Modèles connexionnistes de l'apprentissage

Y. Le Cun, Françoise Fogelman-Soulié

MODELES CONNEXIONNISTES DE L'APPRENTISSAGE.

Yann Le Cun:
Groupe IAAI, ESIEE,
89 rue Falguière,
75015 PARIS

Françoise Fogelman-Soulié:
LIA, EHEI, Univ. Paris V,
45 rue des Saint Pères,
75006 PARIS

et
Laboratoire de Dynamique des Réseaux,
c/o CESTA
1 rue Descartes 75005 PARIS

1 - Introduction

L'apprentissage est à la fois l'un des problèmes les plus anciens et les moins bien résolus de l'Intelligence Artificielle. Ceci pourrait expliquer le nombre et la diversité des méthodes mises en oeuvre pour le résoudre. De nombreuses méthodes actuelles font référence à des travaux datant du début des années soixante [5]. Une nouvelle classe de méthodes, dites *connexionnistes* apparue récemment, n'échappe pas à la règle, et est souvent présentée comme l'héritière d'idées ayant germé à la fin des années cinquante. Elle s'inspire en partie des modèles d'apprentissage statistique dont le représentant le plus simple est le classifieur linéaire. De nombreux travaux on montré que les capacités d'un classifieur linéaire seul sont malheureusement très limitées [6,7].

Cette limitation est à la base de l'échec du perceptron proposé par Rosenblatt [30], étudié par Nilsson [27], Minsky & Papert [25]. Le perceptron est une machine adaptative basée sur un classifieur linéaire. Ces mésaventures du perceptron, en même temps que le développement des modèles basés sur la logique, ont conduit à l'abandon relatif des méthodes statistiques en IA. Ceci s'explique par le fait que le problème essentiel de l'IA aujourd'hui est celui de la représentation des connaissances, problème

que les méthodes statistiques n'abordaient pas du tout. les connaissances dans un modèle statistique sont simplement représentées par un ensemble de paramètres qu'il s'agit de déterminer. Les performances de ces modèles sont donc limitées par la classe de fonctions, en général très restreinte, que les paramètres peuvent engendrer.

Le problème principal de l'apprentissage à partir d'exemples est celui de l'extraction d'une famille de prédicats caractéristiques de la base d'exemples. On considère généralement que l'apprentissage n'est possible que si l'on possède suffisamment de connaissances a-priori sur le domaine [24], ce qui se traduit, pour le programmeur, par une phase initiale de détermination de la famille de représentations internes signifiantes. Les modèles connexionnistes proposés récemment tentent de résoudre ce problème de génération de concepts à partir d'exemples, et ce faisant,s'affranchissent des limitations habituelles des classifieurs statistiques.

Les modèles connexionnistes utilisent un mode de représentation des connaissances sous forme de réseaux d'éléments de calcul le plus simple possible (processeurs, cellules, automates) dont la capacité à traiter l'information réside dans les connexions entre ces unités. En ce sens le connexionnisme est à rapprocher des premiers travaux inspirés du cerveau. L'étude de ces systèmes profite de travaux récents provenant de la physique (théorie des verres de spins, mécanique statistique)[18][28], des mathématiques discrètes (théorie des automates, itérations discrètes)[11,12], de l'informatique (calcul parallèle)... concernant les comportements collectifs des réseaux d'automates.

Deux caractéristiques essentielles sont sans doute à la base du renouveau de ce domaine:

- le parallélisme massif, qui permet d'envisager la réalisation de processeurs dédiés, en particulier pour les tâches de perception.
- les capacités d'apprentissage et de mémoire associative.

Les travaux les plus récents [20,21,31,34] montrent qu'on est aujourd'hui en mesure de concevoir des réseaux capables d'apprendre des tâches "difficiles", i.e. qui dépassent les capacités habituelles de ce genre de modèle. Le meilleur exemple d'algorithme d'apprentissage ayant ces propriétés est le modèle dit de RETRO-PROPAGATION DE GRADIENT, qui concerne des réseaux possédant des unités "cachées", n'ayant pas d'interaction directe avec l'extérieur, et dont le rôle est de synthétiser les prédicats intermédiaires permettant de décomposer la tâche en étapes élémentaires "linéaires".

Entraîné sur une base d'exemples, un tel réseau modifie les poids de ses connexions, pour coder de manière COMPACTE ET DISTRIBUEE, la structure des exemples, ce qui lui permet ensuite d'appliquer à de nouvelles données les représentations internes qu'il s'est ainsi construites et par conséquent de GENERALISER. Le type de généralisation effectué dépend essentiellement de l'architecture du réseau, de la configuration initiale des poids, ainsi que de l'ordre de présentation des exemples.

Plusieurs sytèmes utilisant cette procédure ont été réalisés dont, par exemple, celui mis au point par Sejnowski [34] pour l'apprentissage de la correspondance graphème-phonème, dont on sait que c'est un problème complexe en anglais: NetTalk est un réseau multi-couche qui apprend à lire un texte à haute voix, à partir d'une base d'exemples restreinte (une page de 500 mots et sa transcription phonétique). Sur un texte nouveau, différent de celui ayant servi à l'apprentissage, les performances du réseau sont comparables à celle d'un système programmé "à la main". Ces méthodes peuvent être considérées à la fois comme statistiques et structurelles, selon que l'on se place au niveau de l'élément de base ou du réseau entier. Ceci, ajouté au fait qu'elles utilisent une métrique sur les fonctions générées, permet de les comparer à d'autres techniques [29]

Nous présentons d'abord (§2) les méthodes d'apprentissage, et leurs limites, s'appliquant aux éléments de base des modèles connexionnistes, à savoir les automates à seuil et leurs dérivés. Au §3, nous décrivons les modèles de mémoires associatives par réseaux d'automates, puis aux §4 et 5 présentons les réseaux multi-couches et l' algorithme de rétro-propagation de gradient, que nous illustrons au §6 par quelques exemples.

2 - Machines adaptatives simples

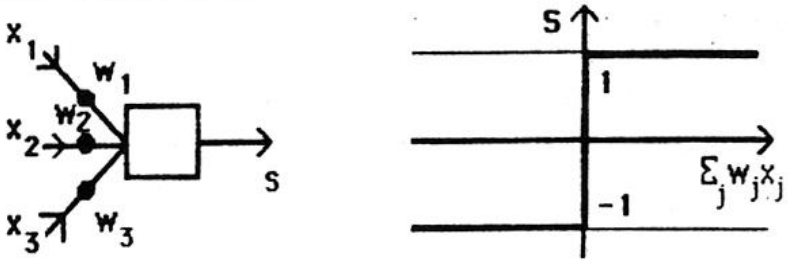
Les premiers travaux sur les machines adaptatives sont issus du modèle du NEURONE FORMEL introduit par McCulloch et Pitts [23]. Un neurone formel (cf Fig 1) est un élément de calcul -un automate à seuil- possédant n entrées représentées par un vecteur **X** et dont la sortie S est calculée par la fonction de changement d'états suivante:

$$S = \begin{cases} 1 & \text{si } \sum_j W_j X_j \geq 0 \\ -1 & \text{sinon} \end{cases}$$

ce qu'on notera: $S = 1[\sum_j W_j X_j]$. Les coefficients W_j s'appellent les POIDS SYNAPTIQUES. En notant **W** le vecteur des poids transposé, on écrira:

$$S = 1[WX] \qquad (1)$$

Figure 1: Automate à seuil.



On considèrera en général que l'entrée n, X_n, est constamment égale à +1, le coefficient $-W_n$ correspondant s'appelle classiquement le SEUIL de l'automate.

L'équation $\sum_j W_j X_j = 0$ est l'équation d'un hyperplan dans l'espace des vecteurs d'entrée **X**, le neurone formel réalise donc une discrimination en deux classes des vecteurs d'entrée Les CLASSIFIEURS LINEAIRES, classiquement utilisés en reconnaissance des formes et classification automatique [6,7] sont basés sur ces éléments. De nombreuses procédures d'apprentissage ont eté proposées pour déterminer les poids convenables réalisant une classification donnée. Ces méthodes sont généralement basées sur la minimisation d'une fonction de coût.

• Nous décrivons tout d'abord la procédure de Widrow Hoff [36,7], qui utilise un critère d'erreur quadratique. Soit un ensemble de vecteurs $\mathbf{X}_k$, k=1,...,m, représentant des formes à classifier en deux classes $\mathcal{C}^+$ et $\mathcal{C}^-$. On notera Y_k la sortie associée au vecteur $\mathbf{X}_k$ avec $Y_k=+1$ si $\mathbf{X}_k$ est dans $\mathcal{C}^+$ et $Y_k=-1$ si $\mathbf{X}_k$ est dans $\mathcal{C}^-$. La fonction de coût choisie est alors:

$$C(\mathbf{W}) = 1/m \sum_k (\mathbf{W}\mathbf{X}_k - Y_k)^2 \qquad (2)$$

Cette fonction peut être minimisée en utilisant les procédures classiques de minimisation de critère quadratique, largement développées dans la littérature [19]. Les plus simples sont des procédures de gradient: l'idée de base en est simple, le gradient de la fonction C «pointe» dans la direction des C croissants. On peut donc écrire une procédure itérative qui à chaque étape modifie les poids dans la direction opposée au gradient:

$$\mathbf{W}^{h+1} = \mathbf{W}^h - \varepsilon_h \, \mathrm{GRAD}[C(\mathbf{W}^h)] \qquad (3)$$

Le terme $\mathrm{GRAD}[C(\mathbf{W}^h)]$ est le vecteur ligne du gradient de C par rapport à $\mathbf{W}^h$. ε_h est un paramètre positif qui détermine la longueur du pas effectué en direction du minimum à chaque itération. Avec la fonction C choisie, on a donc la procédure:

$$\mathbf{W}^{h+1} = \mathbf{W}^h - \varepsilon_h \sum_k (\mathbf{W}\mathbf{X}_k - Y_k)\mathbf{X}_k^t \qquad (4)$$

où $\mathbf{X}_k^t$ note la transposée de $\mathbf{X}_k$. Cette procédure converge si ε_h est bien choisi (assez petit).

L'algorithme du GRADIENT STOCHASTIQUE est une version «bruitée» de l'algorithme précédent. On présente itérativement les couples (X_1,Y_1), ...,(X_m,Y_m), (X_1,Y_1),... ce qui constitue une séquence temporelle qu'on notera, de façon évidente, (X^h,Y^h) et on modifie les poids, forme par forme, selon la loi dite de WIDROW HOFF:

$$W^{h+1} = W^h - \varepsilon_h (W^h X^h - Y^h)(X^h)t \qquad (5)$$

où h est un indice temporel. Lorsque qu'aucune solution exacte n'existe, la suite des valeurs de $C(W^h)$ fluctue autour du minimum, avec des amplitudes d'autant plus grandes que ε_h est grand.

• D'autres fonctions de coût ont été proposées, par exemple le critère du PERCEPTRON [7,26]:

$$C(W) = -1/2 \sum_k Y_k W X_k$$

où la somme porte uniquement sur les formes MAL CLASSEES. La règle d'apprentissage s'écrit alors:

$$W^{h+1} = W^h + \varepsilon_h Y^h(X^h)t \qquad \text{si } X^h \text{ est mal classée}$$

$$W^{h+1} = W^h \qquad \text{sinon} \qquad (6)$$

qui peut s'écrire sous la forme de la règle de Widrow Hoff en introduisant la variable S^h qui est la sortie produite pour la forme X^h:

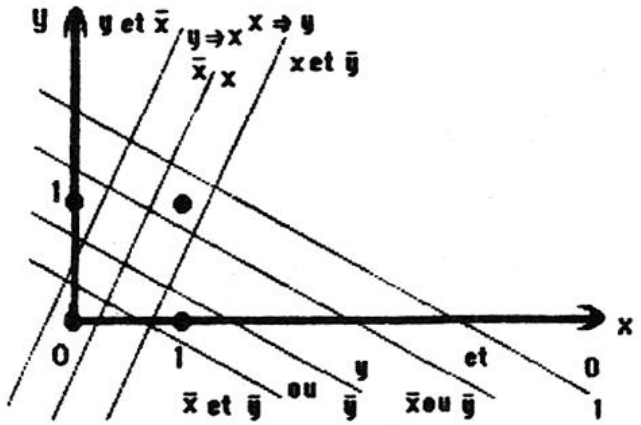
$$W^{h+1} = W^h - \varepsilon_h (S^h - Y^h)(X^h)t \qquad (7)$$

Les capacités d'apprentissage des systèmes d'automates à seuil simples sont liées aux limitations intrinsèques des séparateurs linéaires. En effet, seules les classes linéairement séparables -i.e. séparables par un hyperplan- peuvent être discriminées par de telles méthodes. Or, on sait [6] que la probabilité que m vecteurs de dimension n soient linéairement séparables décroît très rapidement dès que m dépasse 2n. *Ceci implique qu'aucunetâche «intéressante»ne peut être apprisepar un automateà seuil* .Ainsi, il est facile de voir que sur les 16 fonctions booléennes de 2 variables, 14 sont «calculables» par un automate à seuil (cf Fig 2) et les deux autres -le OU EXCLUSIF et l'EQUIVALENCE- ne le sont pas. De plus, la

proportion de fonctions booléennes linéairement séparables de n variables décroît exponentiellement avec n.

Figure 2: Représentation des fonctions booléennes de 2 variables par automate à seuil.

Sur cette figure, on a représenté les fonctions booléennes de 2 variables: toute fonction à valeurs dans {0,1} est équivalente à une fonction à valeurs dans {-1,1} en échangeant les rôles de 0 et -1. Par exemple, la fonction **et** est représentée par la fonction à seuil x+y-3/2 (1+1-3/2 > 0 → **et**(1,1)= 1).



On retrouve ici les limites théoriques du PERCEPTRON [30,27,25], l'une des premières machines adaptatives introduit en 1957 par Rosenblatt. Un perceptron (cf Fig 3) est une machine constituée d'un automate à seuil -l'unité de décision-, prenant ses entrées sur une série d'éléments -les unités d'association- qui extraient les traits caractéristiques des formes présentées à la machine (sur une «rétine»). Il est important de remarquer que les unités d'association sont fixées dès le départ et ne seront *jamais modifiables par l'apprentissage*.Les capacités du perceptron sont donc exactement les mêmes que celles des automates à seuil, sauf si les unités d'association ont été correctement choisies «à la main» (par exemple: la

fonction OU EXCLUSIF peut ainsi être calculée par un perceptron, avec 3 unités d'association convenables, fig4).

Figure 3: Perceptron

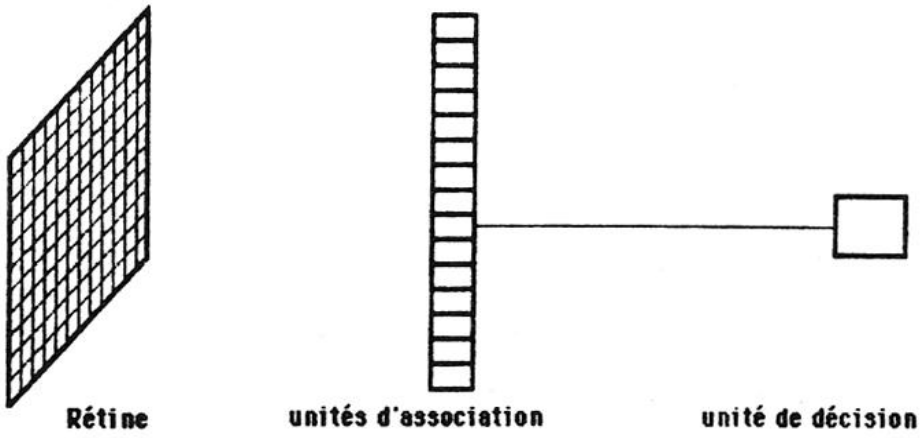
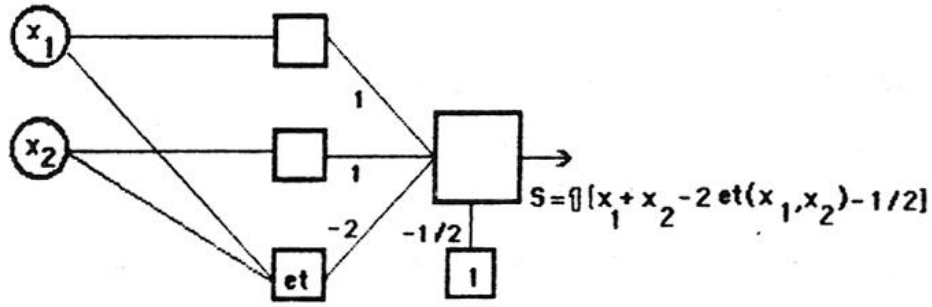


Figure 4: Calcul du OU EXCLUSIF par un perceptron



3 - Réseaux d'automates

Des modèles voisins ont été proposés, QUI NE DEPASSENT PAS ces limitations, mais permettent cependant des performances intéressantes. Il s'agit essentiellement des travaux sur les mémoires associatives développés par de nombreux auteurs [1,26,19,18], ... Tous ces modèles utilisent un RESEAU d'automates, c'est à dire une série d'unités interconnectées, et non plus une seule unité comme précédemment. Les règles de changement d'états de

chacun des automates induisent une DYNAMIQUE globale sur le réseau, qui peut être utilisée pour la représentation de mémoires associatives distribuées. Le modèle utilisé est alors une extension multi dimensionnelle des modèles présentés au §2.

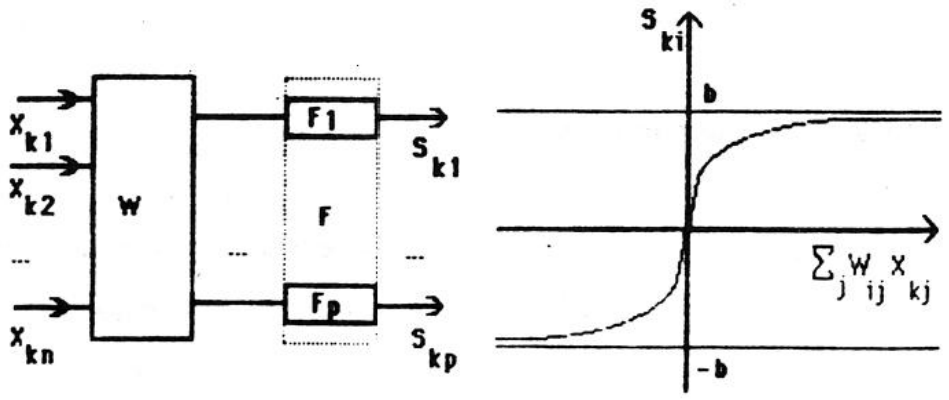
Soit $X_1,\ldots, X_m$ des formes (vecteurs) qu'on désire ASSOCIER à des formes $Y_1,\ldots, Y_m$ (on parlera d'AUTO ASSOCIATION si $X_k=Y_k$ pour tout k, d'HETERO ASSOCIATION sinon). Les X_k constituent les entrées des automates du réseau et les Y_k en sont les sorties désirées. Le réseau, en présence de l'entrée X_k, calcule la sortie S_k par une fonction qui est composée d'une transformation linéaire W éventuellement suivie d'une transformation NON-LINEAIRE F portant sur chaque composante indépendamment. F peut par exemple consister en un seuillage de chaque coordonnée (cf Fig 5):

$$S_k = F[WX_k] \qquad (8)$$

où W est la matrice dont la ligne i est le vecteur des poids associé au i-ème automate.

Figure 5: Schéma d'une mémoire associative.

La figure représente une mémoire associative permettant de réaliser l'association de formes X_k, à n composantes, à des formes Y_k à p composantes.La fonction F (à droite) est une fonction sigmoïde.



Associer les Y_k aux X_k consiste à déterminer la matrice **W** qui satisfasse le système suivant:

$$Y_k = S_k, \text{ pour tout k.}$$

Dans un premier temps, on résoudra, pour simplifier, le système:

$$Y_k = WX_k, \text{ pour tout k.} \qquad (9)$$

La théorie de la pseudo-inverse [13] fournit une solution à cette équation:

$$W = YX^+ + Z(I - XX^+) \qquad (10)$$

où **X** et **Y** sont les matrices dont les colonnes sont respectivement les X_k et les Y_k , X^+ représente la pseudo-inverse de **X**, et **Z** est une matrice arbitraire. La pseudo-inverse est une généralisation de la matrice inverse au cas des matrices non inversibles (non carrées, singulières). Dans le cas où les colonnes de **X** sont linéairement indépendantes: $X^+=(X^tX)^{-1}X^t$. Une solution particulière de (9) peut être obtenue en minimisant le critère d'erreur quadratique $\|Y-WX\|^2$, ce qui fournit une méthode de calcul à l'aide de la procédure de Widrow-Hoff.

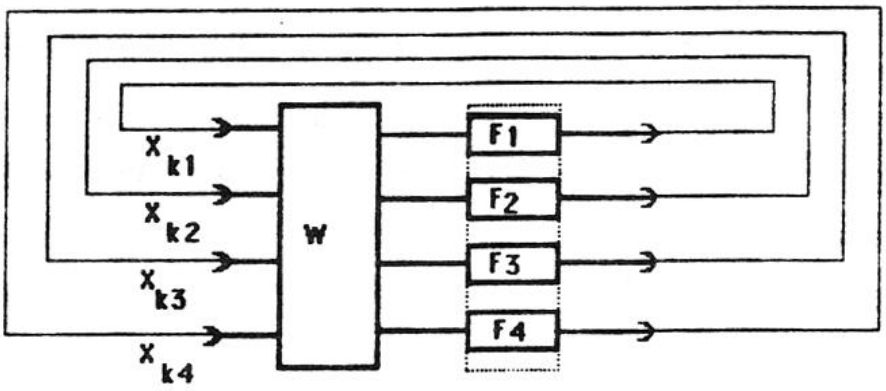
La matrice calculée par (10) possède la propriété de «retrouver» les formes Y_k à partir des X_k -elle est calculée pour ça!- et aussi à partir de versions bruitées ou incomplètes des X_k. Cette propriété est largement due au fait que la matrice XX^+ est la matrice de la projection sur le sous espace vectoriel engendré par les vecteurs X_k. La matrice **Z** doit être choisie afin de tenir compte du type de bruit considéré. Un bruit additif, gaussien, décorrélé, de moyenne nulle correspond à **Z=0** [10].

Les formes à mémoriser sont «codées» dans les connexions du réseau de manière distribuée, chaque pondération contenant une partie de l'information nécessaire à la mémorisation de chaque forme.

Une technique de restauration particulière, ayant fait l'objet de nombreuses études [18,19,1,26,3,8-12,35], est utilisée dans le cas de l'auto-association de vecteurs binaires. Sachant que la transformation effectuée par le réseau tend à réduire le bruit, l'idée est de réitérer le processus, après un

seuillage, jusqu'à élimination totale du bruit. Le modèle considéré est alors un réseau dont les sorties sont rebouclées sur les entrées (cf fig. 6), ce qui est équivalent à un réseau dans lequel tout automate est connecté à tout autre.

Figure 6: Réseau itératif utilisé dans le cas de l'auto-association.



L'évolution temporelle de l'état $\mathcal{X}$ du réseau est gouvernée par l'équation:

$$\mathcal{X}^{h+1} = F[W\mathcal{X}^h] \qquad (11)$$

où h est un indice temporel. Lorsque F est la fonction seuil, et lorsque la matrice W est symétrique à diagonale non négative, on montre que ce processus est toujours convergent [18,11]: le réseau finit par se stabiliser quel que soit l'état initial. Cette propriété est due à l'existence d'une FONCTION ENERGIE H qui décroît le long de la trajectoire:

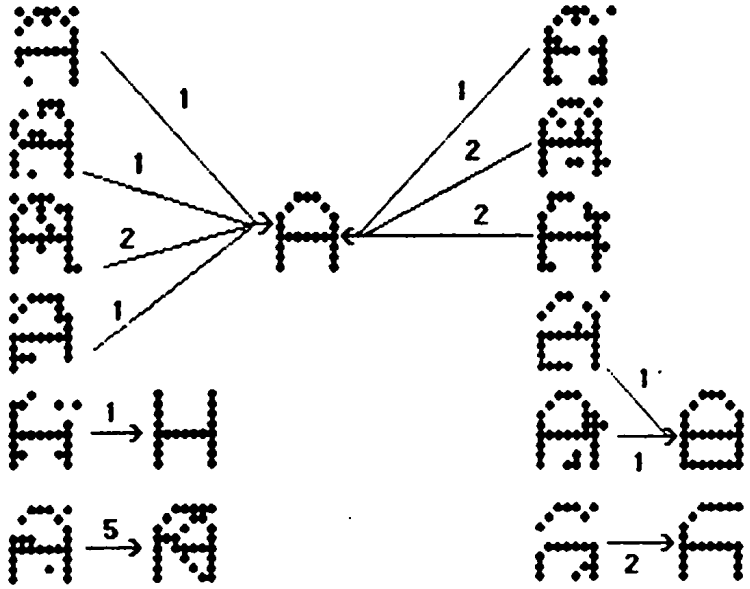
$$H^h = -(\mathcal{X}^h)^t \, W \, \mathcal{X}^{h-1} \qquad (12)$$

$$H^h \geq H^{h+1}$$

Lorsque W est solution du système $X_k = WX_k$ ($\forall$ k), les X_k sont des points stables de la dynamique, ce sont des minima de la fonction énergie: des ATTRACTEURS. Lorsque le réseau est placé dans un état proche de l'un des X_k, on peut espérer qu'il se stabilise dans ce X_k au bout de quelques itérations. Cette propriété est utilisée pour réaliser des mémoires associatives,

restituant un état mémorisé à partir d'une version bruitée ou incomplète de celui-ci. Malheureusement, de nombreuses configurations autres que les X_k peuvent être également stables (par exemple la solution triviale W=I rend tous les états stables) contraignant à choisir une solution pertinente (fig.7).

Figure 7: Reconnaissance de caractères par un réseau auto-associatif.
Un réseau d'automates réalise l'auto-association des 18 premières lettres de l'alphabet, codées en -1,1 sur une grille 8x8. La matrice **W** est calculée selon (10) de façon à ce que la diagonale de **W** soit nulle (cf [10]). La technique de restauration utilisée suit (11) avec une fonction F à seuil.
La figure montre pour différentes formes bruitées (à distance 7 du «A») la configuration stable obtenue, ainsi que le nombre d'itérations nécessaires. D'autres configurations que «A» sont stables.



---

4 - Réseaux multicouches: au delà du perceptron

La plupart des modèles connexionnistes de l'apprentissage sont basés sur le même type d'opérateur: une somme pondérée suivie d'une transformation non-linéaire. Il est clair que toute fonction booléenne peut être calculée a

l'aide d'un ensemble de tels opérateurs assemblés de manière plus ou moins complexe. Malheureusement tous les modèles décrits jusqu'ici ne possèdent qu'un seul étage de pondérations modifiables entre leur entrée et leur sortie. Ils sont par conséquent assujettis aux limitations intrinsèques des séparateurs linéaires. Une solution possible à ce problème est de trouver une procédure d'apprentissage permettant la génération automatique d'un "étage de décodage" pour extraire les traits pertinents des formes d'entrée, rendant le problème plus facilement séparable. Dans les réseaux multicouches, cet extracteur de traits est lui-même constitué d'opérateurs à seuil dont les poids sont modifiés par l'apprentissage. L'apprentissage des poids de ces UNITES CACHEES ne peut être effectué directement car aucun signal extérieur ne spécifie leur état idéal. Il n'existe donc pas de moyen direct permettant de savoir si la sortie d'une unité cachée est correcte ou non pour une forme d'entrée particulière. Il n'est pas étonnant de voir apparaître cette difficulté étant donné que le problème abordé ici est celui de la GENERATION AUTOMATIQUE DE REPRESENTATIONS INTERNES ADEQUATES. Une illustration de ceci peut être trouvée dans l'exemple suivant: considérons un réseau totalement connecté (rebouclé) pour lequel nous désirons que les configurations suivantes soient des attracteurs:

FFFXX...X

FVVXX...X

VFVXX...X

VVFXX...X

Ici F représente une valeur de sortie de -1 pour l'automate correspondant, et V une sortie de 1, XX...X est une séquence de bits arbitraire (la même pour les quatre configurations). Il n'existe aucun ensemble de poids (quelle que soit la méthode employée) qui permette de rendre ces états stables sans en rendre stable un grand nombre d'autres. Ceci est dû au fait que chacun des trois premiers bits est calculé en effectuant un OU EXCLUSIF des deux autres. Calculer le OU EXCLUSIF, ou toute autre fonction non linéairement séparable

nécessite l'utilisation d'unités additionnelles. Celles-ci calculent des prédicats intermédiaires permettant d'exprimer la sortie d'un quelconque automate comme combinaison linéaire (seuillée) des autres. Le problème décrit peut ainsi se résoudre en ajoutant une unité:

FFFFXX...X

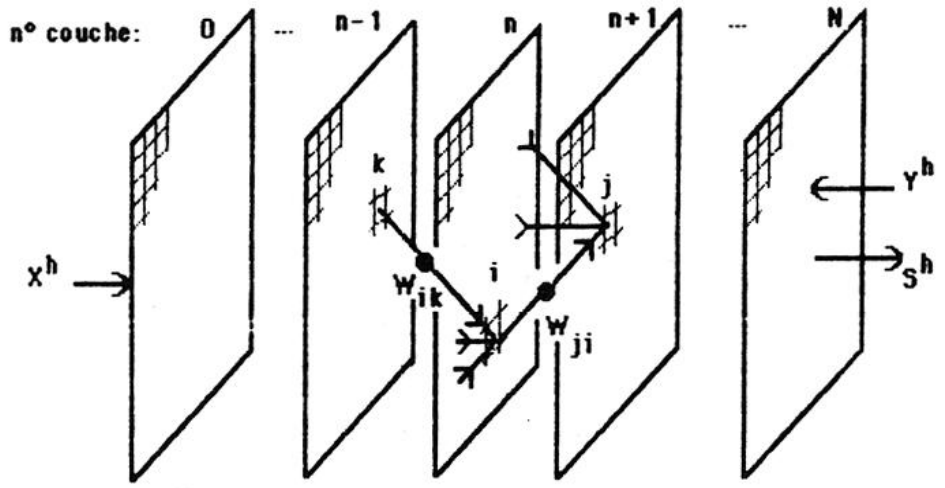
FVVVXX...X

VFVVXX...X

VVFVXX...X

L'état de cette unité CACHEE supplémentaire ne faisant pas partie de la forme à mémoriser, il ne doit pas être spécifé de l'extérieur mais doit être généré par le réseau lui-même. Les unités cachées sont utilisées dans le cas des réseaux multicouches sans boucle, où elles sont le support des prédicats intermédiaires permettant de décomposer la relation entrée-sortie en *une suite d'étapes linéairementséparables* (cf Fig 8).

Figure 8: Réseau multi-couche.



L'apprentissage dans les réseaux multicouches était un problème non résolu jusqu'à très récemment. Certains auteurs [25] ont même suggéré qu'une telle généralisation du paradigme connexionniste pouvait être impossible. Ceci

fournit une explication à la fois pour leur relatif abandon pendant les années 70 ainsi que pour leur récent renouveau En effet, deux types d'algorithmes viennent d'être proposes qui résolvent le problème et semblent générer de "bons" concepts intermédiaires. Le premier est la Machine de Boltzmann [14,15,32,33], développée pour des réseaux probabilistes à matrice de connexion symétrique. Le second algorithme, dit DE RETRO-PROPAGATION DE GRADIENT, a été proposé par plusieurs auteurs sous des formes différentes [20,21], puis [31].

5 - Apprentissage par rétro-propagation de gradient

Dans cette section, nous utilisons un réseau d'automates, structuré en couches successives, la première (d'indice 0) reçoit les entrées du système, la dernière (d'indice N) produit les sorties. Par simplicité, nous supposerons que les seules connexions possibles peuvent relier une couche aux couches d'indice supérieur et qu'il n'y a pas de connexion intra-couche (fig.8)
Chaque automate i du réseau calcule sa sortie comme une fonction DIFFERENTIABLE f de la somme pondérée A_i de ses entrées x_j:

$$x_i = f[A_i] \qquad \text{avec } A_i = \sum_j W_{ij}\, x_j \qquad (13)$$

où W_{ij} est le poids de la connexion de la cellule j vers la cellule i et f est une fonction sigmoïde (fig 5). L'état global du réseau est noté x.
A chaque présentation d'un vecteur d'entrée X^h -qui sera donc l'état de la couche 0- le réseau calcule sa sortie S^h -l'état de la couche N- en appliquant la formule (13), de proche en proche, de la couche 0 à la couche N. Simultanément, une sortie désirée Y^h est présentée sur la couche de sortie N et on va minimiser une fonction de coût, par exemple la fonction quadratique:

$$C(W) = 1/m \sum_k C_k \qquad \text{avec } C_k = \sum_s [S_{ks} - Y_{ks}]^2 \qquad (14)$$

où s indice uniquement les cellules de la DERNIERE COUCHE. Au lieu de minimiser C directement, on préfèrera considérer la moyenne des C^h associés à la

forme présentée au temps h en utilisant un algorithme de type gradient stochastique. Pour cela, on doit connaître le gradient de C par rapport à $\mathbf{W}$, c'est à dire les $\partial C/\partial W_{ik}$, pour tout i,k. Plutôt que de les calculer directement, on évalue d'abord les $\partial C/\partial A_i$, d'où on dérivera ensuite les $\partial C/\partial W_{ik}$ en utilisant (13):

$$\partial C/\partial W_{ik} = \partial C/\partial A_i \,.\, x_k \qquad (15)$$

Les $\partial C^h/\partial A_s$ associés aux cellules de sortie sont facilement calculables en combinant (13) et (15):

$$\partial C^h/\partial A_s = 2\,[S^h_s - y^h_s\,] \cdot \partial S^h_s/\partial A_s$$

or $S^h_s = f(A_s)$, d'où: $\partial C^h/\partial A_s = 2\,[S^h_s - y^h_s\,] \,.\, f'(A_s)$

On notera: $\partial C^h/\partial A_s = \delta_s$

Il n'en va pas de même pour les cellules cachées. La dérivée $\partial C/\partial A_i$ pour la cellule cachée n°i se calcule à partir des dérivées $\partial C/\partial A_j$ correspondant à toutes les cellules j dont elle est une entrée. Les règles de dérivation des fonctions composées donnent:

$$\partial C/\partial A_i = \sum_j \partial C/\partial A_j \,.\, \partial A_j/\partial A_i$$

Or $\partial A_j/\partial A_i = \partial A_j/\partial x_i \,.\, \partial x_i/\partial A_i = W_{ji}\, f'(A_i)$

D'où $\partial C/\partial A_i = f'(A_i) \sum_j W_{ji} \,.\, \partial C/\partial A_j$

ou encore $\delta_i = f'(A_i) \sum_j W_{ji} \,.\, \delta_j$ avec: $\delta_i = \partial C/\partial A_i$

On voit donc que le gradient de la fonction de coût se calcule à partir des δ_i, évalués de proche en proche à partir de la couche N vers les couches internes, et en utilisant les poids "à l'envers" (fig. 9), d'où le nom de "rétro propagation".

En résumé, une itération d'apprentissage comprend trois phases:

- présentation de la forme x^h ,puis calcul de l'état du réseau x par propagation "directe".
- présentation de la sortie désirée y^h et calcul des gradients δ_s sur la dernière couche, puis calcul des gradients δ_i sur les couches internes par rétro-propagation.

• enfin modification des poids en utilisant ces gradients, à l'aide de (15) et d'une méthode de gradient stochastique

si i est une cellule d'entrée: $\mathcal{x}_i = X_i$

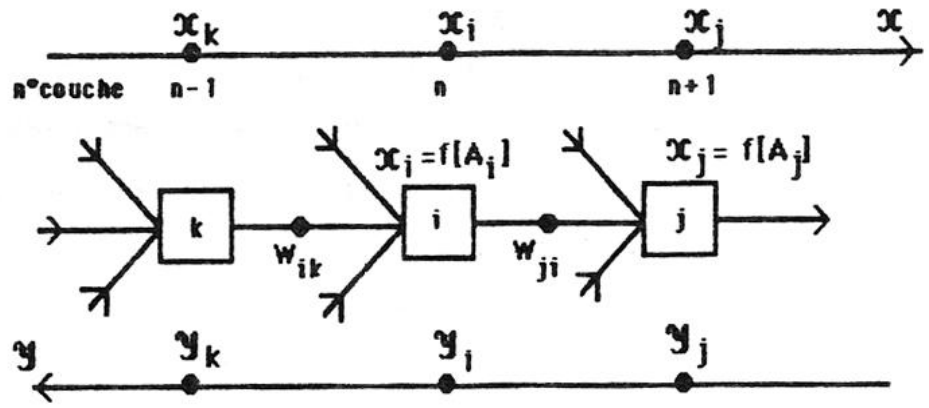
sinon $\mathcal{x}_i = f[A_i]$ avec $A_i = \sum_j W_{ij}\, \mathcal{x}_j$

si s est une cellule de sortie: $\mathcal{y}_s = 2\,[S^h_s - Y^h_s]\,.\,f'(A_s)$

sinon: $\mathcal{y}_i = f'(A_i)\sum_j W_{ji}\,.\,\mathcal{y}_j$

$$W^h_{ij} = W^{h-1}_{ij} - \varepsilon^h \mathcal{y}_i \mathcal{x}_j \qquad (16)$$

Figure 9: Détail du calcul direct des $\mathcal{x}$ et rétrograde des $\mathcal{y}$.



Le raisonnement ayant permis d'obtenir ce résultat reste valable si le graphe de connexion du réseau comporte des boucles, ce qui signifie que l'algorithme de rétro-propagation peut être utilisé pour des réseaux dont la dynamique temporelle n'est pas triviale.

Il est intéressant de remarquer que le critère global minimisé par l'apprentissage N'EST PAS CONVEXE s'il y a des cellules cachées. Ceci implique plusieurs conséquences, la première est que plusieurs configurations de poids peuvent être solution d'un problème, la seconde est que la configuration de poids initiale, ainsi que l'ordre et la fréquence de présentation des exemples influent sur la solution obtenue, la troisième conséquence est que le vecteur de poids peut être piégé dans un minimum

local de la fonction de coût, l'expérience prouve que cette situation ne se produit que lorsque le réseau a tout juste la taille nécessaire pour résoudre le problème.

6- Exemples

• la synthèse de fonctions booléennes

Il s'agit de synthétiser un réseau qui calcule une fonction booléenne, donnée par sa table de vérité. Bien que ne réalisant pas à proprement parler un apprentissage (puisqu'il n'y a pas de généralisation), les exemples que nous présentons permettent de bien illustrer le fonctionnement de la méthode: le réseau obtenu a d'une part "appris par cœur" la fonction, mais en a également réalisé une décomposition sous forme de fonctions élémentaires, et en a extrait une représentation compacte (c'est le problème classique en électronique de la synthèse d'une fonction à partir d'un petit nombre d'éléments différents).

- Nous avons vu qu'un perceptron ne pouvait pas calculer un OU EXCLUSIF, sauf à être doté de cellules d'association adéquates (fig. 4), par contre, il est facile de faire apprendre le OU EXCLUSIF à un réseau multi couche. Il suffit d'un réseau comprenant 2 cellules d'entrée, 2 cellules cachées et 1 cellule de sortie (fig.10). Le réseau calcule, selon les conditions initiales sur les poids, plusieurs décompositions possibles du OU EXCLUSIF obtenues en calculant la fonction booléenne équivalente à la fonction à seuil synthétisée par chacune des cellules du réseau: NON OU (ET, NON OU), ET(NON ET, OU),...
- On peut également synthétiser un réseau qui apprenne les tables de multiplication: pour calculer le produit de deux nombres de 3 bits (produit de 0x0 jusqu'à 7x7), on utilise un réseau dont la couche d'entrée comprend 6 cellules sur lesquelles sont codées les 2 nombres en binaire, la couche intermédiaire contient 30 cellules cachées et la couche de sortie 6

cellules pour coder le résultat du produit (en binaire) Il suffit d'environ 40 présentations (pour des conditions initiales des poids aléatoires) des 64 couples d'exemples pour obtenir 100% de réussite. Ces résultats montrent comment il est possible de trouver une solution parallèle à un problème de nature essentiellement séquentielle.

• **Mémoires associatives:** On a comparé les performances de cinq modèles d'auto-association avec des réseaux à une couche et des réseaux multi-couche. On utilise 18 formes (vecteurs binaires à n=64 composantes, codées en -1,1) X_k, qu'on désire mémoriser et retrouver à partir de formes bruitées. Les formes choisies ici sont des caractères imprimés digitalisés "à la main" dans des matrices 8x8 (cf fig.7).

Les modèles d'apprentissage utilisés sont de deux types:

- réseaux à une couche:

1- le modèle de l'auto-association avec la matrice des poids calculée par (10) avec une technique d'orthogonalisation (Gram-Schmidt) pour le calcul de la pseudo-inverse, soumise à une contrainte de diagonale nulle (la matrice **W** obtenue a une diagonale nulle et la matrice **Z** est différente de 0) [10,11].

2- le modèle de l'auto-association avec la matrice des poids calculée par une technique de gradient stochastique (Widrow Hoff), avec contrainte de diagonale nulle, sur le critère:

$$C(\mathbf{W}) = 1/m \sum_k C_k \quad \text{avec } C_k = \sum_s [F_s(\mathbf{W}X_k) - X_{ks}]^2$$

analogue à (14), où F_s est une fonction sigmoïde (cf fig.5) donnée par:

$$F_s(a) = b[\exp(ka)-1]/[\exp(ka)+1] \qquad (17)$$

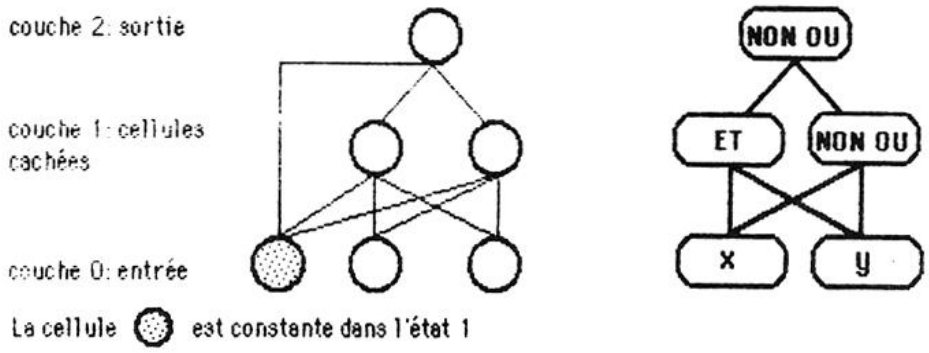
où k est choisi de façon à ce que: $F_s(1) = 1$ et F_s varie entre -b et b (b=1,7 ici).

3- un modèle d'auto-association avec une matrice des poids calculée comme précédemment, mais avec une famille d'exemples BRUITES:

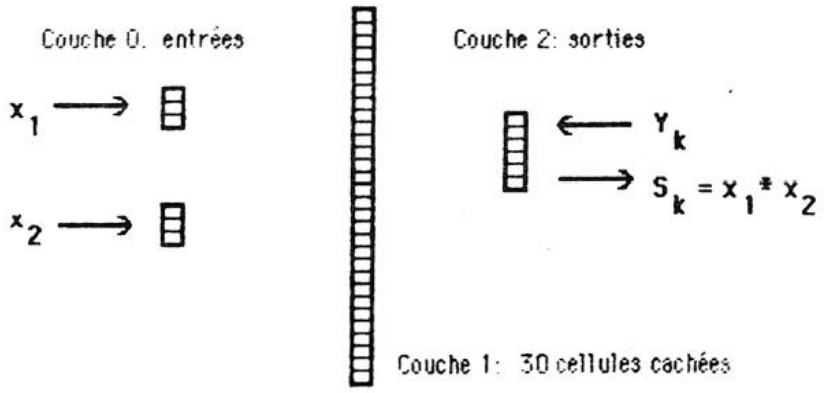
cellules pour coder le résultat du produit (en binaire). Il suffit d'environ 40 présentations (pour des conditions initiales des poids aléatoires) des 64 couples d'exemples pour obtenir 100% de réussite. Ces résultats montrent comment il est possible de trouver une solution parallèle à un problème de nature essentiellement séquentielle.

Figure 10: Synthèse de fonctions booléennes

Réseau pour le calcul de la fonction OU EXCLUSIF:



Réseau pour le calcul de la table de multiplication:



• **Mémoires associatives:** On a comparé les performances de cinq modèles d'auto-association avec des réseaux à une couche et des réseaux multi-couche. On utilise 18 formes (vecteurs binaires à n=64 composantes,

codées en -1,1) X_k, qu'on désire mémoriser et retrouver à partir de formes bruitées Les formes choisies ici sont des caractères imprimés digitalisés "à la main" dans des matrices 8x8 (cf fig.7).

Les modèles d'apprentissage utilisés sont de deux types:

- réseaux à une couche (fig. 11-1):

1- le modèle de l'auto-association avec la matrice des poids calculée par (10) avec une technique d'orthogonalisation (Gram-Schmidt) pour le calcul de la pseudo-inverse, soumise à une contrainte de diagonale nulle (la matrice W obtenue a une diagonale nulle et la matrice Z est différente de 0) [10,11].

2- le modèle de l'auto-association avec la matrice des poids calculée par une technique de gradient stochastique (Widrow Hoff), avec contrainte de diagonale nulle, sur le critère:

$$C(W) = 1/m \sum_k C_k \qquad \text{avec } C_k = \sum_s [F_s(WX_k) - X_{ks}]^2$$

analogue à (14), où F_s est une fonction sigmoïde (cf fig.5) donnée par:

$$F_s(a) = b[\exp(ka)-1]/[\exp(ka)+1] \qquad (17)$$

où k est choisi de façon à ce que: $F_s(1) = 1$ et F_s varie entre -b et b (b=1,7 ici).

3- un modèle d'auto-association avec une matrice des poids calculée comme précédemment, mais avec une famille d'exemples BRUITES: on associe ainsi des formes $\underline{X_k}$ (i.e. la forme X_k dont un certain nombre de composantes, choisies au hasard, ont été inversées) aux formes X_k.

- réseaux multi-couche (fig. 11-2):

4- un réseau à 2 couches de poids comprenant un nombre total de poids égal à celui des réseaux à une couche précédents (64 cellules dans la couche 0, 32 dans la 1ère et 64 dans la dernière et connexions complètes entre couches, soit 4096 poids). On présente une famille d'exemples "purs" X_k.

5- un réseau à 2 couches, identique au précédent, avec une famille d'exemples BRUITES $\underline{X_k}$.

Figure 11: Comparaison des différents modèles de mémoire associative.

Figure 11-1: Réseau auto-associatif à une couche de poids, utilisé pour les courbes 1,2 et 3

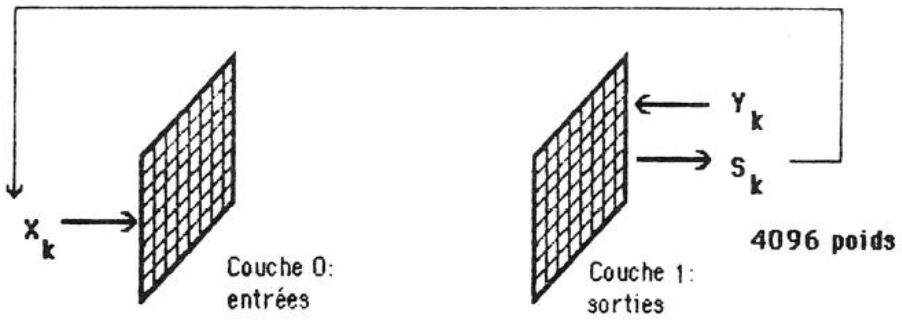
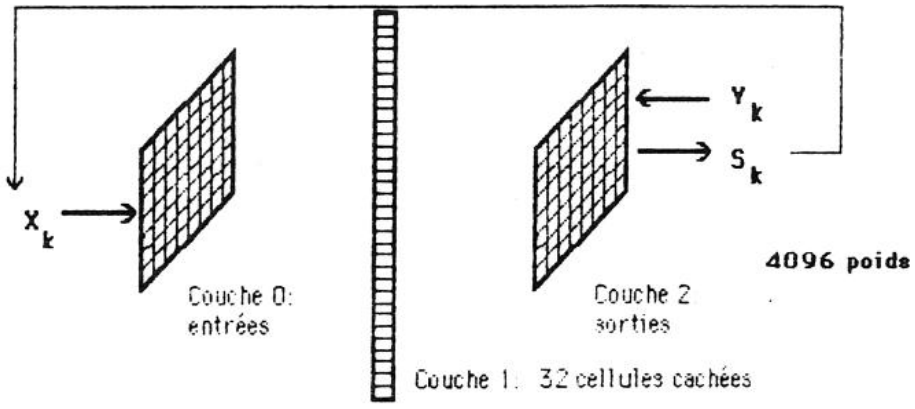
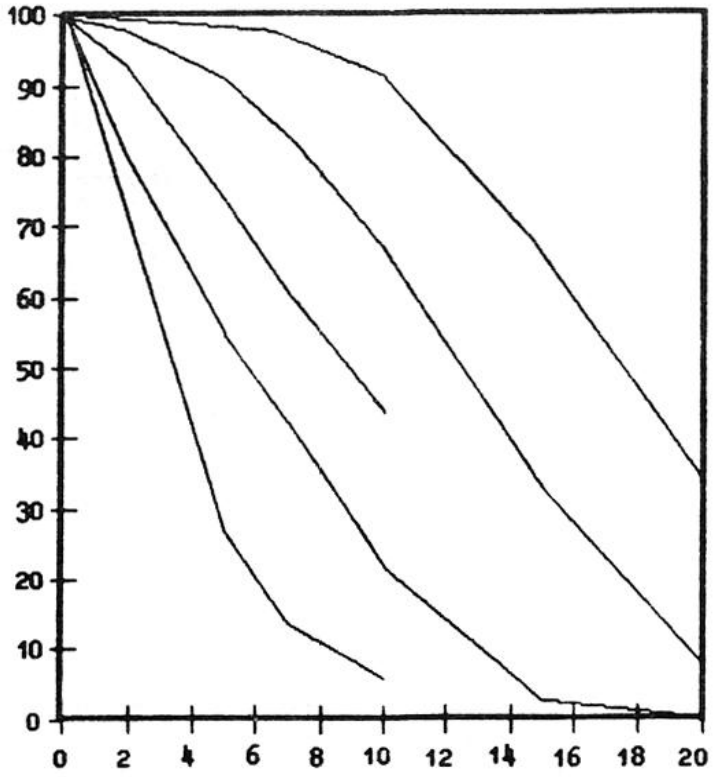


Figure 11-2: Réseau auto-associatif à 2 couches de poids, utilisé pour les courbes 4 et 5.



La technique de restauration utilisée est une itération seuillée: la sortie **S** produite par le réseau est seuillée (1 si $S_S \geq 0$, -1 sinon) et réintroduite en entrée du réseau. On autorise 5 itérations au maximum. La courbe (fig.11-3) représente le taux de succès (i.e. le pourcentage de formes $\underline{X_k}$ à une distance donnée d'un X_k, qui restaure correctement ce X_k) en fonction de la distance à la forme mémorisée X_k. Chaque point de la courbe est une moyenne sur 50 tirages de formes $\underline{X_k}$ pour chacune des 18 formes X_k.

Figure 11-3: Courbes des performances comparées: en abscisse, la distance (en nombre de bits) à la forme mémorisée et en ordonnée le taux de formes correctement restaurées. De haut en bas, courbes 5, 3, 1, 2 et 4.



On observe que:

les performances sont toujours meilleures quand un "modèle du bruit" a été fourni pendant la période d'apprentissage, ce qui contraint la solution **W** à "coder" ce modèle.

- le meilleur modèle multi-couche est largement supérieur au meilleur modèle à une couche. Avec un même nombre de paramètres, le réseau multi-couche, grâce à son architecture, est donc capable de mieux extraire les régularités présentes dans les données.
- les mauvaises performances du modèle a 2 couches sur une base d'exemples "purs" tient à l'absence de modèle du bruit Dans le modèle a 1

couche, il y a un modèle de bruit sous-jacent: le bruit additif gaussien décorrélé. Les différentes courbes obtenues (1,2 et 3) correspondent à 3 modèles de bruit différents (cf [11] pour une discussion du choix optimal de **W** en fonction du bruit).

• Jeu de tic-tac-toe

Le jeu de tic-tac-toe est joué sur un échiquier 3x3: il s'agit, pour chacun des 2 joueurs, d'aligner le premier 3 de ses pions (en ligne, colonne ou diagonale), chaque joueur pose un pion à tour de rôle. Le problème consiste à "apprendre" la nature d'une configuration de l'échiquier: gagnante, perdante ou nulle (nous nous placerons toujours du point de vue des Blancs et on supposera que la stratégie employée par les joueurs est la meilleure possible).

On utilise un réseau multi-couche dont l'entrée est une configuration (i.e. un vecteur à n=27 composantes, 3 groupes de 3x3 cellules codant l'état de la case: vide, Blanc, Noir) et la sortie donne la nature de la configuration d'entrée (i.e. un vecteur à 3 composantes: gagnante, perdante ou nulle). La couche de sortie est connectée à une couche intermédiaire de 50 cellules, elle-même totalement connectée à la couche d'entrée. Le taux de réussite, avec un ensemble d'apprentissage comprenant 50% des 5478 configurations légales, sur cet ensemble croît régulièrement avec le nombre d'itérations (i.e. de passages de la base d'exemples): au bout de 10 passes, il est de 90%, le taux de généralisation à l'autre moitié de la base d'exemples est alors de 76,5%.

Les taux de réussite et de généralisation dépendent du nombre de passages des exemples, du nombre de cellules cachées et du paramètre ε^h utilisé dans la règle d'apprentissage (16). (fig.12)

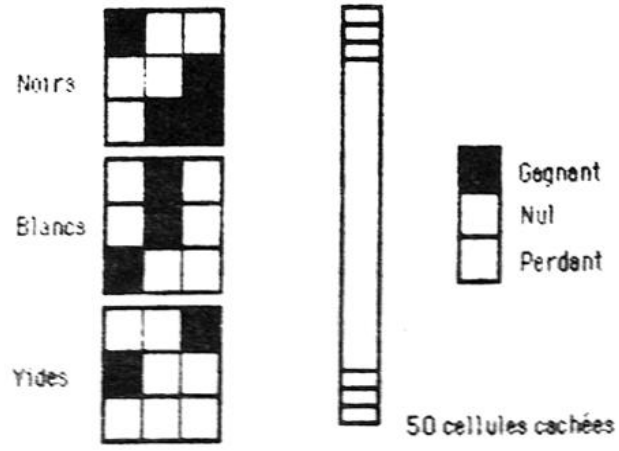
Remarquons que ces performances sont obtenues à partir d'un algorithme et d'une architecture "standards" pratiquement aucune connaissance a priori du domaine n'a été introduite. Ces performances peuvent être améliorées

(temps de calcul) si l'architecture est choisie de façon à tenir compte des régularités du problème (symétrie, détecteurs de lignes, ...)

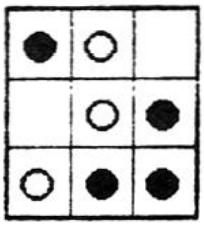
Ces exemples permettent de bien comprendre le fonctionnement de la méthode. Des exemples "en vraie grandeur" sont en cours de développement sur des bases d'exemples de taille plus importante et débouchant sur des applications pratiques.

Figure 12: le jeu de tic tac toe

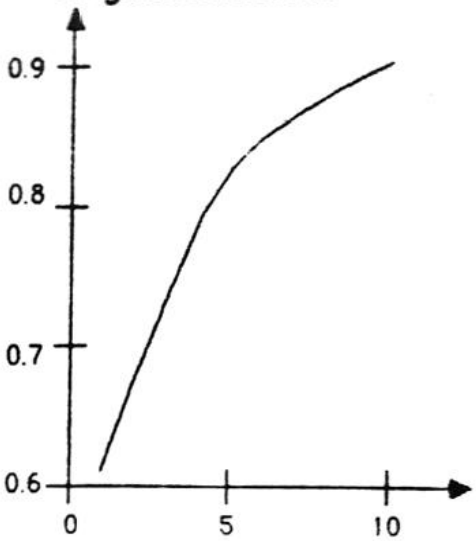
Structure du réseau



Configuration de l'échiquier



Courbe d'apprentissage et taux de généralisation



(en abscisse : nombre de passes, en ordonnée, taux de reconnaissance)

7- Conclusion

Nous avons montré que les limitations intrinsèques des modèles statistiques de l'apprentissage basés sur les associateurs linéaires pouvaient être dépassées. L'introduction dans l'architecture des réseaux de CELLULES CACHEES -i.e. l'utilisation de réseaux MULTI COUCHES- permet de décomposer des problèmes difficiles en étapes intermédiaires plus simples, solubles par des associateurs linéaires. Un algorithme d'apprentissage général, l'algorithme de rétro propagation de gradient, permet en effet d'extraire d'une base d'exemples des représentations internes qui permettent ensuite de GENERALISER.

La représentation des connaissances utilisée par ces modèles est portée par les connexions du réseau de manière compacte et distribuée, ce qui permet d'envisager l'utilisation de processeurs dédiés, d'où des temps de calcul largement réduits.

Les exemples présentés démontrent les performances du modèle. Ils illustrent également certains points fondamentaux: l'importance de la pédagogie (ordre de présentation des exemples) et de la connaissance initiale du domaine (qui permet d'optimiser l'architecture du réseau). Loin de constituer une solution à tous les problèmes d'apprentissage sans connaissance préalable, ce qui semblerait tout à fait irréaliste, cette approche nous permet d'aborder tous les intermédiaires entre un apprentissage sans connaissance a priori, et par conséquent sans bonne généralisation, et une situation où la connaissance du domaine est suffisante pour qu'on puisse définir une architecture de réseau et une configuration initiale de poids conduisant à l'extraction de représentations internes adéquates.

Ces recherches, très prometteuses, ont cependant le défaut par rapport aux méthodes habituelles en IA de ne pas fournir d'explication des résultats fournis. On peut cependant envisager qu'une partie des sorties puisse constituer une argumentation du résultat, pouvant être utilisée à la fois lors de l'apprentissage pour en faciliter le déroulement, et pendant l'utilisation du système en tant que moyen d'évaluation. Dans cette optique, il semble intéressant de mettre en oeuvre des systèmes mixtes utilisant à la fois des représentations de type connexionniste, et des techniques bien maitrisées utilisées couramment dans les systèmes à base de connaissance

---

Nous remercions: G. Hinton, pour avoir suggéré les problèmes du tic tac toe et de la multiplication, F. Manchon et H. Paugam Moisy, qui ont programmé le tic tac toe, M. Cartereau et J.C. Marie, pour la multiplication, P. Gallinari et S. Thiria, qui ont réalisé certaines simulations.

8- Références

[1] S.I AMARI: Learning patterns and patterns sequences by self-organizing net of threshold elements. **IEEE Trans. Com**. Vol C-21, No 11, Nov 72.

[2] D.H. ACKLEY, G.E. HINTON, T.J. SEJNOWSKI: A Learning Algorithm for Boltzmann Machines. **Cognitive Science**, 9, pp 147-169, 1985.

[3] J.A ANDERSON Cognitive capabilities of a parallel system In «Disordered systems and biological organization», E. Bienenstock, F. Fogelman Soulié, G. Weisbuch Eds, **Springer Verlag**, NATO Asi series in systems and computer science, n°20, pp 109-226, 1986.

[4] E. BIENENSTOCK, F. FOGELMAN SOULIE, G. WEISBUCH Eds: «Disordered systems and biological organization», **Springer Verlag**, NATO Asi Series in Systems and Computer Science, n°F20, 1986.

[5] N. BONGARD: Pattern Recognition. **Spartan Books**, New-York, 1970

[6] T.M COVER: Geometrical and statistical properties of systems of linear inequalities with applications to pattern recognition, **IEEE Trans. Electronic Computer,** EC-14 (3) p. 326-334, 1965.

[7] R.O. DUDA, P.E. HART: Pattern classification and scene analysis. **Wiley,** 1973.

[8] F. FOGELMAN SOULIE: Pattern Recognition by Threshold Networks. In «Actes du Colloque International d'Intelligence Artificielle, Marseille», 1984.

[9] F. FOGELMAN SOULIE: Cerveau et machines: des architectures pour demain? «Cognitiva 85», **CESTA-AFCET Ed.,** actes du forum, 1985.

[10] F. FOGELMAN SOULIE, P. GALLINARI, S. THIRIA: Linear learning and associative memory, in "Pattern Recognition, Theory and Applications", NATO ASI Series, P. Devijver Ed., **Springer Verlag,** à paraître.

[11] F. FOGELMAN SOULIE, P. GALLINARI, Y. LE CUN, S. THIRIA: Automata networks and artificial intelligence. In «Computation on automata networks», **Manchester Univ. Press,** à paraître.

[12] F. FOGELMAN SOULIE, G. WEISBUCH: Random iterations of threshold networks and associative memory. **SIAM J. on Computing,** à paraître.

[13] T.N.E. GREVILLE: Some applications of the pseudo inverse of a matrix. **SIAM Rev.** 2, pp 15-22, 1960.

[14] G.E. HINTON: Optimal perceptual inference. **IEEE Conf. on Computer vision and pattern recognition**, pp 448-453, 1983.

[15] G.E. HINTON: Learning in Parallel Networks. **Byte,** pp 265-273, april 1985.

[16] G.E. HINTON: Learning to recognize shapes in a parallel network. In the proceedings of the **Fyssen meeting** on vision, Paris, march 1986.

[17] G.E. HINTON, J.A. ANDERSON (Eds): Parallel Models of Associative Memory Hillsdale, **Erlbaum**, 1981.

[18] J.J. HOPFIELD Neural Networks and Physical Systems with Emergent Collective Computational Abilities, **P.N.A.S. USA** , vol 79, pp 2554-2558, 1982.

[19] T. KOHONEN: Self-Organization and Associative Memory. Springer Series in Information Sciences, vol 8, **Springer Verlag**, 1984.

[20] Y. LE CUN: A learning scheme for asymmetric threshold network. In «Cognitiva 85», **CESTA-AFCET Ed.**, pp 599-604, 1985.

[21] Y. LE CUN: Learning process in an asymmetric threshold network. In [4], pp 233-240, 1986.

[22] J.L. MC CLELLAND, D.E. RUMELHART: Distributed memory and the representation of general and specific information. **J. Exp. Psychology**, Vol 114, n°2, pp 159-188, 1985.

[23] W.S MAC CULLOCH, W. PITTS: A logical calculus of the ideas immanent in nervous activity. **Bull. Math. Biophysics**, 5, pp 115-133, 1943.

[24] R.S. MICHALSKI, L.G. CARBONELL, T.M. MITCHELL: Machine Learning. **Tioga**, 1983.

[25] M. MINSKY, S. PAPERT: Perceptrons, an Introduction to Computational Geometry. Cambridge, **MIT Press**, 1969.

[26] K. NAKANO : Associatron, a model of associative memory. **IEEE Trans Syst. Man Cyb.**, Vol SMC -2, No 3, July 1972.

[27] N.J. NILSSON: Learning machines. **McGraw Hill**, 1965.

[28] P. PERETTO: Collective properties of neural networks: a statistical physics approach. **Biol. Cybern.**, 50, pp 51-62, 1984.

[29] J. QUINQUETON, J. SALLANTIN, H. SOLDANO, S. BOUCHERON: dans ce numéro.

[30] F. ROSENBLATT: Principles of Neurodynamics. **Spartan**, 1962.

[31] D.E. RUMELHART, G.E. HINTON, R.J. WILLIAMS: Learning internal representations by error propagation. In «Parallel Distributed Processing: Explorations in the microstructure of cognition», Vol 1:Foundations, D.E Rumelhart, J.L. McClelland Eds. **MIT Press**, 1986.

[32] T.J. SEJNOWSKI, G.E. HINTON: Separating figure from ground with a Boltzmann machine. In «Vision, brain and cooperative computation», Arbib M.A., Hanson A.R. Eds, Cambridge, **MIT Press**, 1985.

[33] T.J. SEJNOWSKI, P.K. KIENKER, G.E. HINTON: Learning symmetry groups with hidden units: beyond the perceptron. to appear in **Physica D**.

[34] T.J. SEJNOWSKI, C.H. ROSENBERG: NETtalk: a parallel network that learns to read aloud. Johns Hopkins Technical report **JHU/EECS-86/01**.

[35] G. WEISBUCH, F. FOGELMAN SOULIE: Scaling laws for the attractors of Hopfield networks. **J. Phys. Lett.**, 46, 623-630, 1985.

[36] B. WIDROW, M.E. HOFF: Adaptive switching circuits. **IRE WESCON Conv. Record**, part 4, pp 96-104, 1960.