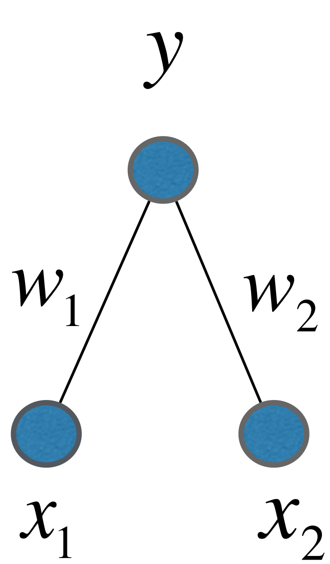
4I702 : TME 8

Apprentissage non-supervisé et réseaux récurrents

ACP par un neurone

ex_acp.py

Exercice :



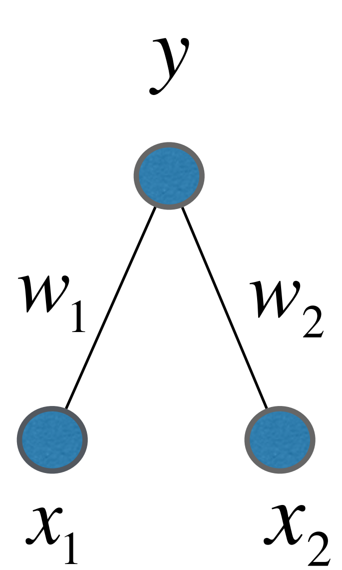
- Ajouter le calcul de la la matrice de covariance (lignes 20)
- Ajouter la règle d'apprentissage hebbien (ligne 57)
- Lancer le script. Que se passe-t-il avec le vecteur de poids ? Quel est le problème avec l'apprentissage hebbien pure ?
- Quelle règle d'apprentissage rénormalise automatiquement le vecteur de poids ? Remplacer la règle hebbienne par cette règle et observer le vecteur de poids résultant.

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

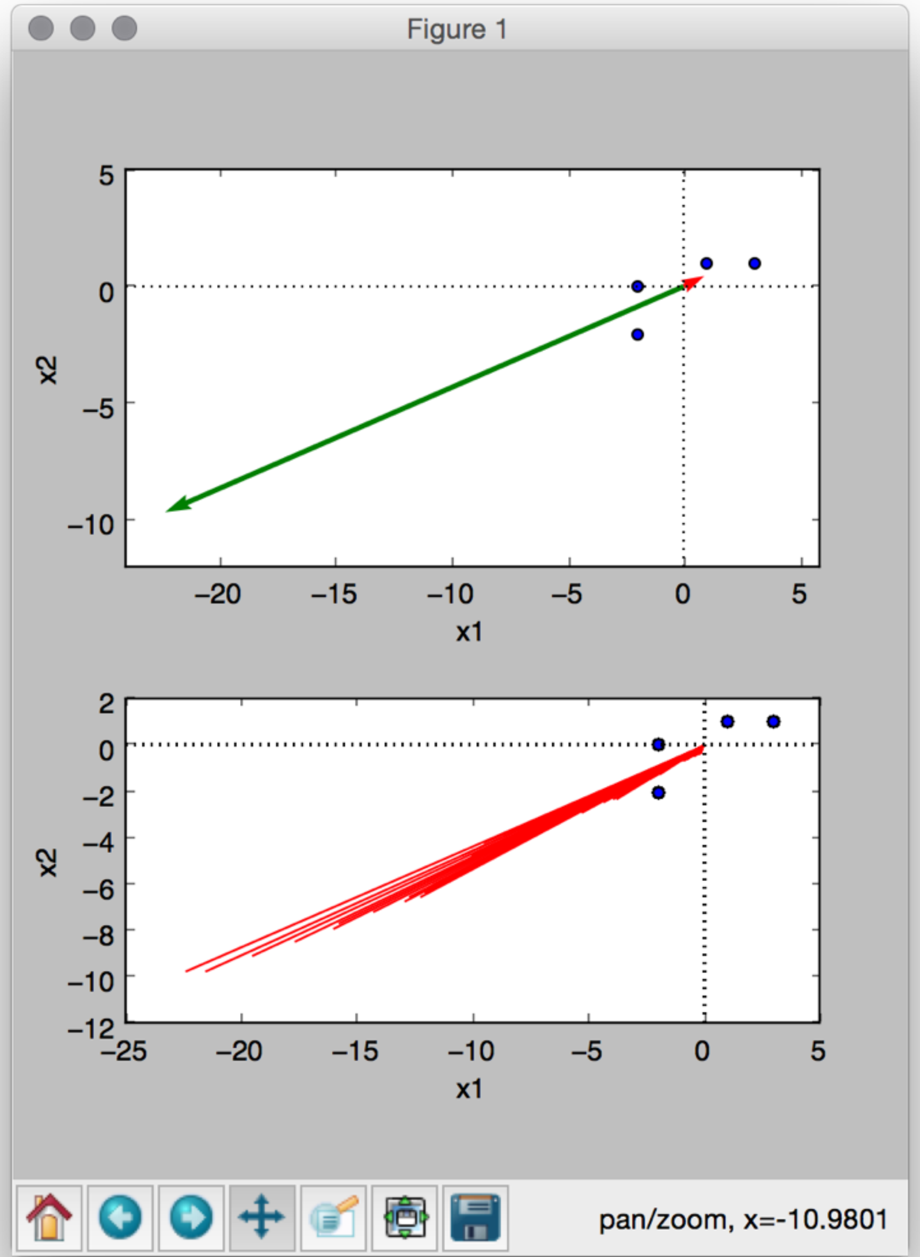
ACP par un neurone

ex_acp.py

Règle hebbienne pure



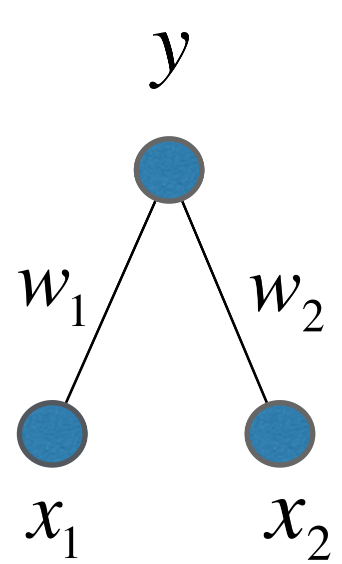
$$\Delta w_i = \eta y x_i$$



ACP par un neurone

ex_acp.py

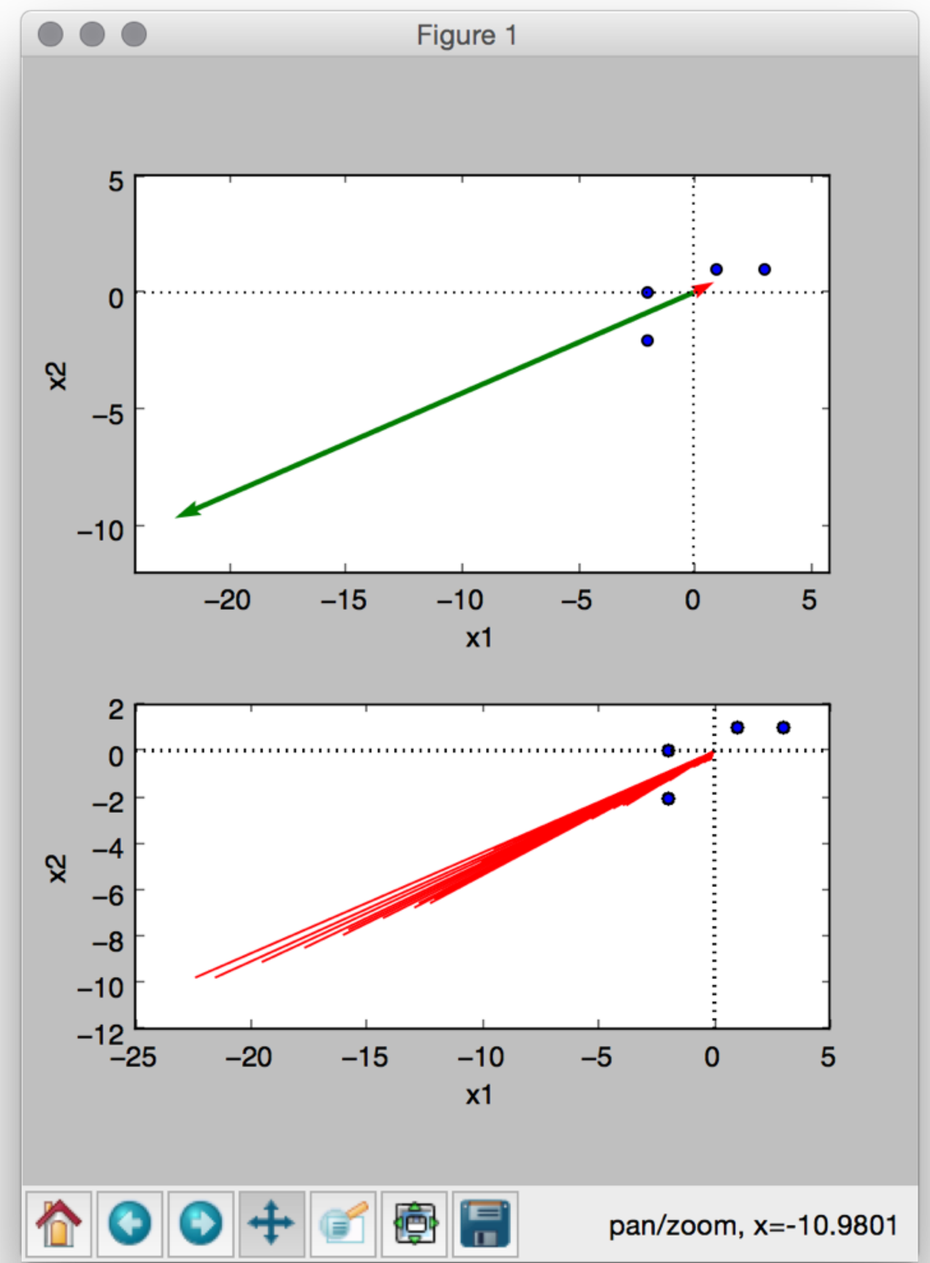
Règle hebbienne pure



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

1. La direction du vecteur de poids coïncide avec le 1er axe principal

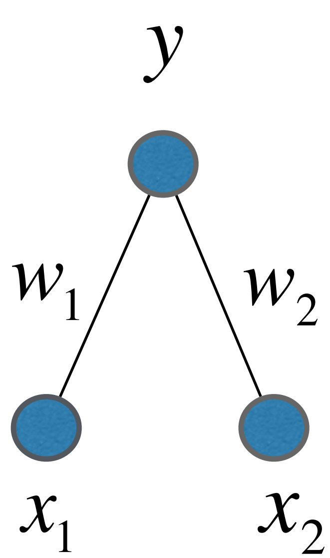
2. La norme du vecteur de poids augmente sans limite



ACP par un neurone

ex_acp.py

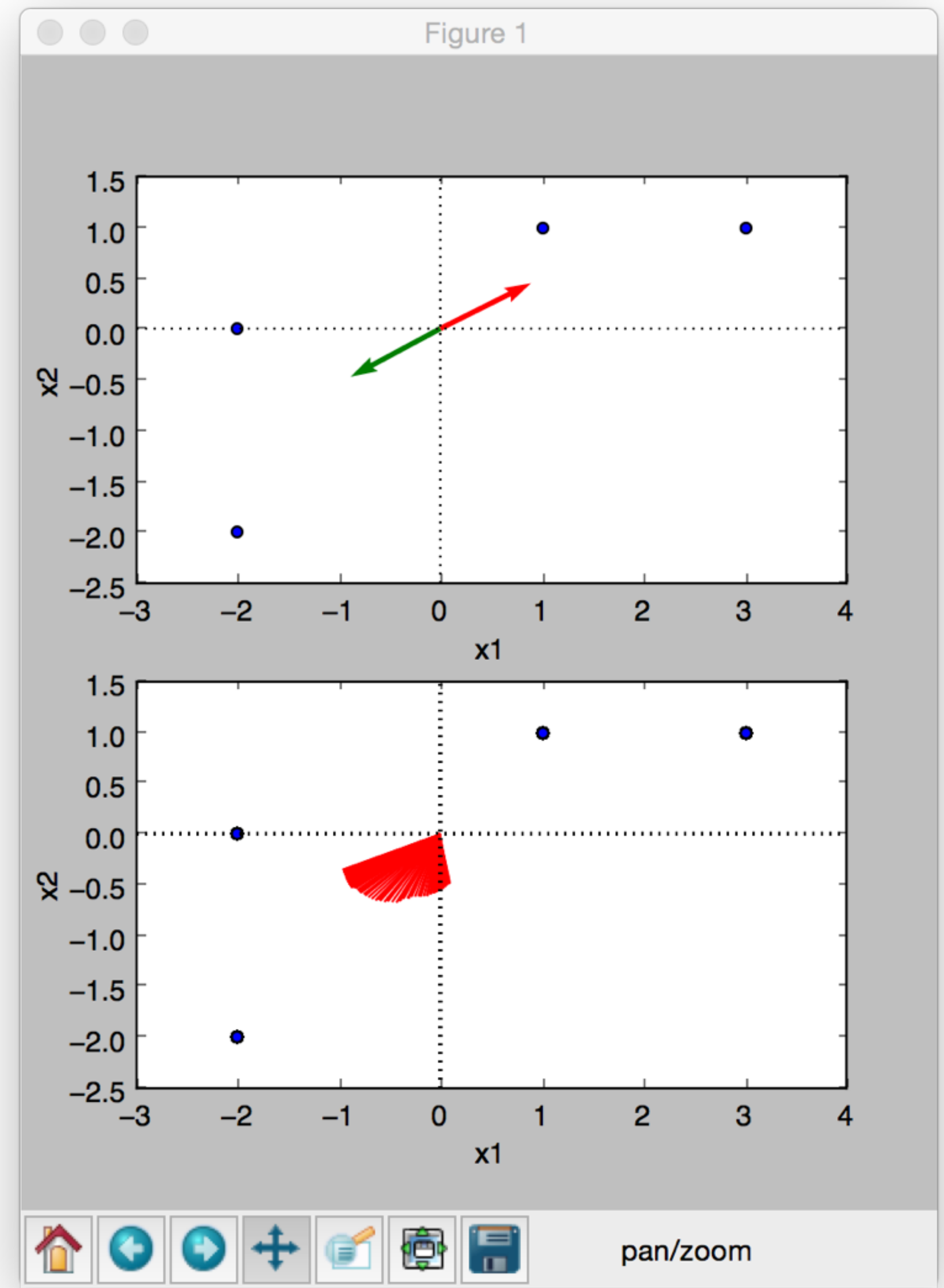
Règle de Oja



1. La direction du vecteur de poids coïncide avec le 1er axe principal

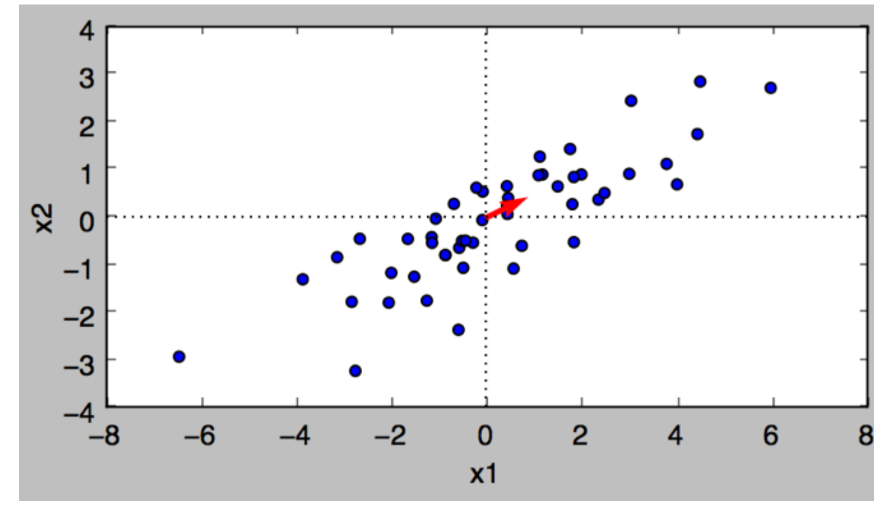
2. La norme du vecteur de poids égale à 1

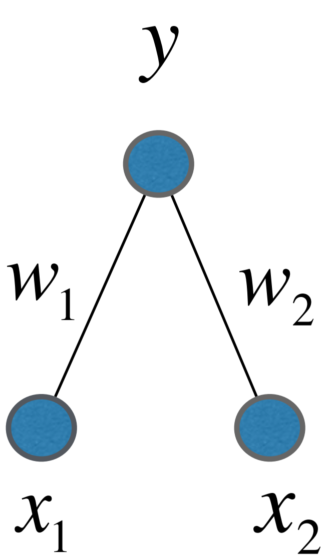
$$\Delta w_i = \eta y (x_i - y w_i)$$



ACP par un neurone

ex_acp.py





Vérifiez que l'algorithme marche bien avec d'autres ensembles de données

- decommenter les lignes 11 - 14

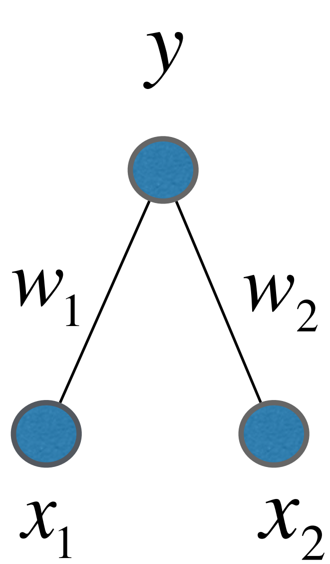
- lancer l'apprentissage

$$\Delta w_i = \eta y(x_i - y w_i)$$

ACP par un neurone

ex_acp.py

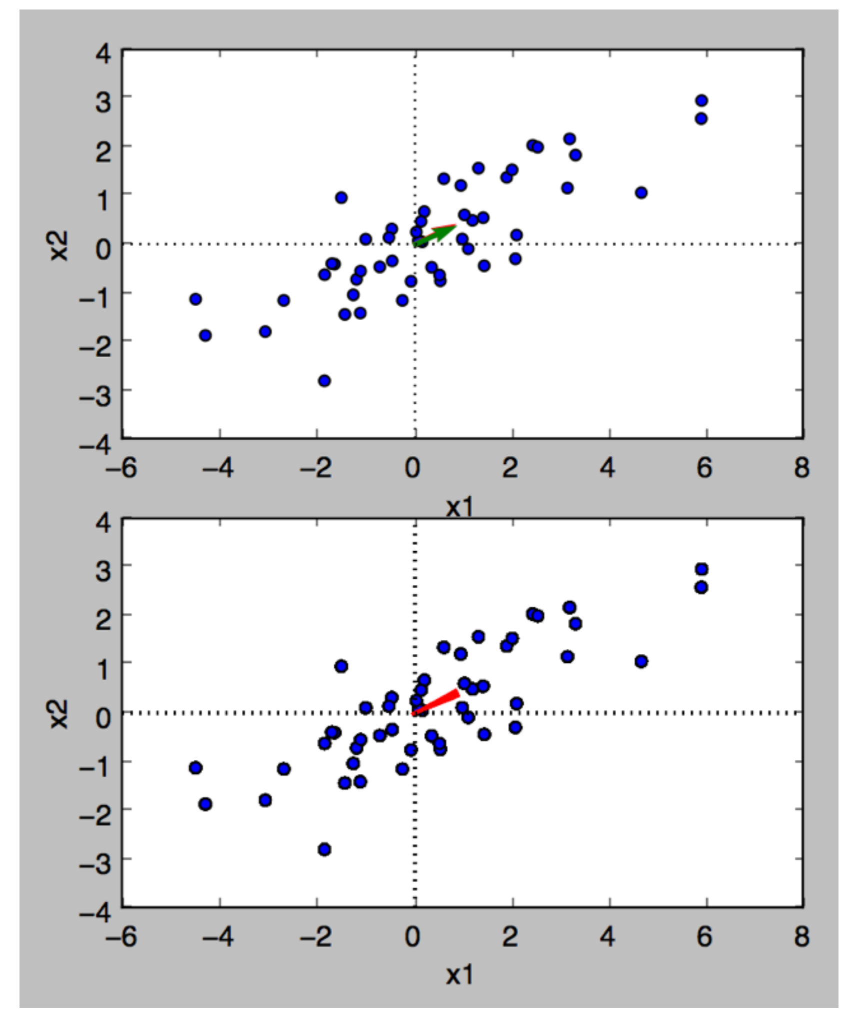
Règle de Oja



1. La direction du vecteur de poids coïncide avec celle de la 1ère composante principale

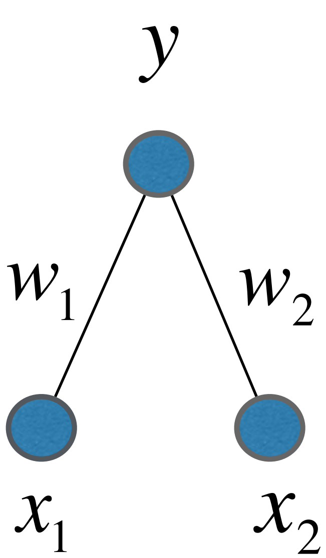
2. La norme du vecteur de poids égale à 1

$$\Delta w_i = \eta y(x_i - y w_i)$$



ACP par un neurone

ex_acp.py



Que se passe si les données ne sont pas centrées ?

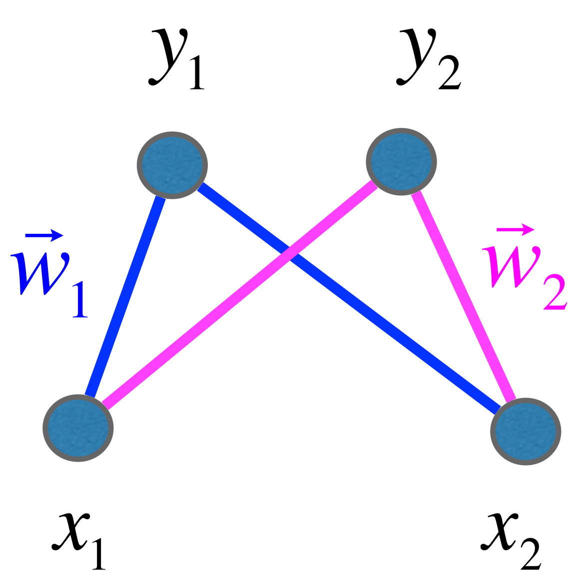
$$\Delta w_i = \eta y (x_i - y w_i)$$

Apprentissage compétitif

ex_compet.py

Exercice :

- Ajouter l'initialisation aléatoire du vecteur de poids (ligne 24)
- Ajouter la règle d'apprentissage compétitif (ligne 37)
- Lancer le script plusieurs fois. Marche-t-il toujours bien ? Pourquoi ? Proposer une solution.



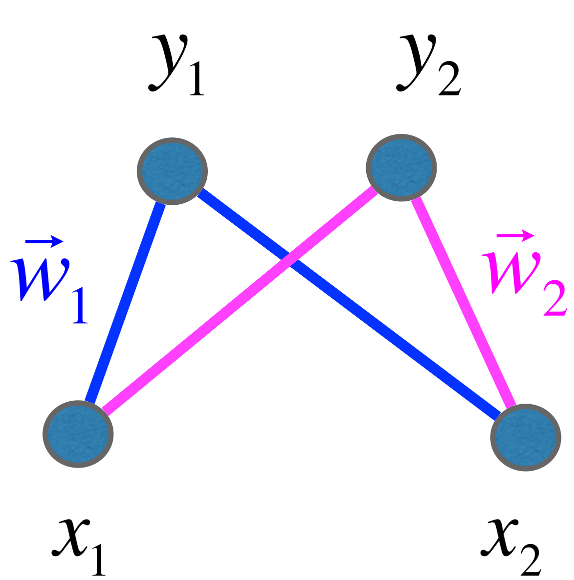
$$\vec{y} = W \cdot \vec{x}$$

$$\Delta w_{ki} = \eta y_k^*(x_i - y_k^* w_{ki})$$

$$\begin{cases} y_k = 1, & \text{si } k \text{ est l'index de } y_{max} \\ y_k = 0, & \text{sinon} \end{cases}$$

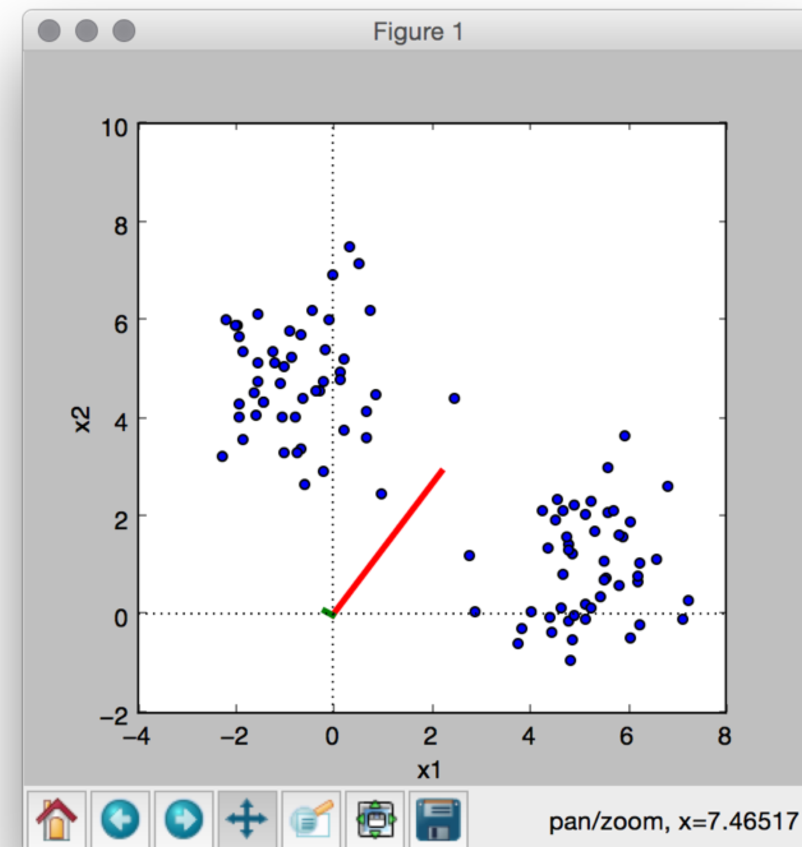
Apprentissage compétitif

ex_compet.py



1. La performance du réseau dépend de l'initialisation des poids

2. Afin d'améliorer l'algorithme, initialiser les vecteurs de poids par des motifs qui appartiennent déjà à l'ensemble



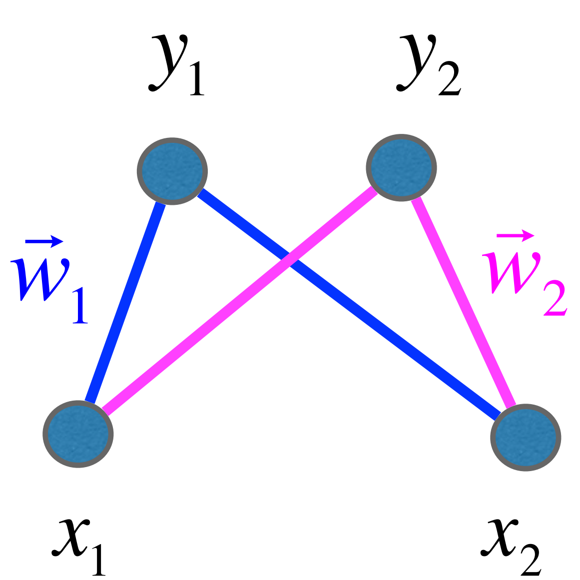
$$\vec{y} = W \cdot \vec{x}$$

$$\Delta w_{ki} = \eta y_k^* (x_i - y_k^* w_{ki})$$

$$\Delta \vec{w}_k = \begin{cases} \eta(\vec{x} - \vec{w}_k), & \text{si } k \text{ est l'index de } y_{max} \\ 0, & \text{sinon} \end{cases}$$

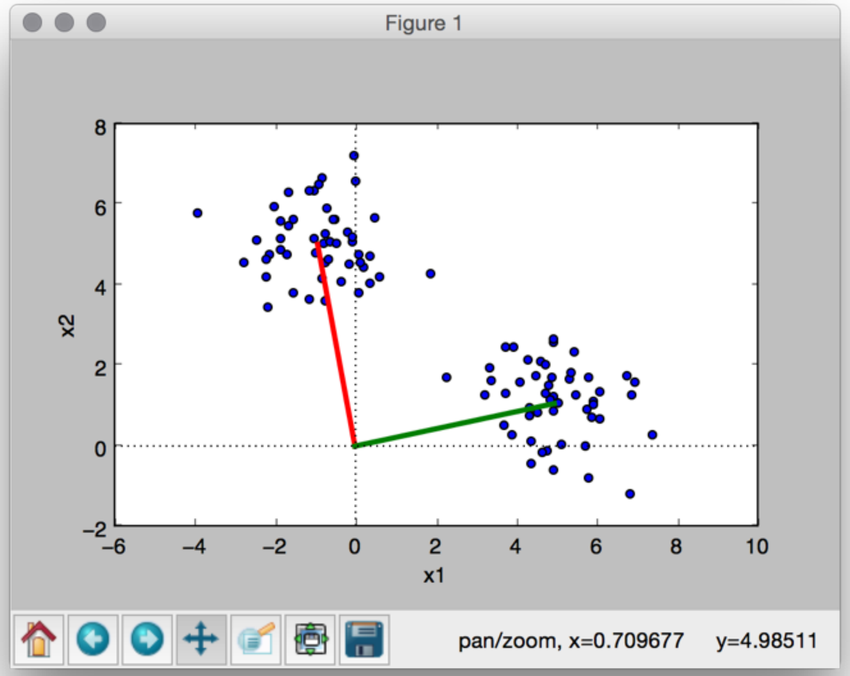
Apprentissage compétitif

ex_compet.py



1. La performance du réseau dépend de l'initialisation des poids

2. L'algorithme peut être amélioré par initialisation de deux vecteurs de poids par des motif de l'ensemble



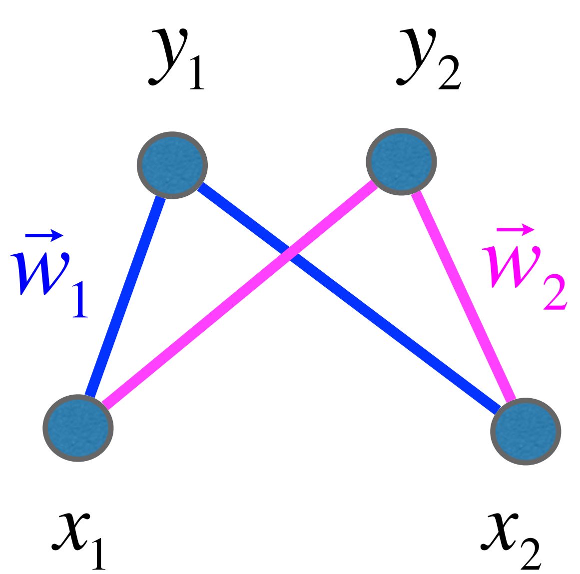
$$\vec{y} = W \cdot \vec{x}$$

$$\Delta w_{ki} = \eta y_k^* (x_i - y_k^* w_{ki})$$

$$\Delta \vec{w}_k = \begin{cases} \eta(\vec{x} - \vec{w}_k), & \text{si } k \text{ est l'index de } y_{max} \\ 0, & \text{sinon} \end{cases}$$

Apprentissage compétitif

ex_compet.py



Comment ce réseau peut être utilisé pour classifier des nouvelles données ?

Sur présentation d'un vecteur de données, les deux neurones de sortie calculent les projections de ce vecteur sur le vecteur de poids correspondant.

Où, à votre avis, passe la frontière de separation ? Quelle est la forme de cette frontière ?

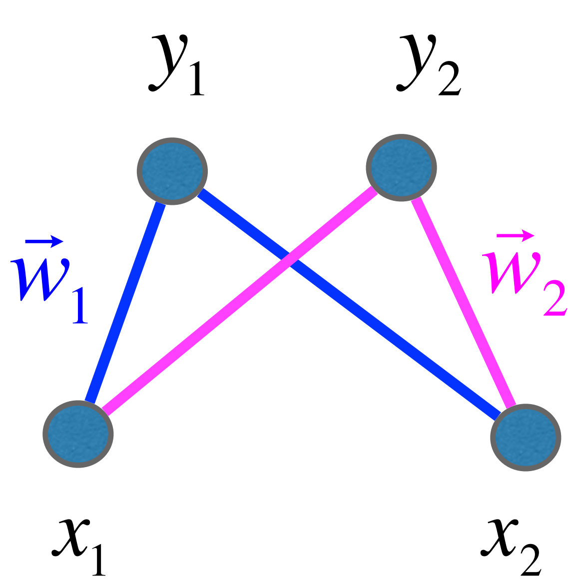
$$\vec{y} = W \cdot \vec{x}$$

$$\Delta w_{ki} = \eta y_k^* (x_i - y_k^* w_{ki})$$

$$\Delta \vec{w}_k = \begin{cases} \eta(\vec{x} - \vec{w}_k), & \text{si } k \text{ est l'index de } y_{max} \\ 0, & \text{sinon} \end{cases}$$

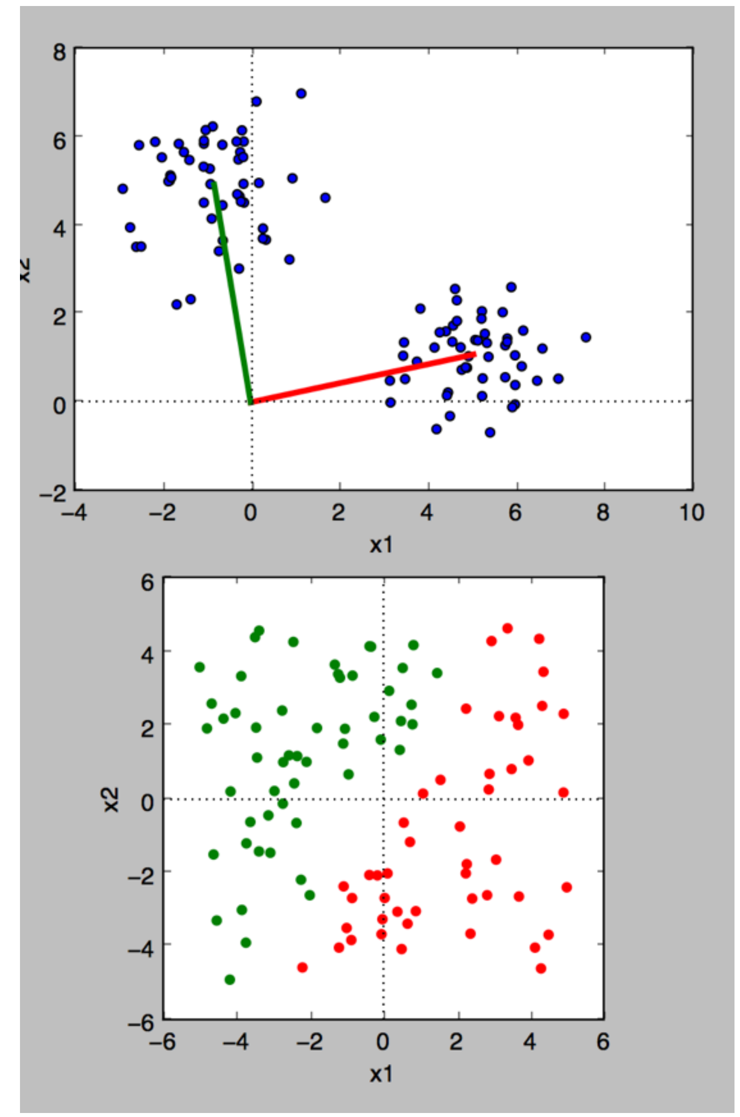
Apprentissage compétitif

ex_compet.py



Vérifiez votre réponse :

décommentez les lignes 57-75 et lancez le script.



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

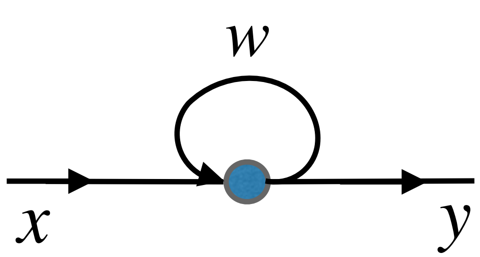
$$\Delta w_{ki} = \eta y_k^* (x_i - y_k^* w_{ki})$$

$$\Delta \vec{w}_k = \begin{cases} \eta(\vec{x} - \vec{w}_k), & \text{si } k \text{ est l'index de } y_{max} \\ 0, & \text{sinon} \end{cases}$$

Réseaux récurrents

Un neurone avec une synapse récurrent

ex_autapse.py



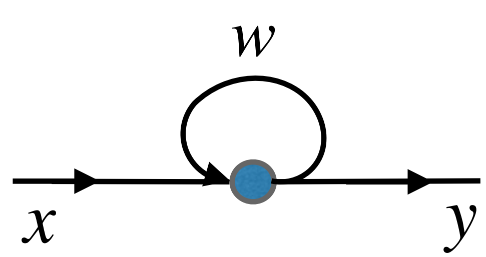
$$\dot{y} = -y + \frac{1}{1+e^{-wy-x}}$$

Exercices :

- Choisir les valeurs de l'entrée $\boldsymbol{x}$ et du poids synaptique $\boldsymbol{w}$ (lignes 6, 7) pour avoir trois états stationnaires du système (Voir TD)
- Utiliser le schéma d'Euler (TD6) pour trouver numériquement la solution de l'équation différentielle de ce système (ligne 24)
- Étudier la solution pour différentes conditions initiales (ligne 19). Déterminer les points fixes, séparatrices, bassins d'attraction

Un neurone avec une synapse récurrent

ex_autapse.py



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

Attracteurs :

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

Bassins d'attraction :

$$y < 0.5$$

$$y > 0.5$$

Point fixe instable :

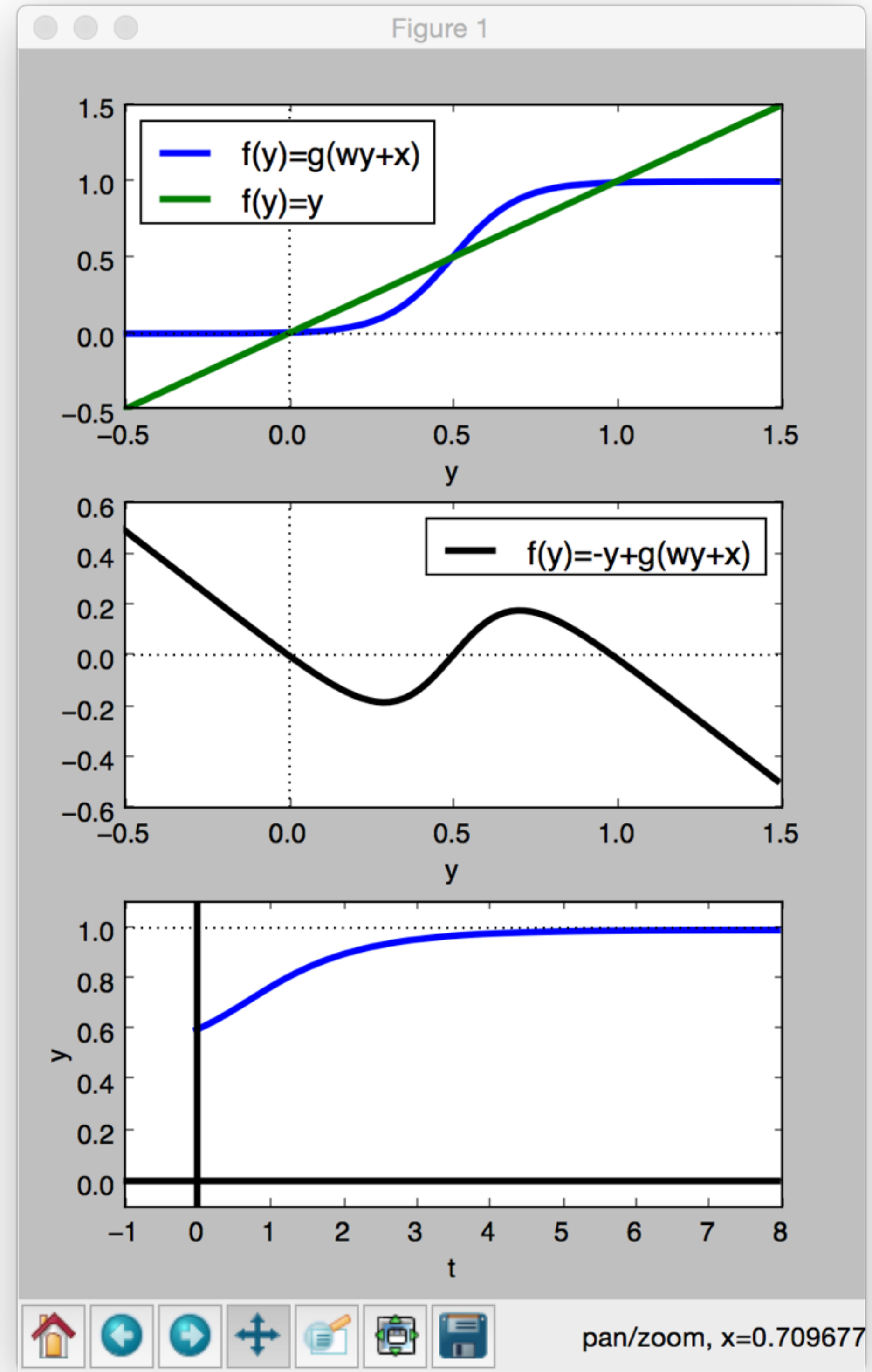
$$y = 0.5$$

Séparatrice :

$$y = 0.5$$

$$w = 10$$

$$x = -5$$

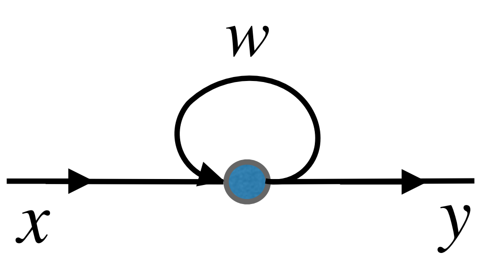


Un neurone avec une synapse récurrent

ex_autapse.py

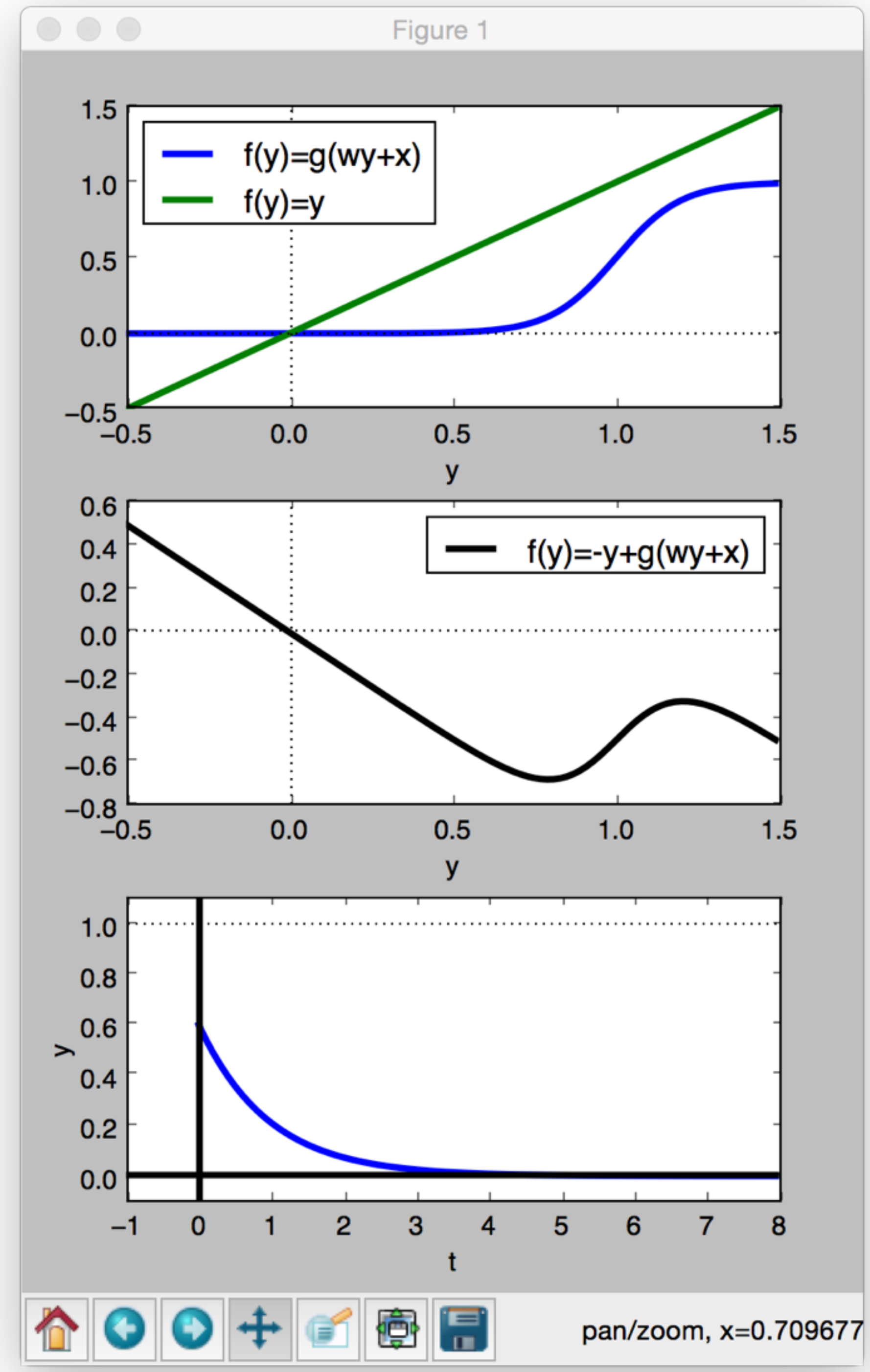
$$w = 10$$

$$x = -10$$



Valeurs de w et x déterminent les points fixes du système

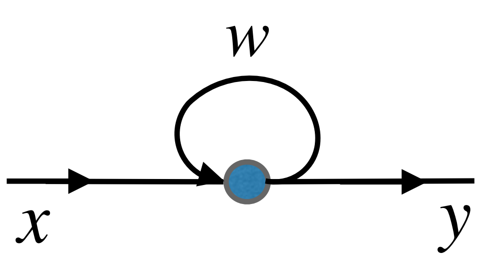
$$\dot{y} = -y + \frac{1}{1 + e^{-wy-x}}$$



Fonction énergie

ex_energie.py

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



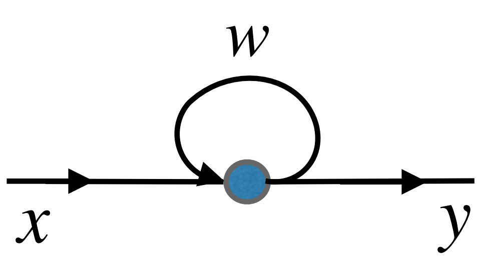
$$\dot{y} = -y + \frac{1}{1+e^{-wy-x}}$$

Exercice :

- Ecrire la fonction-énergie pour l'autapse (ligne 14, voir TD)
- A quoi correspond le point rouge sur le graphique ?
- Quelle sera la position du point rouge pour le point fixe instable ? Vérifiez.
- Que va se passer avec la fonction-énergie si l'on met par exemple ***x=-10*** et ***w = 10*** ? Proposer la réponse, puis vérifiez par simulation.

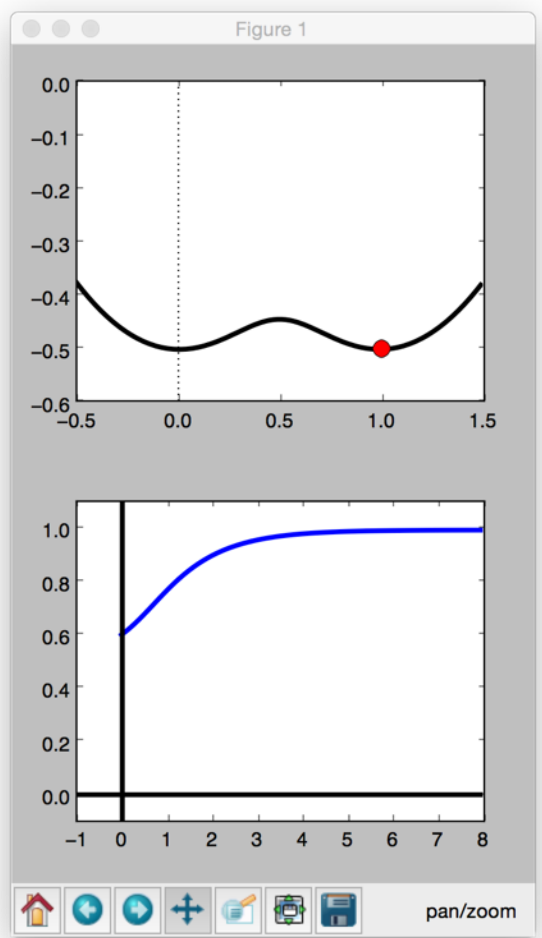
Fonction énergie

ex_energie.py

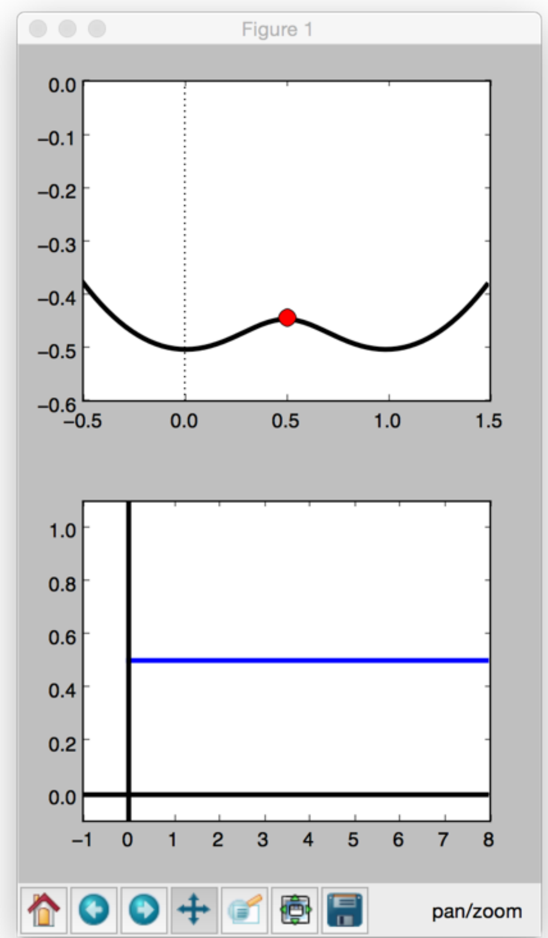


$$\dot{y} = -y + \frac{1}{1+e^{-wy-x}}$$

$y(0) = 0.6$

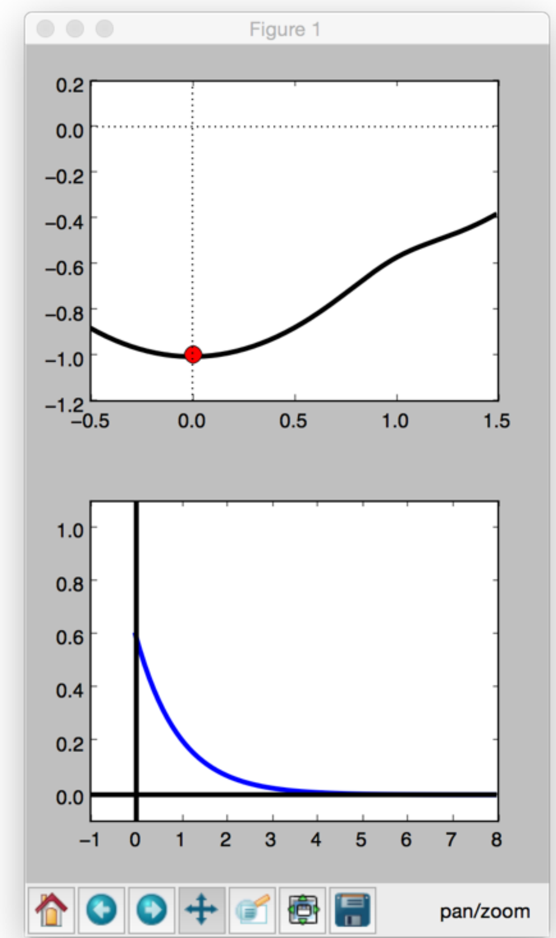


$y(0) = 0.5$



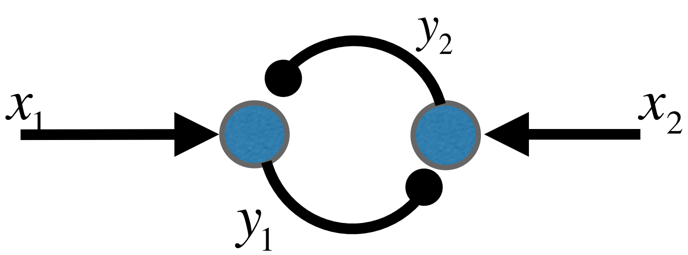
$w = 10$

$x = -10$



Deux neurones

ex_deux_neurones.py



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

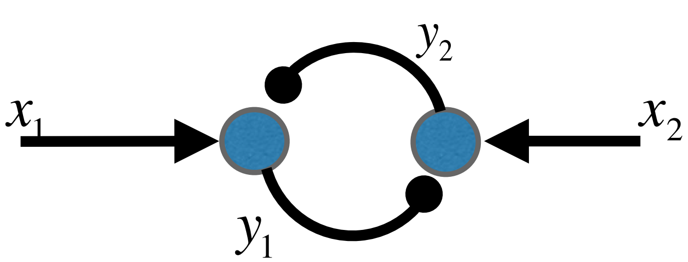
$$\dot{\vec{y}} = -\vec{y} + g(W \cdot \vec{y} + \vec{x})$$

Exercice :

- Ajouter les nullclines du système dynamique (lignes 14, 15, voir TD)
- Écrire la matrice des poids ***W*** (ligne 24)
- Implementer le schéma d’Euler pour résoudre le système d’équations (ligne 32). Déterminer les points fixes, bassins d’attraction, séparatrices
- Quel est le type de stabilité de tous les points fixes ?

Deux neurones

ex_deux_neurones.py



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

$$\dot{\vec{y}} = -\vec{y} + g(W \cdot \vec{y} + \vec{x})$$

Attracteurs :

$$\vec{y}^{(1)} = (0,1)$$
$$\vec{y}^{(2)} = (1,0)$$

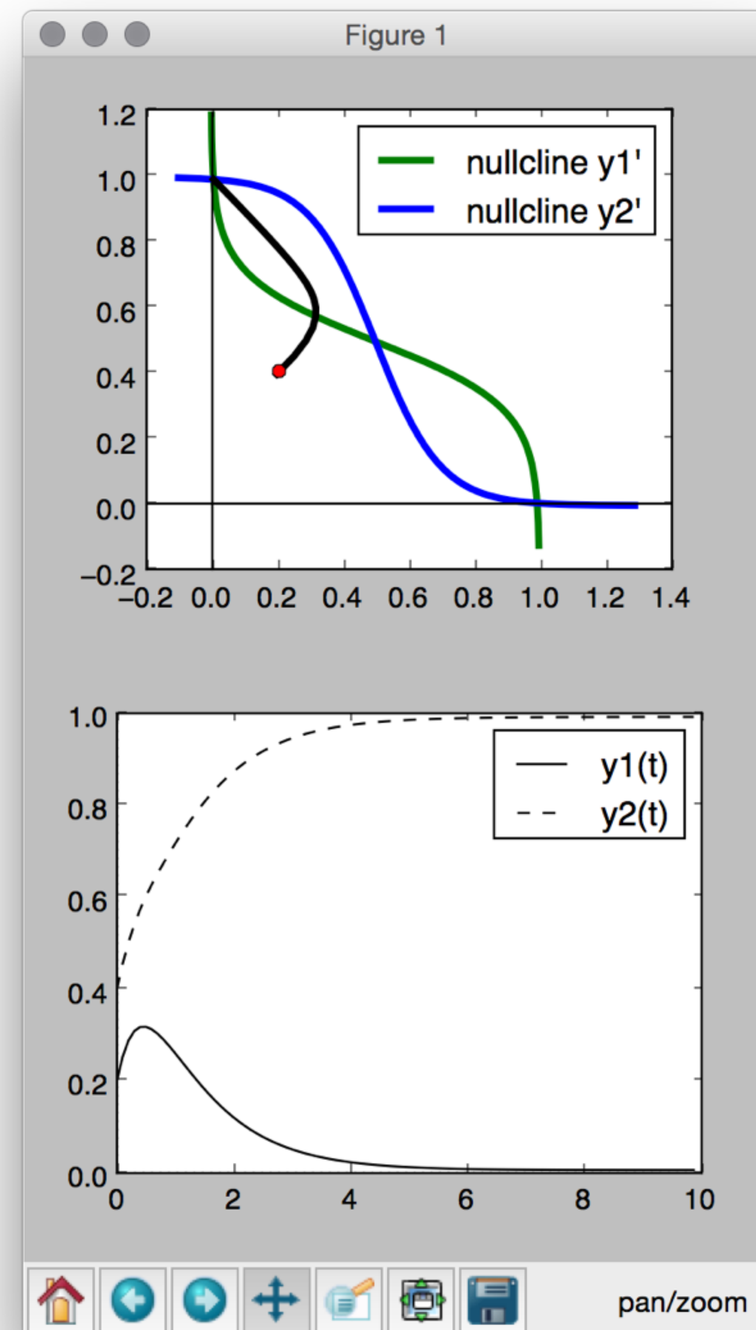
Séparatrice :

$$y_1 = y_2$$

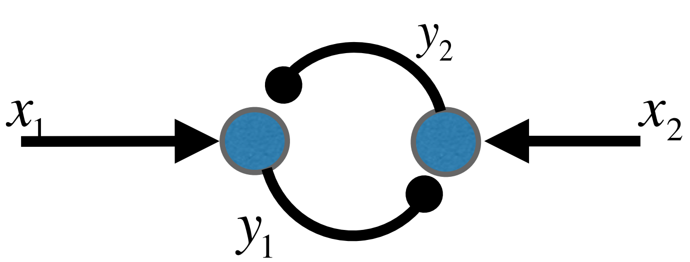
Bassin d'attraction :

$y_2 > y_1$ pour $\vec{y}^{(1)}$

$y_2 < y_1$ pour $\vec{y}^{(2)}$



Deux neurones



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

$$\dot{\vec{y}} = -\vec{y} + g(W \cdot \vec{y} + \vec{x})$$

On peut considérer ce réseau comme un modèle de mémoire pour deux motifs :

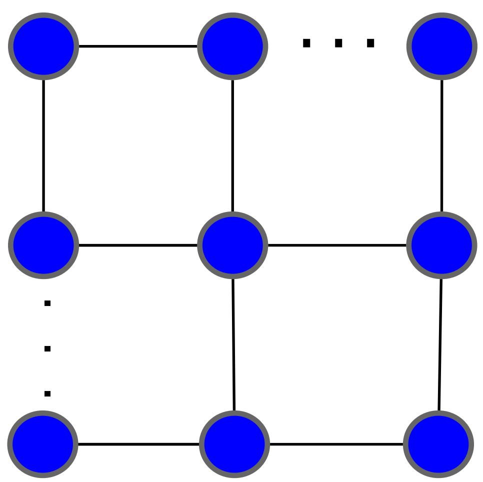
$$\vec{y}^{(1)} = (0,1)$$
$$\vec{y}^{(2)} = (1,0)$$

- Est-ce que ce réseau est capable à corriger des erreurs de mémoire ? Donner un exemple

Réseau de Hopfield

64 neurones

hopfield.py



Poids synaptiques hebbiens

$$w_{ij} = \eta y_i y_j$$

Pas d'entrées externes :

$$\vec{x} = 0$$

Dynamique :

$$\dot{\vec{y}} = -\vec{y} + g(W \cdot \vec{y} + \vec{x})$$

Motif mémorisé
(vecteur binaire)

Matrice de poids

Motifs "rappelé" à
partir d'un motif
dégradé