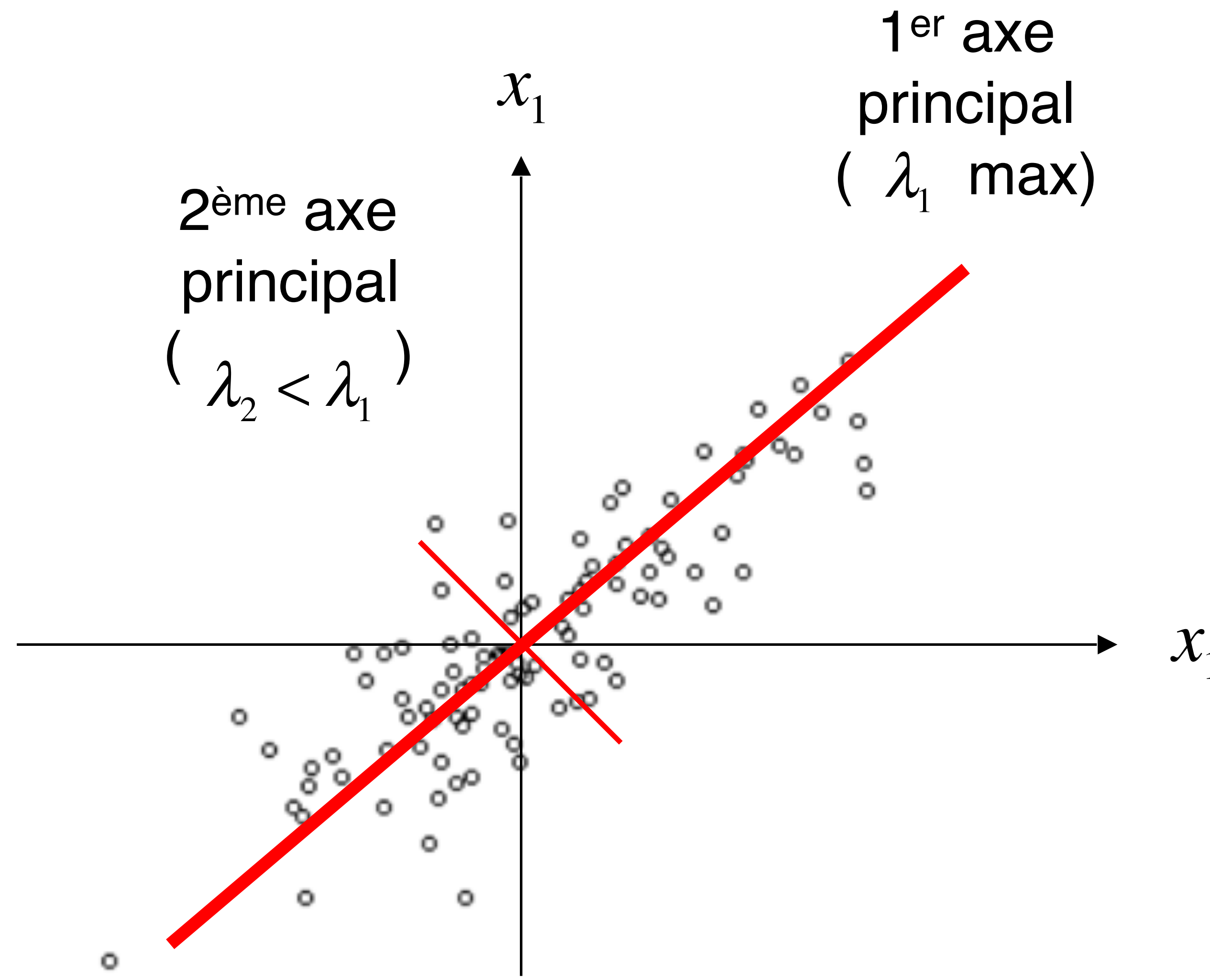
Rappel : analyse en composantes principales (ACP)

Définitions :

- **1$^{\text{ère}}$ axe principal** - direction de la variance maximale des données

- 1$^{\text{ère}}$ axe principal = vecteur propre de la matrice de covariance correspondant à la valeur propre maximale.

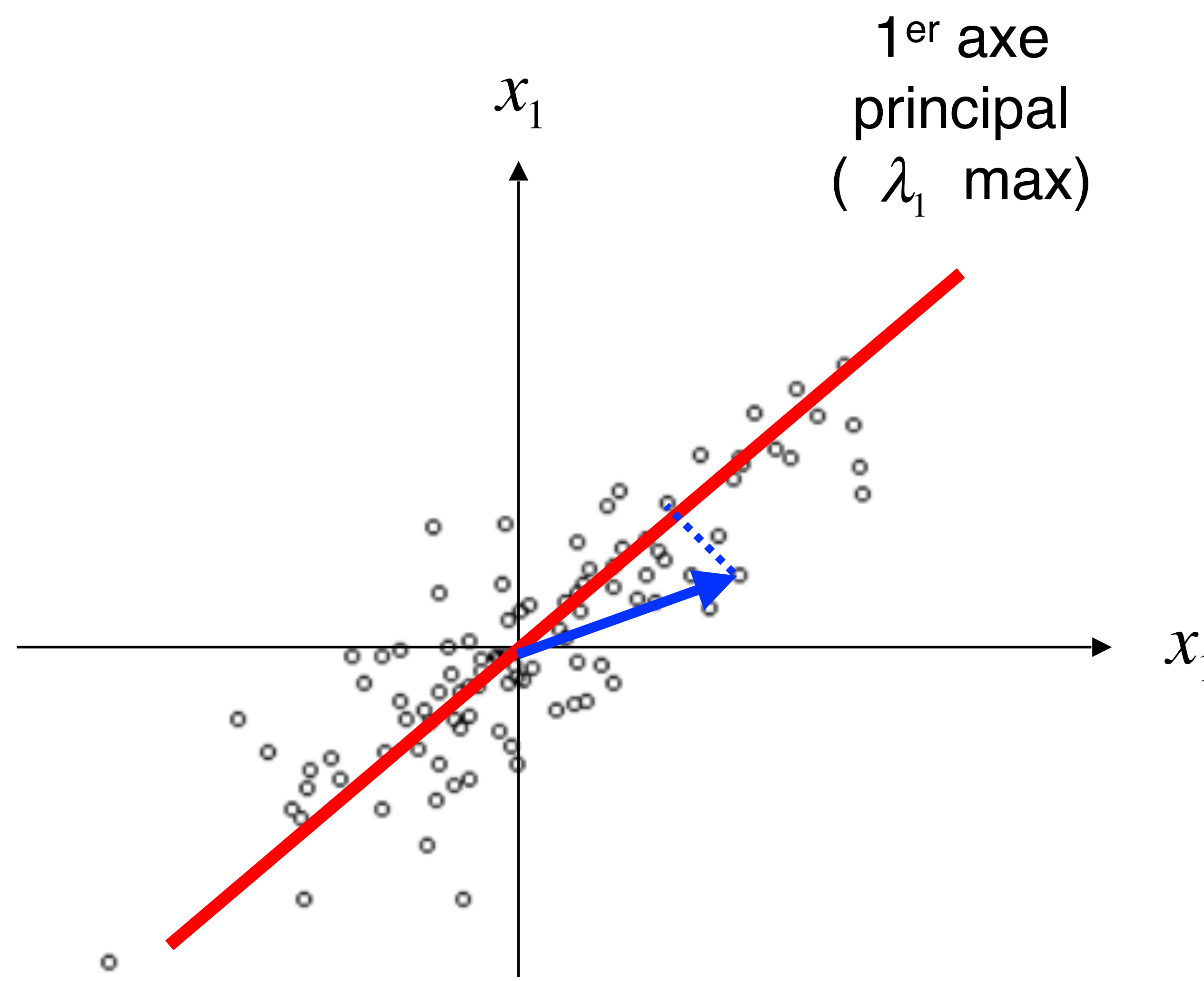
- Coordonnées des vecteurs de données sur les axes principaux = **composantes principales**



Rappel : analyse en composantes principales (ACP)

Utilité :

- 1[ère] axe principale - **l'axe de projection** le long la variance maximale
- Projection des données sur l'axe principal diminue la dimensionnalité en minimisant la perte d'information



Rappel : analyse en composantes principales (ACP)

Extension à plusieurs dimensions :

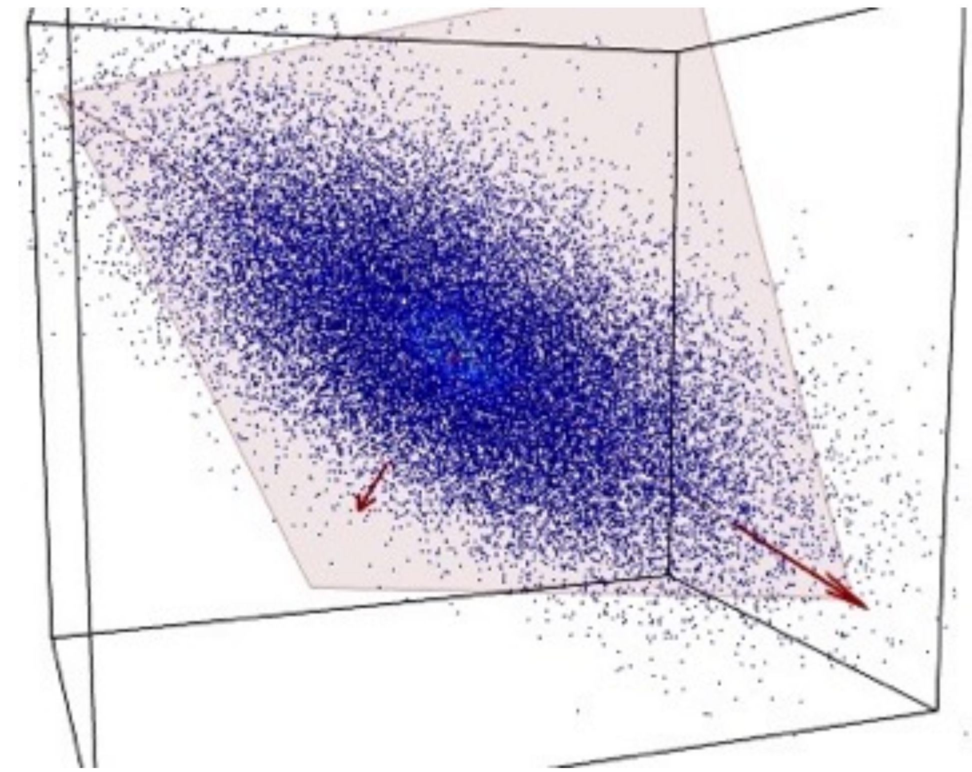
- Un ensemble des vecteurs

$$\vec{x}^{(p)} = (x_1^{(p)}, x_2^{(p)}, \ldots x_D^{(p)})$$

- Matrice de covariance

$$C = \begin{bmatrix} \sigma_{X_1}^2 & \sigma_{X_1X_2} & \ldots & \sigma_{X_1X_D} \\ \sigma_{X_1X_2} & \sigma_{X_2}^2 & & \\ \ldots & & \ldots & \\ \sigma_{X_1X_D} & & & \sigma_{X_D}^2 \end{bmatrix}$$

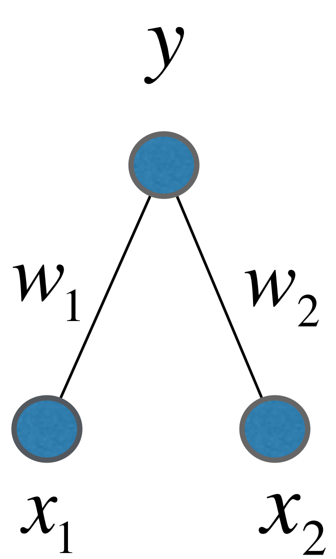
- Projection sur le plan des axes principaux : le plan de projection qui minimise la perte d'information



Réduction de dimensionnalité par l'ACP

Apprentissage non-supervisé hebbien

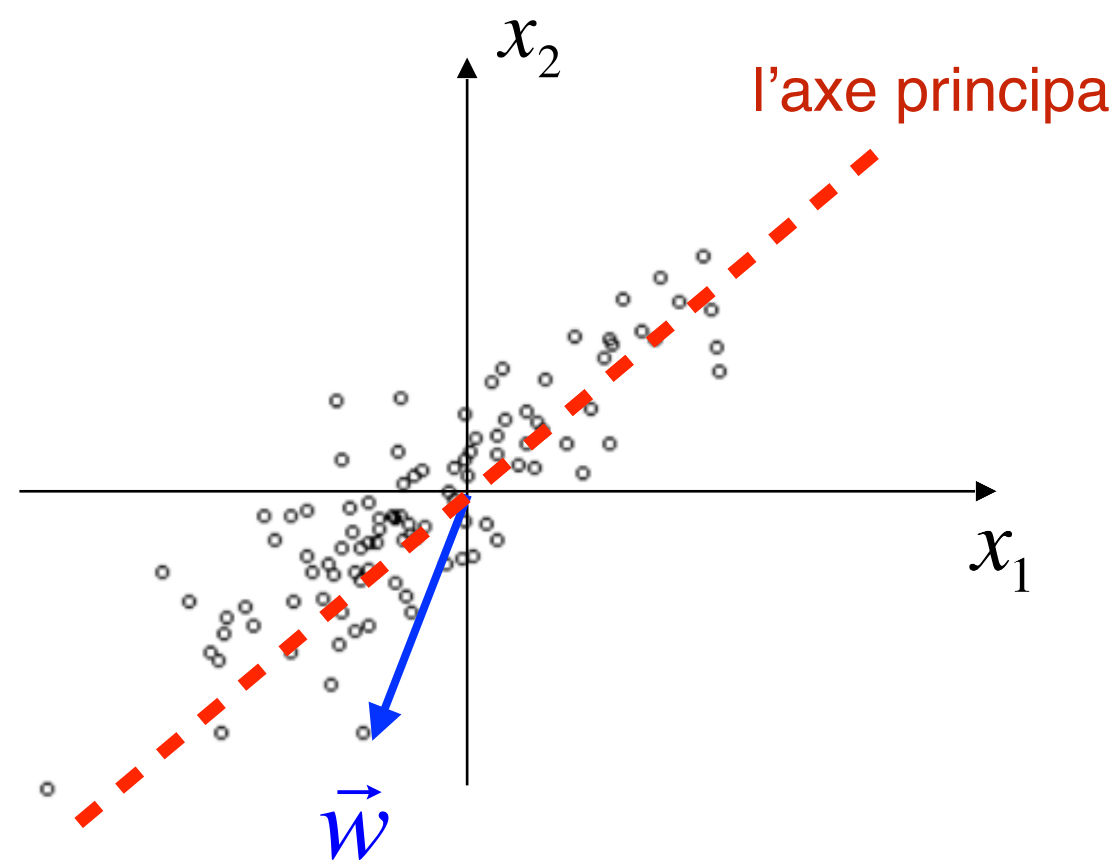
Apprentissage non-supervisé Hebbien



$$y = \sum_{i=1}^{D} w_i x_i = \vec{w} \cdot \vec{x}$$

$$\Delta \vec{w} = \eta y \vec{x}$$

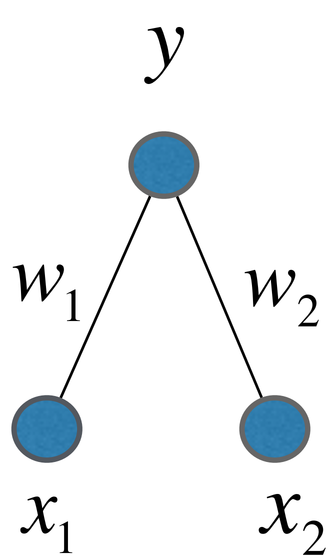
règle d'apprentissage
"hebbienne"



Avant l'apprentissage :

vecteur $\vec{w}$ initialisé aléatoirement

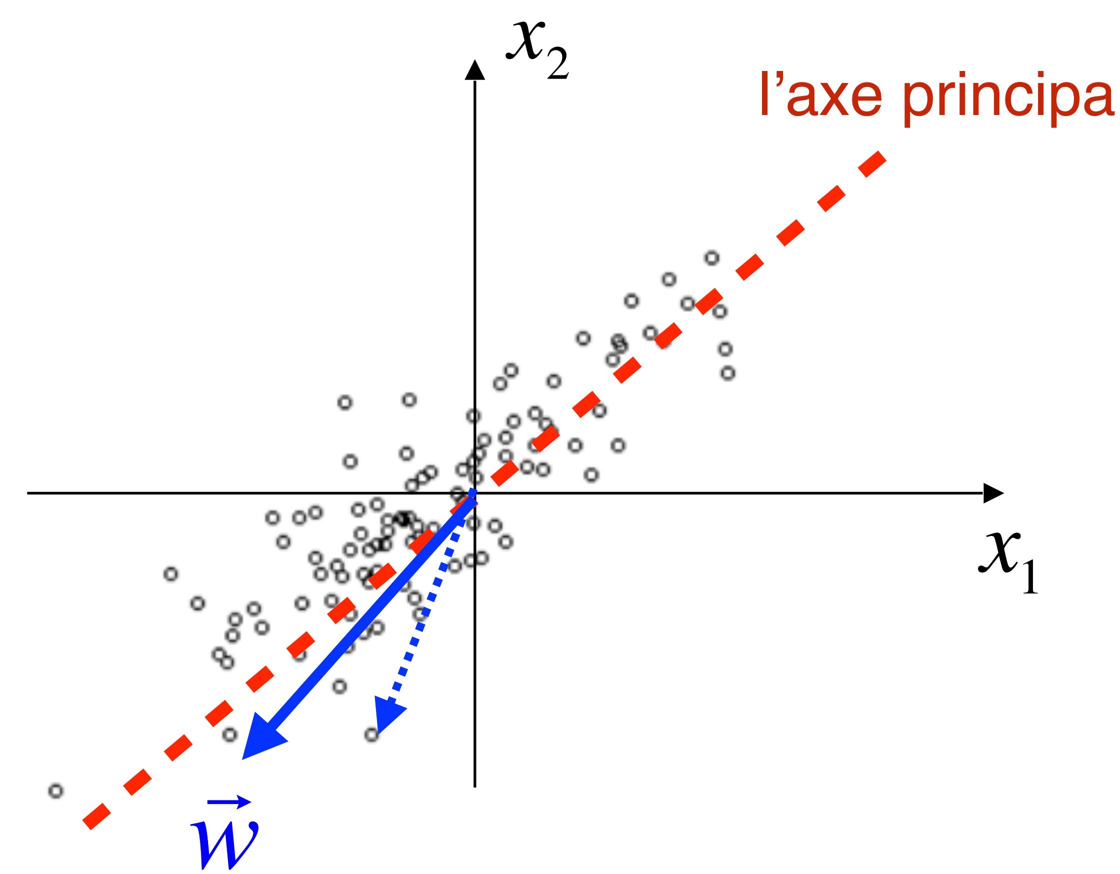
Apprentissage non-supervisé Hebbien



$$y = \sum_{i=1}^{D} w_i x_i = \vec{w} \cdot \vec{x}$$

$$\Delta \vec{w} = \eta y \vec{x}$$

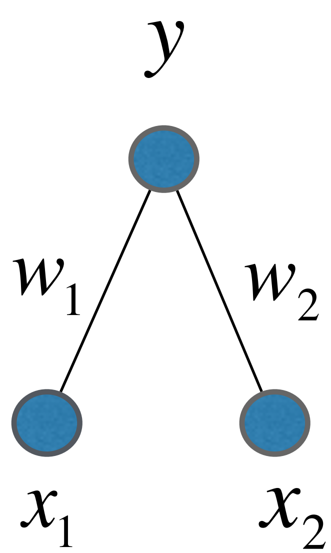
règle d'apprentissage
"hebbienne"



Pendant l'apprentissage

1. On choisie un vecteur de données $\vec{x}^{(p)}$
2. On calcule la sortie $y^{(p)} = \vec{w} \cdot \vec{x}$
3. On ajuste les poids selon la règle Hebbienne
4. On répète 1-3 pour chaque vecteur dans l'ensemble

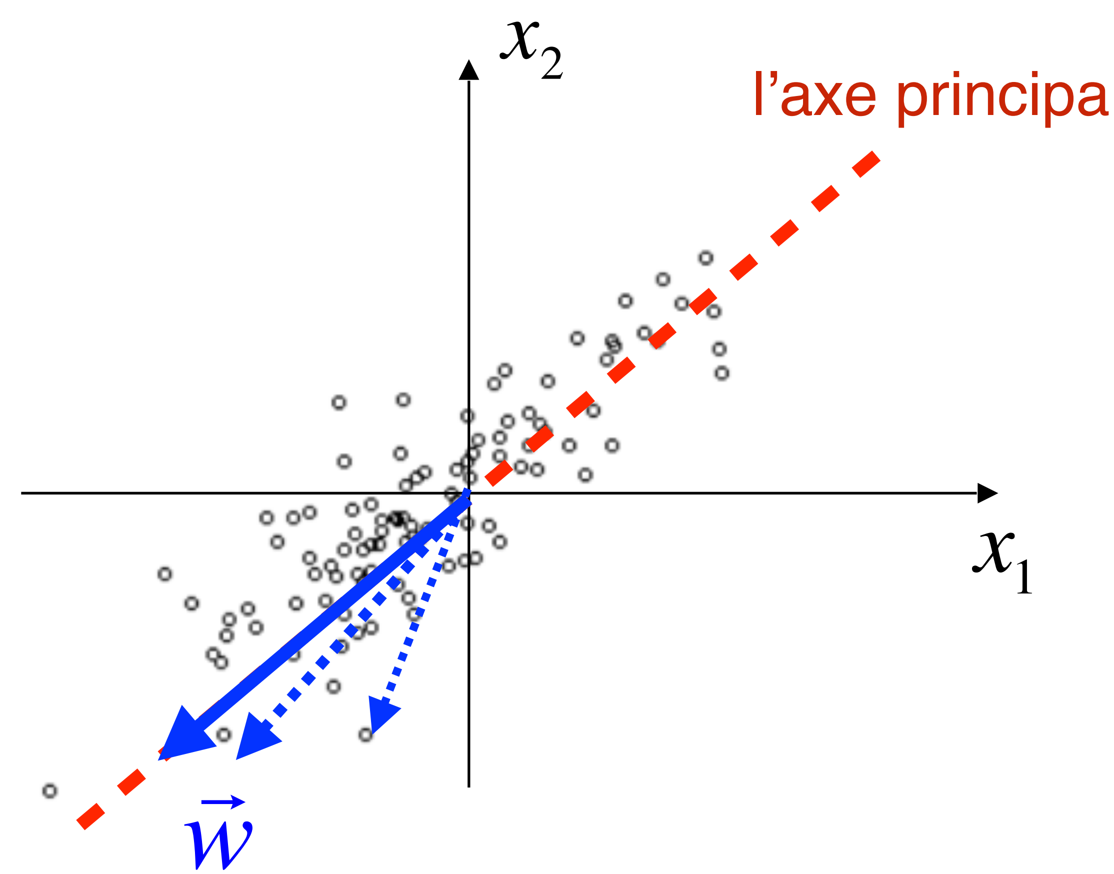
Apprentissage non-supervisé Hebbien



$$y = \sum_{i=1}^{D} w_i x_i = \vec{w} \cdot \vec{x}$$

$$\Delta \vec{w} = \eta y \vec{x}$$

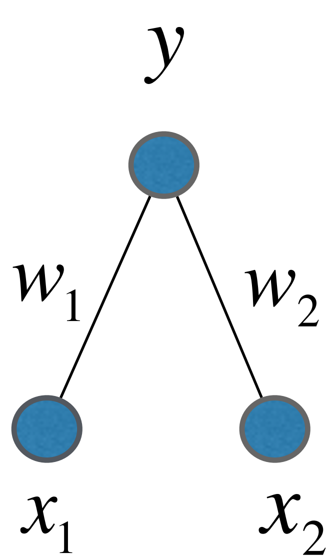
règle d'apprentissage "hebbienne"



Après l'apprentissage

La direction du vecteur de poids correspond à la direction de l'axe principal

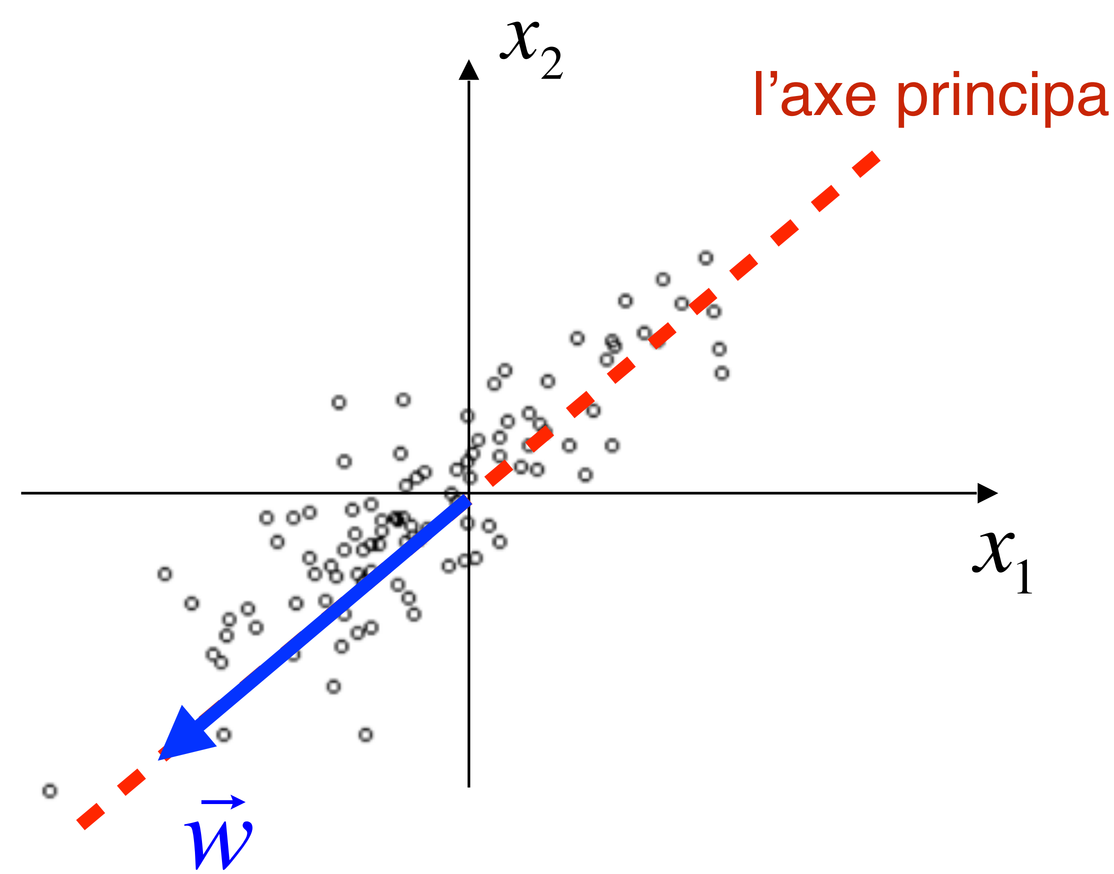
Apprentissage non-supervisé Hebbien



$$y = \sum_{i=1}^{D} w_i x_i = \vec{w} \cdot \vec{x}$$

$$\Delta \vec{w} = \eta y \vec{x}$$

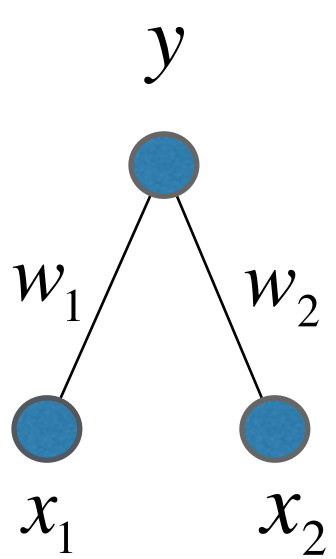
règle d'apprentissage
"hebbienne"



Apprentissage Hebbien sur l'ensemble de données résulte en un **vecteur de poids** qui **correspond au 1^er^ axe principale** de l'ensemble en 2 dimensions.

Le calcul fait par le réseau : projection du vecteur de données sur l'axe principal (càd **la 1ère composante principale** du vecteur)

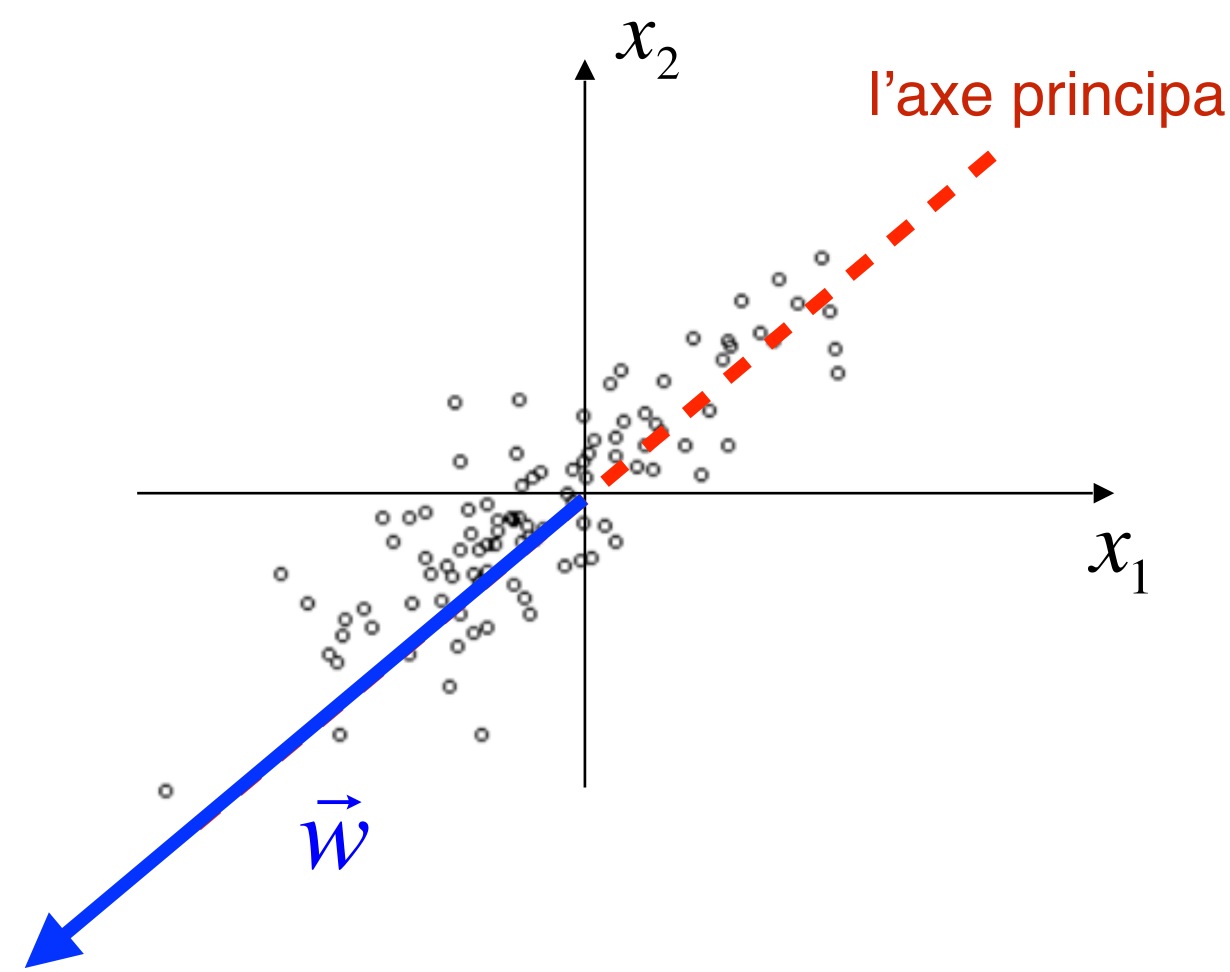
Apprentissage non-supervisé Hebbien



$$y = \sum_{i=1}^{D} w_i x_i = \vec{w} \cdot \vec{x}$$

$$\Delta\vec{w} = \eta y \vec{x}$$

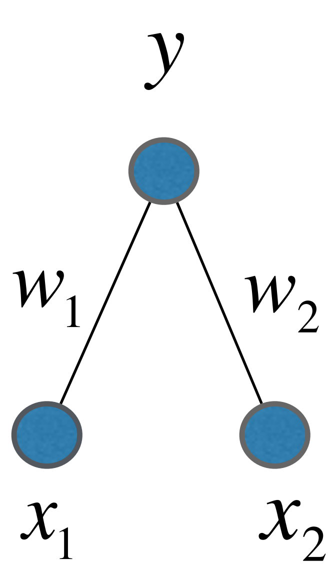
règle d'apprentissage "hebbienne"



Deux problèmes de la règle Hebbienne:

- La norme de $\vec{w}$ augmente sans limite
- Pour que la sortie du réseau effectue une **projection** sur l'axe principal, il faut que le vecteur de poids soit normalisé ($||\vec{w}|| = 1$)

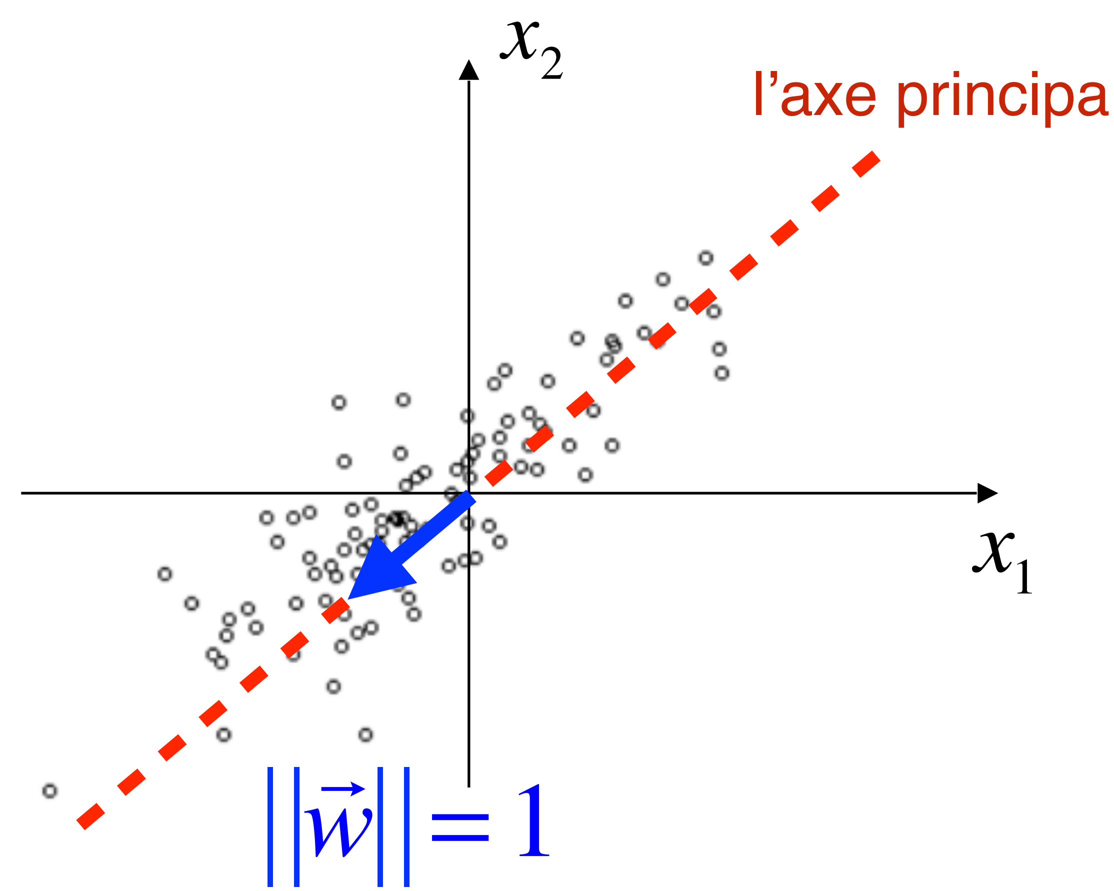
Apprentissage non-supervisé Hebbien



$$y = \sum_{i=1}^{D} w_i x_i = \vec{w} \cdot \vec{x}$$

$$\Delta \vec{w} = \eta y (\vec{x} - y \vec{w})$$

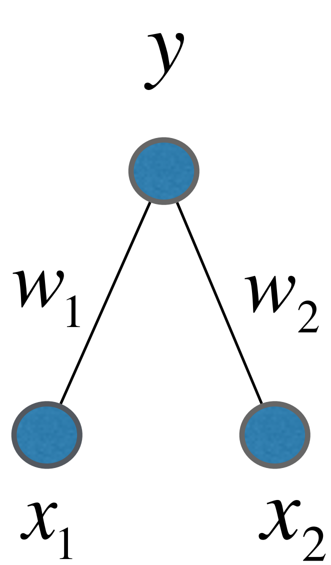
règle d'apprentissage
dite "règle d'Oja"



La règle Hebbien modifié (appelé **la règle d'Oja)** normalise automatiquement le vecteur des poids.

E. Oja (1982) *A simplified neuron model as a principal component analyzer*. *J. Mathematical Biology* **15**, pp. 267–273

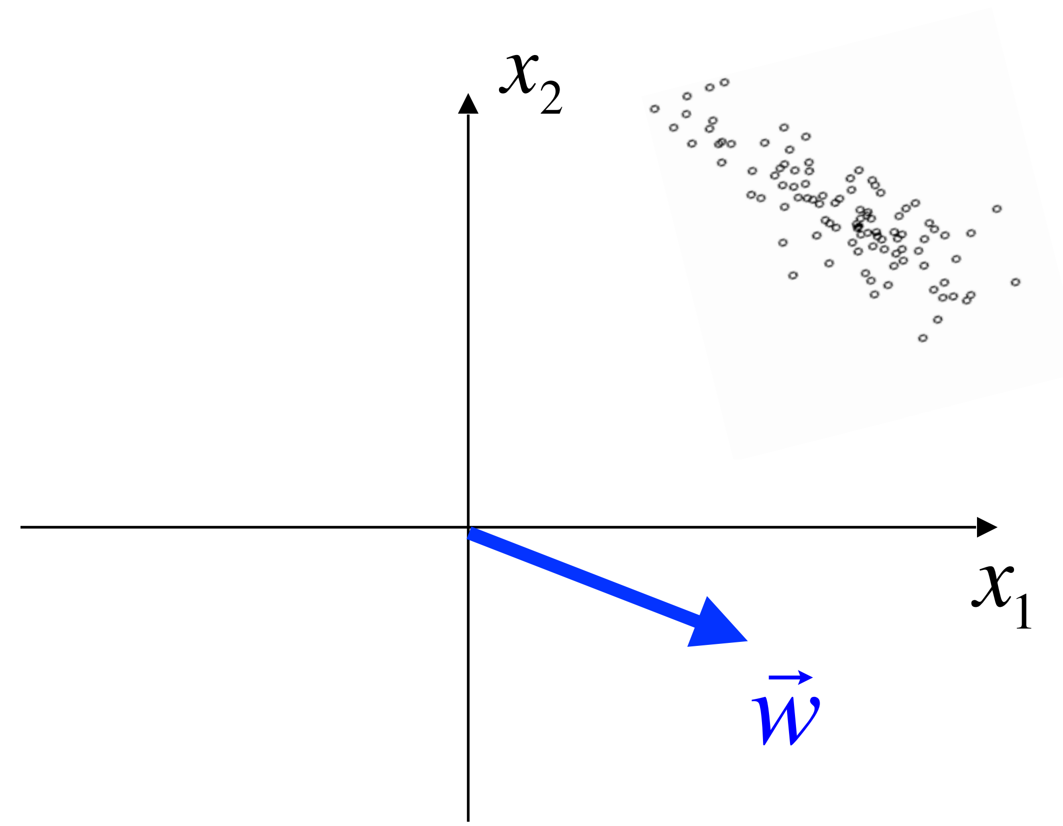
Apprentissage non-supervisé Hebbien



$$y = \sum_{i=1}^{D} w_i x_i = \vec{w} \cdot \vec{x}$$

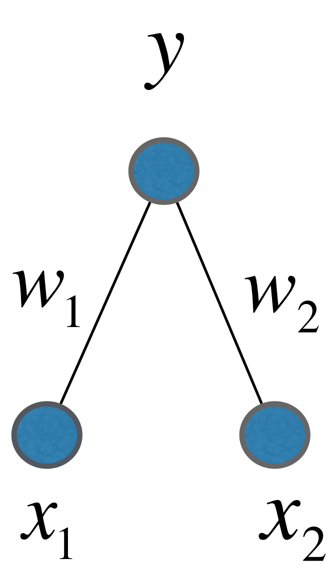
$$\Delta \vec{w} = \eta y \vec{x}$$

règle hebbienne



Que se passe si les données ne sont pas centrées ?

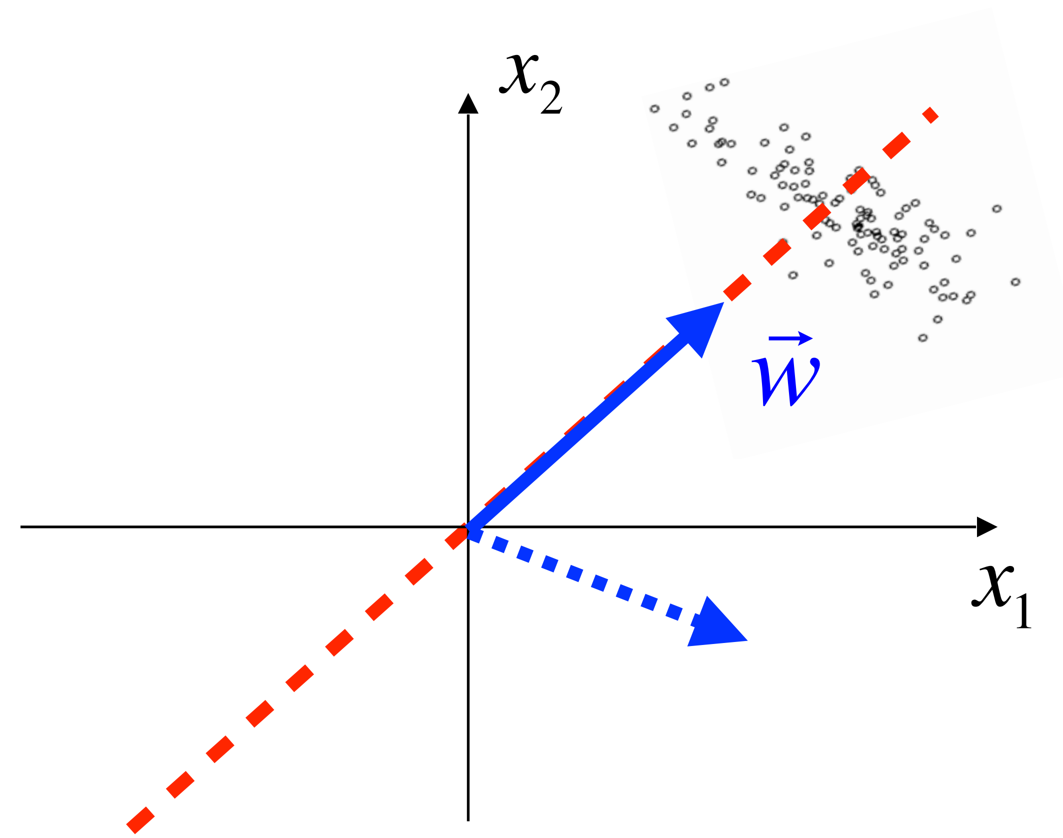
Apprentissage non-supervisé Hebbien



$$y = \sum_{i=1}^{D} w_i x_i = \vec{w} \cdot \vec{x}$$

$$\Delta \vec{w} = \eta y \vec{x}$$

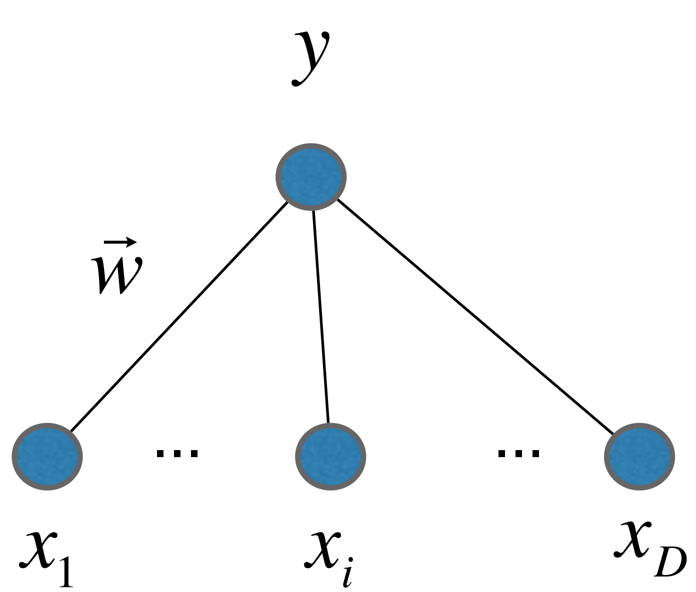
règle hebbienne



Que se passe si les données ne sont pas centrées ?

Le vecteur de poids est dirigé vers le centre de masse de l'ensemble (car l'axe principal est calculé par rapport à l'origine)

Apprentissage non-supervisé Hebbien avec un neurone de sortie : Synthèse



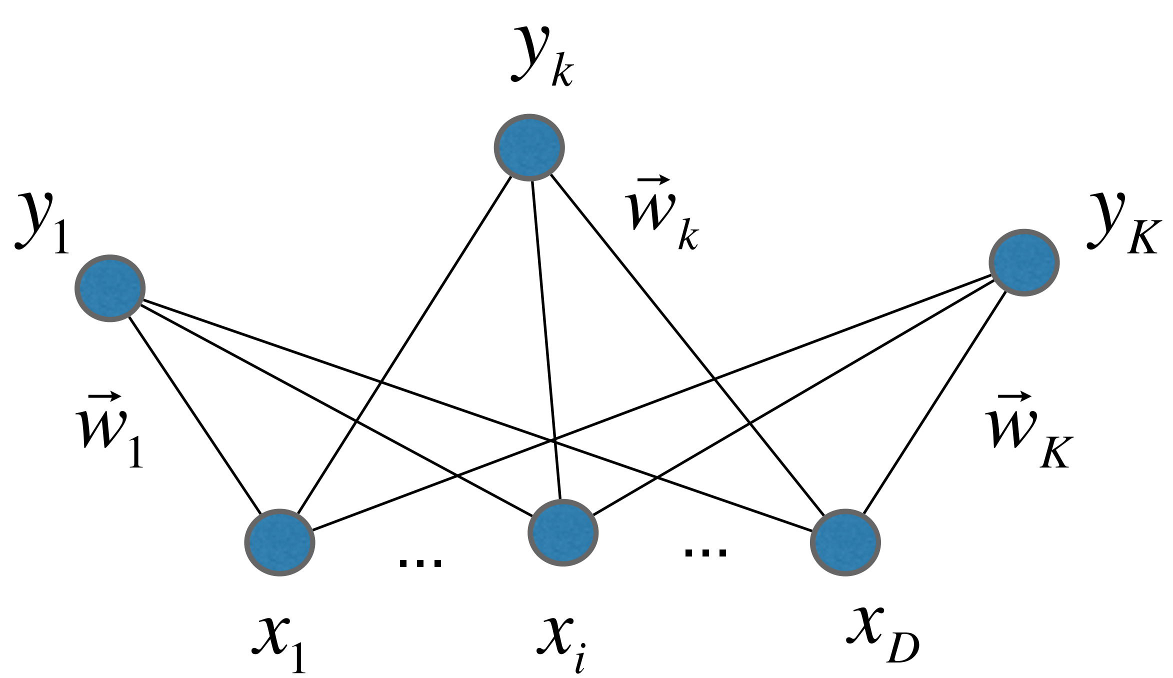
Le réseau avec un neurone de sortie, D entrées et l'apprentissage hebbien extrait le 1er axe principal des données dans D dimensions. La sortie du réseau après l'apprentissage correspond à la 1ère composante principale du vecteur d'entrée

$$y = \vec{w} \cdot \vec{x}$$

Règle Hebbien "pure" $\quad \Delta\vec{w} = \eta y \vec{x}$

Règle de Oja $\quad \Delta\vec{w} = \eta y (\vec{x} - y\vec{w})$

Apprentissage non-supervisé Hebbien avec un neurone de sortie : Synthèse



Le réseau avec un neurone de sortie, D entrées et l'apprentissage hebbien extrait le 1er axe principal des données dans D dimensions. La sortie du réseau après l'apprentissage correspond à la 1ère composante principale du vecteur d'entrée

$$y_k = \vec{w}_k \cdot \vec{x}$$

Que se passe si on ajoute d'autres neurones de sortie ?

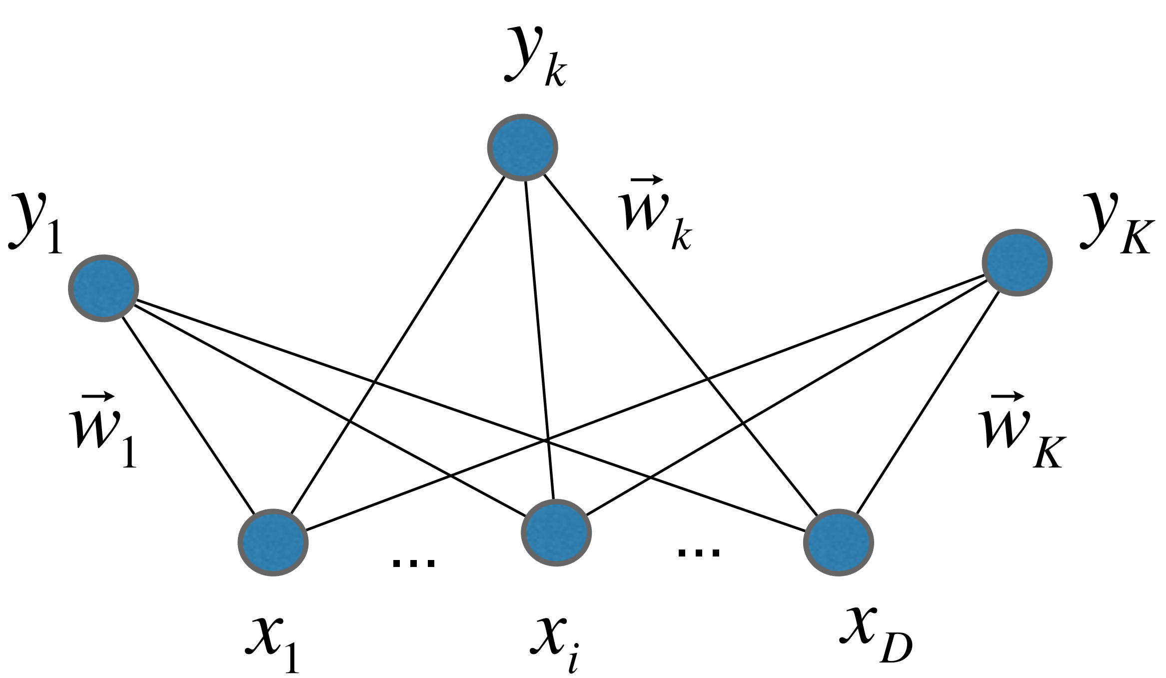
Règle Hebbien "pure"

$$\Delta \vec{w}_k = \eta y_k \vec{x}$$

Règle de Oja

$$\Delta \vec{w}_k = \eta y_k (\vec{x} - y_k \vec{w}_k)$$

Apprentissage non-supervisé Hebbien avec un neurone de sortie : Synthèse



Le réseau avec un neurone de sortie, D entrées et l'apprentissage hebbien extrait le 1er axe principal des données dans D dimensions. La sortie du réseau après l'apprentissage correspond à la 1ère composante principale du vecteur d'entrée

$$y_k = \vec{w}_k \cdot \vec{x}$$

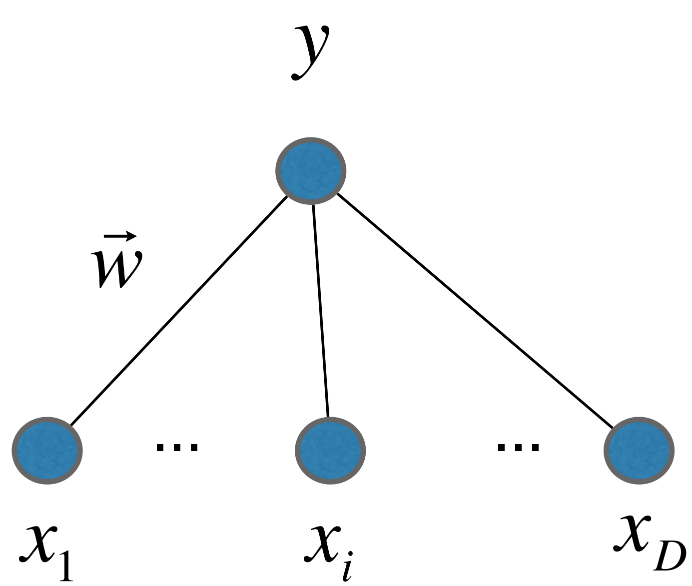
Règle Hebbien "pure" $$\Delta \vec{w}_k = \eta y_k \vec{x}$$

Règle de Oja $$\Delta \vec{w}_k = \eta y_k (\vec{x} - y_k \vec{w}_k)$$

Car tous le neurones de sortie "voient" le même ensemble, ils apprennent tous la même chose.

Alors tous les vecteurs de poids seront dirigés vers la même direction

Apprentissage non-supervisé Hebbien avec un neurone de sortie : Synthèse



Le réseau avec un neurone de sortie, D entrées et l'apprentissage hebbien extrait le 1er axe principal des données dans D dimensions. La sortie du réseau après l'apprentissage correspond à la 1ère composante principale du vecteur d'entrée

$$y = \vec{w} \cdot \vec{x}$$

Règle Hebbien "pure"

$$\Delta\vec{w} = \eta y \vec{x}$$

Règle de Oja

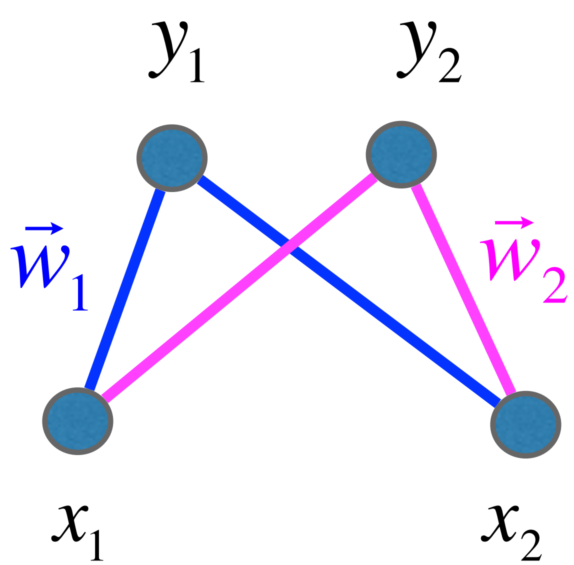
$$\Delta\vec{w} = \eta y(\vec{x} - y\vec{w})$$

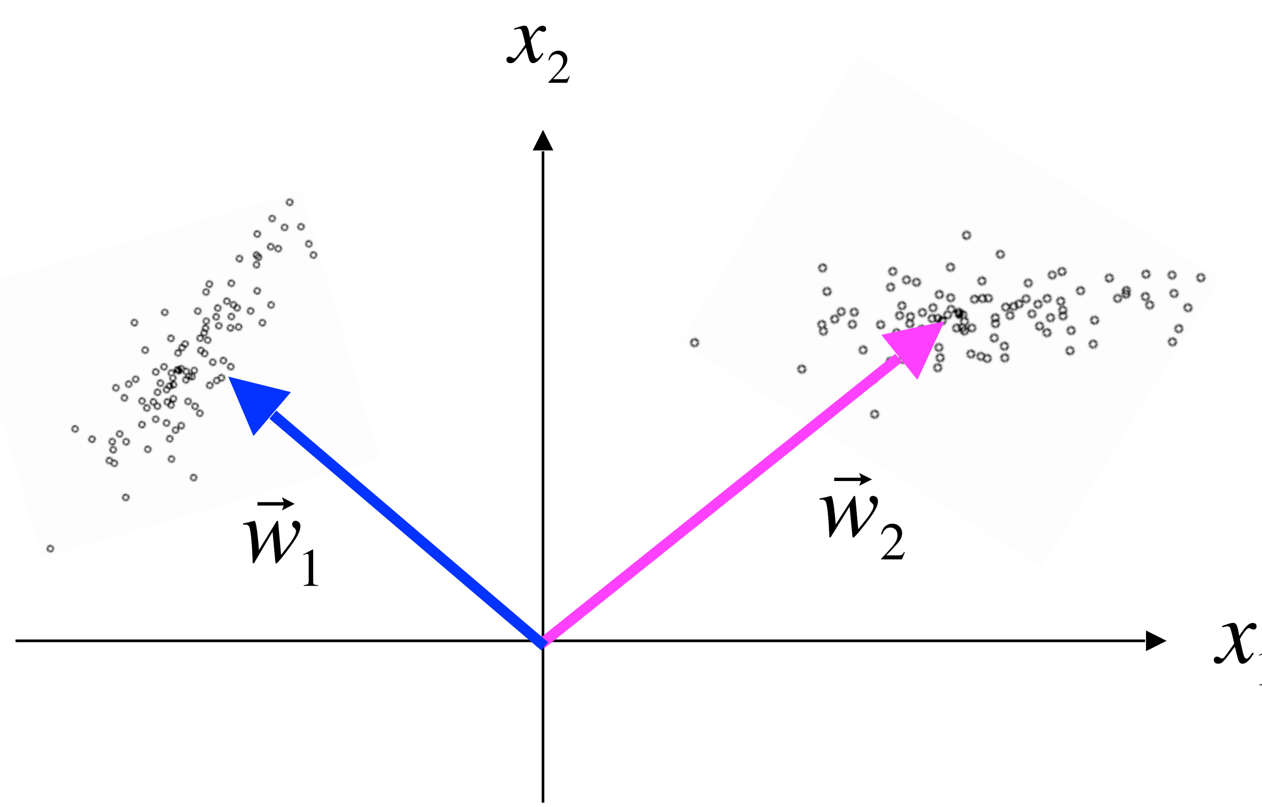
Si l'on remplace la fonction d'activation linéaire par la fonction sigmoïde, cela ne changera pas le résultat d'apprentissage

(Bishop, 1995, Neural networks for pattern recognition)

Apprentissage compétitif

Apprentissage compétitif



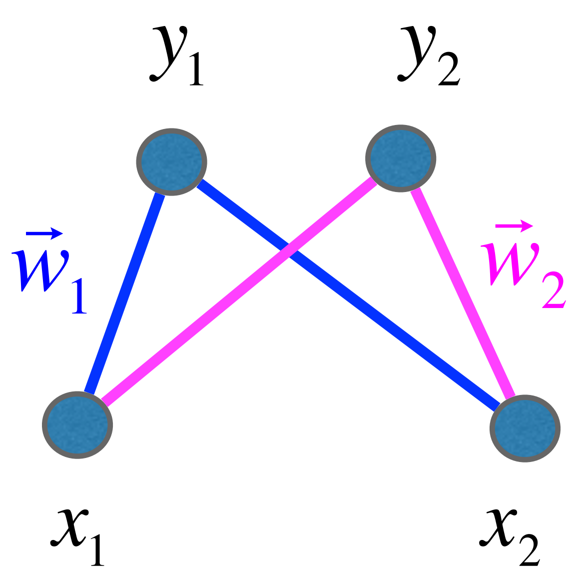


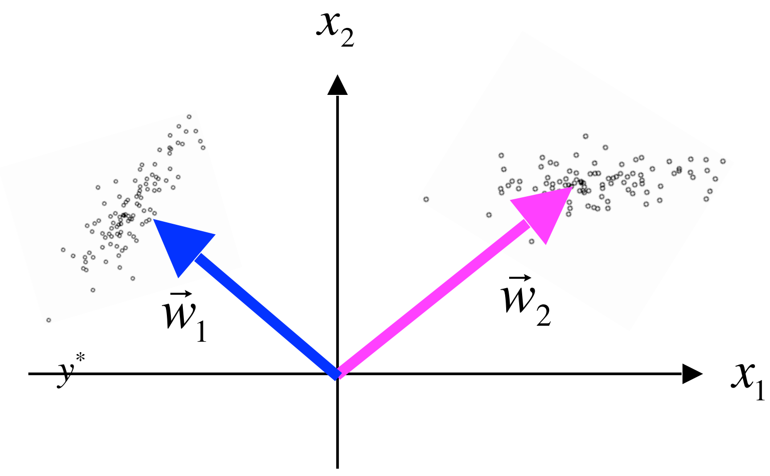
$$y_k = \vec{w}_k \cdot \vec{x}$$

L'idée de l'apprentissage compétitif est de forcer une **compétition** entre les neurones de sortie :

seulement les poids du neurone "gagnant" (avec l'activité maximale) met sont mis à jour pendant l'apprentissage

Apprentissage compétitif





$$y_k = \vec{w}_k \cdot \vec{x}$$

$$\Delta \vec{w}_k = \eta y_k^* (\vec{x} - y_k^* \vec{w}_k)$$

où

$$y_k^* = \begin{cases} 1, \text{ si } y_k \text{ est le “winner”} \\ 0, \text{ sinon} \end{cases}$$

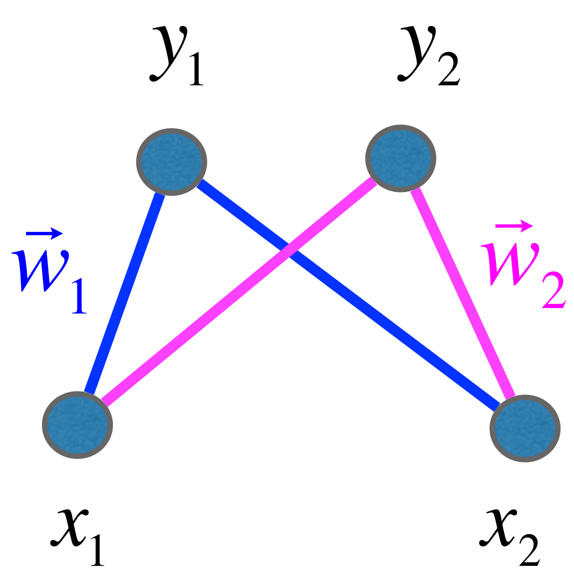
règle d'apprentissage compétitif (la règle d'Oja modifiée)

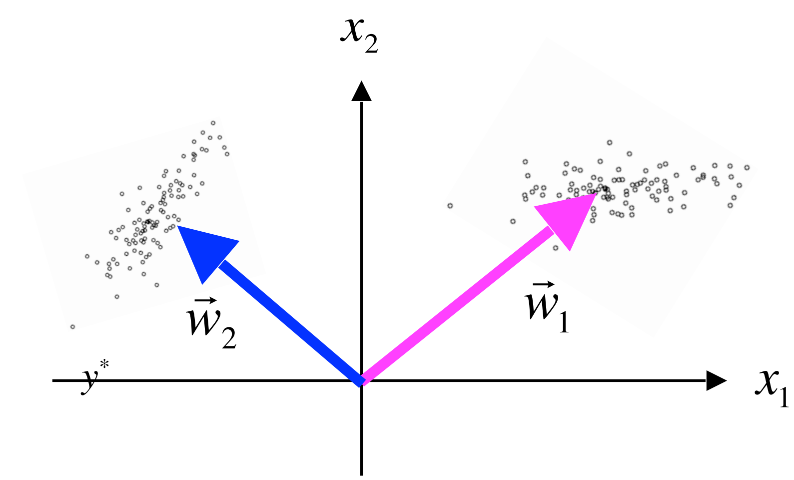
Algorithme :

1. Choisir un exemple de l'ensemble d'entrainement, le présenter au réseau et calculer les 2 sorties
2. Déterminer le neurone de sortie avec l'activité maximale (neurone gagnant, ou “winner neuron”)
3. Mettre à jour les poids du neurone gagnant selon la règle d'apprentissage compétitif
4. Répéter 1-4 jusqu'à ce que les vecteurs de poids ne changent plus

Réseau de type “**Winner take all (WTA)**”

Apprentissage compétitif





$$y_k = \vec{w}_k \cdot \vec{x}$$

$$\Delta \vec{w}_k = \eta y_k^* (\vec{x} - y_k^* \vec{w}_k)$$

où

$$y_k^* = \begin{cases} 1, \text{ si } \; y_k \text{ est le “winner”} \\ 0, \text{ sinon} \end{cases}$$

règle d'apprentissage compétitif (la règle d'Oja modifiée)

Résultat d'apprentissage (clustering automatique) :

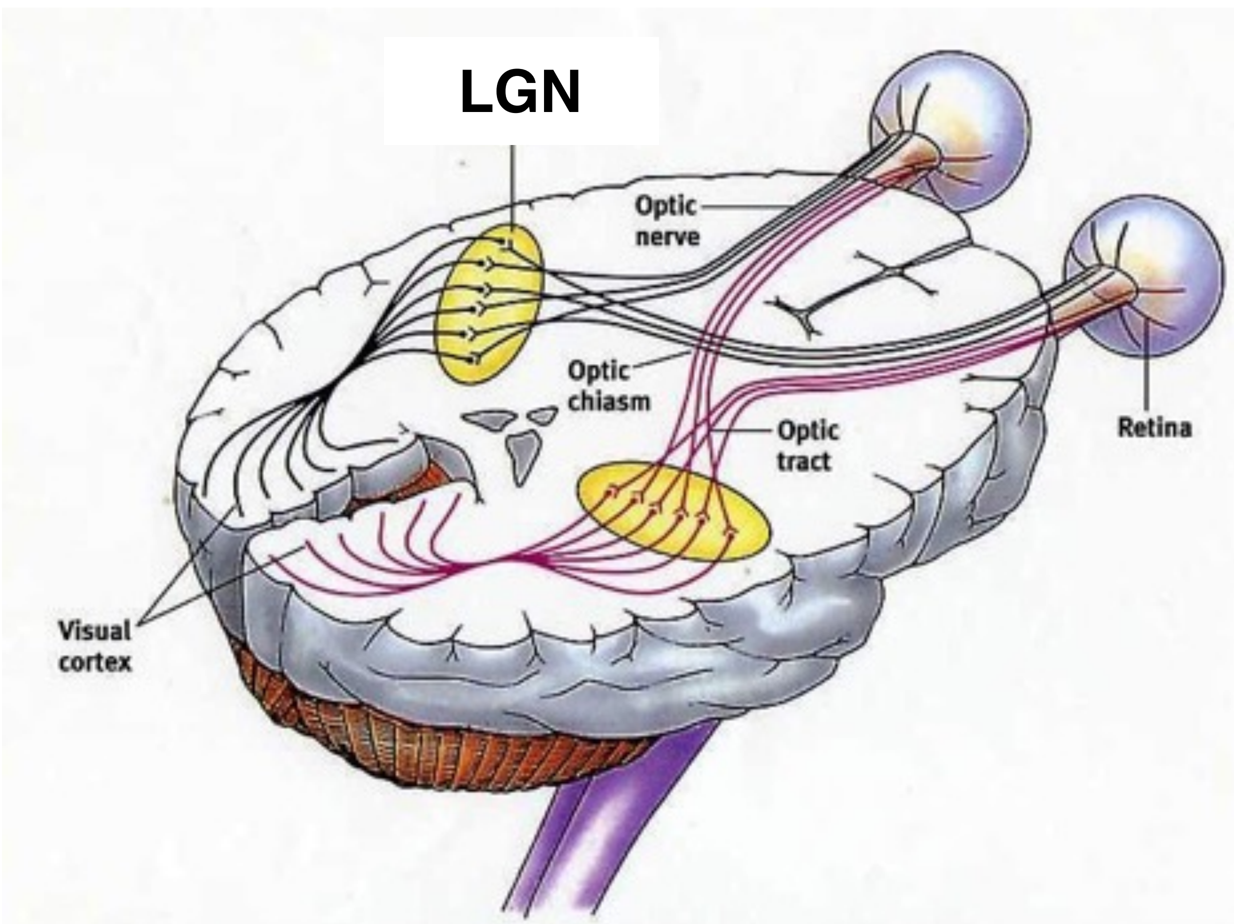
1. Les vecteurs de poids correspondent aux centres de masse de différents sous-ensembles de points

3. Application du réseau à un vecteur de données calcule la projection du vecteur sur les vecteurs de poids. Le neurone gagnant correspond alors à la classe (cluster) de ce vecteur.

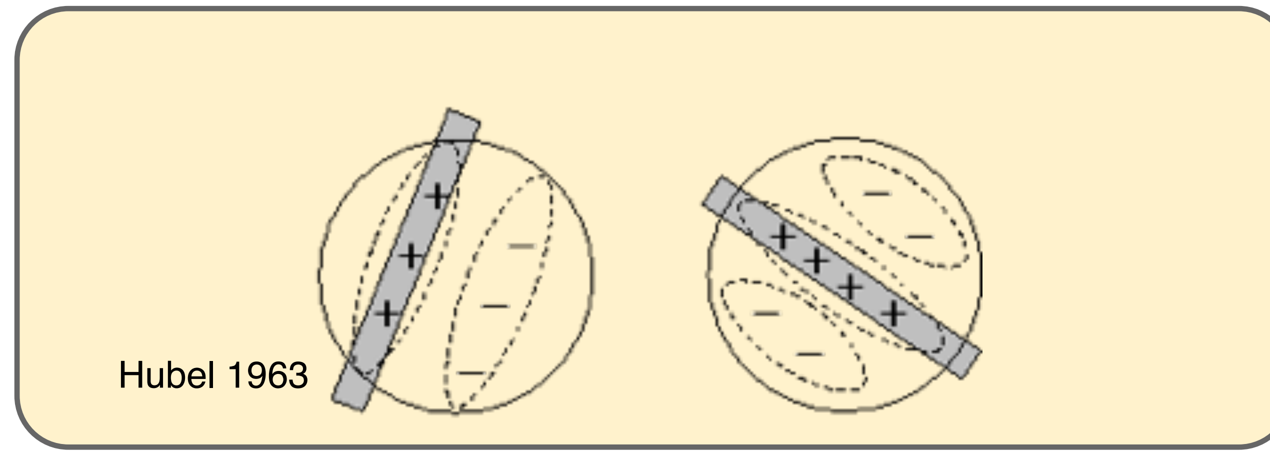
Réseau de type “**Winner take all (WTA)**”

Cas d'étude : apprentissage non-supervisé de champs récepteurs visuels

corps géniculé latéral

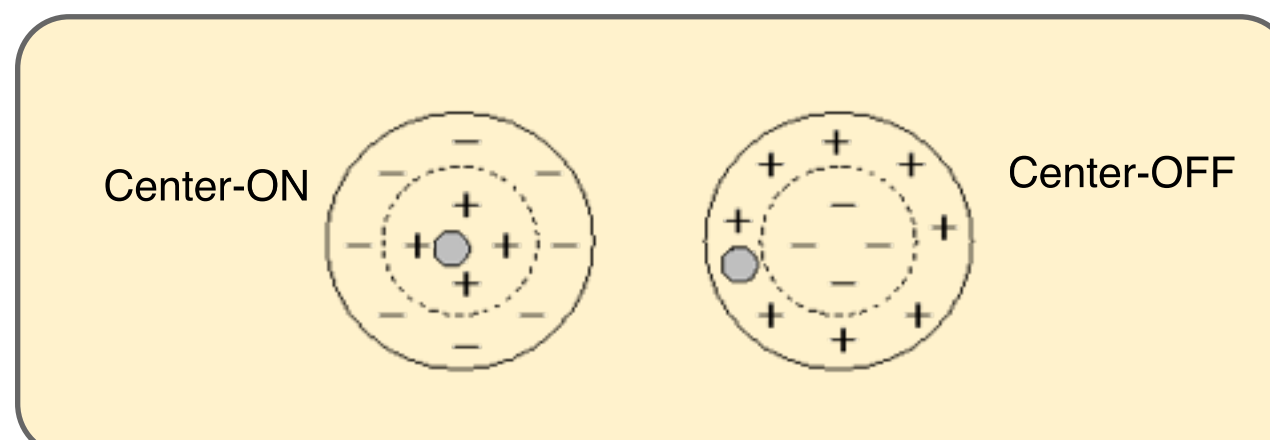


Champs récepteur dans le cortex visuel (V1)



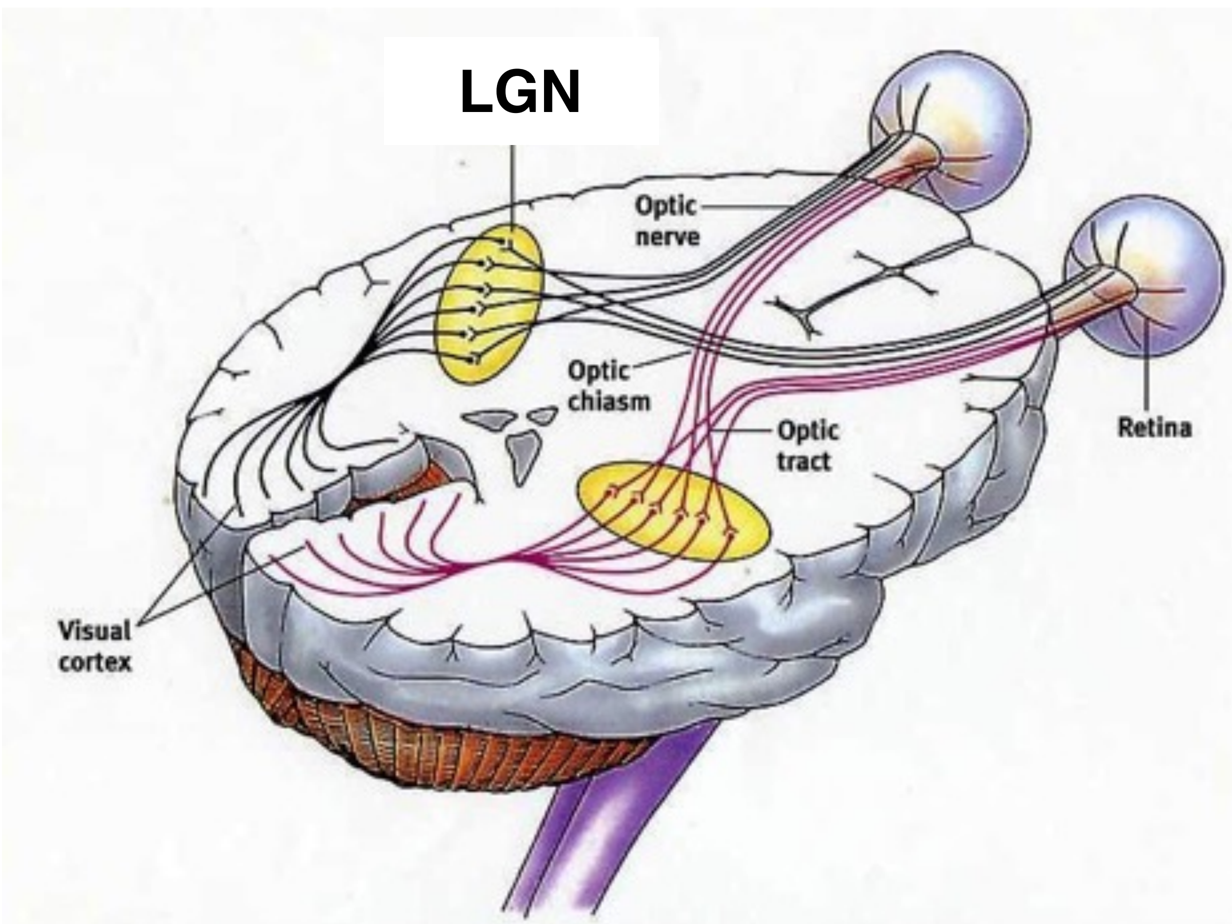
Règle d'apprentissage ?

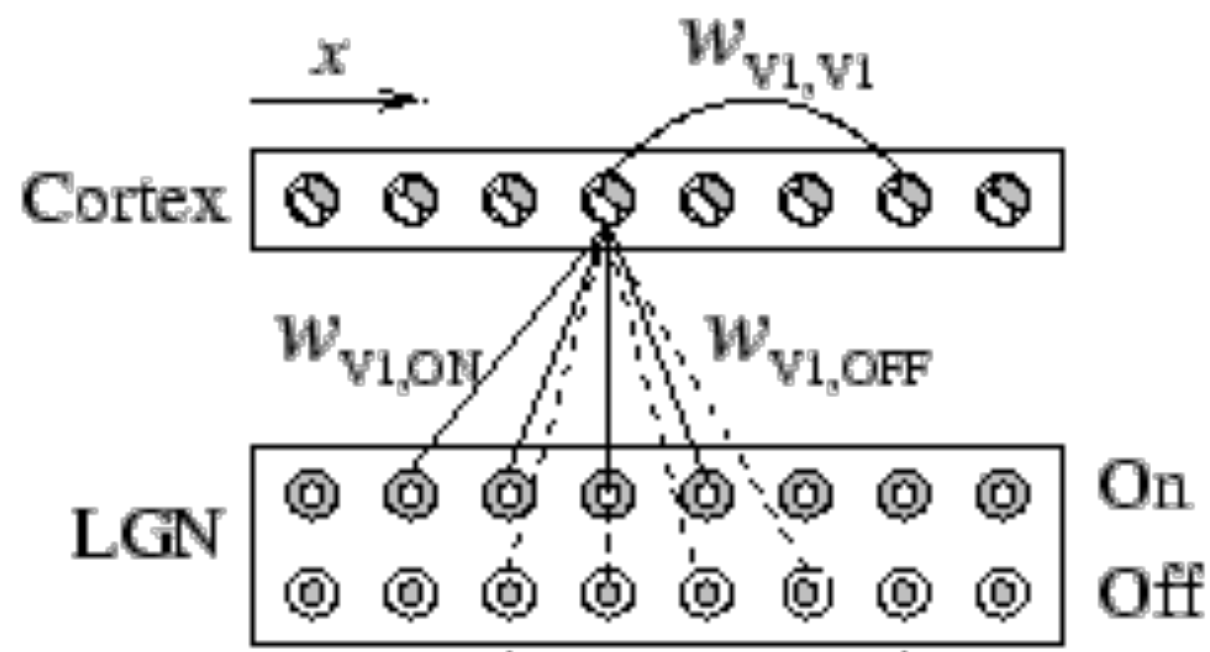
Champs récepteur dans la rétine et le LGN



Cas d'étude : apprentissage non-supervisé de champs récepteurs visuels

corps géniculé latéral

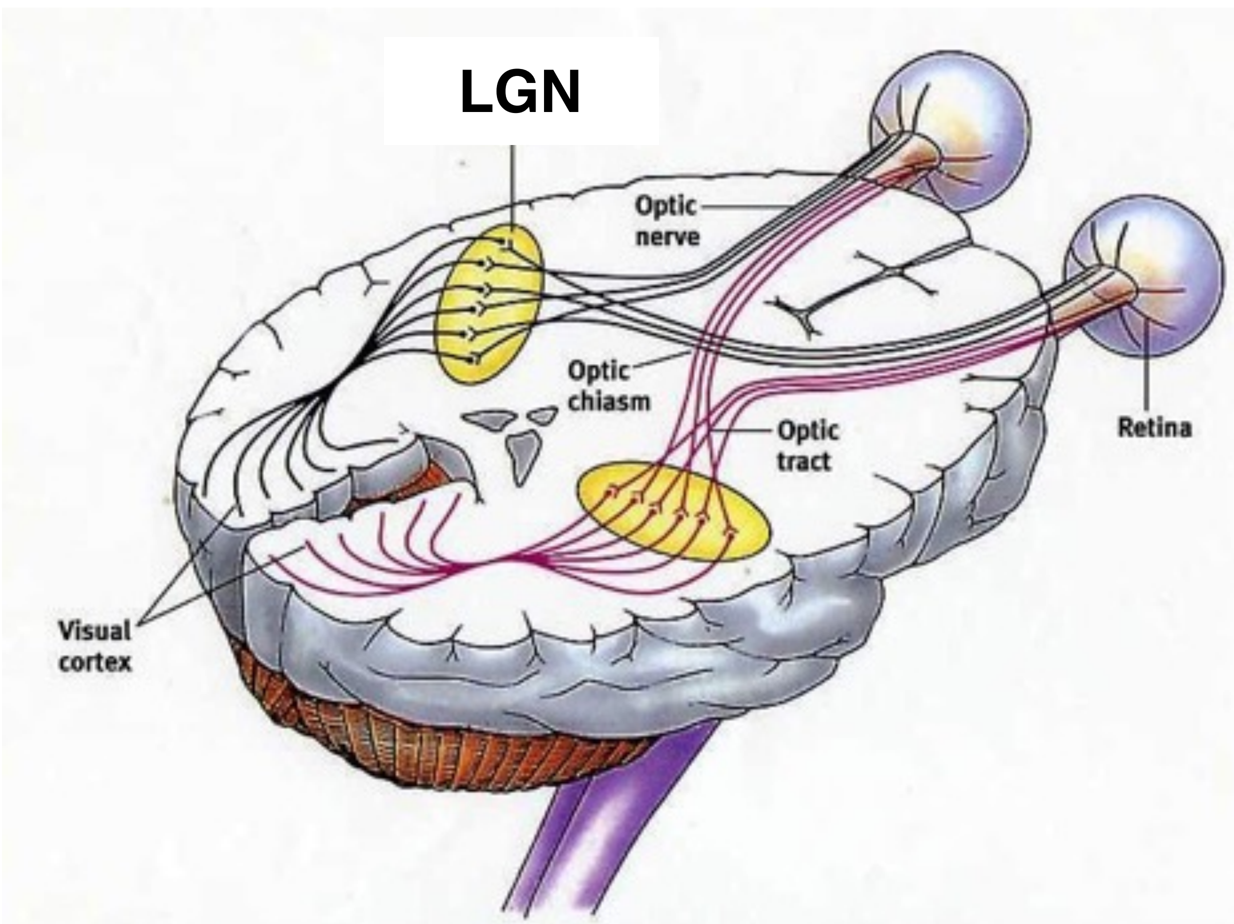




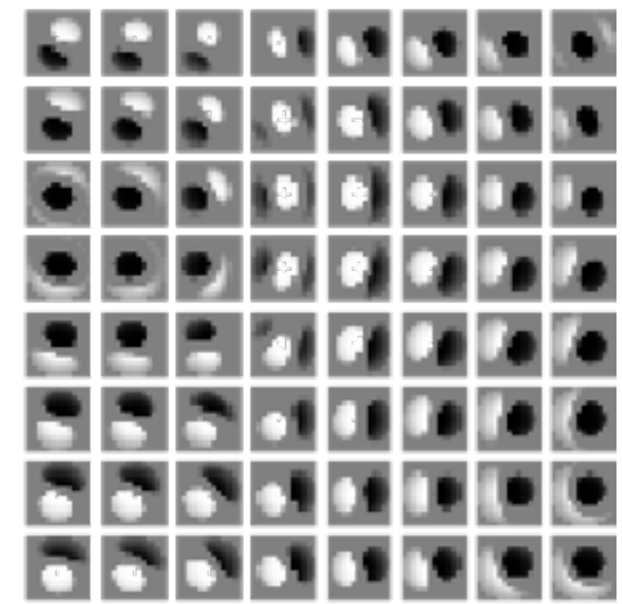
Miller, K. D. (1994). A model for the development of simple cell receptive fields and the ordered arrangement of orientation columns through activity dependent competition between ON- and OFF-center inputs. J. Neurosci., 14:409-441.

Cas d'étude : apprentissage non-supervisé de champs récepteurs visuels

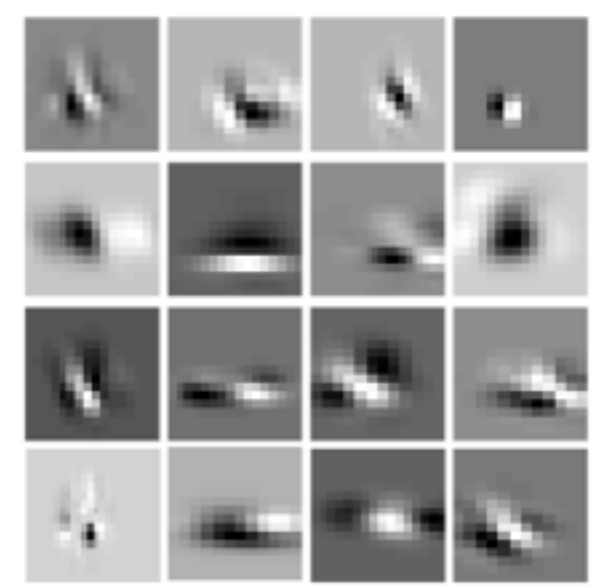
corps géniculé latéral



Modèle d'apprentissage compétitif



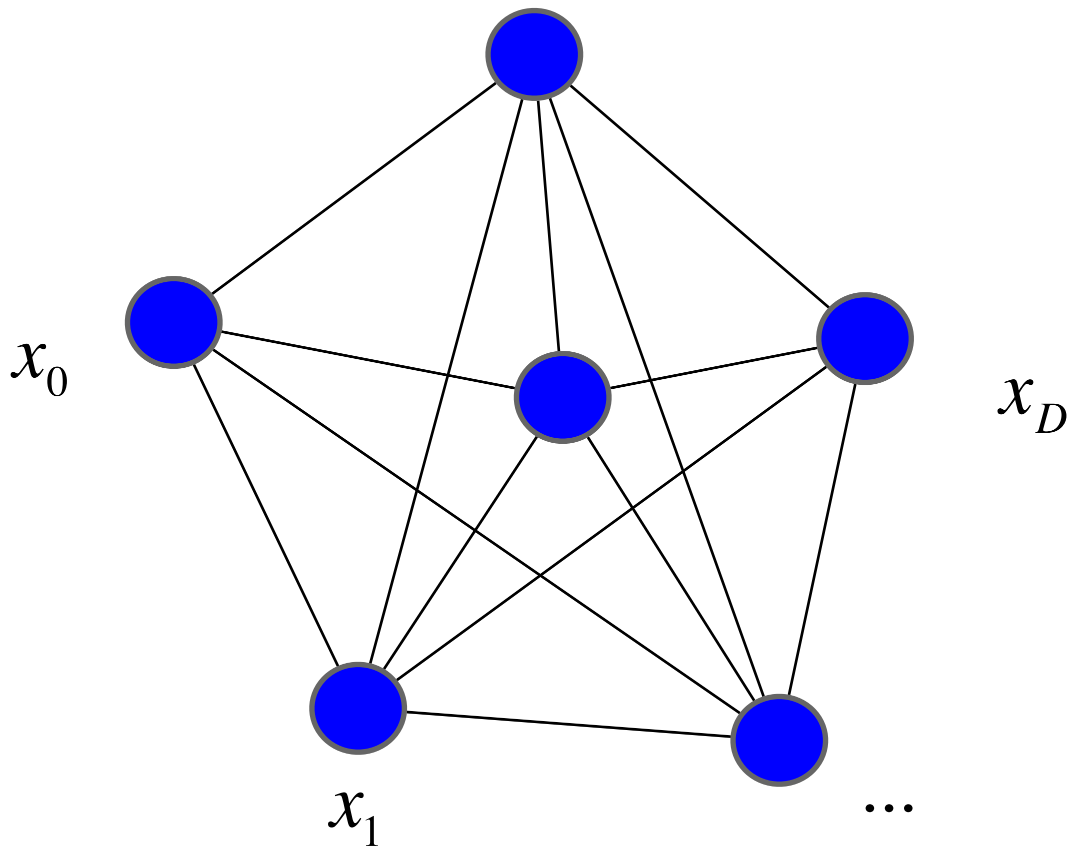
Enregistrements des champs récepteur dans le cortex visuel



Réseaux récurrents

Réseau **récurrent**

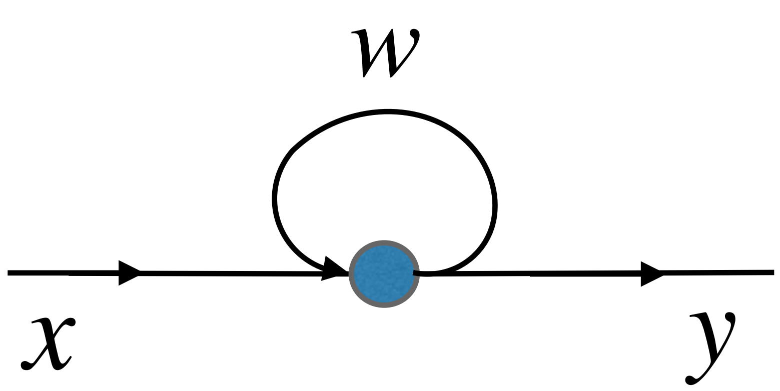
$$x_k = f(x_0, x_1 ..., x_D)$$



Exemple : cortex, hippocampe

Un neurone avec une synapse récurrente

"Autapse"



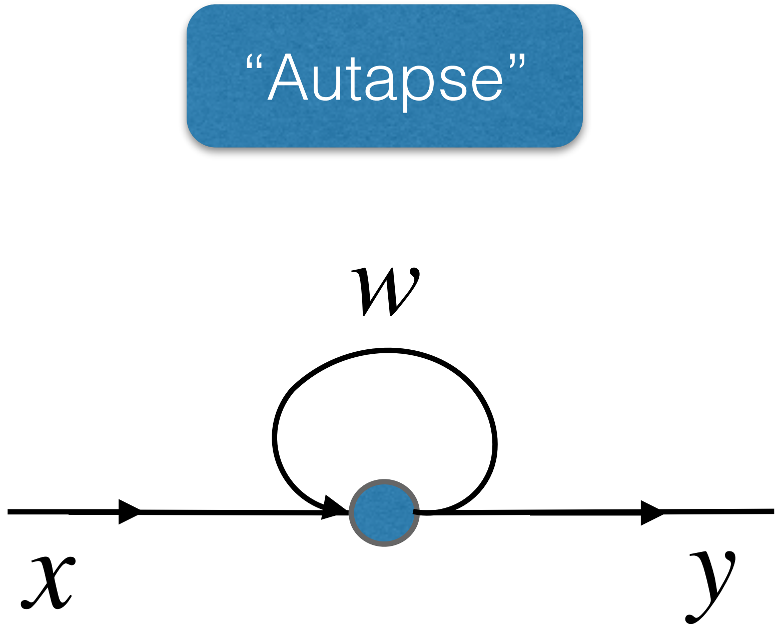
Activation $a = wy + x$

Fonction d'activation $g(a) = \dfrac{1}{1+e^{-a}}$

Sortie du réseau $y = g(wy + x)$

Que fait ce réseau ?

Un neurone avec une synapse récurrente



Activation $a = wy + x$

Fonction d'activation $g(a) = \dfrac{1}{1+e^{-a}}$

Sortie du réseau $y = g(wy + x)$

- L'autapse peut être analysée sous forme d'une équation différentielle suivante

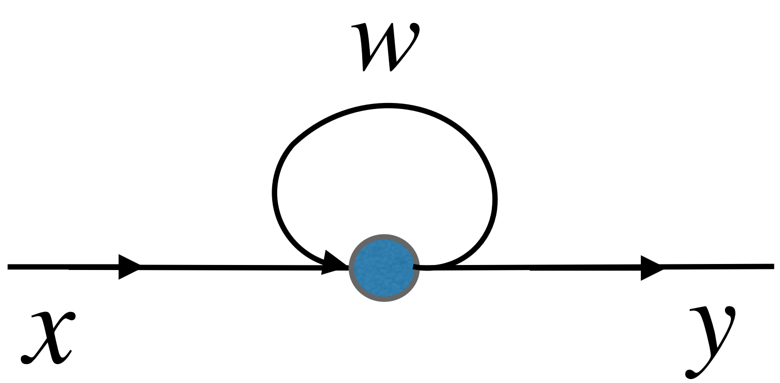
$$\dot{y} = -y + g(wy + x)$$

- La sortie du réseau correspond à l'état stable du système dynamique

$$\dot{y} = 0 \quad \Rightarrow \quad y = g(wy + x)$$

Un neurone avec une synapse récurrente

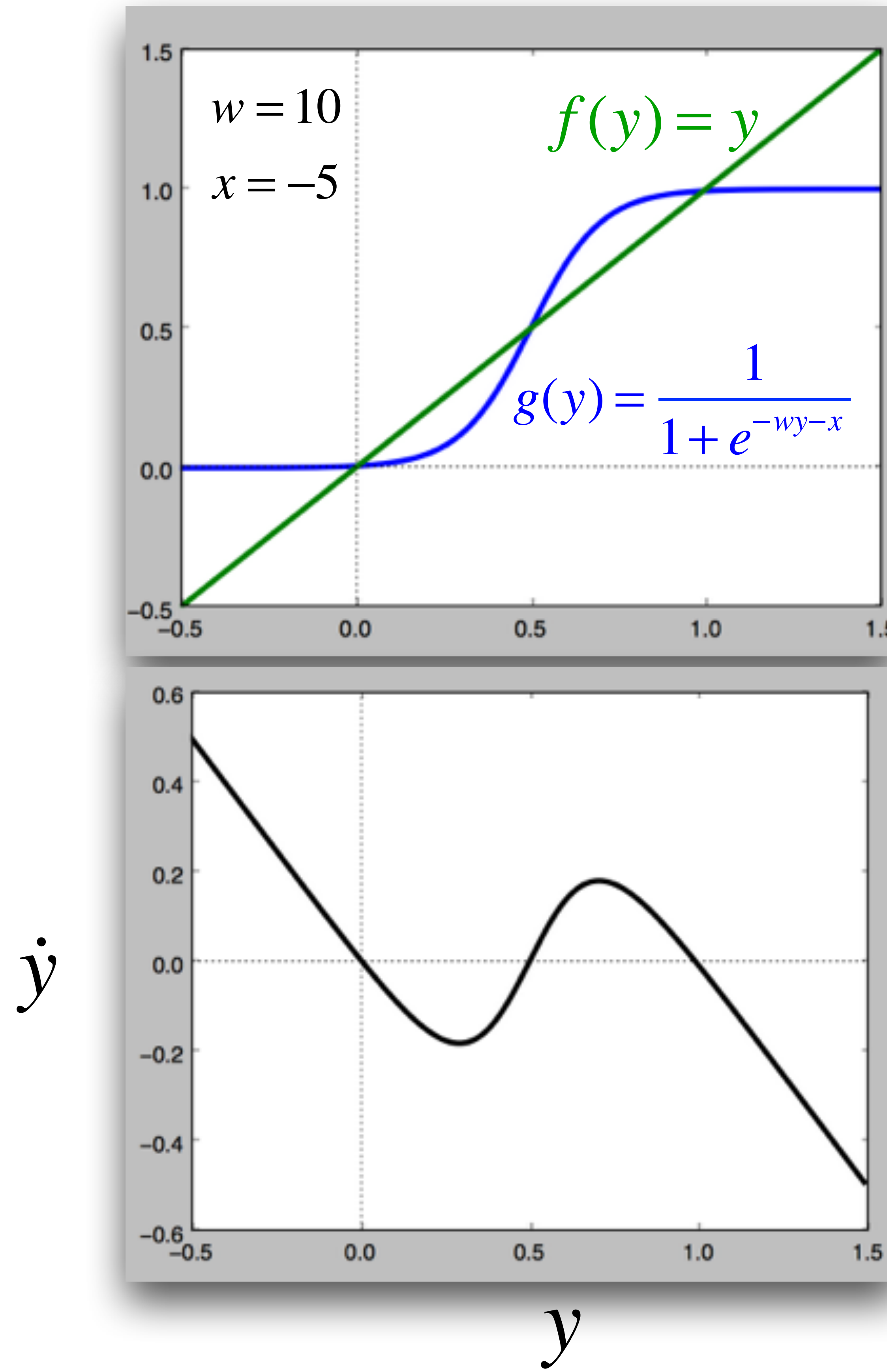
"Autapse"



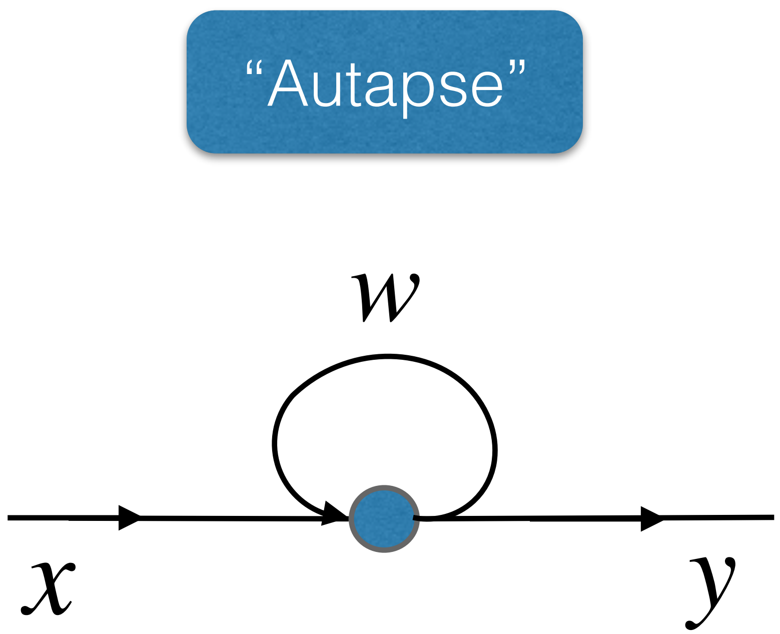
$$\dot{y} = -y + \frac{1}{1+e^{-wy-x}}$$

Synapse excitatrice $w = 10$

Entrée constante inhibitrice $x = -5$



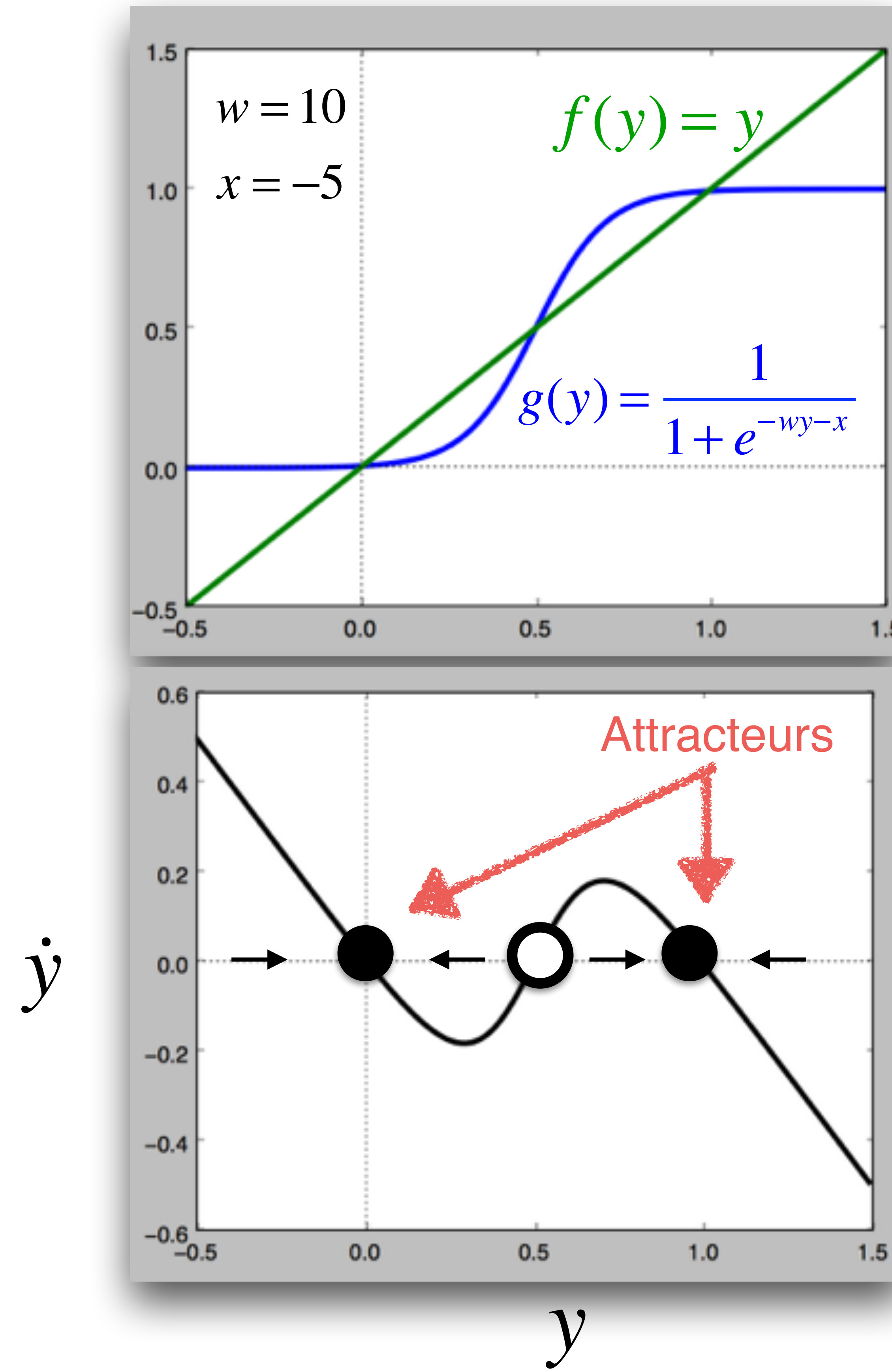
Un neurone avec une synapse récurrente



$$\dot{y} = -y + \frac{1}{1+e^{-wy-x}}$$

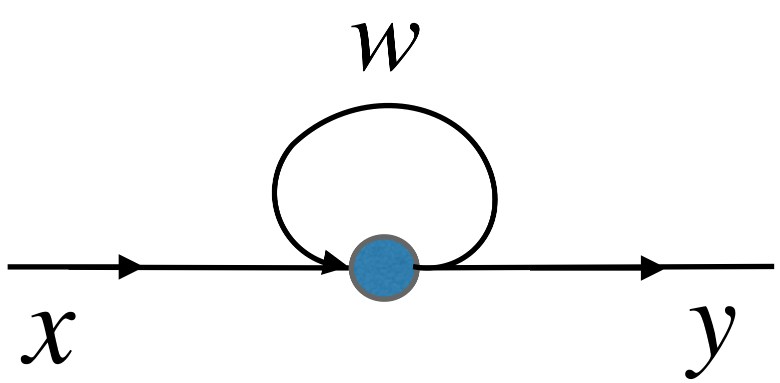
Synapse excitatrice $w = 10$

Entrée constante inhibitrice $x = -5$



Un neurone avec une synapse récurrent

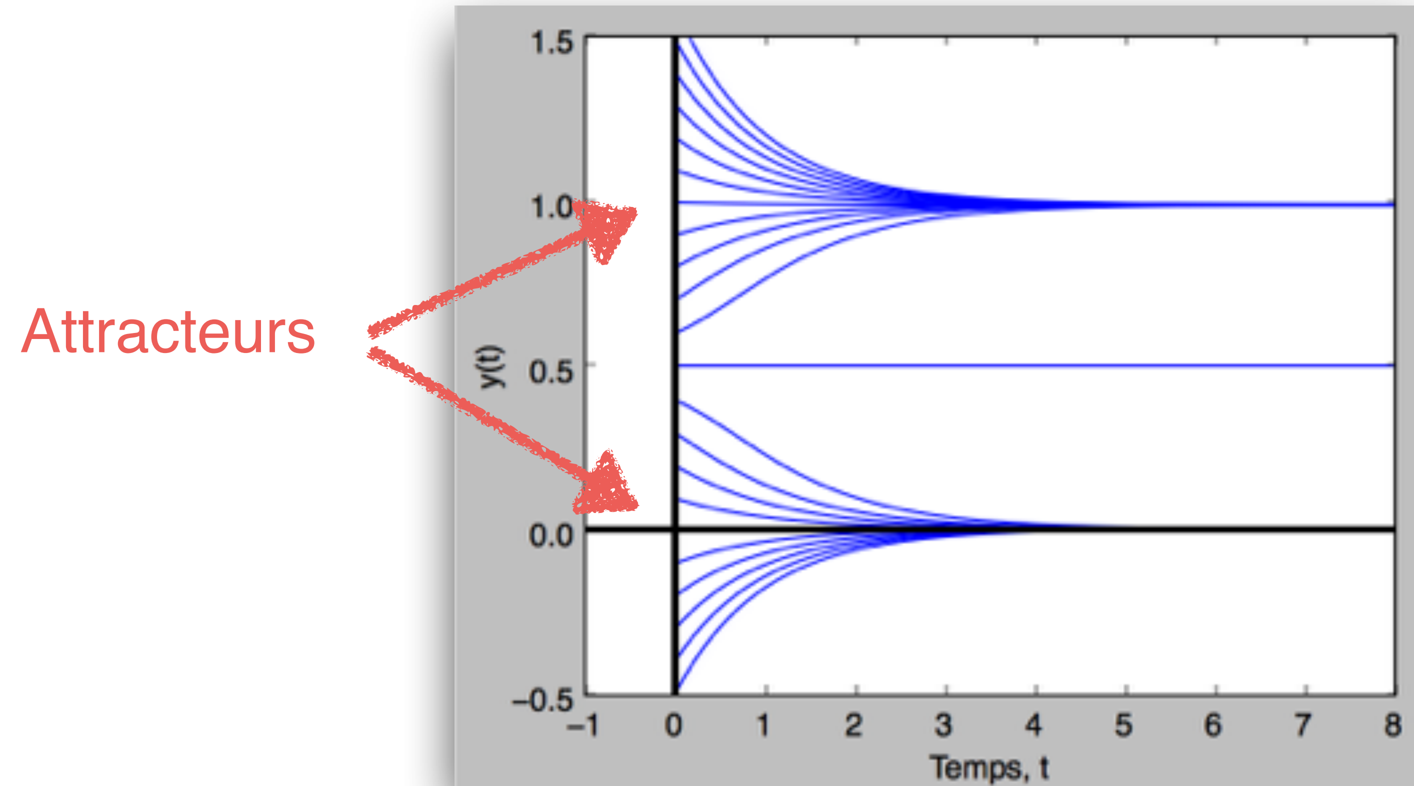
"Autapse"

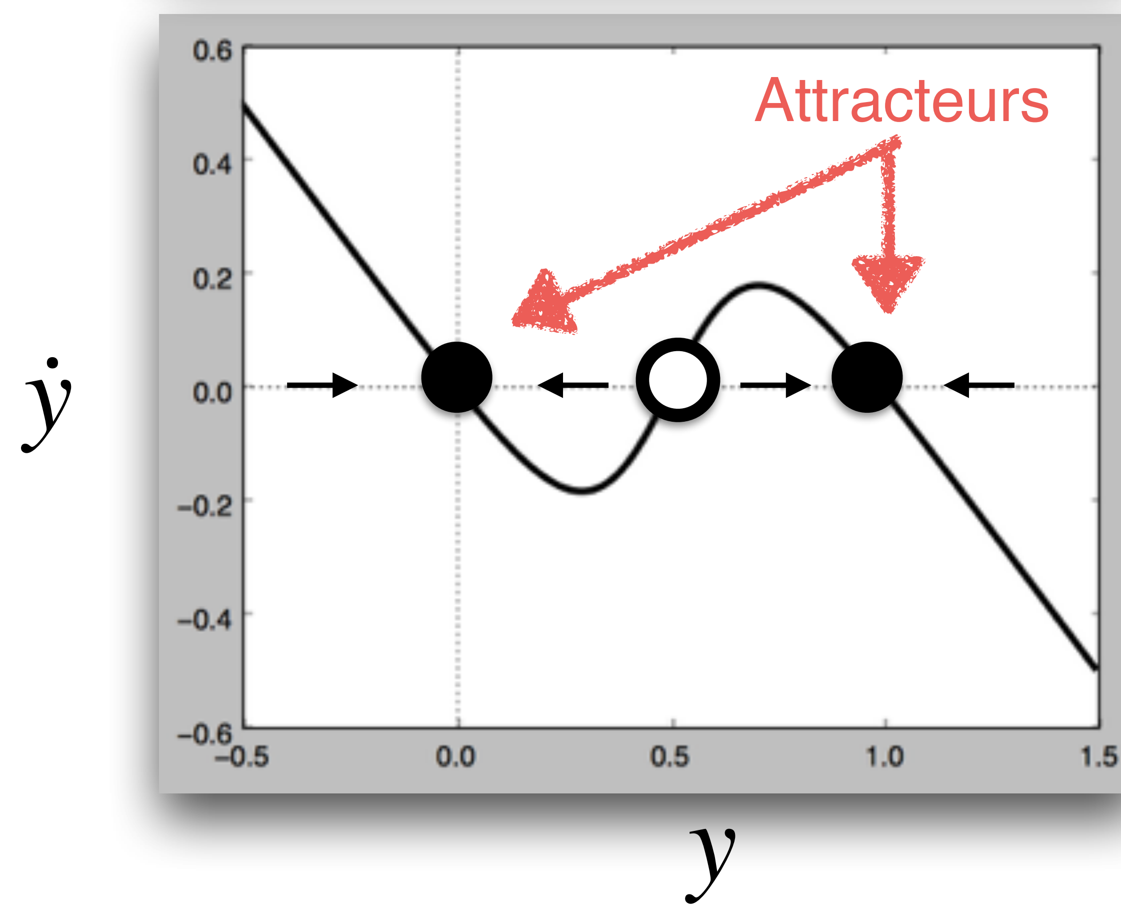


$$\dot{y} = -y + \frac{1}{1+e^{-wy-x}}$$

Deux états stables :

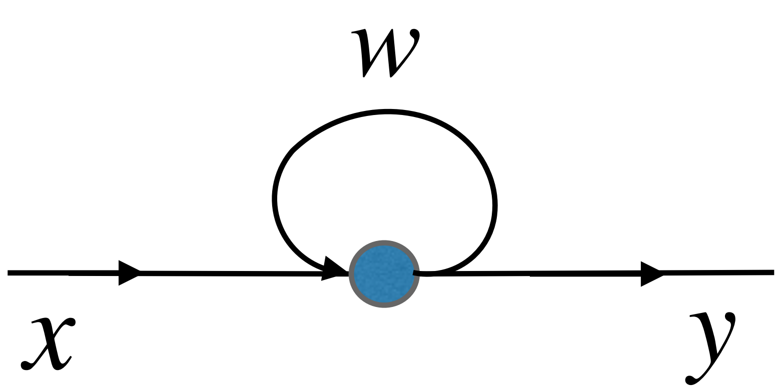
$\boxed{y=0}$ $\boxed{y=1}$





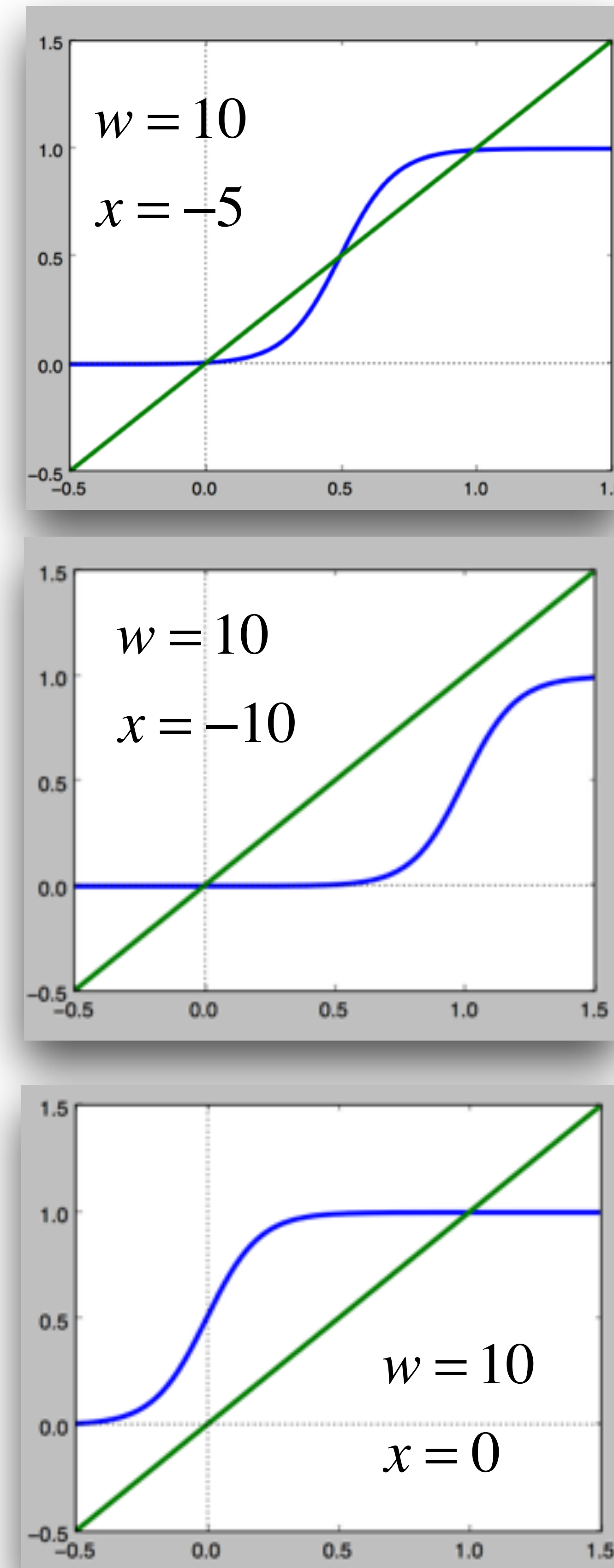
Un neurone avec une synapse récurrente

"Autapse"



$$\dot{y} = -y + \frac{1}{1+e^{-wy-x}}$$

valeurs de ***x*** et ***w*** détermine le nombres des points fixes du système



$f(y) = y$

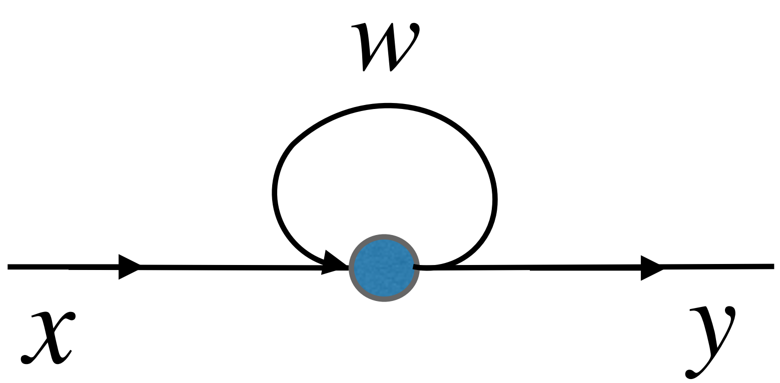
$g(y) = \frac{1}{1+e^{-wy-x}}$

?

?

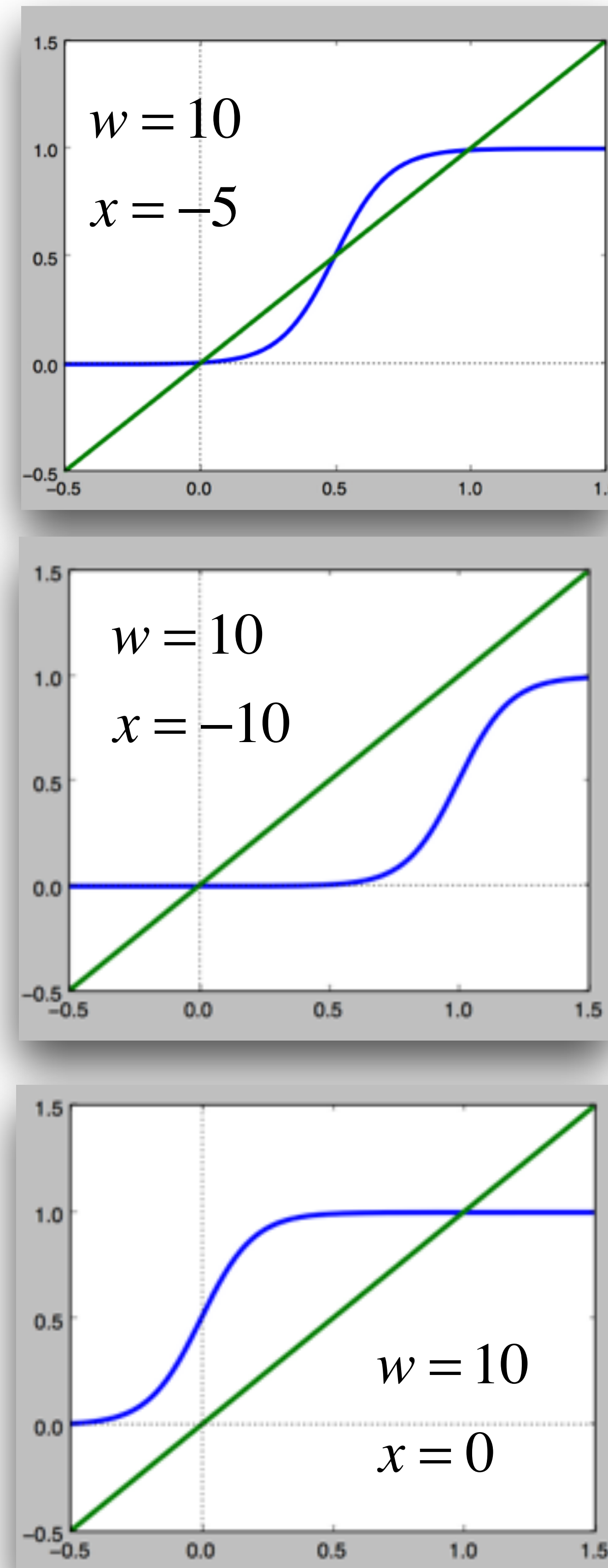
Un neurone avec une synapse récurrente

"Autapse"



$$\dot{y} = -y + \frac{1}{1+e^{-wy-x}}$$

- **Sortie du réseau correspond à l'état stable du système dynamique correspondant**



$f(y) = y$

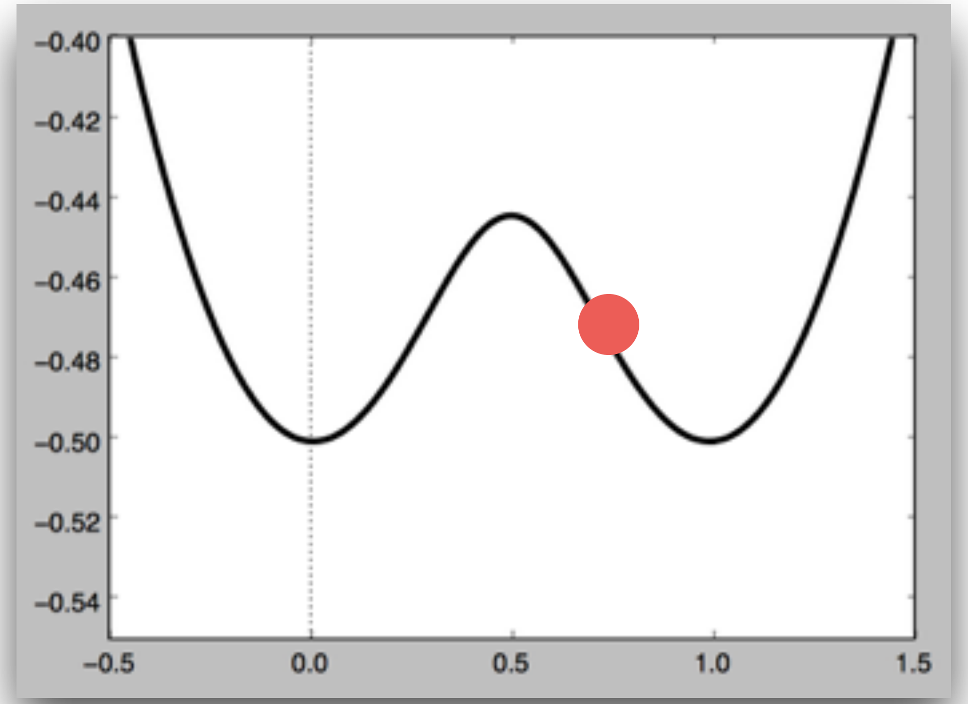
$g(y) = \frac{1}{1+e^{-wy-x}}$

?

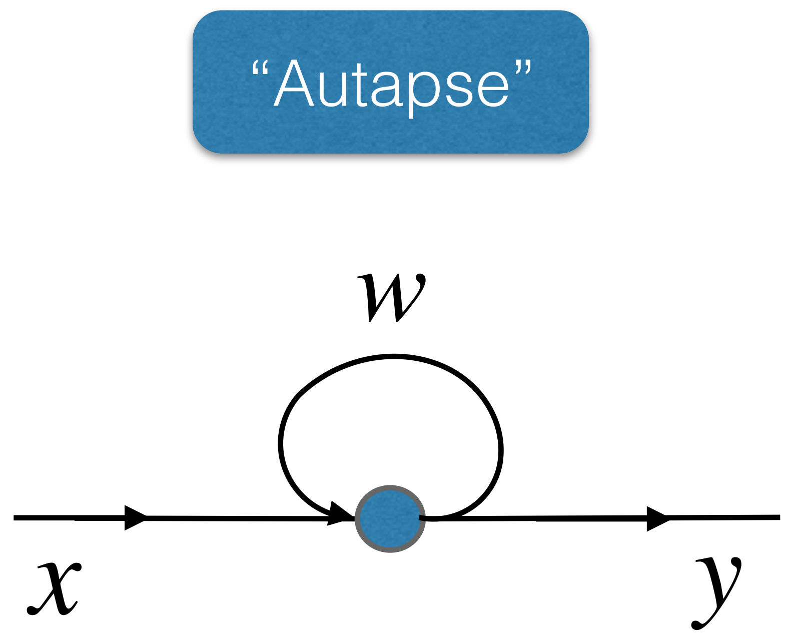
?

Fonction-énergie

$E(y)$



Fonction-énergie



$$\dot{y} = -y + \frac{1}{1 + e^{-wy - x}}$$

- Pour des systèmes dynamiques dans une dimension

$$\dot{y} = f(y)$$

la **fonction-énergie** $E(y)$
est la fonction telle que

$$\frac{dE}{dy} = -\dot{y}$$

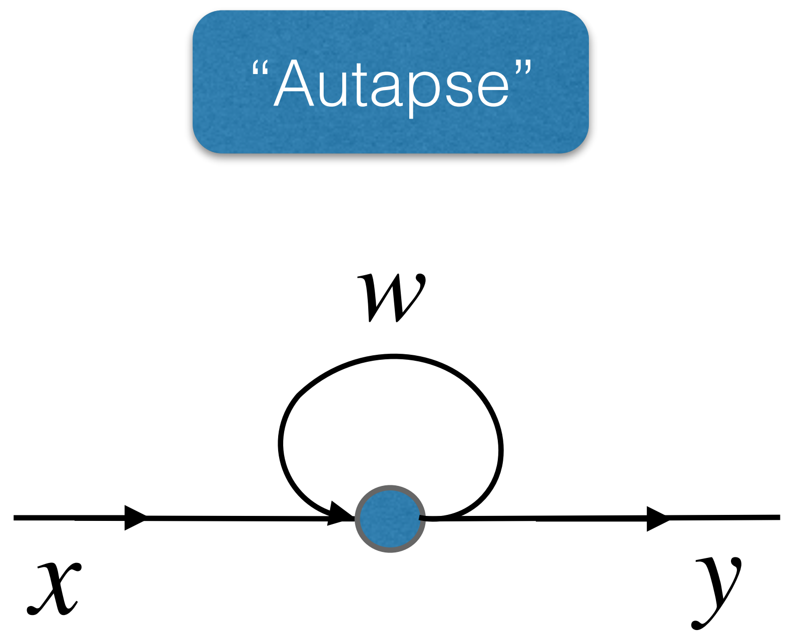
- Fonction-énergie décroit toujours avec le temps :

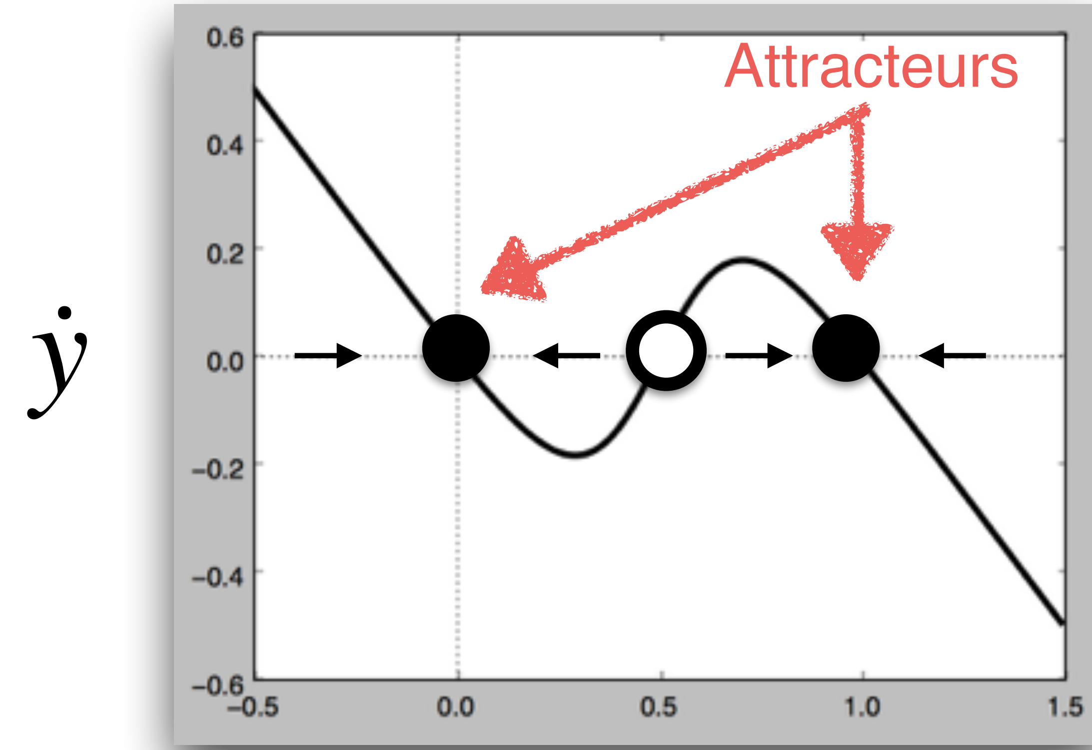
$$\frac{dE}{dt} = \frac{dE}{dy}\frac{dy}{dt} = -\dot{y}\dot{y} = -\dot{y}^2 \leq 0$$

- Elle reste constante en tout point fixe du système dynamique

$$\dot{y} = 0 \quad \Rightarrow \quad E = const$$

Fonction-énergie

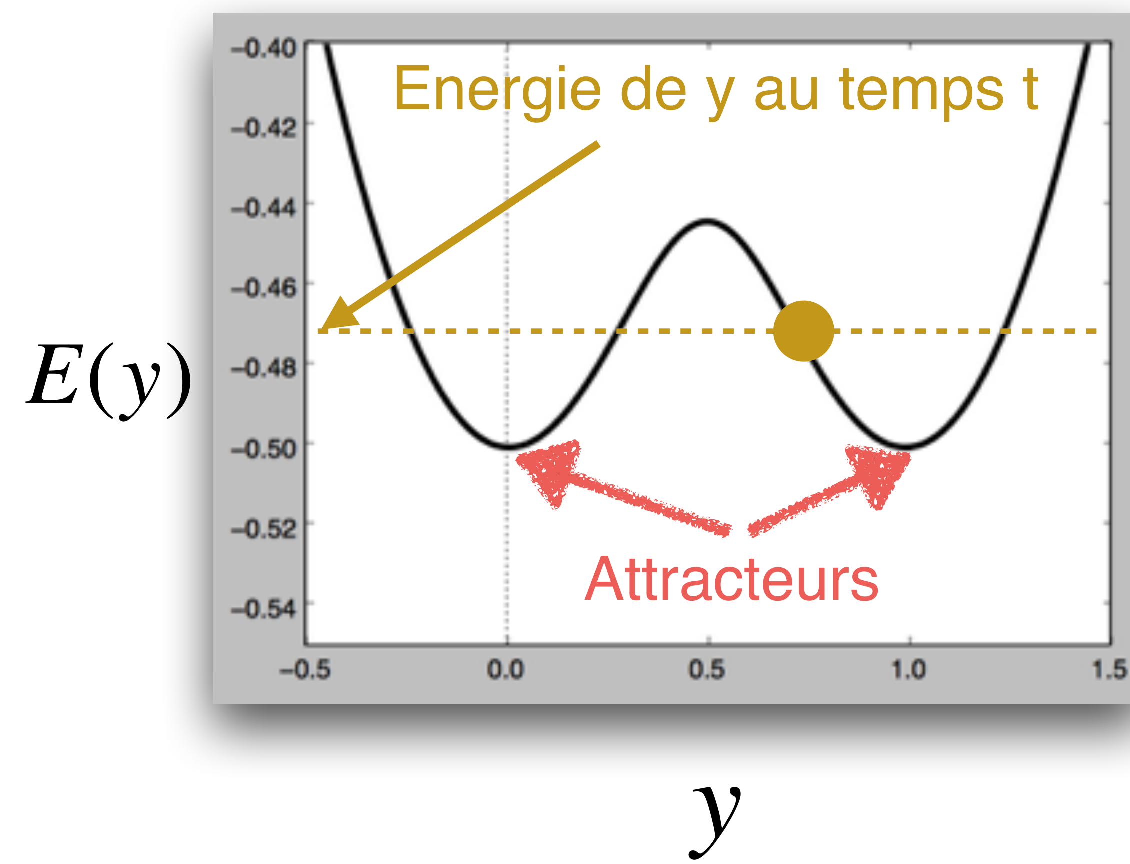




- Fonction-énergie est utile pour **visualiser** le comportement du système

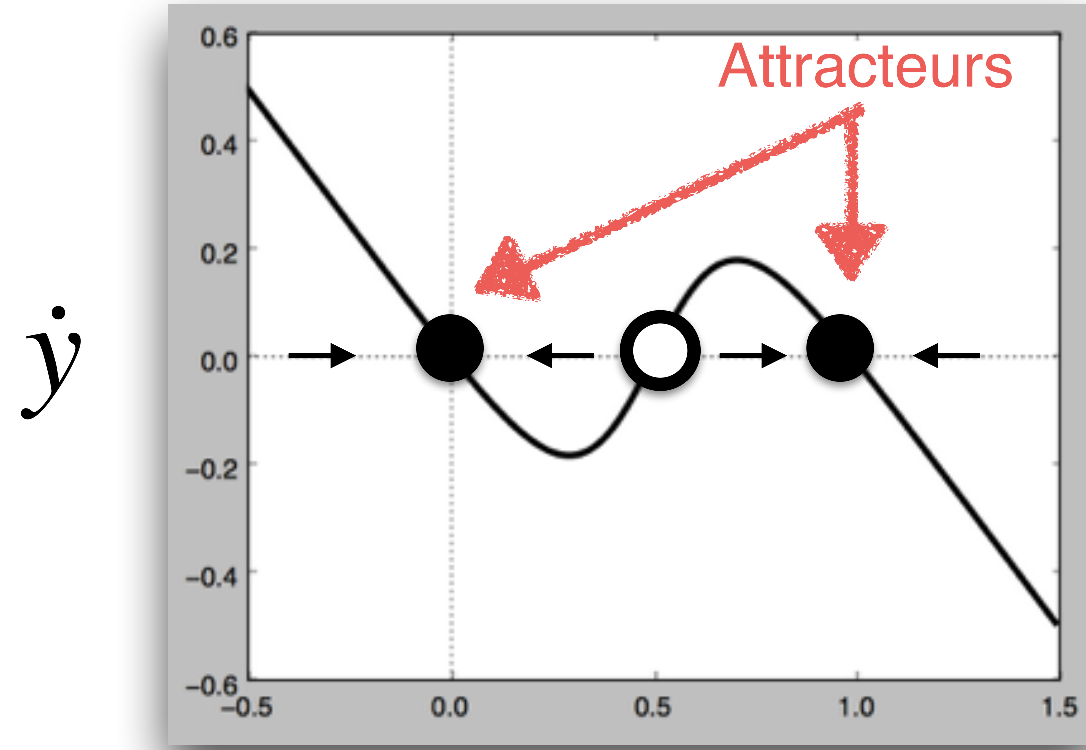
- Pour l'autapse (voir TD) :

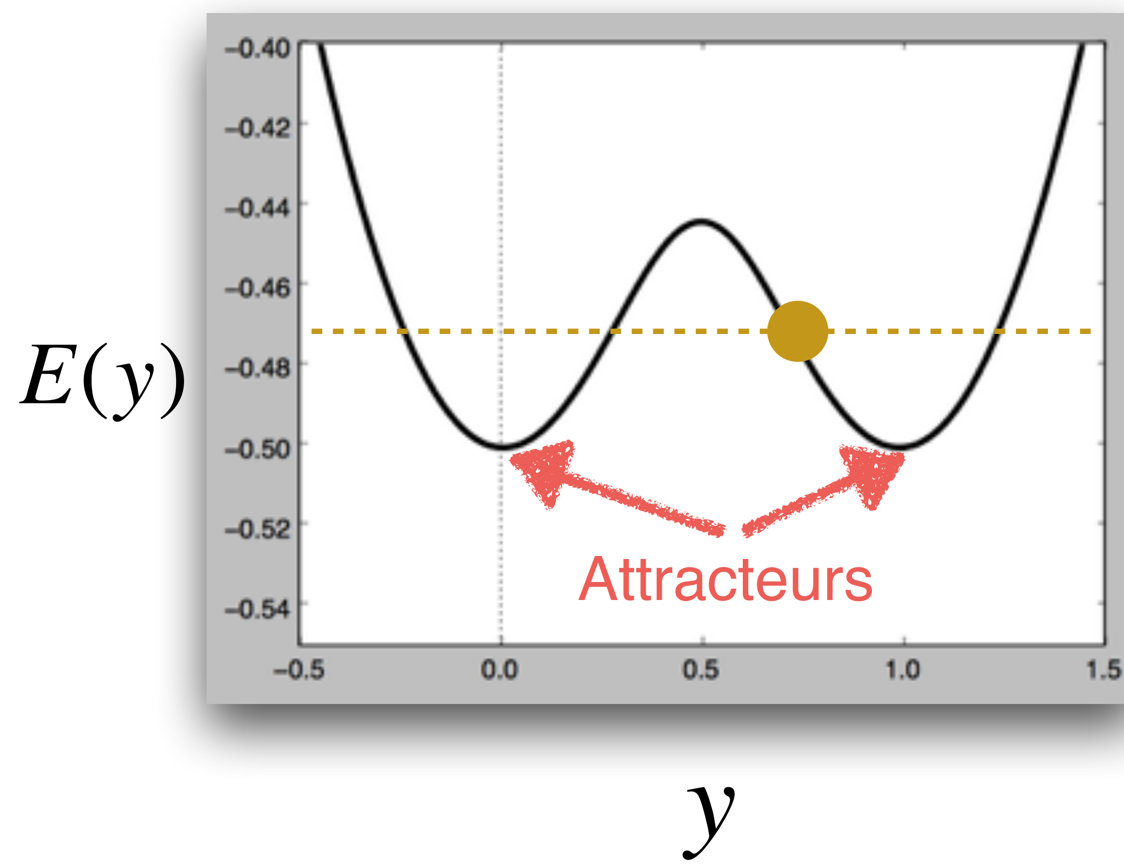
$$E(y) = \frac{y^2}{2} - y - \frac{\ln(1+e^{-wy-x})}{w}$$



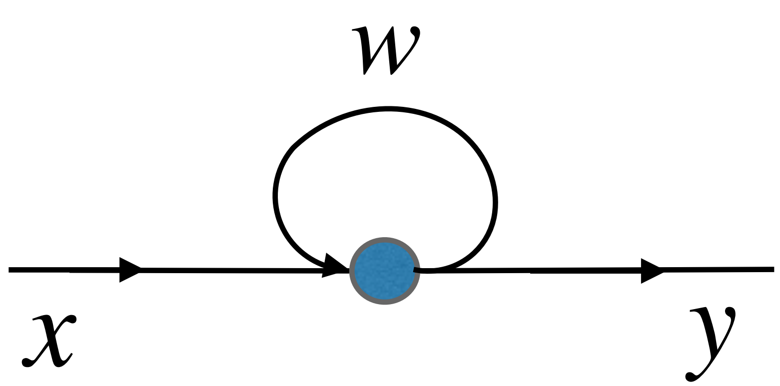
Fonction-énergie

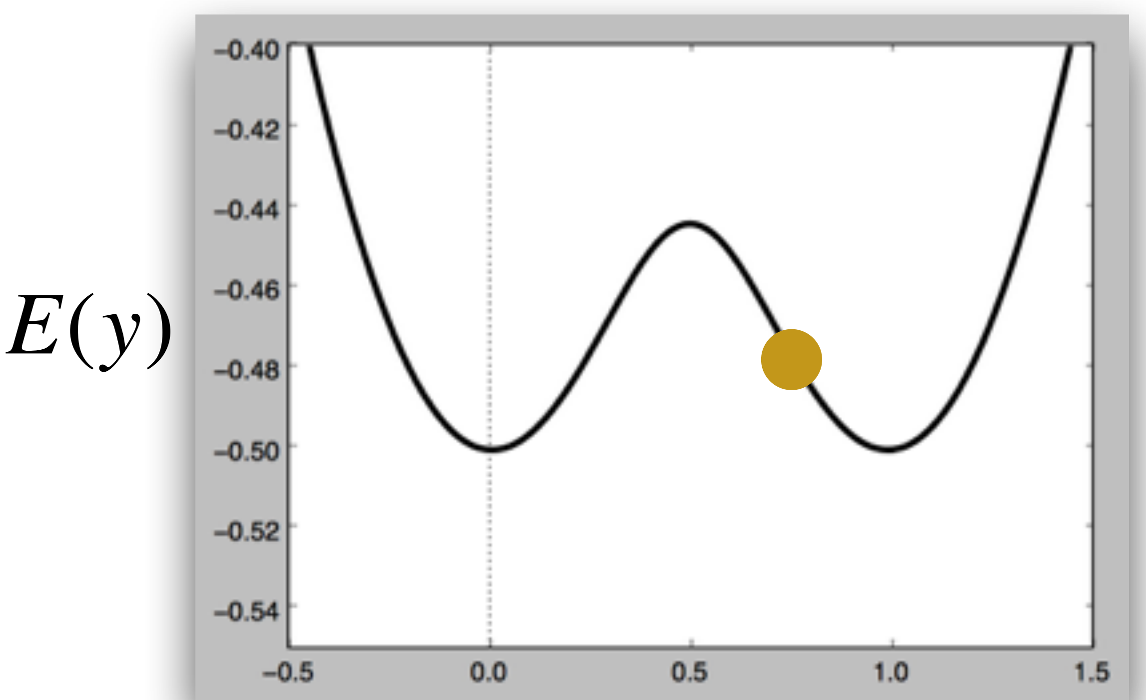
- Où est la séparatrice ?
- .. bassin d'attraction ?





Un neurone avec une synapse récurrente

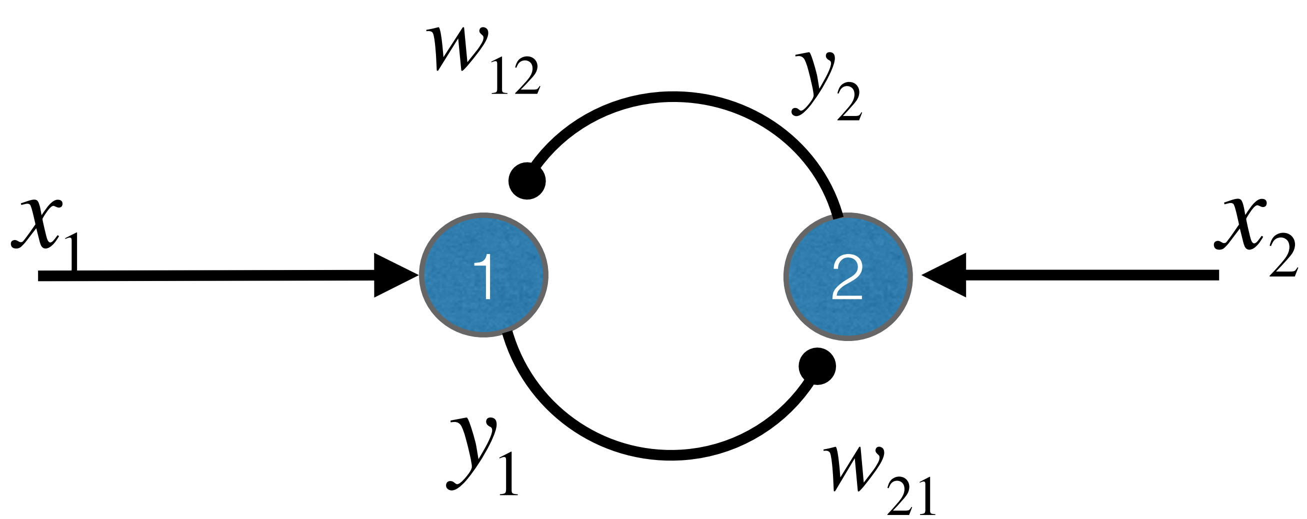




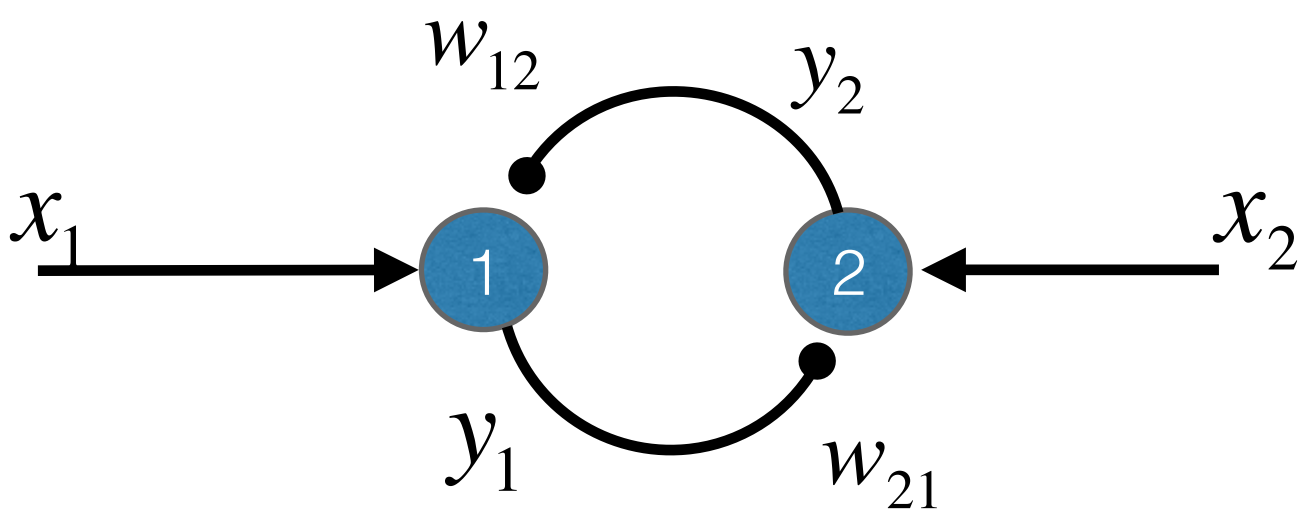
Que fait ce réseau ?

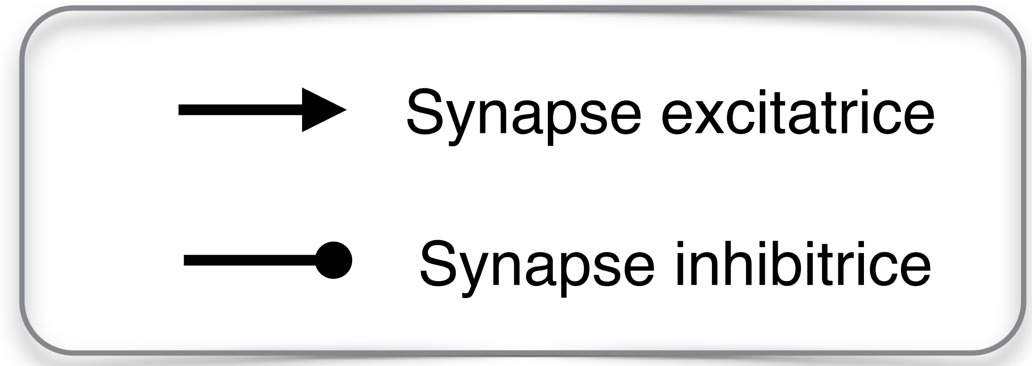
- **Sortie du réseau correspond à l'état stable du système dynamique correspondant**
- **On peut représenter la solution par un point qui descend la surface de la fonction-énergie du système**
- **Minimums locaux de la fonction-énergie correspondent aux points stables du système (attracteurs)**
- **Maximums locaux correspondent aux séparatrices**

Réseau récurrent : Deux neurones



Deux neurones avec synapses récurrentes





$$\dot{y}_1 = -y_1 + g(w_{12}y_2 + x_1)$$
$$\dot{y}_2 = -y_2 + g(w_{21}y_1 + x_2)$$

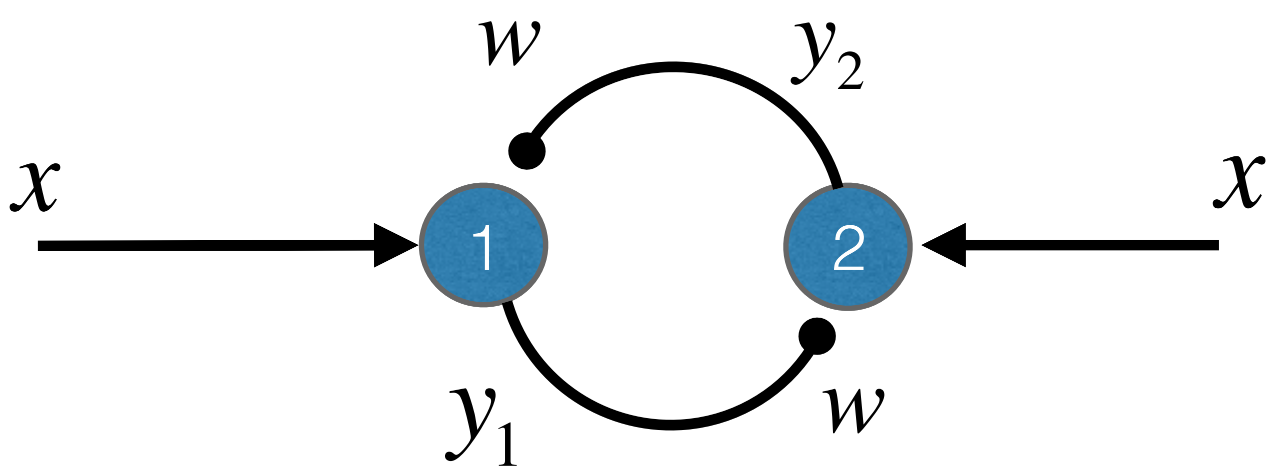
Pour simplifier :

$w_{12} = w_{21} = w < 0$ Synapses inhibitrice

$x_1 = x_2 = x > 0$ Entrées constantes excitatrices

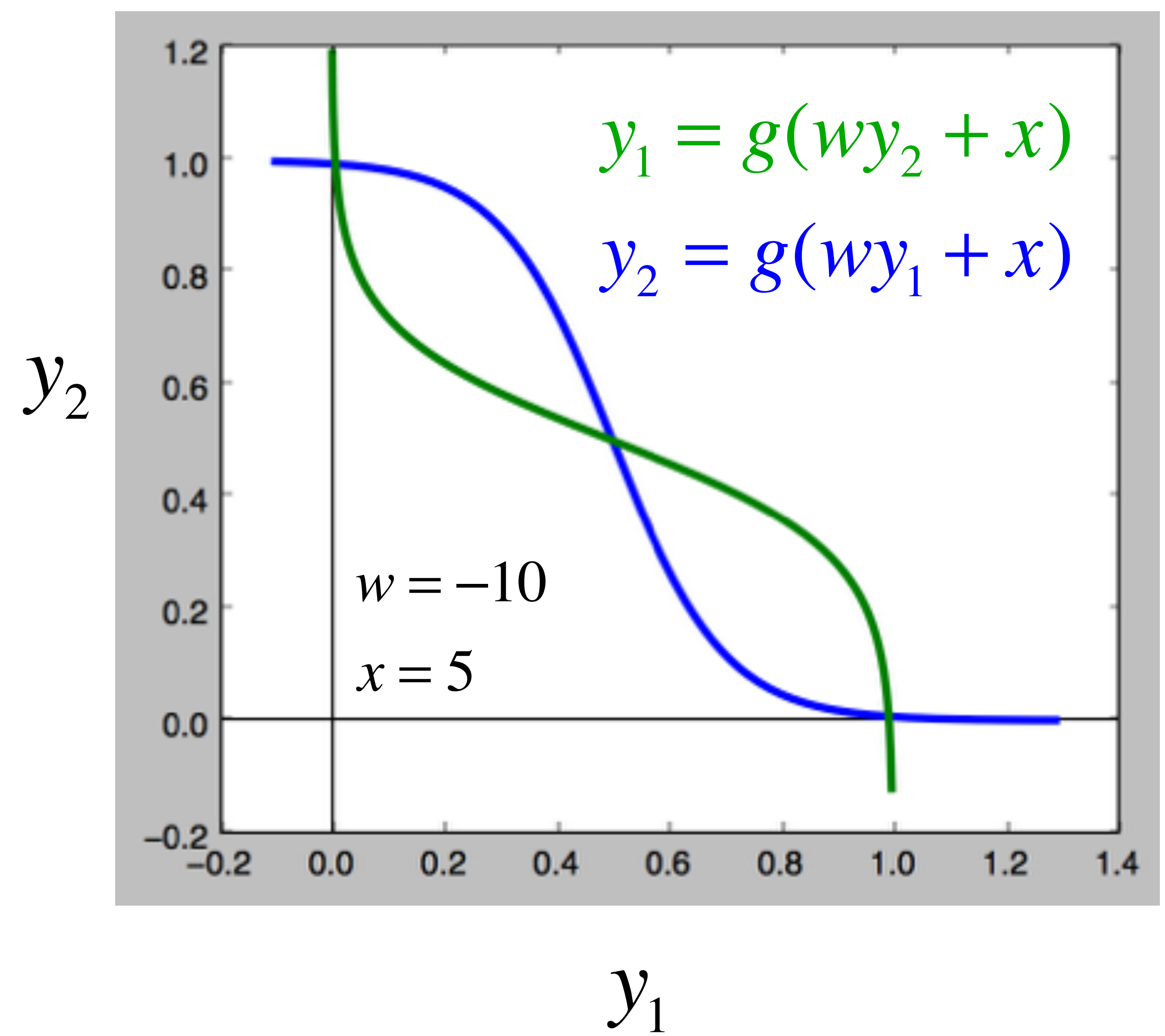
$$\boxed{\begin{aligned}\dot{y}_1 &= -y_1 + g(wy_2 + x)\\ \dot{y}_2 &= -y_2 + g(wy_1 + x)\end{aligned}}$$

Deux neurones avec synapses récurrentes

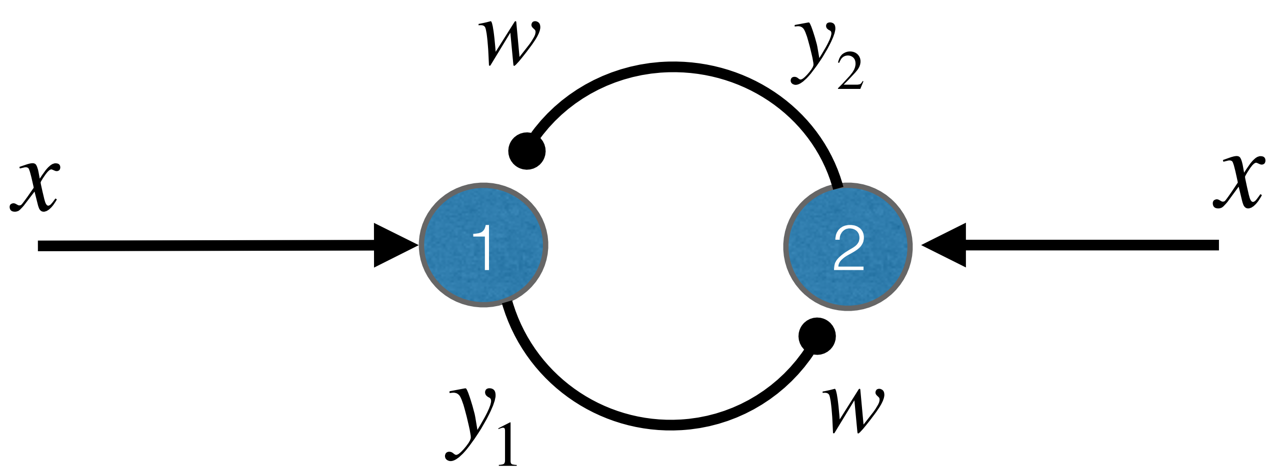


$$\dot{y}_1 = -y_1 + g(wy_2 + x)$$
$$\dot{y}_2 = -y_2 + g(wy_1 + x)$$

Plan de phase, nullclines :



Deux neurones avec synapses récurrentes

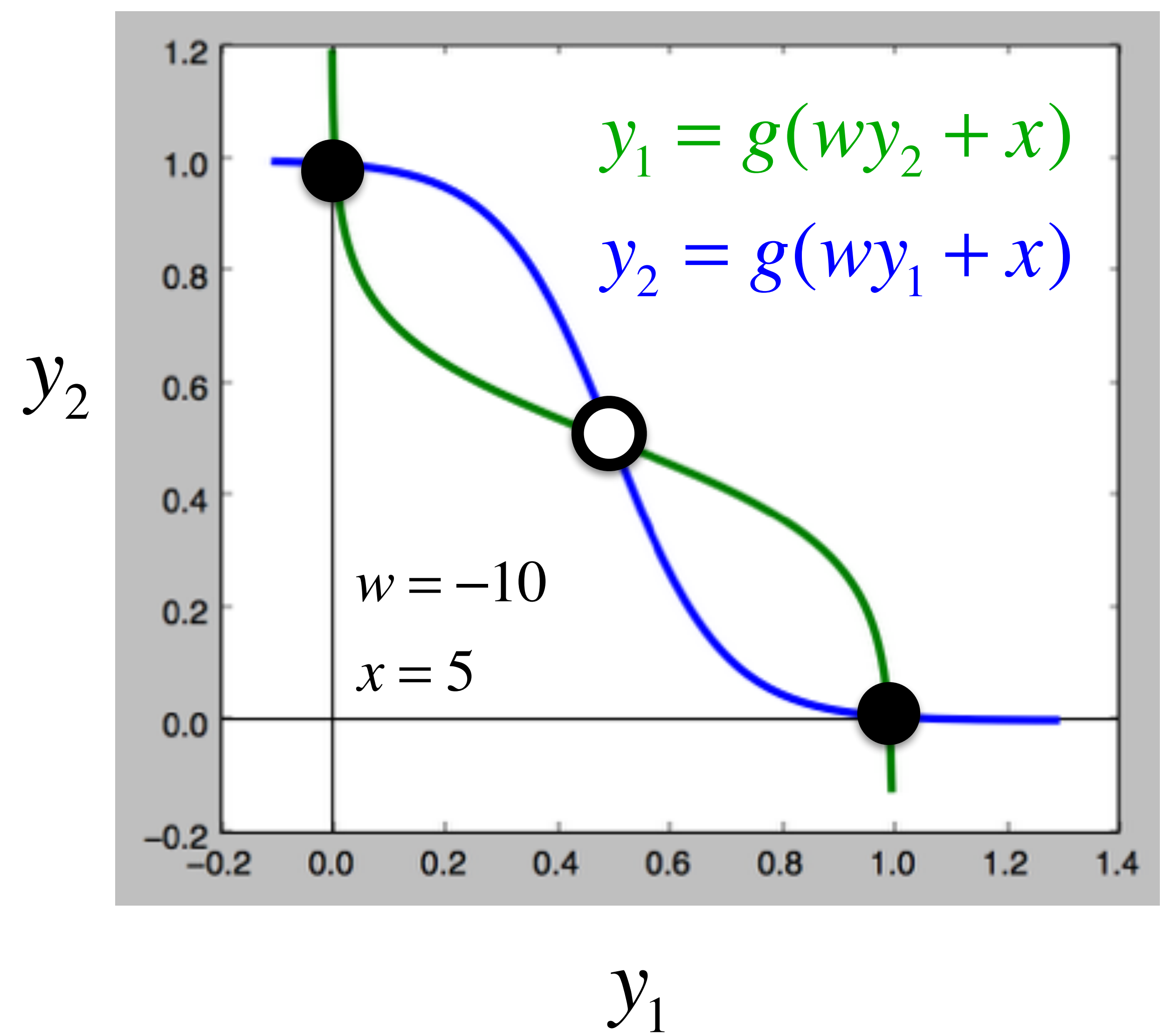


$$\dot{y}_1 = -y_1 + g(wy_2 + x)$$
$$\dot{y}_2 = -y_2 + g(wy_1 + x)$$

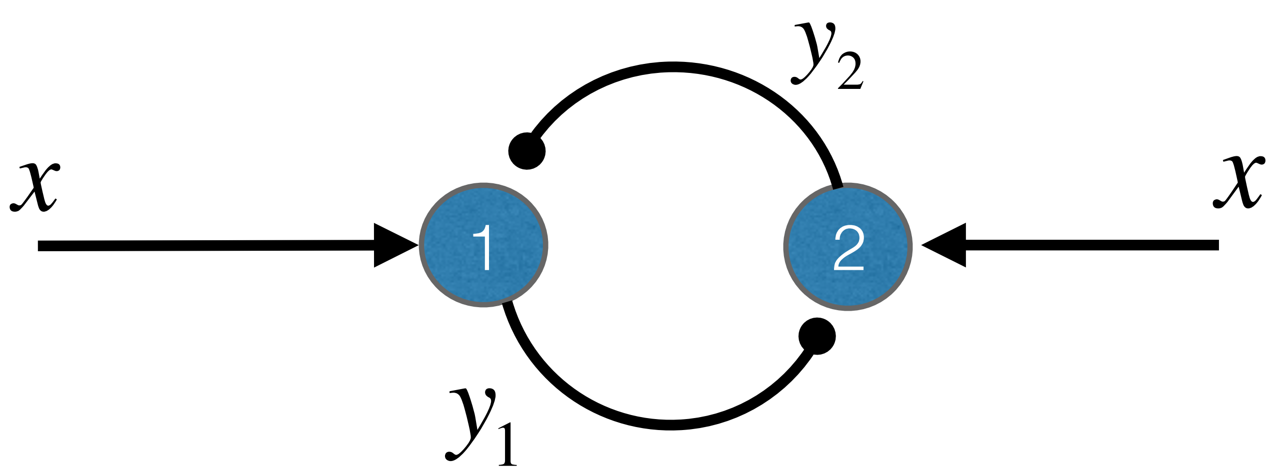
Deux états stables :

$$\boxed{\begin{array}{l} y_1 = 1 \\ y_2 = 0 \end{array}} \qquad \boxed{\begin{array}{l} y_1 = 0 \\ y_2 = 1 \end{array}}$$

Plan de phase, nullclines :



Deux neurones avec synapses récurrents

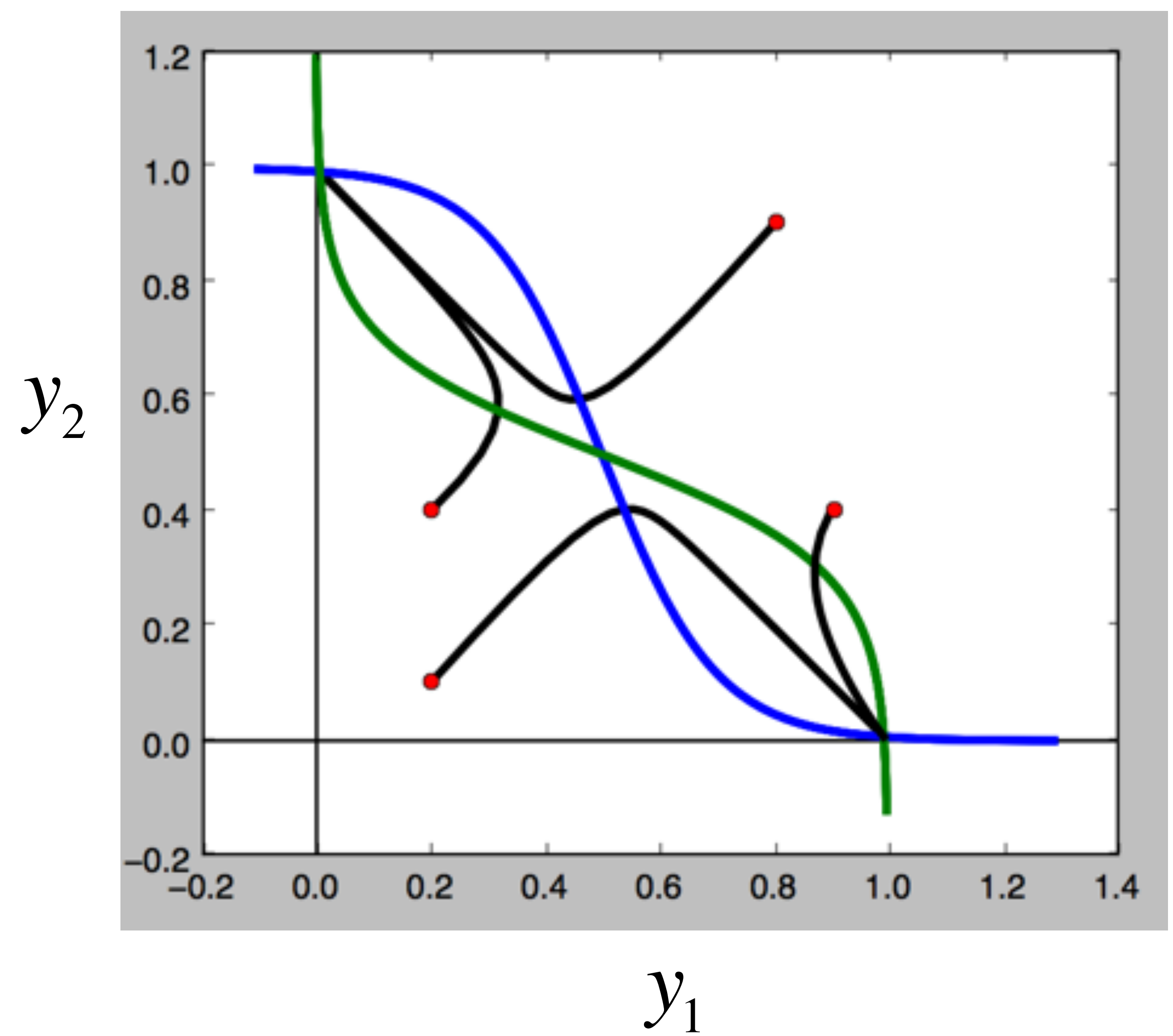


$$\dot{y}_1 = -y_1 + g(wy_2 + x)$$
$$\dot{y}_2 = -y_2 + g(wy_1 + x)$$

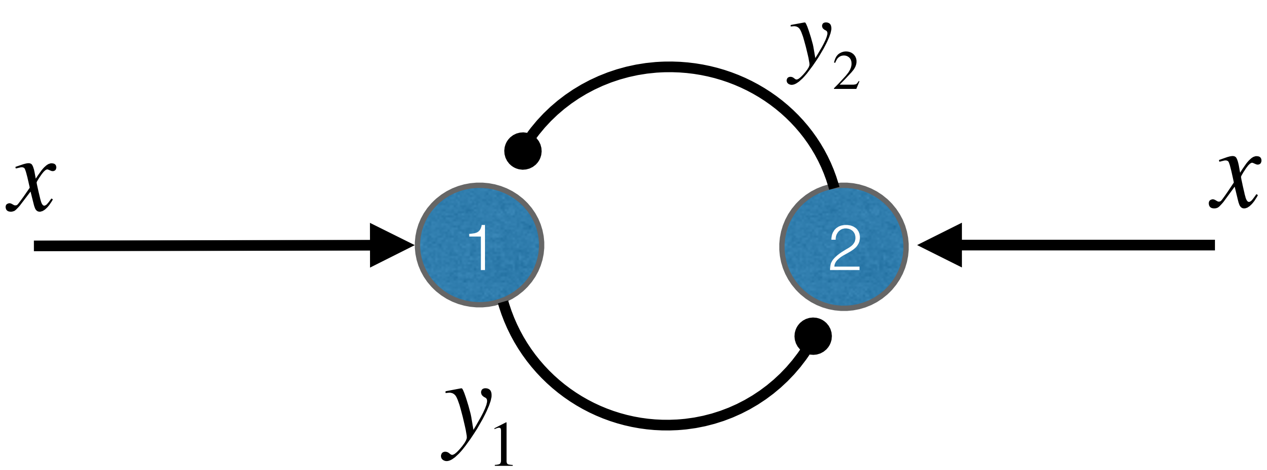
Deux états stables :

$\boxed{\begin{matrix} y_1 = 1 \\ y_2 = 0 \end{matrix}}$ $\boxed{\begin{matrix} y_1 = 0 \\ y_2 = 1 \end{matrix}}$

Plan de phase, solutions :



Deux neurones avec synapses récurrents

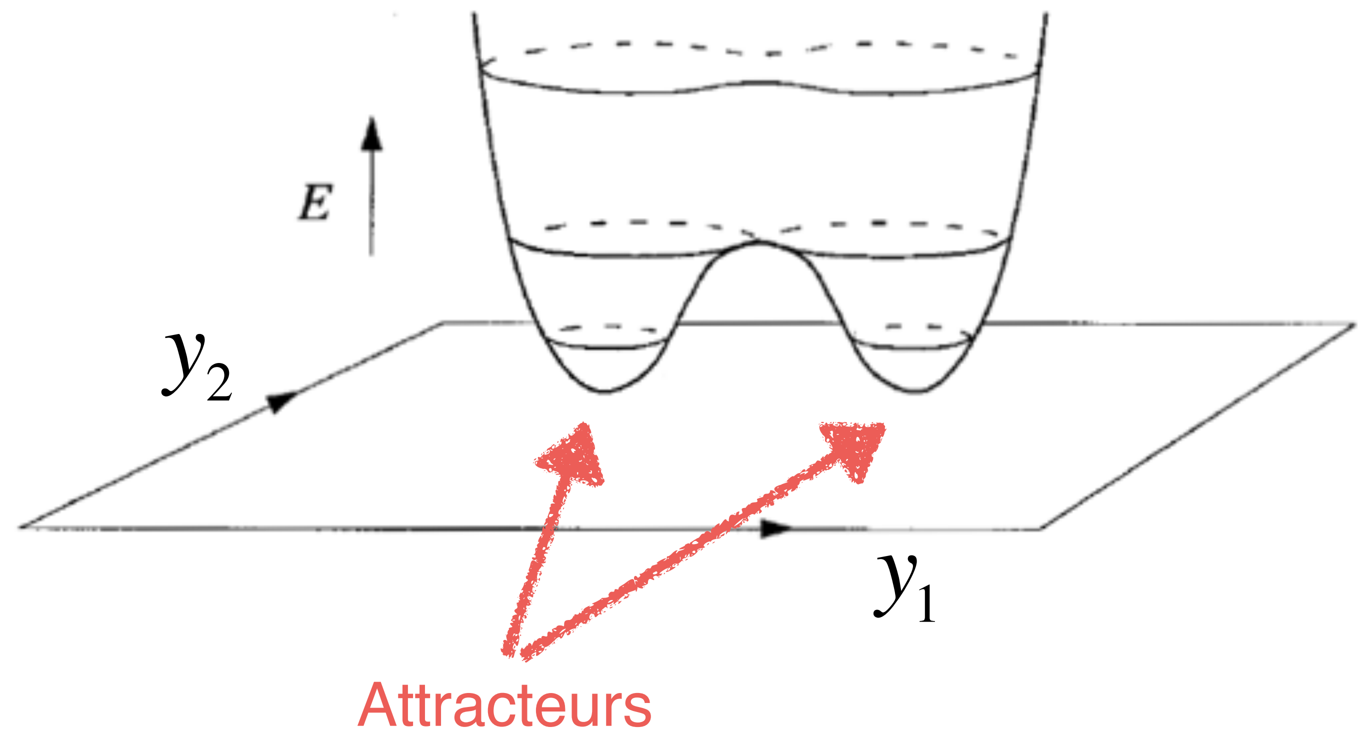


$$\dot{y}_1 = -y_1 + g(wy_2 + x)$$
$$\dot{y}_2 = -y_2 + g(wy_1 + x)$$

Deux états stables :

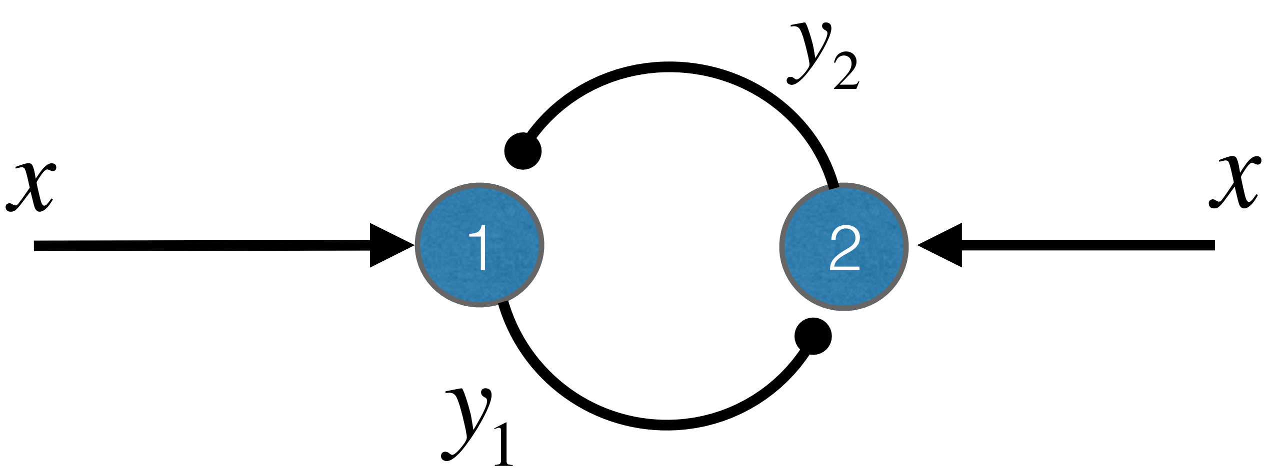
$\boxed{y_1 = 1,\ y_2 = 0}$ $\boxed{y_1 = 0,\ y_2 = 1}$

Fonction-énergie :



- Les états stables (attracteurs) correspondent au minima de la fonction énergie (si elle existe)

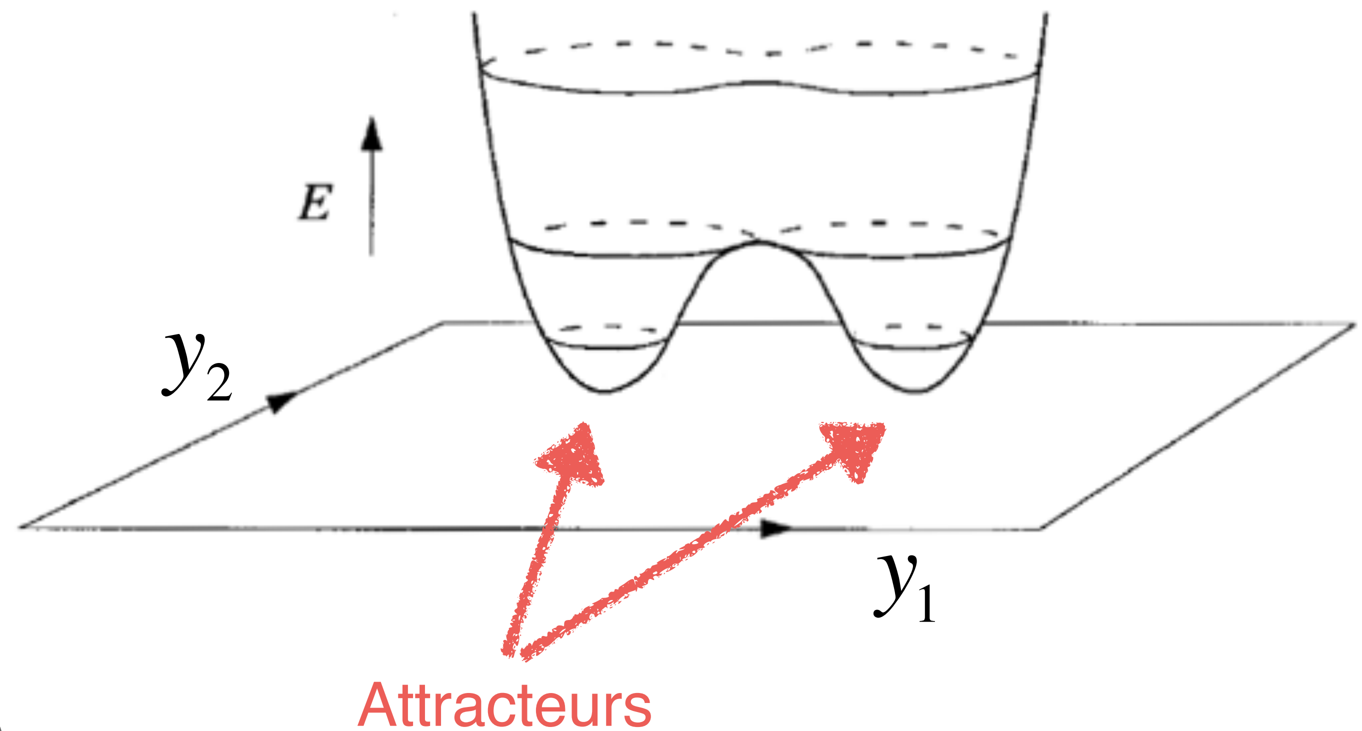
Deux neurones avec synapses récurrents



Deux états stables :

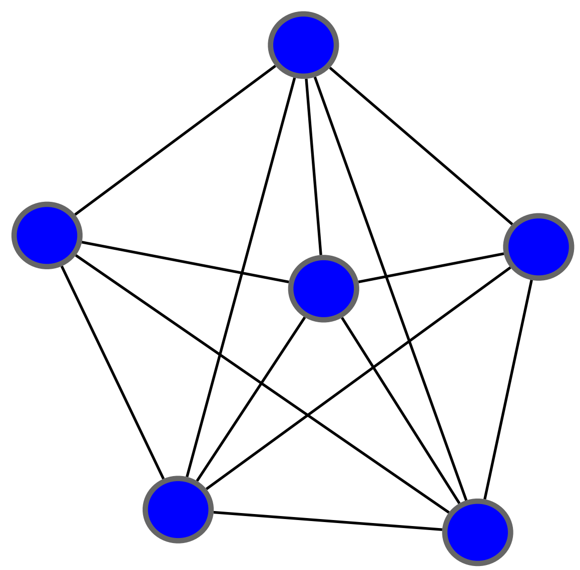
$\boxed{\begin{matrix} y_1 = 1 \\ y_2 = 0 \end{matrix}}$ $\boxed{\begin{matrix} y_1 = 0 \\ y_2 = 1 \end{matrix}}$

Fonction-énergie :

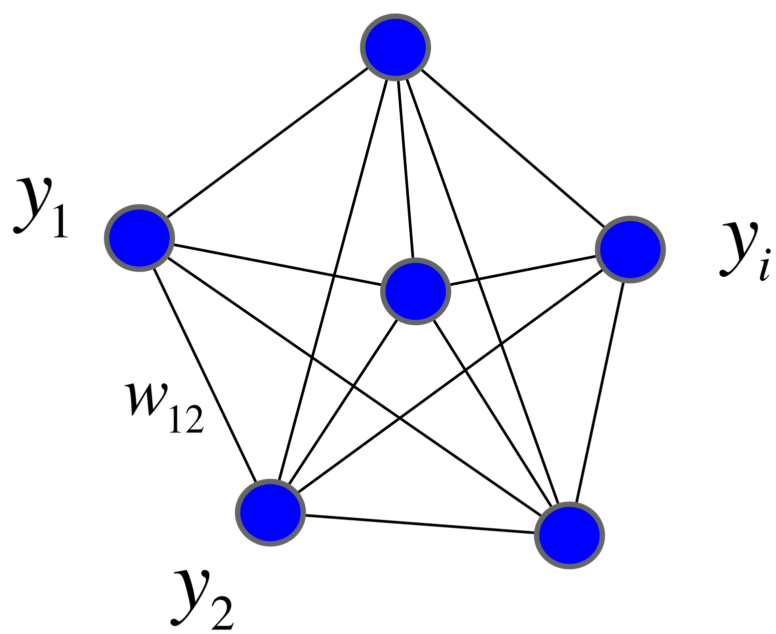


- **On peut considérer le réseau récurrent avec deux neurones comme un modèle de mémoire qui est capable de mémoriser deux motifs binaires**
- **La capacité de la mémoire correspond au nombre d'états stables du système dynamique**

Réseaux récurrents : N neurones



Réseau de Hopfield



***Hopfield, J. J. (1982)**. Neural networks and physical systems with emergent collective computational abilities. PNAS, 79(April), 2554–2558.*

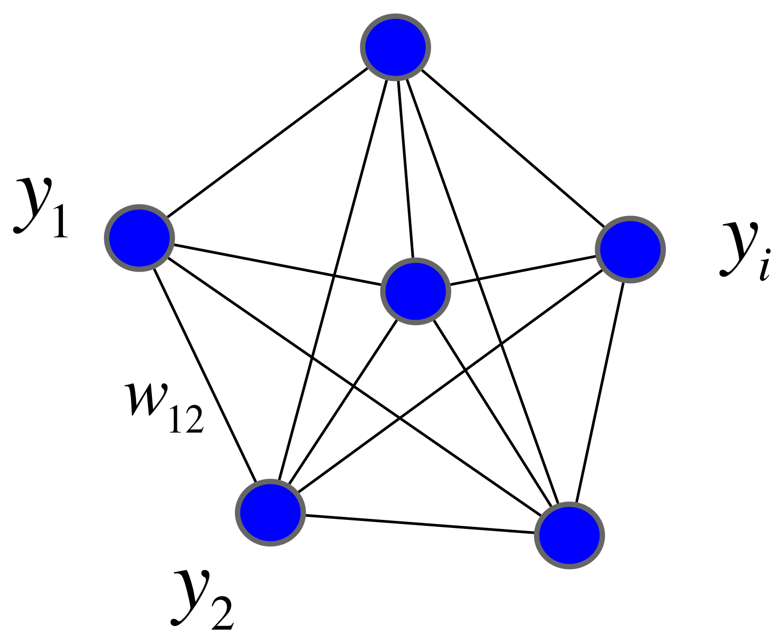
- N neurones
- Connections symétriques, bidirectionnelles

$$w_{ij} = w_{ji}$$

- Activité d'un neurone

$$y_i = g(a_i) = g\left(\sum_{i=1}^{N} w_{ij} y_j\right)$$

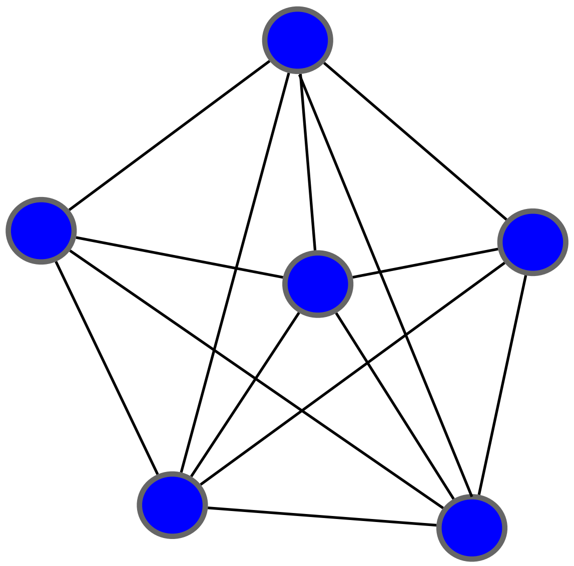
Réseau de Hopfield



***Hopfield, J. J. (1982)**. Neural networks and physical systems with emergent collective computational abilities. PNAS, 79(April), 2554–2558.*

- Le réseau de Hopfield est équivalent à un système dynamique dans N dimensions
- L'état du système dynamique au temps t est le vecteur des activités $\vec{y}(t) = (y_1, \ldots y_N)$
- Le résultat de computation est un état stable du système $\vec{y}^*$
  Il est déterminé par les conditions initiales et par les poids

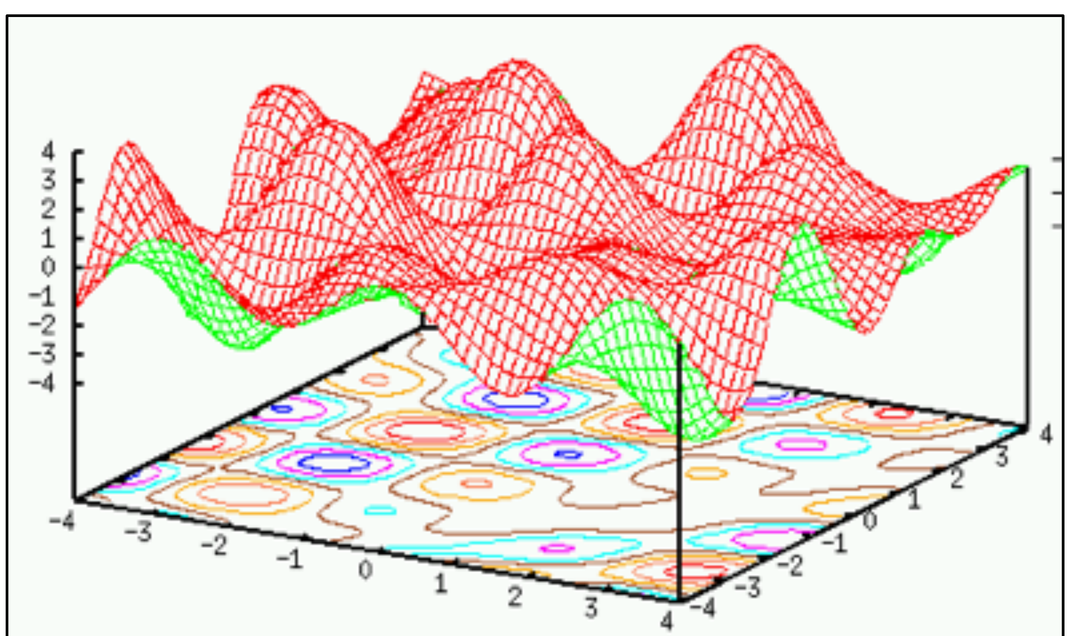
Réseau de Hopfield

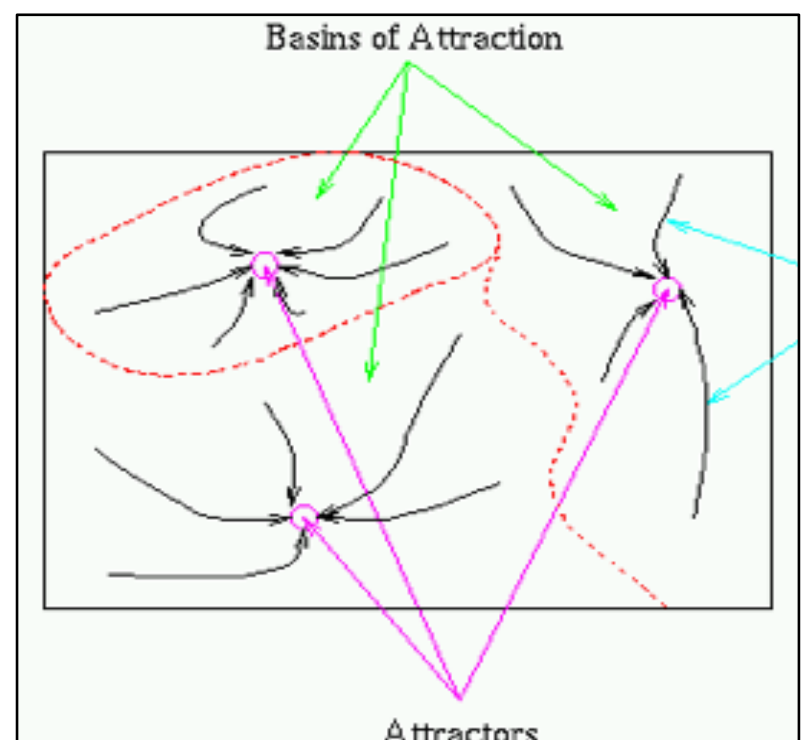


- J. Hopfield a démontré que si les poids sont symétriques, il existe une fonction-énergie associée à ce réseau

- Les attracteurs multidimensionnels peuvent être considérés comme des motifs mémorisés par le réseau

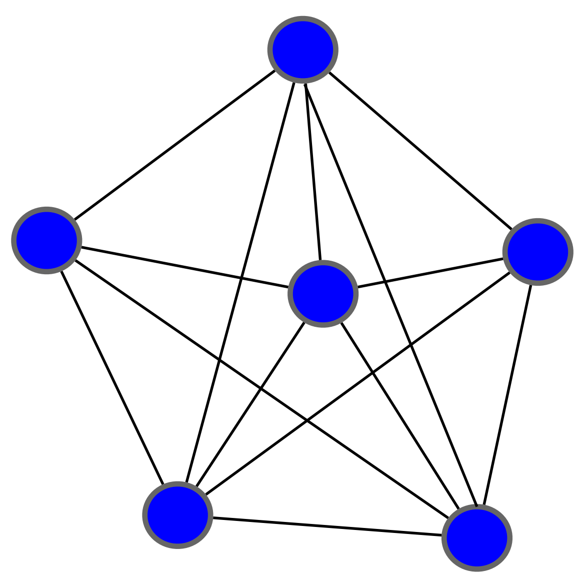
Fonction d'énergie





Espace de phases

Réseau de Hopfield

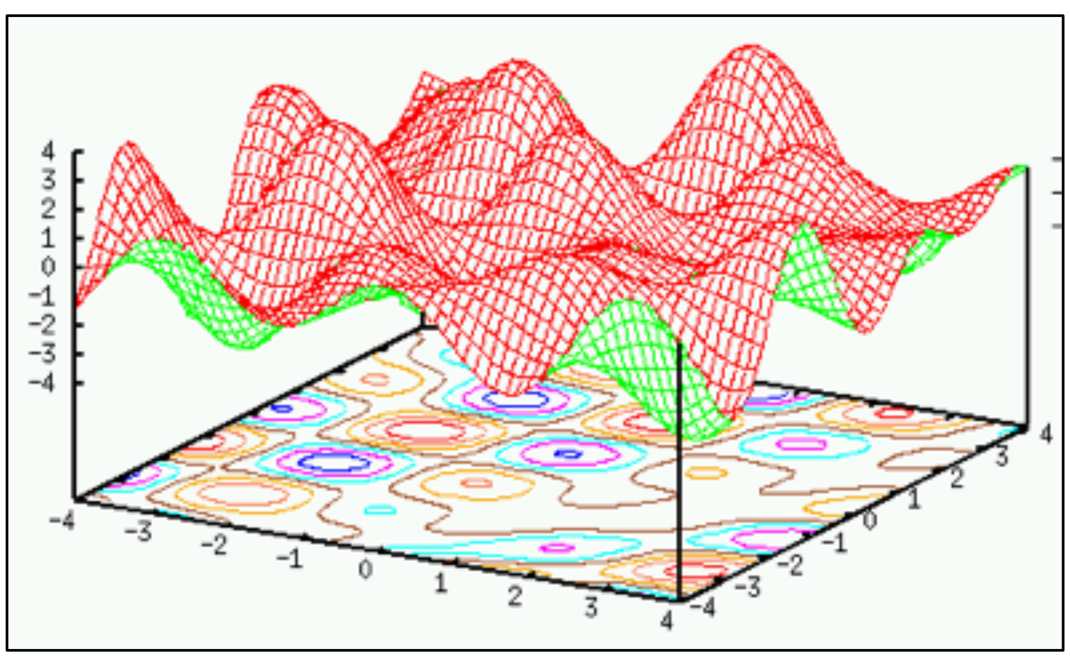


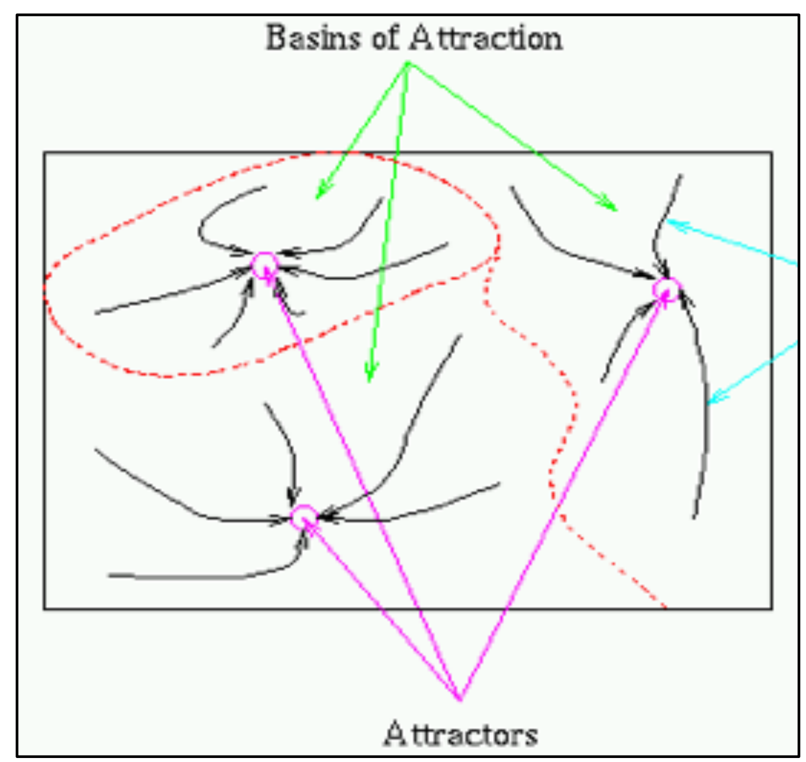
- Afin d'apprendre les motifs, il suffit de changer les poids synaptiques selon la règle d'apprentissage hebbienne :

On présent les motifs $\vec{y}^{(p)}$

On met à jour les poids $\Delta w_{ij} = \eta y_i^{(p)} y_j^{(p)}$

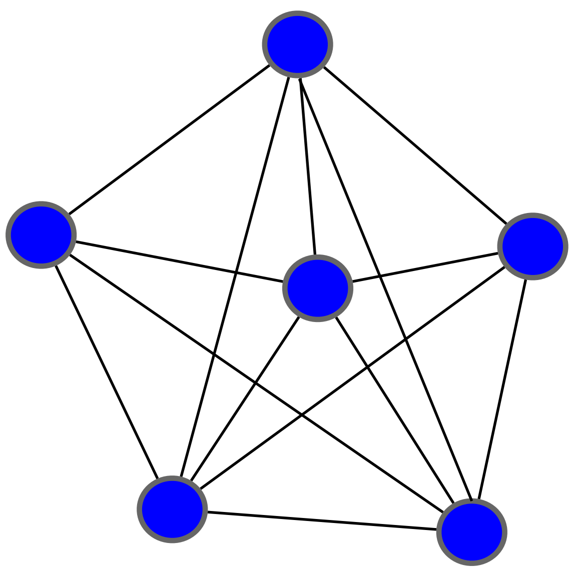
Fonction d'énergie



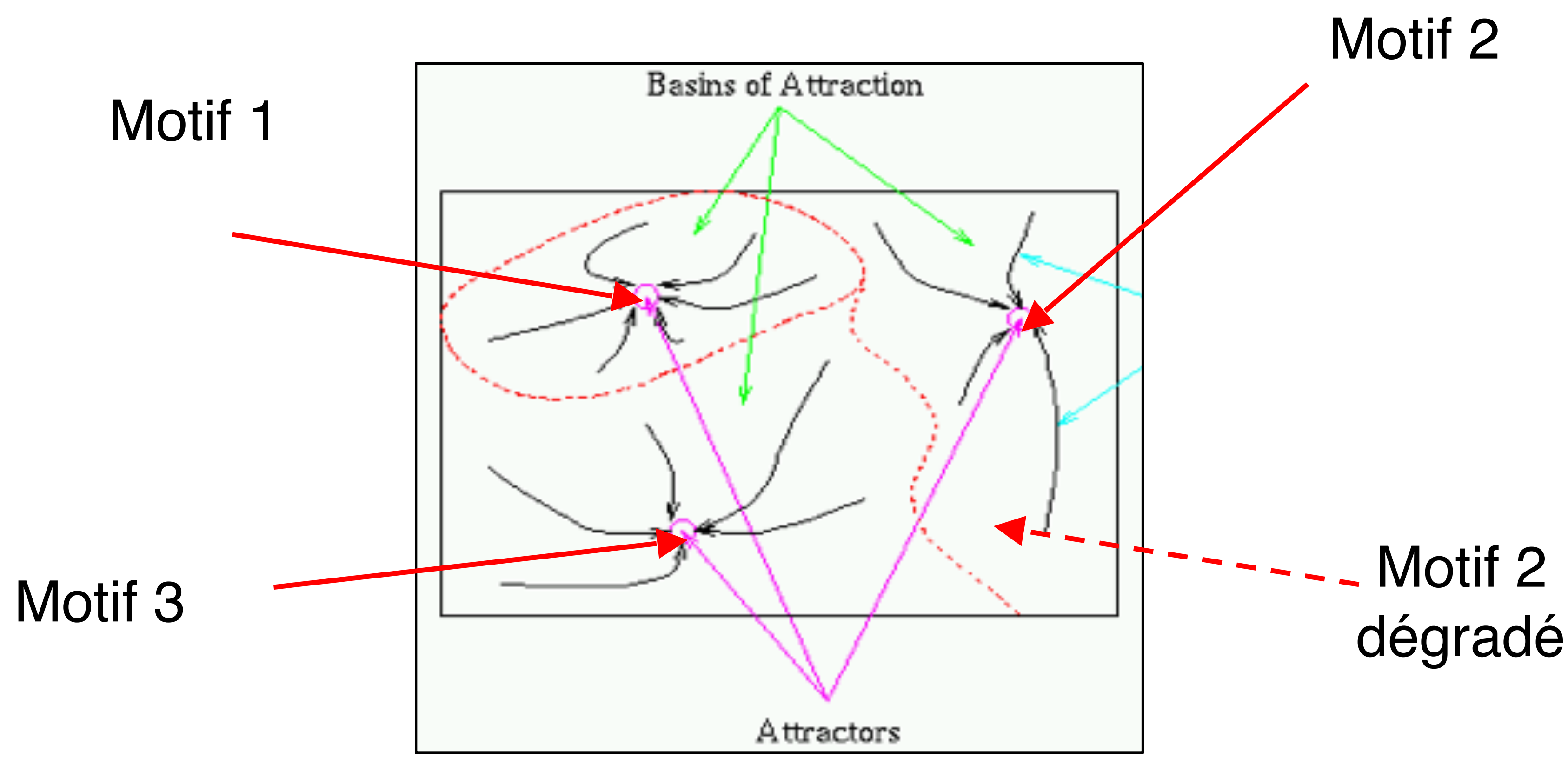


Espace de phases

Réseau de Hopfield et la mémoire associative

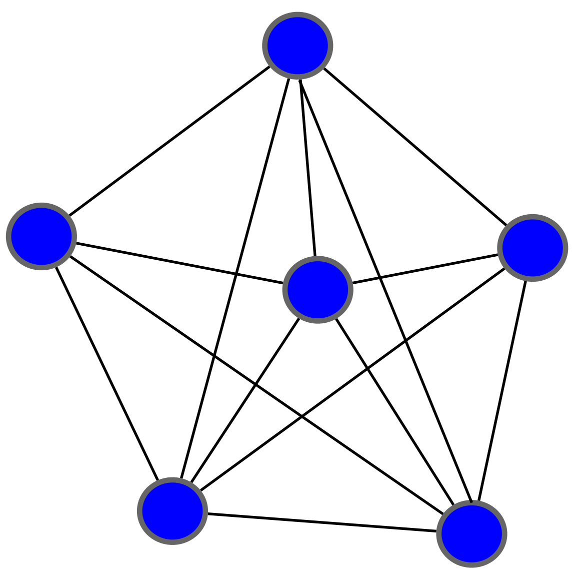


Les motifs mémorisés correspondent aux attracteurs de la dynamique



Les bassins d'attraction représentent la capacité de corriger les motifs erronés

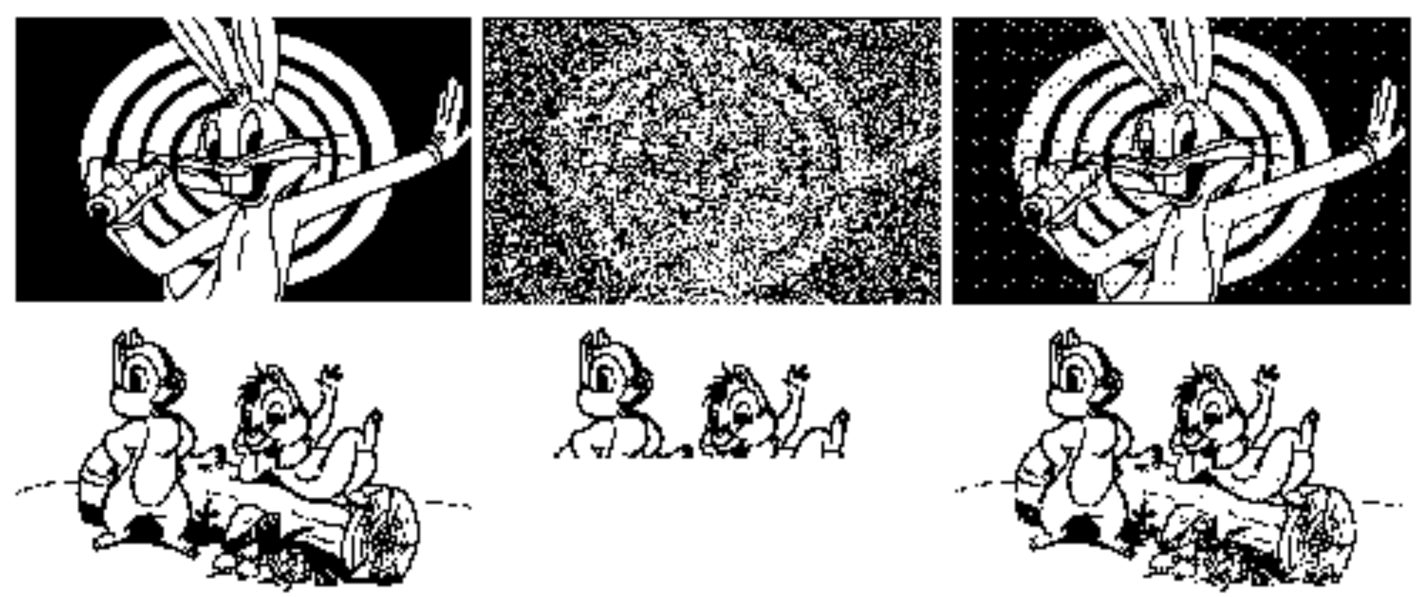
Réseau de Hopfield et la mémoire associative



Motif appris par le réseau

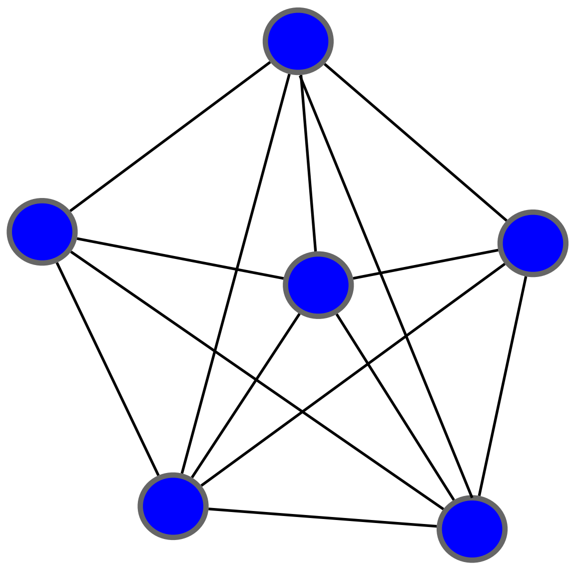
Motif dégradé présenté au réseau

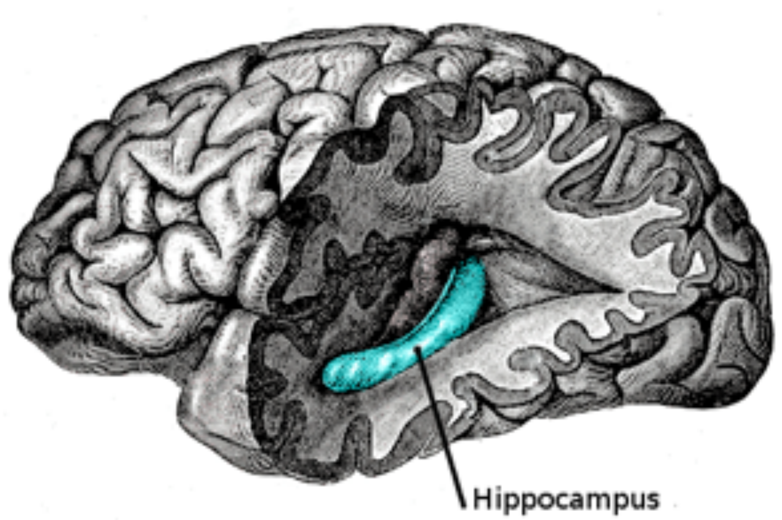
Motif reconstruit par le réseau



Original Degraded Reconstruction

Réseau de Hopfield et la mémoire associative





- L'hippocampe est la structure du cerveau qui joue le rôle central dans la mémoire
- L'aire CA3 de l'hippocampe est un énorme réseau récurrent
- Le réseau de Hopfield est le modèle actuel de la mémoire épisodique dans l'hippocampe