

INTELLIGENCE ARTIFICIELLE

Partie 5: Le Perceptron

Tarik AL ANI, Département Informatique

ESIEE-Paris

E-mail : t.alani@esiee.fr

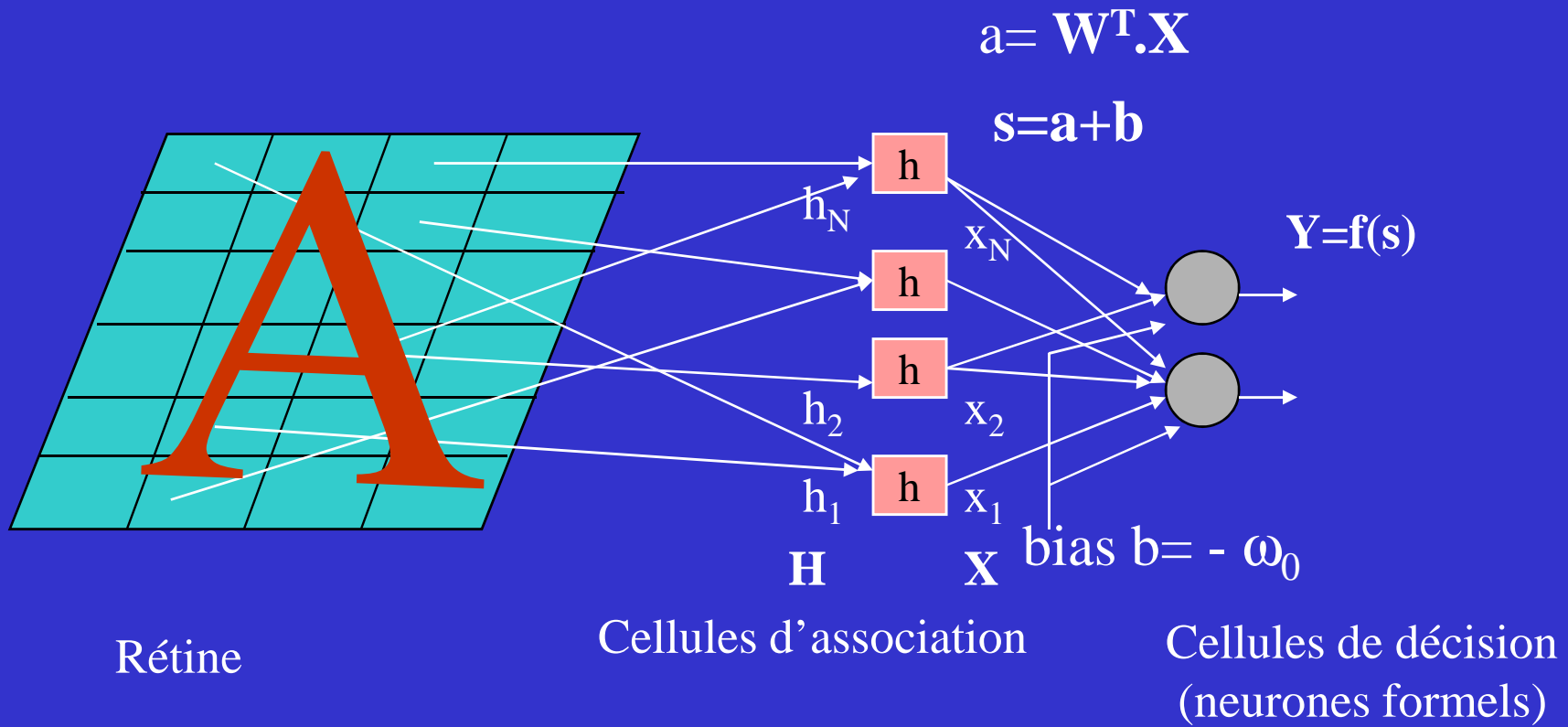
Url: <http://www.esiee.fr/~alanit>

Réseaux de neurones : Les modèles à couches : le perceptron

les *modèles à couches* ou « *perceptrons* »

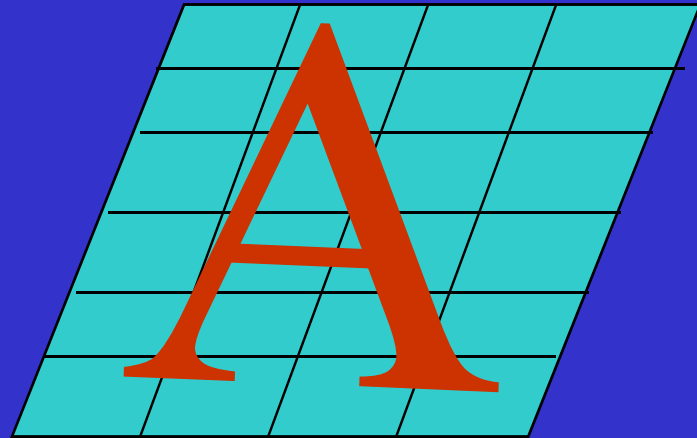
En 1958 Rosenblatt propose un modèle monocouche largement inspiré des travaux de neuro-anatomie et neurophysiologie, notamment sur les systèmes visuels.

Ce modèle est le premier système permettant un apprentissage automatique à partir d'exemples à fins de classification.



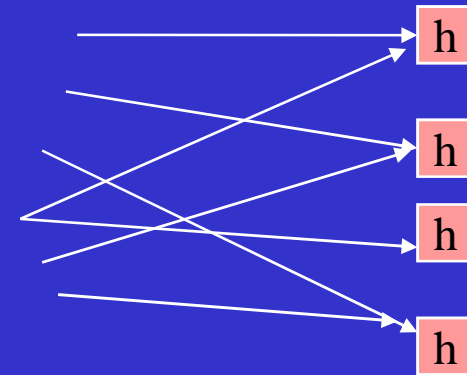
Architecture

Formé de trois couches :



Une *rétilne* : constituée de cellules, on y inscrit l'exemple à faire apprendre. Ces cellules jouent le rôle de capteurs. Elle reçoit les exemples ou formes à classer. Chaque élément de la rétine peut être considéré comme un pixel prenant des valeurs binaires 1 ou 0.

Une couche de *cellules d'associations* : constituée de cellules fixes d'association qui sont connectées totalement ou de façon aléatoire aux cellules de la rétine. Généralement, chaque cellule est dotée d'une fonction h d'association (booléenne ou linéaires) des stimuli envoyés par les cellules de la rétine qui lui sont connectées. La sortie pondérée de chaque cellule est connectée totalement ou partiellement aux cellules de la couche de décision.

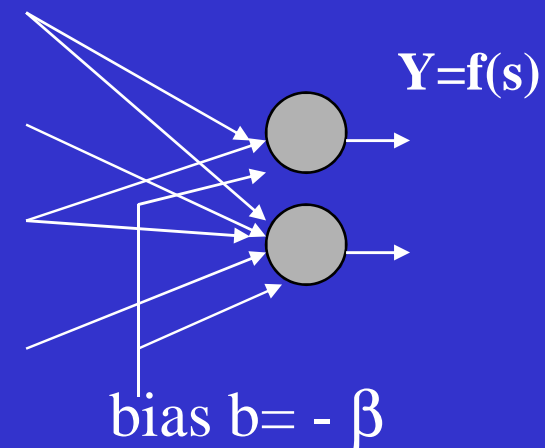


Une couche de *cellules de décision* :

Chaque cellule (neurone formel) est un automate à seuil de fonction d'activation f_i qui délivre la sortie binaire Y_i . La combinatoire de toutes les configurations possibles est presque infini si l'on influe sur les connexions et la nature des fonctions f et h .

$$a = \mathbf{W}^T \cdot \mathbf{X}$$

$$s = a + b$$



Utilisation des perceptrons comme des classificateurs

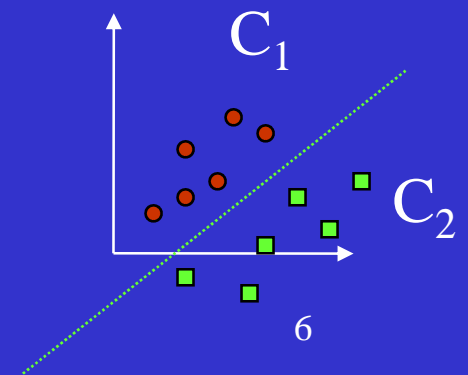
Cas de 2 classes linéairement séparables : C_1 , C_2

En utilisant l'algorithme d'apprentissage de perceptron, nous pouvons alors utiliser le perceptron pour classifier deux classes linéairement séparables C_1 et C_2 .

Exemple : Reconnaissance des caractères

C_1 : alphabets

C_2 : chiffres



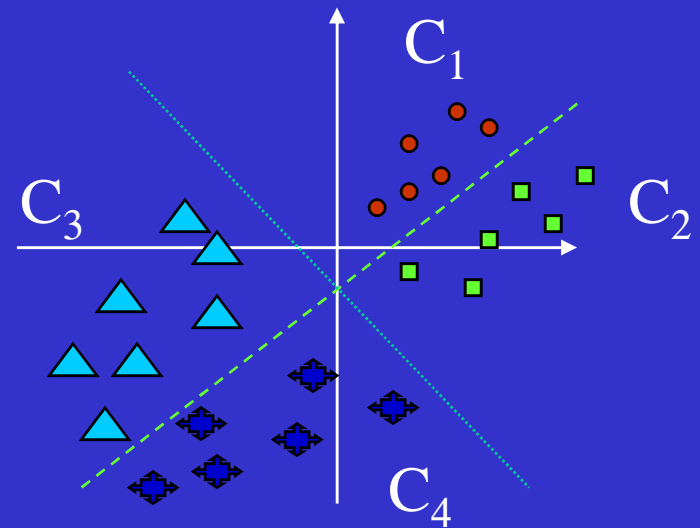
- Cas de 4 classes linéairement séparables : C_1 , C_2 , C_3 et C_4

Nous pouvons dans ce cas utiliser 2 neurones formels pour effectuer ces deux classifications :

	1	0
y_1	(c1, c2)	(c3, c4)
y_2	(c1, c3)	(c2, c4)

Table de classification

06/11/2011



y_1	y_2	classe
0	0	C_3
0	1	C_4
1	0	C_2
1	1	C_1

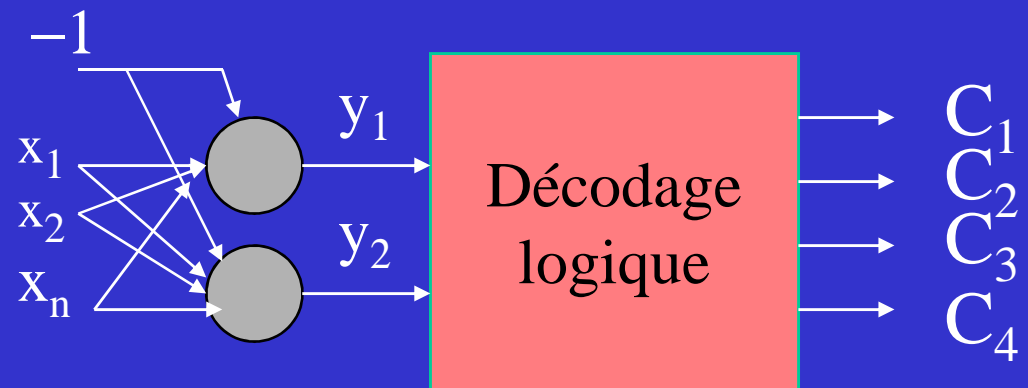
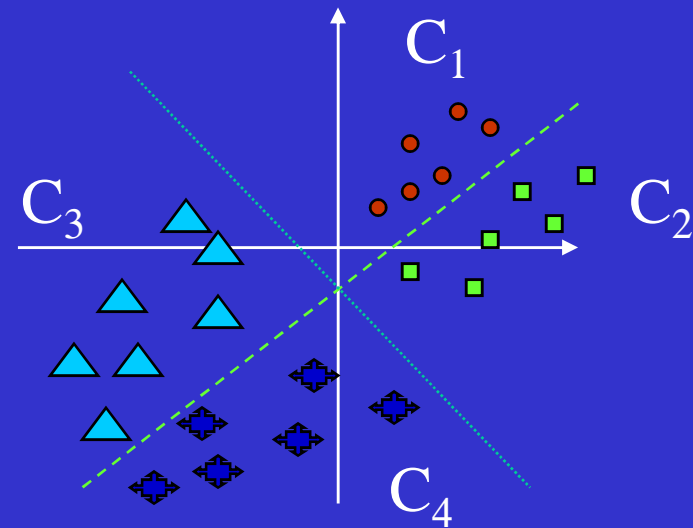


Table de codage

Ainsi pour effectuer l'apprentissage nous avons deux critères inspirés par la connaissance a priori :

1. Les quatre classes peuvent être séparées par deux hyperplans;
2. (C_1, C_2) est linéairement séparable de (C_3, C_4) et (C_1, C_3) est linéairement séparable de (C_2, C_4) .



Il est plus pratique de réaliser l'architecture précédente si nous pourrions se dispenser de **critère 2**. Il est nécessaire alors d'introduire un nouveau algorithme d'apprentissage basé sur une approche légèrement différente pour **éviter le besoin de connaître a priori la nature des hyperplans**.

Règles d'apprentissage

Les stratégies de modification des poids synaptiques sont dérivées des règles générales suivantes:

Règle simple :

- Règle de *Hebb*

Règles basée sur les techniques d'optimisation :

- Règle des *moindres carrés*
- Règle de *Widrow-Hoff* ou règle delta ou règle de gradient stochastique
- Règle de *Widrow-Hoff* généralisée ou règle delta généralisée

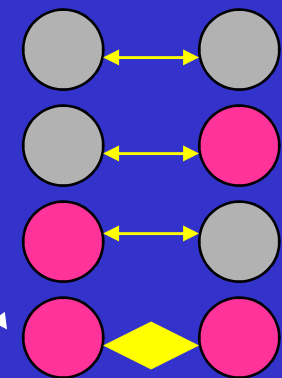
1. Règle de Hebb

Le poids des connexions entre deux neurones m et n élémentaires est renforcé si les deux neurones élémentaires sont activés simultanément.

Mécanisme d'adaptation :

$$\Delta \omega_{mn} = \omega_{mn}(t+dt) - \omega_{mn}(t) = \alpha S_m S_n$$

$\alpha > 0$ est une constante mesurant l'intensité de l'apprentissage.



Règle de Hebb

La règle de Hebb a la propriété de trouver automatiquement des corrélations entre neurones.

$$\Delta \omega_{mn} = \omega_{mn}(t+dt) - \omega_{mn}(t) = \alpha (d_s(t) - y_s(t)) x_s(t)$$

Où :

- $y_s(t)$ désigne la valeur calculée par le réseau sur la sortie s ,
- d_s correspond à la valeur désirée,
- α est le pas (ou coefficient) d'apprentissage servant à moduler l'impact de l'erreur sur la connexion.

Règles basées sur la technique d'optimisation

2. Règle des moindres carrés : Minimisation de l'erreur au sens de critère des moindres carrés pour un neurone donné

Étant donné

$\mathbf{X}=[\mathbf{X}^1 \ \mathbf{X}^2 \ \dots \ \mathbf{X}^P]$: une matrice : $((N+1) \times 1) \times P$

$e=1, 2, \dots, P$ où P : nombre d'exemples

$\mathbf{X}^e=[-1 \ x_1 \ x_2 \ \dots \ x_N]^T$: vecteur exemple e : $(N+1) \times 1$

$\mathbf{D}=[d^1 \ d^2 \ \dots \ d^P]^T$: vecteur sorties désirées : $(P \times 1)$

Où $d^e = d^e(\mathbf{X}^e)$

$\mathbf{W}^e=[\omega_0 \ \omega_1 \ \omega_2 \ \dots \ \omega_N]^T$: vecteur poids : $(N+1) \times 1$

Critère des moindres carrés : Minimiser
l'erreur totale

$$E(W) = \frac{1}{2} \sum_{e=1}^P (d^e - X^{eT} W^e)^2 = \frac{1}{2} \|D - X^T W\|^2$$

Cette méthode de minimisation revient à déterminer le vecteur de poids \mathbf{W}^* qui minimise la fonction de coût $E(\mathbf{W})$. Le calcul du vecteur gradient $\nabla(W)$ conduit à

$$\nabla(W) = -\alpha \frac{\partial E(W)}{\partial W} = \alpha (\mathbf{X}\mathbf{X}^T\mathbf{W} - \mathbf{X}\mathbf{D})$$

Pour $\nabla(W)=0$ nous obtenons

$$\mathbf{W}^* = (\mathbf{X}\mathbf{X}^T)^{-1} \mathbf{X}\mathbf{D}$$

Inconvenient de cette solution :

- Inversion $(\mathbf{X}\mathbf{X}^T)^{-1}$
- traitement simultané de toutes les informations : contraintes mémoire, difficultés numérique

Solution : Moindres carrées récursifs

Avantage :

- pas d'inversion
- Stockage minimale des informations à chaque itération.
- Le vecteur de poids s'effectue de façon récursive

- Le calcul de W_i s'effectue de façon récursive, ainsi à l'itération $i+1$ et en présentant l'exemple $e(i+1)$:

$$W_{i+1}^{e(i+1)} = W_i^{e(i)} + \alpha_{i+1} \chi_{i+1}^{e(i+1)} (\chi_{i+1}^{e(i+1)T} W_i^{e(i)} - D_{i+1}^{e(i+1)})$$

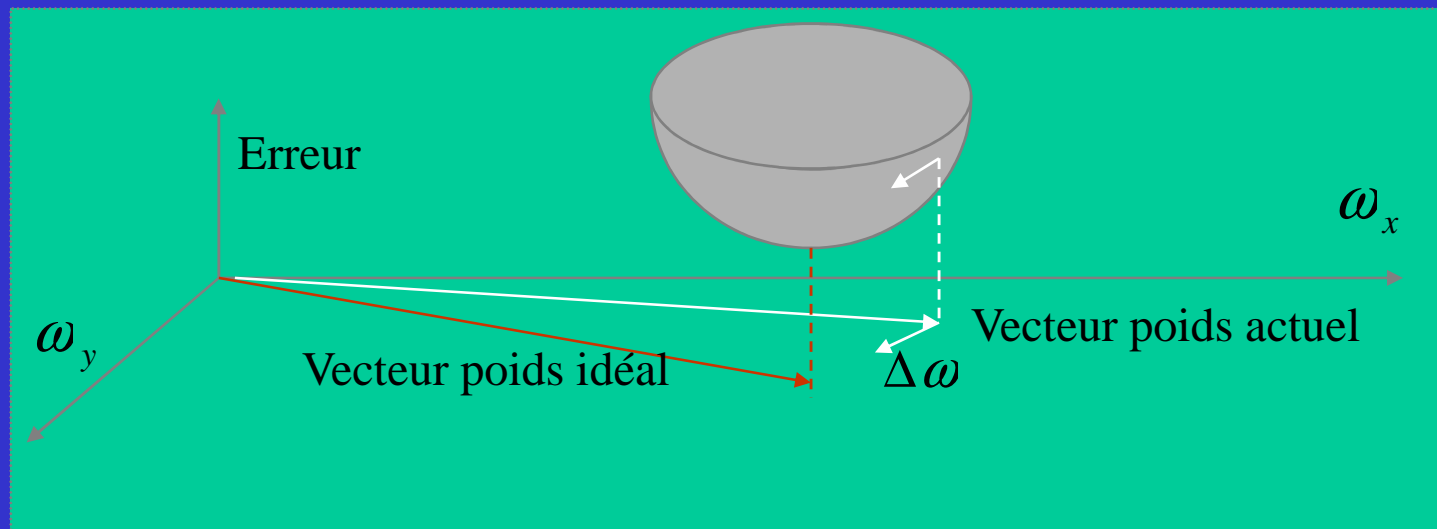
Où $\chi_{i+1}^{e(i+1)} = [X_1^{e(1)} \quad X_2^{e(2)} \quad \dots \quad X_i^{e(i+1)}]$ une matrice :

$((N+1) \times 1) \times (i+1)$ pour $i=0, 1, 2, 3, \dots, I_{\max}$

$W_i^{e(i)} = [\omega_0 \quad \omega_1 \quad \omega_2 \quad \dots \quad \omega_N]^T$: vecteur poids : $(N+1) \times 1$
 quand l'exemple e est présenté au réseau à l'itération i

$D_{i+1}^{e(i+1)} = [d^{e(1)} \quad d^{e(2)} \quad \dots \quad d^{e(i+1)}]^T$ vecteur $(i+1) \times 1$ de
 valeurs désirées pour les exemples présentés au réseau jusqu'à
 l'itération $i+1$.

On montre que l'erreur est une fonction quadratique du vecteur poids. Sa représentation graphique en fonction des vecteurs poids possibles est un **paraboloïde**. Le fond de ce paraboloïde correspond au vecteur poids idéal pour l'ensemble des entrées.



Minimisation de l'erreur par évolution du vecteur poids (de dimension 2).

Le constant α est une **mesure de la convergence** du vecteur poids vers la position d'erreur minimum:

- Une valeur proche de 1 provoque une convergence rapide mais risque aussi d'entraîner une oscillation autour du minimum.
- Une faible valeur de α évite l'oscillation mais ralentit le processus de convergence.

3. Minimisation de l'erreur au sens de gradient stochastique (règle de Widrow-Hoff, règle delta, ou règle de gradient stochastique) pour un neurone donné

Cette approche diffère sensiblement de l'approche précédente car la modification des poids est effectuée à chaque exemple présenté en minimisant un critère partiel pour le vecteur d'entrée \mathbf{X}^e avec $1 \leq e \leq P$.

Critère partiel pour le vecteur d'entrée X^e
à l'itération i :

$$E_i^e(W) = \frac{1}{2}(d_i^e (X_i^e) - s_i^e)^2$$

$$\text{où } s_i^e = a_i^e + b = X_i^{eT} W$$

a_i^e est l'activation du neurone pour l'exemple e
à l'itération i

Le vecteur gradient :

$$\nabla_i^e(W) = -\alpha \frac{\partial E_i^e(W)}{\partial W} = -\alpha (d_i^e (X_i^e) - X_i^{eT} W_i) X_i^e$$

en scalaire: $\frac{\partial E_i^e}{\partial \omega^j} = \frac{\partial E_i^e}{\partial s_i^e} \bullet \frac{\partial s_i^e}{\partial \omega^j} = - (d_i^e(x_i^e) - s_i^e) x_i^j$

$$1 \leq j \leq (N+1).$$

L'algorithme de Widrow-Hoff donne la loi d'adaptation des poids qui évoluent dans la direction opposée au vecteur gradient :

$$W_{i+1} = W_i + \alpha_{i+1} (d_{i+1}^e - X_{i+1}^{eT} W_i) X_{i+1}^e$$

Remarque: si $\alpha^i = 1/i$, cette règle converge vers la règle des moindres carrés.

La différence entre la règle delta et la règle de perceptron est que l'erreur dans la deuxième n'étendra jamais la valeur nulle. Ainsi la possibilité de «pas de changement de poids » dans l'algorithme général d'apprentissage n'est plus satisfaite : il y 'aura toujours une certaine adaptation de poids.

Pour effectuer la descente de gradient, l'erreur doit être une fonction continue de poids et dérivable à chaque point. Deux solutions:

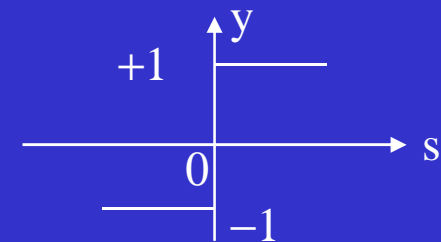
1. Utiliser l'activation (a) à la place de la sortie (y).
2. Utiliser la sortie (y) avec une fonction d'activation (f) continue et dérivable (sigmoïde par exemple).

La première solution a été adoptée par Widrow et Hoff [Widrow-Hoff 60] où la sortie y est définie entre $+1$ et -1 au lieu de 1 et 0 .

Le réseau obtenu est appelé:

Adaptive Linear Elements (AdaLinE)

[Widrow-Hoff 60], [Widrow et al 87], [Widrow and Stearns 85].



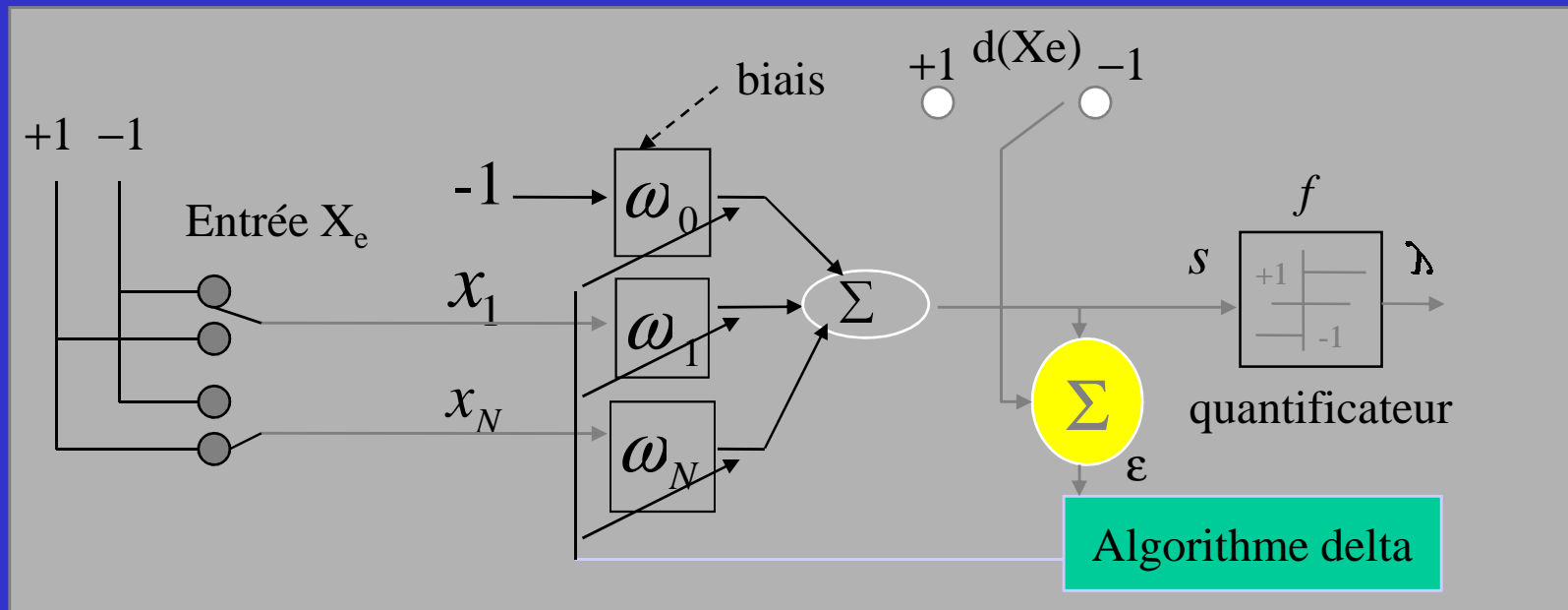
Adaptive Linear Elements (AdaLinE)

C'est une généralisation très importante de la règle de Widrow-Hoff.

La principale différence fonctionnelle avec l'apprentissage par la règle de perceptron est la manière par laquelle la sortie du système est utilisée dans la règle d'apprentissage.

- La règle d'apprentissage de perceptron utilise la sortie de la fonction d'activation $(-1, +1)$ pour l'apprentissage.
- La règle delta utilise le sommateur de réseau: $s=a+b$.

Schéma synoptique d'une unité d'AdaLinE



Un réseau à plusieurs sorties parallèles peut être construit en utilisant plusieurs modules unitaires AdaLinE.

L'objectif de l'AdaLinE est de reproduire une valeur cible $d(\mathbf{X}^e)$ quand \mathbf{X}^e est présenté à l'entrée.

Objectif : déterminer W de tel manière que la réponse à une entrée donnée est correcte pour un grand nombre d'entrées choisies arbitrairement dans un ensemble de test.

Remarques :

1. La fonction de coût (critère) étant quadratique par rapport aux poids synaptiques. On démontre qu'il n'existe qu'un seul minimum (l'unicité de la solution est assurée).

2. Si l'ensemble d'apprentissage est parfaitement connu, il est possible de mettre en œuvre les nombreuses méthodes puissantes de programmation non linéaire:
- Méthodes directes : Rosenbrock, Hooke et Jeeves, Nelder et Mead.
 - Méthodes à base de gradient conjugué convergeant en n itérations.
 - Méthodes de second ordre de Newton Raphson utilisant la matrice Hessienne des dérivées secondes qui a la propriété de converger en une itération.

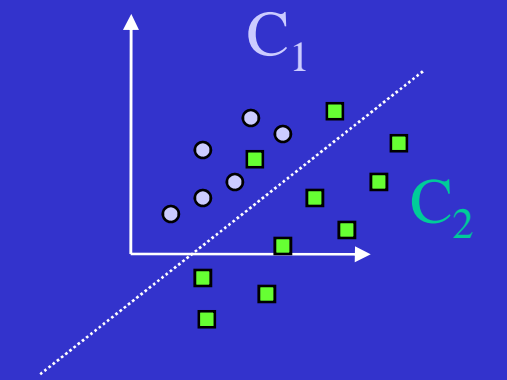
3. La règle delta est formellement équivalente à la règle de perceptron (voir partie 4). Cependant, la seconde est obtenue à partir de point de départ théorique.

La règle de perceptron a été obtenue en considérant la manipulation de l'hyperplan tandis que la règle delta est obtenue par la descente de gradient sur l'erreur quadratique.

4. La règle delta modifie les poids quelques soit les signes des entrées ou des cibles et quelques soit le type des entrées et des sorties binaire ou continue.
5. Un inconvénient de la règle delta est la vitesse de convergence lente.

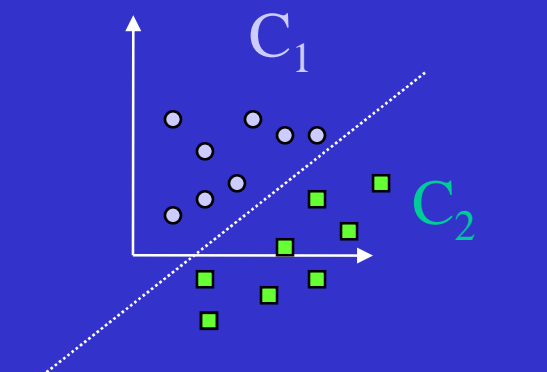
Performances des règles précédentes

- Si les deux classes C_1 et C_2 ne sont pas séparables, les règles précédentes convergent vers une solution optimale qui séparera linéairement l'espace des formes.
- Si C_1 et C_2 sont linéairement séparables, les résultats dépendront du rapport du nombre P_1 des exemples utilisés de C_1 et du nombre P_2 des exemples utilisés de C_2 .

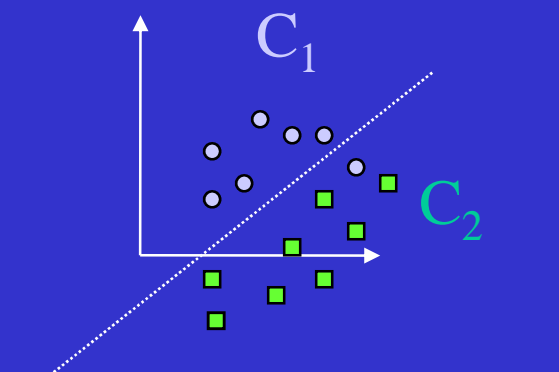


Classes non séparables

06/11/2011



Classes séparables $P_1=P_2$



Classes séparables $P_1 < P_2$

37

3. Règle delta généralisée pour un neurone donné

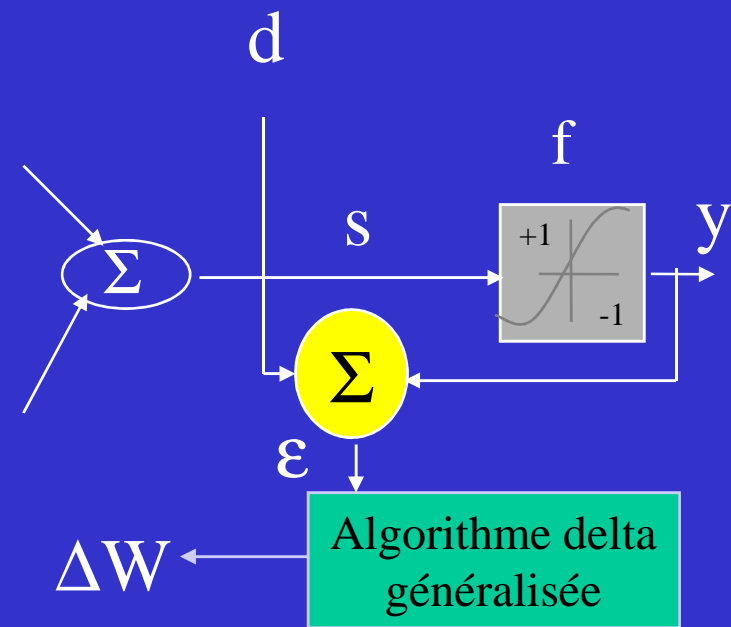
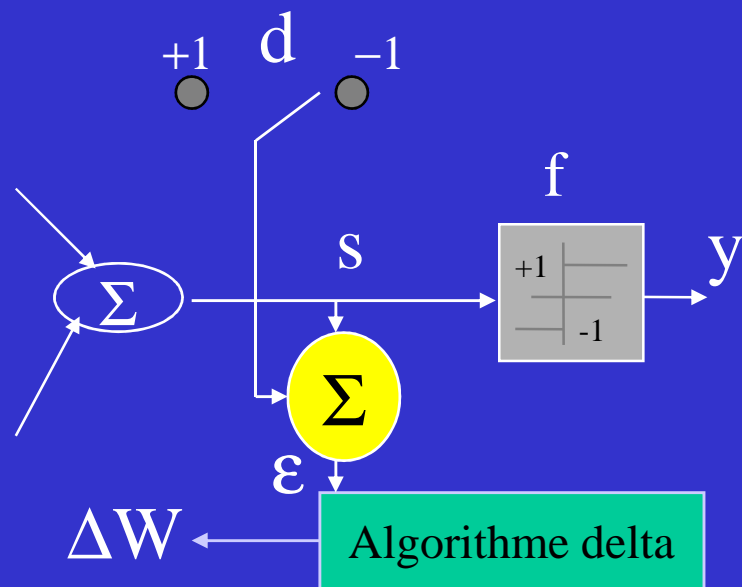
Dans ce cas, la minimisation est effectuée au sens des moindres carrés en utilisant une fonction d'activation dérivable, sigmoïde par exemple.

Dans ce cas, à l'itération i

$$s_i = \sum_{j=0}^N \omega_{i-1}^j x_i^j, \quad y_i = f(s_i)$$

où x_i^j une composantes i de \mathbf{X}^e . La règle delta généralisée minimise le critère partiel $E_i(\mathbf{W})$ à l'itération i

$$E_i(W) = \frac{1}{2}(d_i - y_i)^2 = \frac{1}{2}(d_i - f(\sum_{j=0}^N \omega_{i-1}^j x_i^j))^2$$



La minimisation du critère conduit à calculer :

$$\nabla (W) = -\alpha \frac{\partial E(W)}{\partial W}$$

Nous obtenons alors

$$W_{i+1} = W_i + \alpha_{i+1} (d_i - y_{i+1}) f' (s_{i+1}) x_i^j$$

$$y_i = f(s_i) \text{ et } f' (s_i) = \frac{\partial f}{\partial s_i}$$

Remarque: la règle de Widrow-Hoff est un cas particulier de la règle Delta généralisée puisque dans ce cas $f(s_i) = s_i$ avec $f'(s_i) = 1$.

Conclusion sur les différentes règles liées aux réseaux de classification linéaire : perceptron et AdalinEs [Zwin 95]

Règle de Widrow-Hoff (règle Delta)

- L'algorithme d'apprentissage converge toujours vers la solution unique définie par la méthode de moindres carrés.
- Si les deux classes de l'apprentissage sont linéairement séparables, la ligne de séparation ne réalise pas nécessairement leur séparation : le résultat dépend principalement de la distribution des exemples choisis dans les deux classes.

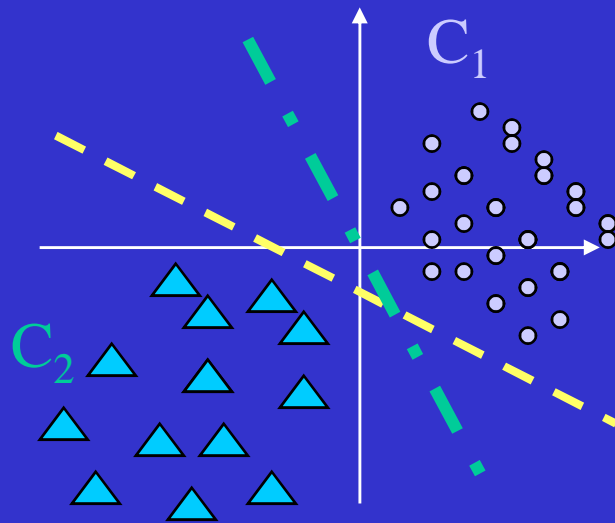
Règle Delta généralisée

- L'algorithme d'apprentissage converge toujours vers une solution. Si les deux classes de l'apprentissage sont linéairement séparables, on obtient une infinité de solutions les séparant.
- En général, la solution est plus optimale que celle fournie par la règle Delta.

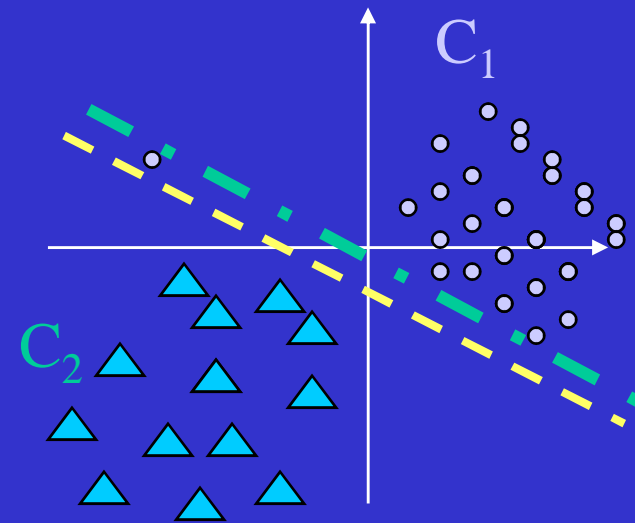
Règle du perceptron

- L'algorithme d'apprentissage ne converge pas si les deux classes de l'apprentissage ne sont pas linéairement séparables,
- La solution est proche de celle de la règle Delta mais elle est en général de qualité moins bonne.

Règles perceptron-Delta

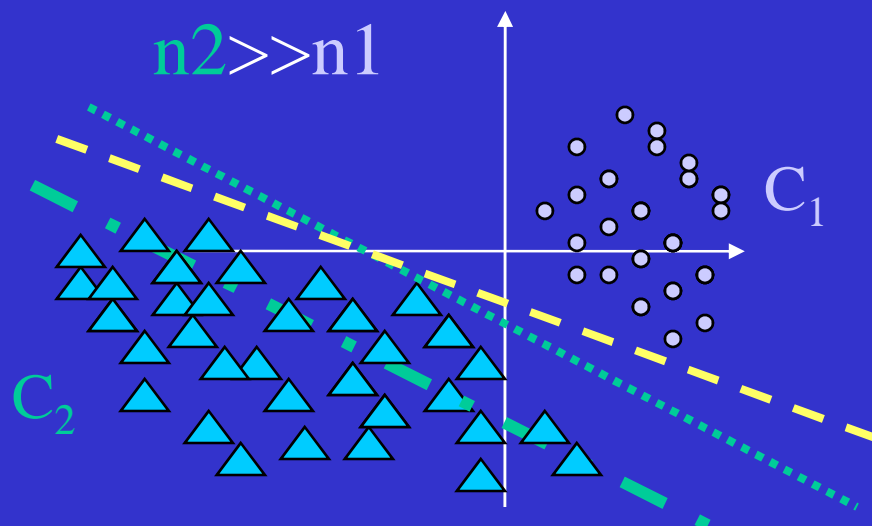


(a) 2 classes linéairement séparables
- - - Perceptron - - - Delta



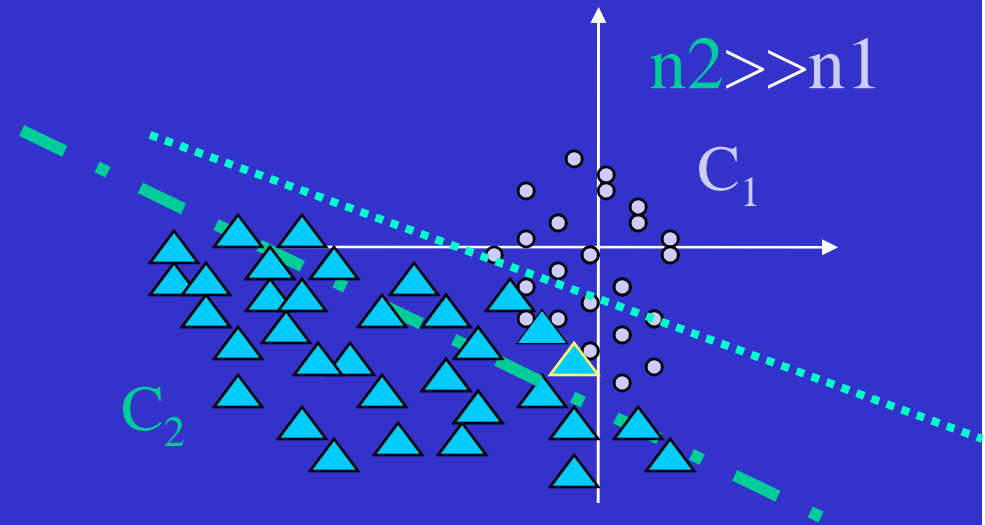
Classe (•) possède un individu isolé
La règle de perceptron est plus efficace

Règles perceptron-Delta-Delta généralisé



(a) 2 classes linéairement séparables. Seuls les règles perceptron et Delta généralisée sont satisfaisantes.

— — — — — Perceptron - . - . - Delta
..... Delta généralisé



(b) 2 Classes non linéairement séparables. L'algorithme de perceptron ne converge pas. Les autres sont les plus satisfaisantes

Algorithme d'apprentissage

- Initialisation : Fixer à $t=0$, les valeurs des w_i et w_0 une valeur aléatoire proche de 0.
- Présentation : Présenter un exemple $X_e(t)$ à apprendre au réseau, $d_e(t)$ est la sortie désirée, $e=1,2, \dots, P$.

- **Propagation** : Propager le signal de la couche d'association vers la couche de décision et calculer la réponse $y(t+1)=f(\mathbf{X}^T\mathbf{W})$.
- Présenter un nouvel exemple jusqu'à stabilisation de \mathbf{W} pour tous les exemples à apprendre.

Ordinateurs conventionnels contre Mémoires associatives

- **Ordinateurs conventionnels (machine de von Neumann)**

Un traitement séquentielle des informations qui sont codées sous forme binaire et rangées dans des mémoires accessibles par leur adresse indépendamment de leur contenu.

L'ordinateur effectue d'une façon répétitive le cycle suivant d'événements :

1. Chercher en mémoire une instruction,
2. Chercher en mémoire les données utilisées par cette instruction ,
3. Exécuter l'instruction (traiter les données),
4. Stocker les résultats dans la mémoire,
5. Revenir à l'étape 1.

Quels types de problèmes peut résoudre facilement cet ordinateur?

Formaliser beaucoup de problèmes en terme d'algorithme. Cet algorithme peut ensuite être décomposé en un ensemble d'instructions plus simples qui, à leur tour, peuvent être décomposés éventuellement, en des instructions exécutées par l'ordinateur.

En particulier, il est possible de traiter des chaînes de symboles qui obéissent à des règles d'un certain système formel et qui peuvent être interprétées (par l'homme) comme des « idées » ou des « concepts ». C'était le souhait du programme de l'IA que toutes les connaissances puissent être formalisés de cette façon : elles pourraient être alors réduites à des manipulations de symboles selon des règles et ces manipulations seraient implémentées sur la machine de von Neumann (ordinateur conventionnel).

Caractéristiques essentielles :

- Les séries exactes d'étapes nécessaires pour exécuter un algorithme ainsi que le type et le format exact de données (données non bruitées), doivent être fournis a priori à l'ordinateur (programme de l'ordinateur),

- Une défaillance d'une partie de la mémoire peut provoquer l'interruption de la chaîne de traitement séquentiel des opérations.
- Il existe une correspondance claire entre les objets sémantiques utilisés (nombres, mots, accès aux base de données, etc.) et le matériel de l'ordinateur.

Le succès de l'**approche symbolique** en IA dépend directement des conséquences du premier point qui suppose que nous pouvons trouver un algorithme qui décrit la solution du problème ce qui n'est pas le cas pour beaucoup d'applications pratiques où les informations sont partiellement connues (par exemple une partie d'image à reconnaître).

Caractéristiques des réseaux de neurones comparées à l'ordinateurs conventionnels

- La mémoire est distribuée et tout ou partie des poids de connexion du réseau constitue la mémoire du réseau.

Ainsi, un exemple d'apprentissage sera mémorisé (et non pas programmé) par un sous-ensemble particulier des cellules qui contient également la mémoire de certains autres exemples de la base d'apprentissage.

- Ils sont **robustes en présence du bruit** à leurs l'entrée.
- Ils sont **robustes en présence d'une défaillance matérielle**: un changement du poids peut éventuellement affecter la sortie pour certains exemples d'entrée.

- Des concepts de haut niveau seront représentés sous forme d'activités à travers plusieurs noyaux au lieu de les représenter sous forme de contenus d'une petite partie de la mémoire de l'ordinateur.

- Les réseaux peuvent **généraliser** à des exemples non présentés dans la base d'apprentissage.
- Les réseaux sont intéressants pour des tâches « perceptuelles » et associatifs. Ces tâches sont difficiles à traiter par l'approche symbolique.

Mémoires associatives

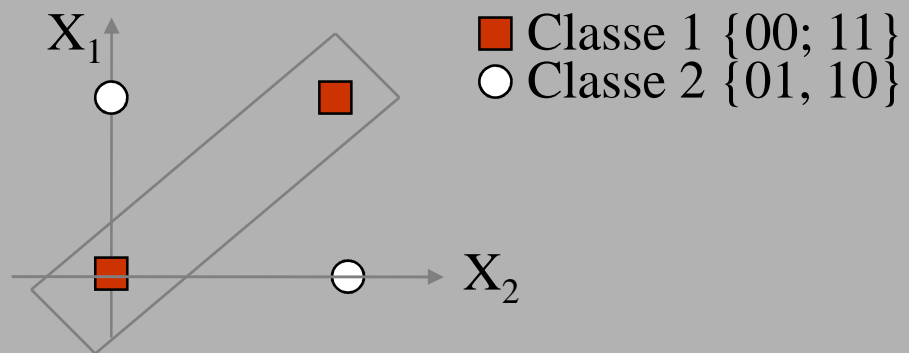
Deux types:

Mémoires auto-associatives : elles reproduisent en sortie, l'image apprise de l'entrée à partir d'un exemple incomplet ou bruité.

Mémoires hétéro-associatives : elles apprennent une relation entre un exemple présenté et proche de l'exemple appris, il restituera la sortie désirée .

Limitations du perceptron

Du fait de l'utilisation d'un **automate linéaire à seuil** sur la couche de décision, Minsky et Papert [Minsky 69] ont souligné que le perceptron ne peut séparer des exemples non linéairement séparables. Ainsi, il ne peut pas réaliser la fonction booléenne « ou exclusif XOR », et ce, quel que soient les poids de ses connexions. De plus, l'existence d'un hyperplan séparateur est une condition **et nécessaire et suffisante** à la convergence de l'algorithme d'apprentissage.

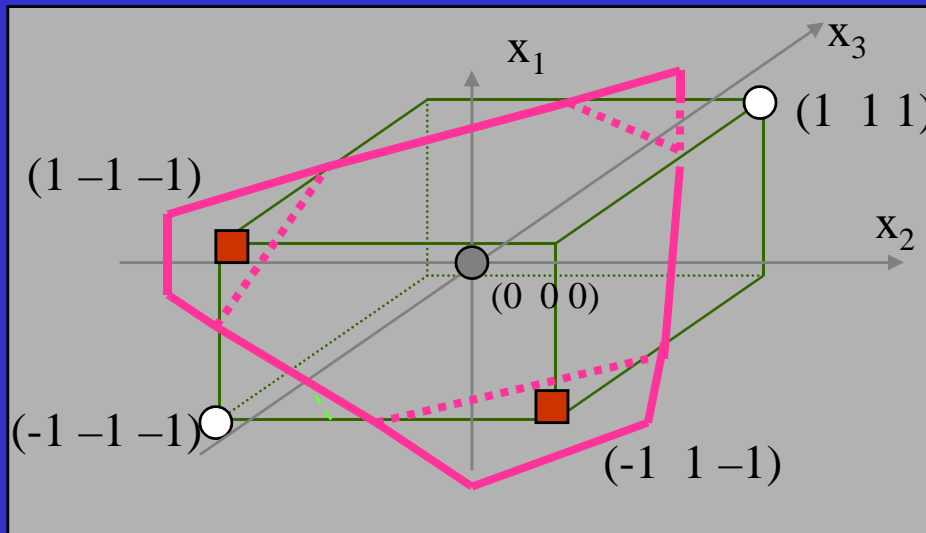


Le problème de XOR.

Afin de lever ces limitations il faut envisager des couches intermédiaires entre la couche d'association et la couche de décision : **Architecture multi-couches.**

Minsky et Papert [Minsky 69] ont montrés que 2 couches statiques non bouclées (*Feed-Forward Network*) peut éviter beaucoup de restrictions mais il n'ont pas présentés une solution au problème d'apprentissage des poids synaptiques des couches cachées.

Solution : Réseau à deux couches



x_1	x_2	$x_1 \text{ ET } x_2$	d
-1	-1	-1	-1
-1	1	-1	1
1	-1	-1	1
1	1	1	-1

Extension en trois dimensions du problème XOR (la troisième dimension étant obtenue comme le ET des deux premières)

Réseau à deux couches pour le OU Exclusive

$$\omega_0 = 0.5$$

$-\omega_0$

entrées

x_1

x_2

$$\omega_i = 1, \quad i = 1, 2$$

$$\omega_{12} = -2$$

