INTELLIGENCE ARTIFICIELLE

Partie 4: NEURONES FORMEL (PERCEPTRON)

Tarik AL ANI, Département Informatique
ESIEE-Paris
E-mail : t.alani@esiee.fr
Url: http://www.esiee.fr/~alanit

Modèles d 'apprentissage

- ***Apprentissage supervisé*** : au cours duquel on cherche à imposer au réseau une certaine fonctionnalité en forçant, à partir des entrées qui lui sont présentées, les sorties correspondantes par modification des poids synaptiques. Le réseau apparaît comme un filtre dont les paramètres de la fonction de transfère sont définis par les couples entrée/sorties présentés (***perceptrons***, ***Adaline***).

- ***Apprentissage non Supervisé*** **(** ***Apprentissage compétitif, Apprentissage associatif*** **) :**

  Au cours duquel on présente que des exemples en entrée et on laisse le réseau s'auto-organiser uniquement au moyen de lois locales qui régissent l'évolution des synapses (exemple: modèle de Kohonen).

Les étapes de fonctionnement d’un réseau de neurone

Deux phases:

Phase d’apprentissage : au cours de laquelle on présente au réseau la représentation du problème à résoudre et, dans le cas d’un apprentissage supervisé, la solution attendue. On aura donc au préalable codé le problème afin de le présenter au réseau. Le processus se répète sur tous les exemples à apprendre jusqu’à ce que les réponses données par le réseau soient jugées satisfaisantes.

Phase de reconnaissance : au cours de laquelle on présente au réseau un problème qu'il a appris afin d'obtenir la solution. On peut aussi présenter un problème nouveau mais voisin des exemples appris afin d'obtenir une solution. Cette possibilité de **généralisation** est une des caractéristiques les plus importantes des modèles connexionnistes.

neurone formel (perceptron) - règles simples d’apprentissage

Interprétation géométrique de l’action d’un neurone formel :

Ce neurone permet de séparer ou « partitionner » l’espace des données à l’entrée en deux parties séparées par un hyperplan selon le résultat de classification de l’entrée en « 1 » ou « 0 ».

Classification des données: espace d'entrée = 2

Considérons un neurone formel à deux entrées: $X=[x_1\ x_2]^T$

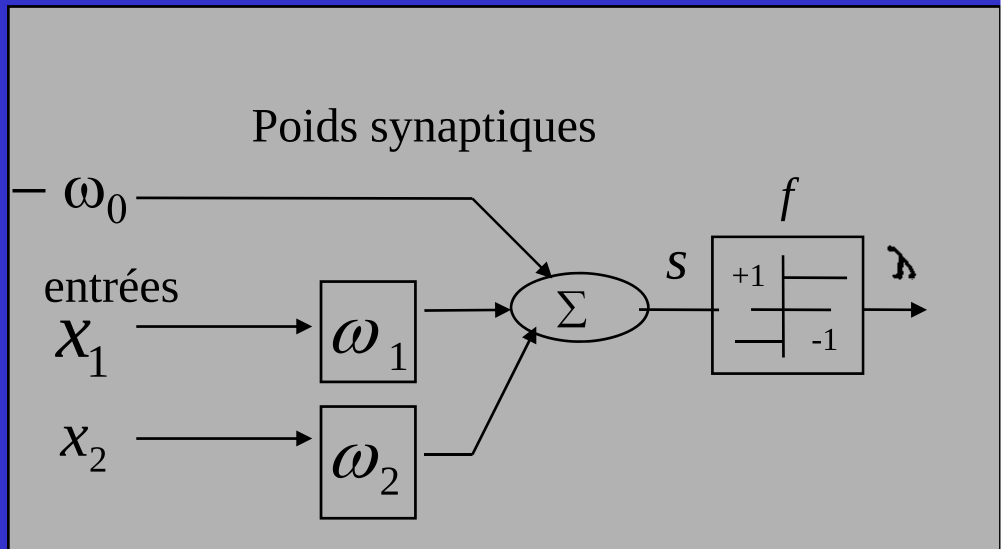
b = biais = - ω_0, ω_0 est un seuil

$$\text{activation} \quad : a(x) = X^T \bullet W = \sum_{j=1}^{2} \omega_j x_j$$

$$W = \text{vecteur de poids} = [\omega_1\ \omega_2]^T$$

$$s = X^T \bullet W + b$$

$$y = f(s)$$



Séparation linéaire des classes

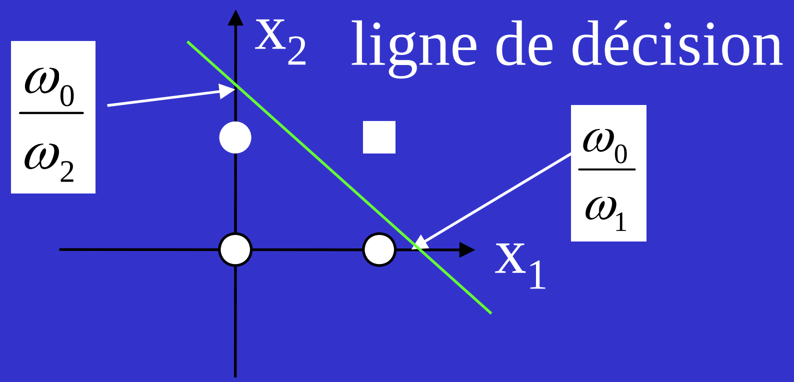
Condition critique de séparation : ***activation = seuil***

$$b = biais = -\omega_0$$

$X^T \bullet W = \omega_0$ équation d'une ligne droite de décision

- Dans l'espace 2D si $\omega_0 > 0$ nous obtenons une équation d'une line droite avec une pente $p = -\frac{\omega_1}{\omega_2}$

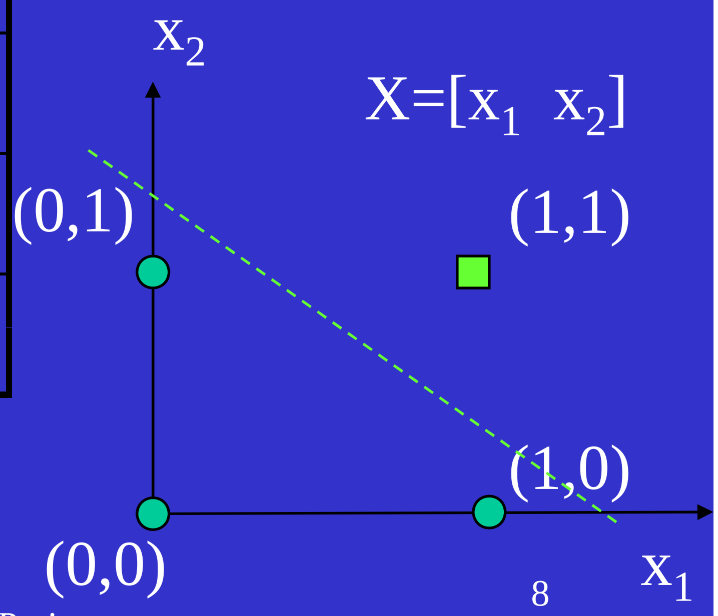
$$x_2 = -\frac{\omega_1}{\omega_2} x_1 + \frac{\omega_0}{\omega_2}$$



Exemple d’un espace d’entrée = 2 avec 2 classes (classe 1: ● , classe 2: ■)

$\omega_1 = \omega_2 = 1$, seuil $\omega_0 = 1{,}5$

x_1	x_2	activation	sortie
0	0	0	0 ●
0	1	1	0 ●
1	0	1	0 ●
1	1	2	1 ■



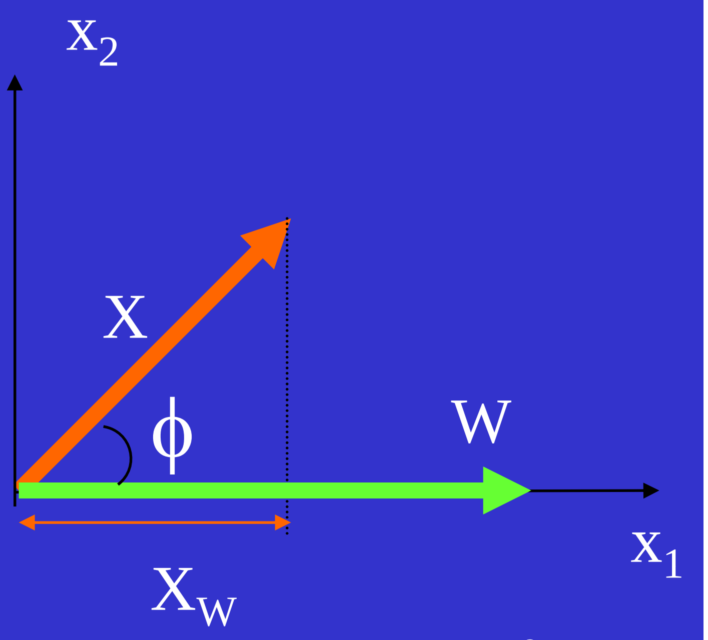
Le produit scalaire permet de trouver le degré de l'alignement entre deux vecteurs.

Projection de X sur W :

$$X_W = \frac{X^T \bullet W}{\|W\|}$$

Condition critique de séparation :

$$X^T \bullet W = \omega_0$$

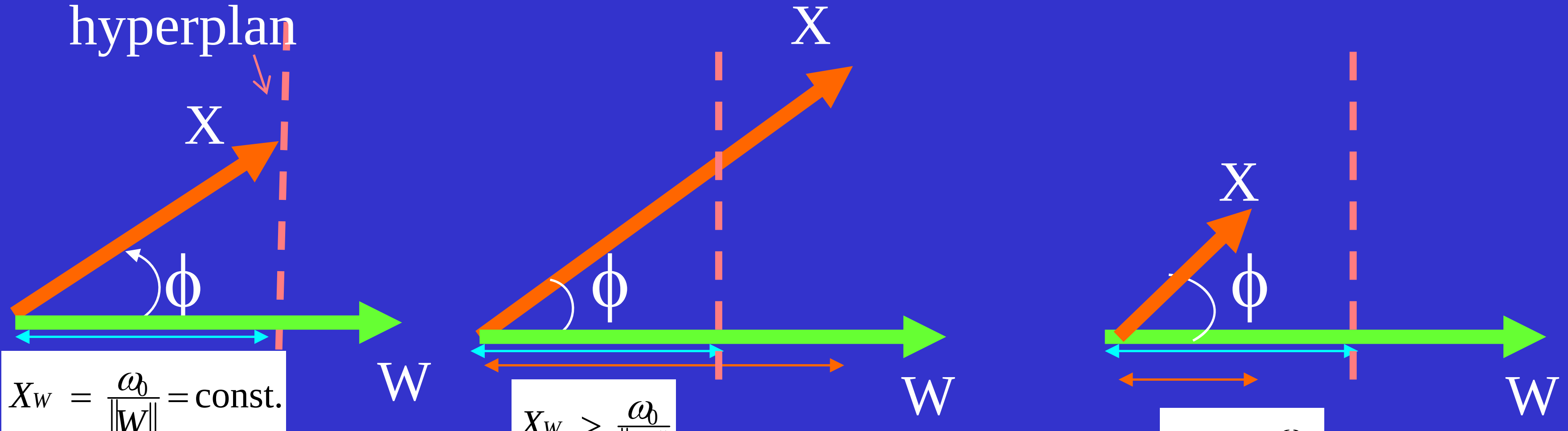


- En 3-D la ligne de décision devient un plan (***hyperplan***)

Supposons que W et ω_0 sont constants

$X_W = \text{const.}$	$X_W > w_0$	$X_W < w_0$
$X^T \bullet W = w_0$	$X^T \bullet W > w_0$	$X^T \bullet W < w_0$

X est sur le « hyperplan »



$$X_W = \frac{\omega_0}{\|W\|} = \text{const.}$$

$$X_W > \frac{\omega_0}{\|W\|}$$

$$X_W < \frac{\omega_0}{\|W\|}$$

- En n-D le plan de décision devient un n-cubes(***hypercube***)

Classification des données : X et W augmentés
Considérons un neurone formel à trois entrées:
$X=[-1\ x_1\ x_2]^T$: espace d'entrée = 3

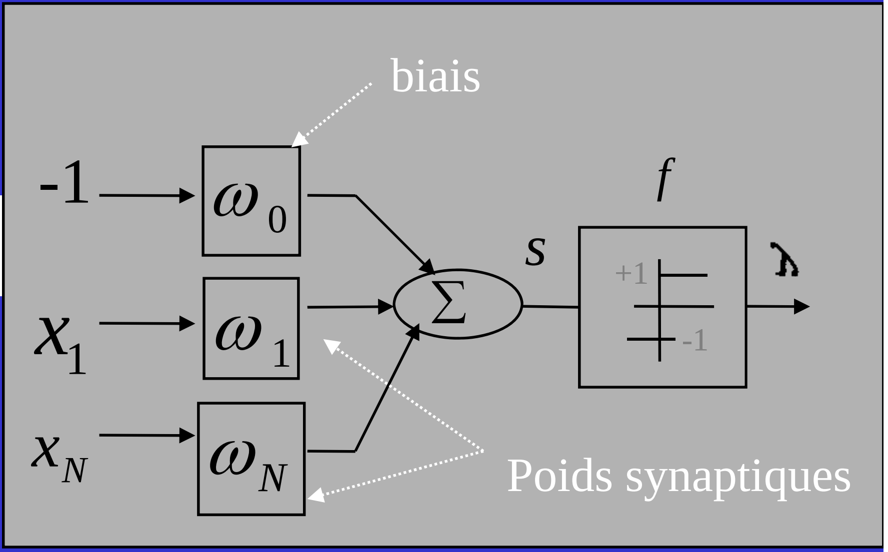
$$b = biais = -\omega_0$$

$$activation\ \ a(x) = \sum_{j=1}^{2} \omega_j x_j$$

W = vecteur de poids augmenté $= [\omega_0\ \omega_1\ \omega_2]^T$

$s = X^T \bullet W$

$y = f(s)$

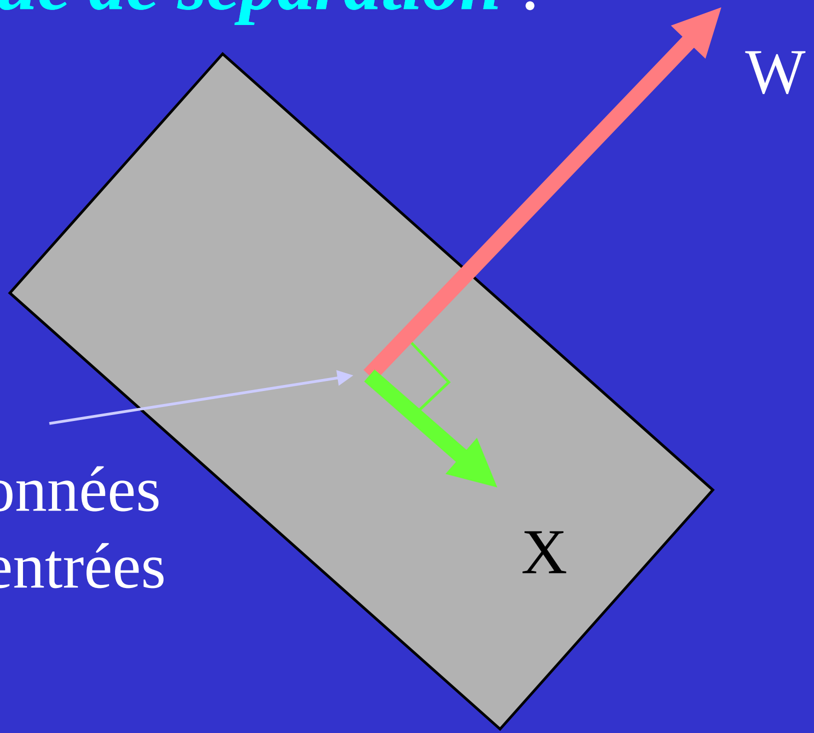


Condition critique de séparation :

$$X^T \bullet W = 0$$

L'hyperplan est toujours orthogonal au vecteur de poids augmenté et passe par l'origine de l'espace de représentation de données augmentées.

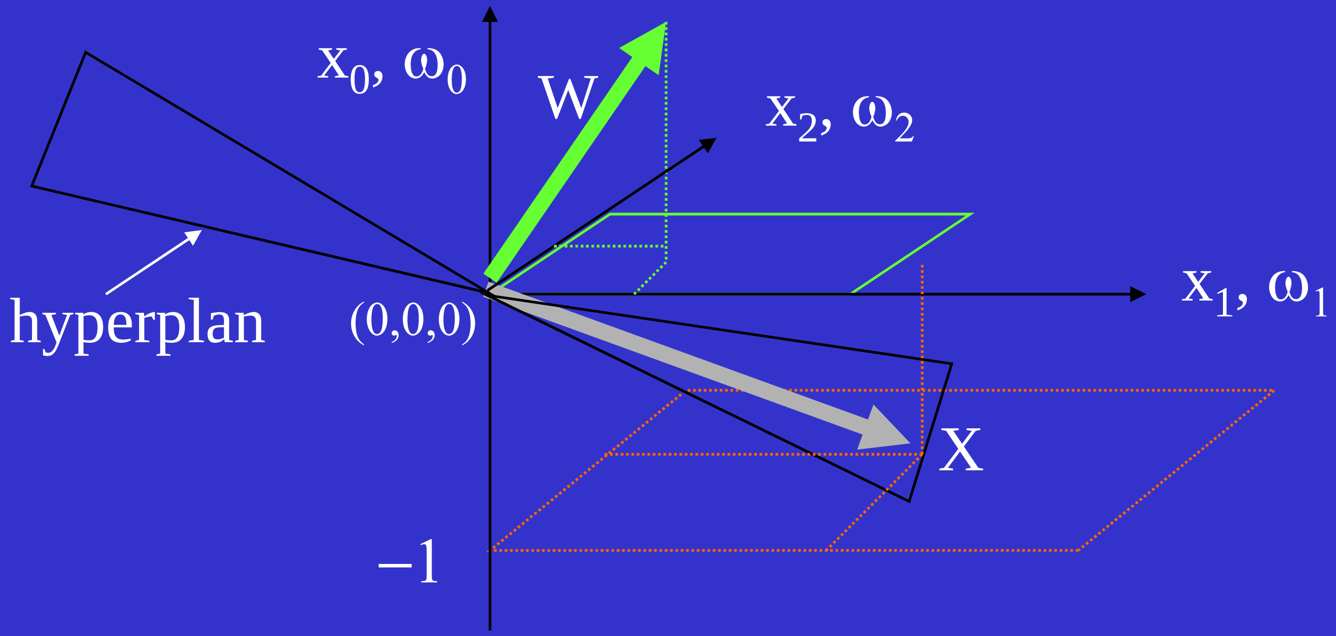
Condition critique de séparation :



Origine des coordonnées
dans l'espace des entrées

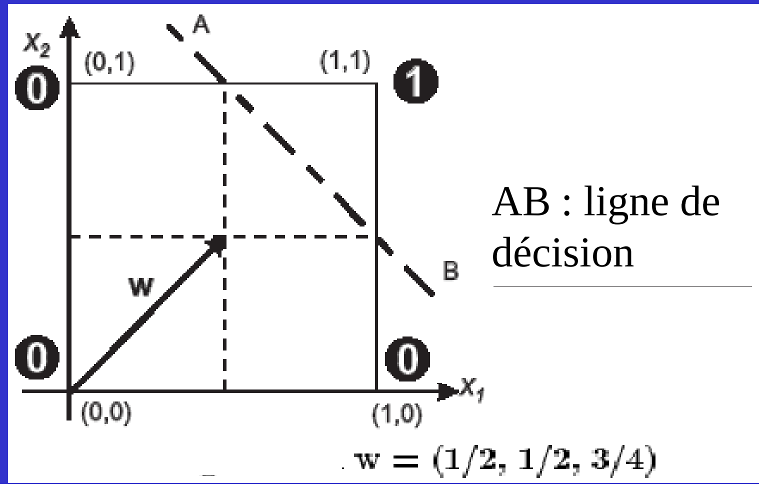
Le vecteur de poids doit être orthogonal au plan de décision et passe par l'origine.

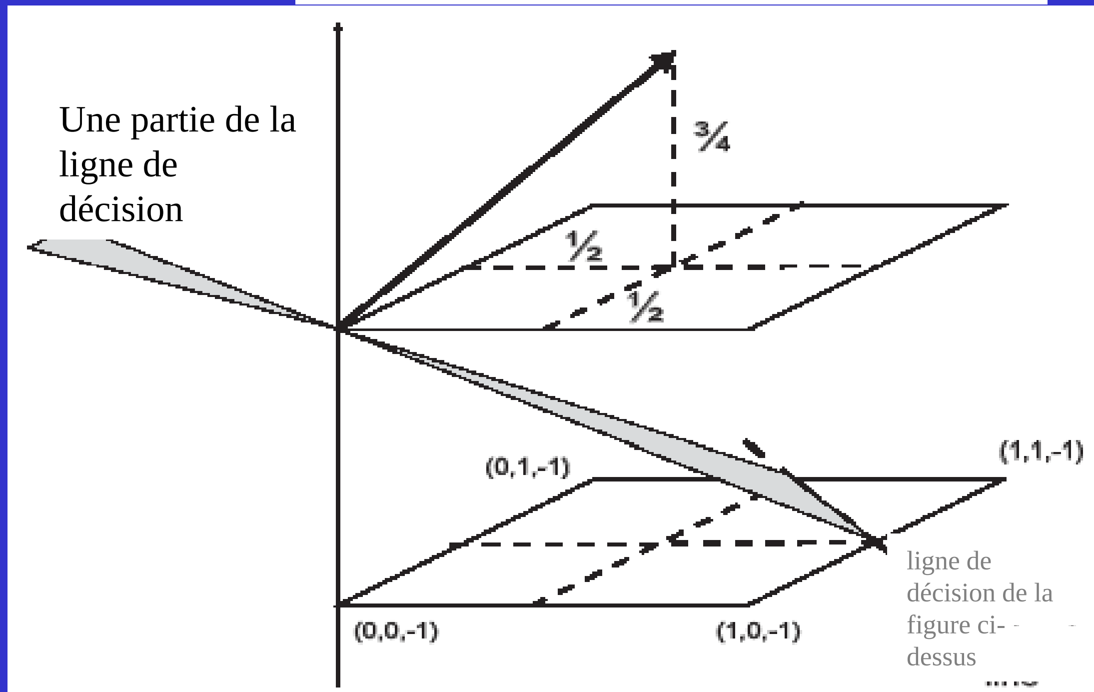
RESEAUX DE NEURONES FORMELS



Exemple :

Fonction ET logique

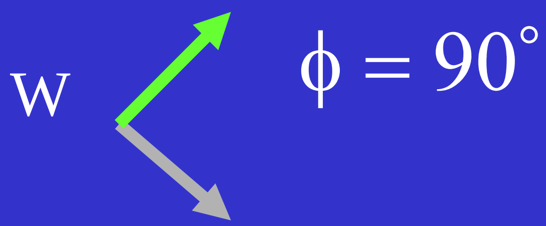


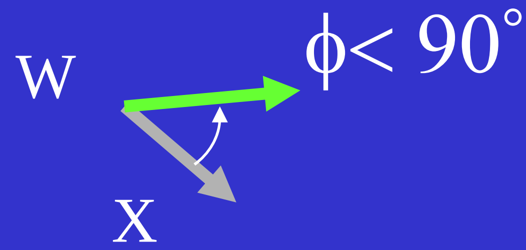


RESEAUX DE NEURONES FORMELS

$$X^T \bullet W \geq 0$$

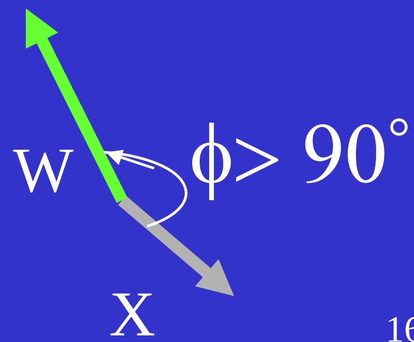
$$y = +1$$





$$X^T \bullet W < 0$$

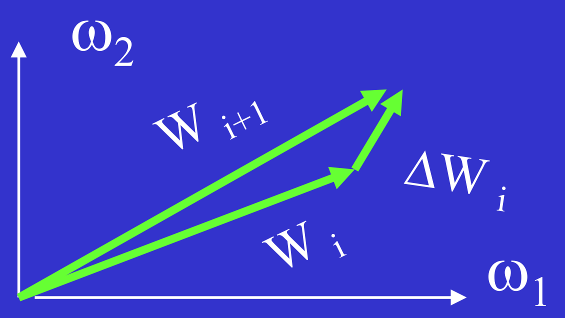
$$y = 0 \text{ (ou } -1)$$



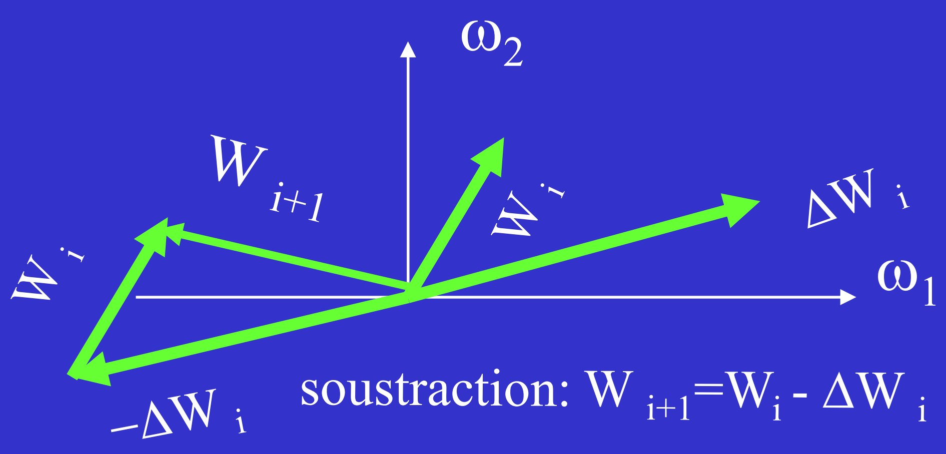
Apprentissage d'un neurone formel :

Ajuster W pour que la classification désirée soit obtenue.

Changer W peut être considéré comme une addition avec un autre vecteur.



addition: $W_{i+1}=W_i + \Delta W_i$



soustraction: $W_{i+1}=W_i - \Delta W_i$

Apprentissage supervisé

Etant donnée un ensemble de couples d'apprentissage {X,d}

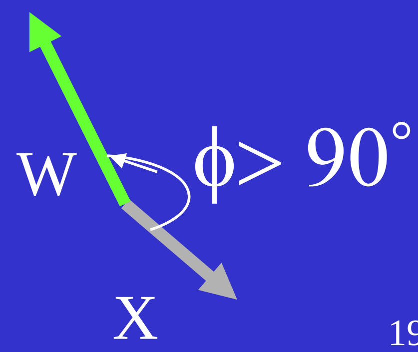
X: vecteur augmenté d'entrée (exemple plus (-1))

d: valeur scalaire désirée (***cible*** ou ***consigne***) correspondant à X.

Supposons qu'un couple (X,d) est présenté à l'entrée d'un neurone possédant un vecteur augmenté de poids W.

- ***Cas 1*** : classification erronée (la sortie réelle y=0 au lieu de y = d = 1)

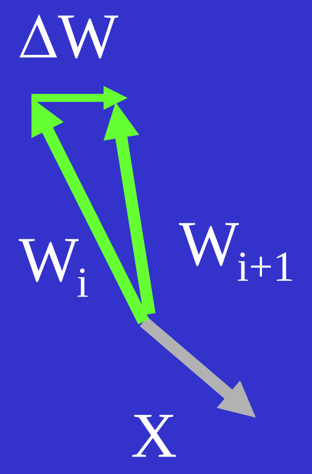
$X^T \bullet W < 0$ au lieu de $X^T \bullet W > 0$



e= d - y = 1

Pour corriger la situation précédente, il faut faire tourner W dans la direction de X tout en évitant trop de modifications importantes qui peuvent provoquer un bouleversement de l’apprentissage précédent.

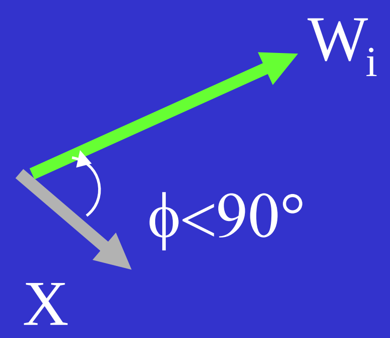
Solution: pour obtenir W_{i+1} il faut rajouter une fraction $\Delta W=\alpha X^T$ à W_i : $W_{i+1} = W_i + \alpha X^T$, $0<\alpha<1$



Le vecteur d'entrée X est additionné au vecteur de poids w. Cela fait le vecteur de poids s'orienter vers le vecteur d'entrée, augmentant la chance que le vecteur d'entrée sera classifié comme 1 dans l'avenir

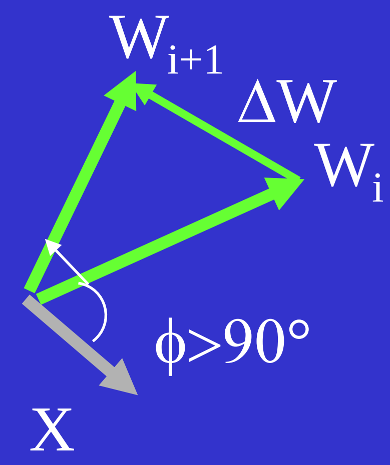
- ***Cas 2*** : classification erronée (la sortie réelle y=1 au lieu de y = d = 0)

$$\boxed{X^T \bullet W > 0} \text{ au lieu de } \boxed{X^T \bullet W < 0}$$



e= d - y = -1

Solution: pour obtenir W_{i+1} il faut rajouter une fraction $\Delta W = -\alpha X^T$ de W_i : $W_{i+1} = W_i - \alpha X$, $0<\alpha<1$



Le vecteur d'entrée X est soustrait du vecteur de poids w. Cela fait le vecteur de poids s'orienter plus loin du vecteur d'entrée, augmentant la chance que le vecteur d'entrée sera classifié comme 0 dans l'avenir

Règle de Widrow-Hoff (ou ***règle de perceptron***)

Les deux équations précédentes peuvent être mise sous la forme d'une règle (mécanisme d'apprentissage) :

$W_{i+1} = W_i + \Delta W = W_i + \alpha(d-y)X, \quad 0<\alpha<1$

y=0 ou 1; d=0 ou 1

Le paramètre α est appelé ***taux d'apprentissage*** (ou ***pas d'adaptation***).

Théorème de convergence

S'il existe un ensemble de poids de synaptiques W* capable de réaliser la transformation y=d(X), la règle d'apprentissage de perceptron convergera vers une solution (qui peut être la même que W*) en un nombre fini d'itérations à partir d'un choix initial arbitraire de vecteur de poids.

***Algorithme général d'apprentissage supervisé d'un neurone*:**

Nous possédons une base de donnée contenant des couple de vecteurs d'entrée **X** et des valeurs correspondantes de sorties désirées d.

Pour ajuster les poids il faut que le vecteur de poids soit orthogonal au plan de décision (hyperplan) et que ce plan doit passer par l'origine des coordonnées dans l'espace d'exemples **X** ($\mathbf{X}^T .\mathbf{W}=\mathbf{0}$).

Initialiser un vecteur du poids W et un pas d'apprentissage α.

répéter k=1, 2, ...

pour chaque vecteur de couple (**X**,d)

évaluer la sortie réelle y quand **X** est présenté à un neurone;

si y $\neq$ d alors

calculer un nouveau vecteur du poids **W**(k) selon un règle d 'apprentissage convenablement choisie;

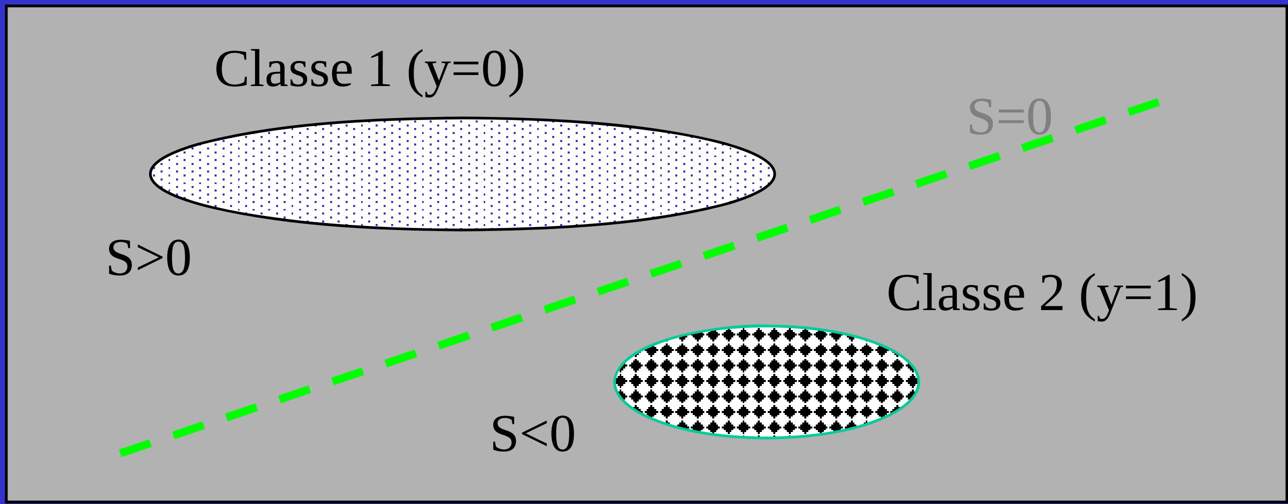
sinon

garder (ou diminuer) les poids;

finsi

fin pour

jusqu'à ce que **y**=d pour tous les vecteurs de couples



Séparation de deux classes ($s=X'W$)

Exemple

Un neurone possédant un vecteur poids **W**=[0 0.4] et un seuil ω_0=0.3. Il doit apprendre une fonction de ET logique : toutes les sorties sont « 0 » sauf avec les entrées (1,1). Le taux d'apprentissage α = 0.25.

La règle d'apprentissage utilisée est celle définie précédemment.

Exemple (suite)

$$activation\ \ a(x)=\sum_{j=1}^{2}\omega_j x_j$$

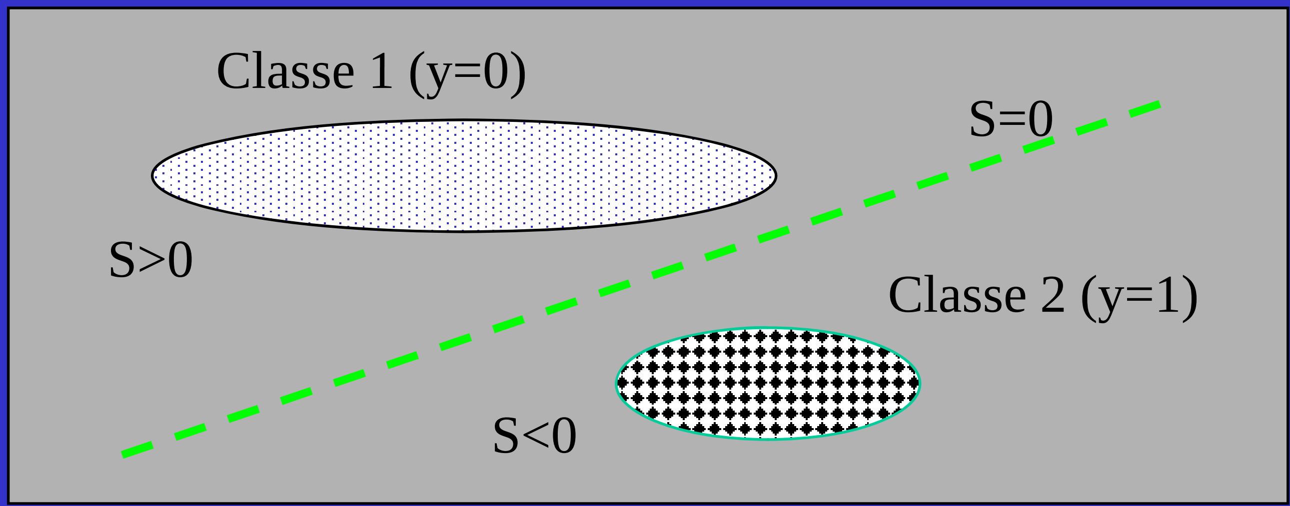
$$X=[-1\ \ x_1\ \ x_2]^T$$

$$W=[\omega_0\,\omega_1\,\omega_2]^T$$

$$\Delta W=[\delta\omega_0\ \delta\omega_1\,\delta\omega_2]^T=\alpha(d-y)X^T$$

$$W_{i+1}=W_i+\Delta W$$

i	ω_0	ω_1	ω_2	x_1	x_2	a	s	y	d	α(d-y)	$\delta\omega_0$	$\delta\omega_1$	$\delta\omega_2$
0	0.3	0.0	0.4	0	0	0	-0.3	0	0	0	0	0	0
1	0.3	0.0	0.4	0	1	0.4	0.1	1	0	-0.25	0.25	0	-0.25
2	0.55	0.0	0.15	1	0	0	-0.55	0	0	0	0	0	0
3	0.55	0.0	0.15	1	1	0.15	-0.40	0	1	0.25	-0.25	0.25	0.25
4	0.3	0.25	0.4	0	0	0	-0.3	0	0	0	0	0	0
5	0.3	0.25	0.4	0	1	0.4	0.1	1	0	-0.25	0.25	0	-0.25



Séparation de deux classes ($s=X^TW$)