MEMOIRES ASSOCIATIVES DISTRIBUEES: UNE COMPARAISON

DISTRIBUTED ASSOCIATIVE MEMORIES: A COMPARISON

F. Fogelman Soulié *,P. Gallinari**, Y. Le Cun ***, S. Thiria **

1- INTRODUCTION

Les modèles dits "connexionnistes" ont été utilisés en tant que mémoires associatives pour résoudre de nombreux problèmes [3,4,5,7,10,12,13,14,21,22,23,24]

Ces techniques dont les origines remontent aux travaux sur le perceptron et les associateurs linéaires connaissent actuellement un grand regain d'intérêt, ce qui fait qu'on se trouve aujourd'hui en présence d'un grand nombre de modèles Bien que conçus dans des buts différents, ils possèdent de nombreuses propriétes de base en commun pour lesquelles on possède peu d'éléments de comparaison Il apparaît donc nécessaire de réaliser une synthèse de ces différents modèles et une comparaison de leur aptitude pour la réalisation de tâches précises C'est ce que l'on se propose de faire ici en testant les capacités de memoire associative de quelques modèles représentatifs

Après avoir décrit le problème (§2) de la réalisation de ces mémoires sur des réseaux, nous passons en revue les principales approches existantes (§3) Puis nous étudions le comportement et les performances de différents réseaux sur un problème d'association d'ensembles de formes aléatoires qui est un problème de memorisation pure

2- LE PROBLEME

Nous rappelons ici brièvement le problème Une **mémoire associative** est un systeme qui permet de realiser une association entre deux ensembles de formes X et Y, en faisant correspondre une forme de Y à une forme de X Nous nous intéressons ici à la réalisation de telles mémoires par des reseaux d'automates

Un **réseau d'automates** est un ensemble d'éléments en interaction, chacun capable de réaliser des calculs élémentaires Ces automates échangent des informations au moyen de connexions qui les relient. Un des principaux intérêts de ces réseaux est qu'ils constituent des mémoires adressables par contenu et des mémoires distribuées une information est codee non pas à un endroit précis mais sur l'ensemble des automates du reseau. Ils présentent en plus des qualités de robustesse au bruit i e que si (x y) est une des associations apprises, le réseau retrouvera egalement y à partir de versions bruitées de x

Pour constituer une telle mémoire sur un réseau on distingue trois phases successives dont nous allons indiquer les traits principaux

- Il faut tout d'abord déterminer l'architecture du réseau c'est à dire le nombre de ses automates, leur nature et leur interconnexion Certains automates des réseaux que nous utiliserons ont pour rôle de recevoir les données, ils constituent la **couche d'entrée** du réseau On définit de même une **couche de sortie** qui collecte les réponses du réseau Entre ces deux couches peuvent exister d'autres automates qui ne sont pas reliés à l'extérieur et peuvent être disposés en "couches intermédiaires" Ces différents éléments sont reliés entre eux et les données circulent de la couche d'entrée vers la couche de sortie La détermination du nombre d automates par couche et des chemins de communication c'est à dire la présence ou l'absence d'une connexion entre deux automates est fonction des problèmes traités

- On programme ensuite cette architecture, c'est la phase dite d'**apprentissage** pendant laquelle la connaissance permettant l'association est encodee dans les connexions du réseau sous la forme d'une valuation (poids) de ces connexions Ce mode de stockage de l'information est d'ailleurs à l'origine du terme **connexionniste** pour désigner ces modèles Dans ceux que nous étudions, la determination des valeurs des connexions se fera à **partir d'exemples** les couples à associer sont présentés au reseau et celui-ci modifie ses connexions pour realiser l'association demandée De nombreux algorithmes d'apprentissage ont été proposés [1,2,6,15,16,18,20], dont certains, récents, ont résolu le problème de l'apprentissage sur réseau multi-couches, ce qui constituait un problème ouvert depuis les travaux de Minsky et Papert [17]

- la dernière phase est celle de la **reconnaissance** le réseau ayant encodé suffisamment de connaissances sur le problème pendant l'apprentissage, il peut ensuite être utilisé, par exemple en tant que mémoire associative Quand une donnée est présentée sur la couche d'entrée du réseau qui vient d'être construit, celui-ci va appliquer un algorithme de reconnaissance et fournir une réponse sur les cellules de sortie A cette étape on met en oeuvre différentes procédures qui permettent d'une part de calculer par propagation les états du réseau, d'autre part d'interpréter l'état des cellules de sortie

3- MODELES DE MEMOIRES ASSOCIATIVES

Plusieurs architectures et algorithmes ont été proposés pour la réalisation de telles mémoires Des résultats théoriques n'existent que pour les plus simples de ces modèles les modèles **linéaires** ou le **perceptron** Pour une présentation détaillée nous renvoyons à Kohonen [14] qui offre un ouvrage de base sur le sujet, ou à [9] ou sont étudiés et comparés de nombreux modèles Nous présentons les principales méthodes qui ont été utilisées pour résoudre ce problème

1- Pour l'apprentissage, on peut utiliser des algorithmes exacts [11,14,19] Ces techniques demandant souvent une grande quantité de calculs et n'étant pas applicables à tous les modèles, on leur préfère des algorithmes approchés, par exemple de type gradient De plus on exige souvent des mémoires associatives qu'elles soient modifiables pour y incorporer de

MEMOIRES ASSOCIATIVES DISTRIBUEES: UNE COMPARAISON

DISTRIBUTED ASSOCIATIVE MEMORIES: A COMPARISON

F. Fogelman Soulié *,P. Gallinari**, Y. Le Cun ***, S. Thiria **

---

nouvelles données ou changer leur contenu, on utilise alors des algorithmes dits adaptatifs [6,25].

Dans tous les cas les exemples constituant la base d'apprentissage sont présentés au réseau qui doit être modifié pour produire les réponses désirées Pour cela on prend en compte une fonction de coût qui traduit une distance entre la réponse désirée et la réponse calculée par le réseau Certains algorithmes utilisent un coût global calculé sur l'ensemble des associations que l'on veut apprendre, ce qui suppose qu'à un même instant on connaisse toutes ces associations. D'autres, les algorithmes adaptatifs, modifient le réseau en utilisant un coût "local" calculé par exemple après la présentation de chaque forme Parmi les algorithmes adaptatifs les plus connus on peut citer ceux du type perceptron [17] et l'algorithme de Widrow -Hoff [25]

2- De même plusieurs procédures ont été proposées pour les tâches de reconnaissance On peut utiliser

- différentes fonctions pour le calcul de l'état des automates Une des plus utilisées consiste à calculer une fonction linéaire des entrées de l'automate suivie éventuellement d'une fonction non linéaire

- des itérations pour les tâches d'auto-association on boucle la couche de sortie du réseau sur celle d'entrée et on itère un certain nombre de fois

- différentes procédures pour interpréter les résultats de la couche de sortie

Pendant longtemps seuls des modèles de réseaux à deux couches avec connexion totale entre les automates des deux couches ont été étudiés Ces réseaux possèdent un certain nombre de limitations intrinsèques, par exemple le fait qu'ils ne puissent réaliser qu'une séparation linéaire de leurs entrées, et donc leur incapacité à apprendre automatiquement des fonctions simples [17] L'ajout de cellules cachées est apparue très rapidement comme une des voies permettant d'augmenter les capacités de ces réseaux Toutefois, ce n'est que récemment que des algorithmes permettant l'apprentissage sur de tels reseaux ont été proposés [1,15,16,18,20] Ces réseaux permettent la synthèse de fonctions plus complexes Plusieurs expériences ont montré l'intérêt de construire des architectures plus sophistiquées [20] pour, par exemple, décomposer un raisonnement complexe en étapes élémentaires ou utiliser des informations locales.

Nous nous restreignons ici a l'étude de la mémorisation dans ces réseaux Nous avons testé les capacités de mémoire associative de différents modèles à deux couches [8] en mettant en évidence les qualités respectives de différents algorithmes d'apprentissage et de reconnaissance Ces tests ont été effectue sur la mémorisation de lettres de l'alphabet codées sur deux niveaux de gris Les meilleurs résultats ont été obtenus pour l'apprentissage pour l'algorithme exact de Greville [11,19], qui présente toutefois l'inconvénient de nécessiter à un moment donné la connaissance de l'ensemble des formes à mémoriser

Un des moyens d'améliorer la reconnaissance est d'employer des modèles permettant les itérations, dans lesquels on espère que la solution calculée est ameliorée itérativement pour se rapprocher de celle désirée Pour cela, on peut utiliser des automates qui, d'abord calculent une fonction linéaire de leurs entrées, puis font suivre celle-ci d'une transformation non linéaire, par exemple une fonction à seuil, une fonction sigmoide, ou des transformations plus sophistiquées comme dans le modèle "Brain State in the Box" (BSB) de Anderson [2] Dans les tests effectués [8], les meilleures performances ont été obtenues par l'algorithme du BSB, toutefois certains modèles, comme ceux utilisant des fonctions sigmoïdes offrent des performances presque équivalentes avec un nombre d'itérations beaucoup plus faible

On étudie ici l'influence de l'architecture dans la construction de mémoires associatives et plus particulièrement sur leur capacité de mémorisation Pour cette raison nous avons sélectionne des algorithmes d'apprentissage et de reconnaissance qui sont à la fois suffisamment généraux pour être applicables à de nombreux problèmes et qui offrent de bonnes performances Pour l'apprentissage nous avons utilisé l'algorithme de rétro-propagation du gradient (Gradient Back Propagation GBP) qui permet de tester de nombreuses architectures. Par rapport eux autres algorithmes autorisant l'utilisation de cellules cachées celui-ci présente l'avantage d'être moins coûteux en temps de calcul et, associé à un apprentissage avec bruit sur des réseaux à deux couches, il concurrence les meilleurs algorihmes d'apprentissage Pour la reconnaissance nous avons utilisé les même automates que pour l'apprentissage

Nous décrivons maintenant l'algorithme GBP

4- L'ALGORITHME GBP.

Cet algorithme, qui a été proposé par plusieurs auteurs [15,16,18,20], est une version étendue aux réseaux multicouches de l'algorithme de Widrow-Hoff Nous le décrivons, pour ne pas alourdir la présentation, dans le cas simplifié d'un réseau multi-couches où des connexions existent uniquement entre des automates d'une couche i et des automates de la couche $i+1$

Notons (X_l, Y_l) un des couples à associer Les formes $X_1, X_2, \ , X_m$ sont présentées séquentiellement un certain nombre de fois en entrée du réseau On notera par la suite la séquence temporelle (X_1, Y_1), (X_2, Y_2), ... (X_m, Y_m),

MEMOIRES ASSOCIATIVES DISTRIBUEES: UNE COMPARAISON

DISTRIBUTED ASSOCIATIVE MEMORIES: A COMPARISON

F. Fogelman Soulié *,P. Gallinari**, Y. Le Cun ***, S. Thiria **

(X_1,Y_1), (X_2,Y_2) .. par (X^1,Y^1), (X^2,Y^2), .. (X^m,Y^m), (X^{m+1},Y^{m+1}), (X^{m+2},Y^{m+2}). .

On note W_{ij} la valuation de la connexion qui va de la cellule j à la cellule i Soit x_j la sortie d'un automate de la couche k, alors la sortie x_i de l'automate i sur la couche k+1 est calculée de la façon suivante

$$x_i = f(\Sigma_j W_{ij} x_j)$$

ou la somme est faite sur les automates j qui envoient leur sortie vers l'automate i et f est une fonction sigmoïde de la forme $f(x) = a(e^{kx} - 1) / (e^{kx} + 1)$ Par la suite, on notera

$$A_i = \Sigma_j W_{ij} x_j \qquad x_i = f(A_i)$$

Quand une forme X^k est présentée au réseau, celui ci calcule dans un premier temps la sortie S^k qui lui correspond par propagation dans le réseau de la couche d'entrée vers la couche de sortie Dans un second temps, S^k est comparée à la sortie Y^k que l'on désire associer à X^k pour produire un critère d'erreur et les valuations des connexions vont être modifiées pour minimiser ce critère

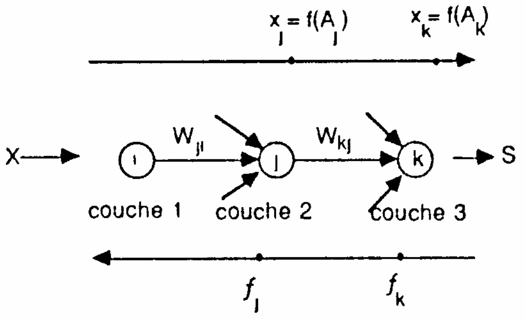

figure 1 Algorithme GBP, on a représenté sur un réseau à trois couches de cellules, le calcul d'une sortie S quand on présente en entree une forme X (en haut) et la propagation des gradients (en bas)

Pour calculer la sortie S^k, quand la forme X^k est présentée sur la couche d'entrée, l'état d'un automate i de cette couche est imposé à X^k_i, puis chaque automate de la deuxième couche va calculer son état en fonction de ses entrées selon la fonction de transition précédemment décrite, l'envoyer vers les automates de la couche supérieure auxquels il est relié qui pourront calculer leur état, et ainsi de suite, jusqu'à ce que le résultat atteigne la couche de sortie dont l'état S^k sera la sortie du réseau Les iterations pour réaliser cette première phase sont donc faites séquentiellement d'une couche à la suivante et parallèlement dans une même couche

On peut alors commencer la phase de modification des poids Cette sortie calculée S^k est comparée à la sortie désirée Y^k pour produire une fonction de coût mesurant l'erreur commise par le réseau

Au lieu d'utiliser une fonction de coût globale à l'ensemble des formes, nous utilisons un critere local à chaque presentation le coût à minimiser à l'étape k est le carré de la distance euclidienne de la sortie calculée S^k à celle désirée Y^k

$$C(W)=\Sigma_i (S^k_i - Y^k_i)^2$$

i decrivant l'ensemble des cellules de la dernière couche

L'algorithme d'apprentissage va modifier les valuations des connexions du réseau, les W_{ij}, de façon à reduire cette erreur Cette modification va être faite cette fois de la couche des poids supérieure jusqu'à la première couche de poids

Soit A_i l'entrée d'une cellule i quelconque du réseau, on a

$$A_i = \Sigma_j W_{ij} x_j \text{ avec } x_j = f(A_j)$$

On veut minimiser C en ajustant les valuations du réseau, pour cela on emploie un algorithme de gradient adaptatif

$$W_{ij}(k) = W_{ij}(k-1) - \varepsilon(k)\, \partial C / \partial W_{ij}$$

ou $\varepsilon(k)$ est le pas du gradient au temps k

On doit calculer pour toutes les connexions i-j le gradient de C

$$\partial C / \partial W_{ij} = \partial C / \partial A_i \;\; \partial A_i / \partial W_{ij} = \partial C / \partial A_i \;\; x_j \qquad (1)$$

- pour une cellule i sur le dernier niveau et la $k^{\text{ème}}$ forme de la séquence, on a, si on note $S^k_i = f(A_i)$ la sortie calculée de cette cellule

$$\partial C / \partial A_i = 2(S^k_i - Y^k_i)\, \partial S^k_i / \partial A_i$$
$$= 2(S^k_i - Y^k_i) \,.\, f'(A_i)$$

Posons $\partial C / \partial A_i = f_i$, on a alors

$$f_i = 2(S^k_i - Y^k_i) \;\; f(A_i)$$

Ceci permet de calculer $\partial C / \partial W_{ij}$ pour tous les poids de la dernière couche Munis de ce résultat, on peut alors calculer (1) pour la couche immédiatement inférieure, puis recommencer l'opération jusqu'à la première couche de poids

- pour une cellule cachée i on a alors

$$\partial C \; / \; \partial A_i = \Sigma_h \; \partial C / \partial A_h \;\; \partial A_h / \partial A_i$$

ou h indice les cellules qui prennent une entrée sur la cellule i

$$\partial C \; / \; \partial A_i = \Sigma_h \; f_h \;\; \partial A_h / \partial x_i \;\; \partial x_i / \partial A_i$$

d'ou $\partial C \; / \; \partial A_i = \Sigma_h \; f_h \;\; W_{h\,i} \;\; f'(A_i)$
i e $f_i = (\Sigma_h \;\; f_h \;\; W_{h\,i}) \;\; f'(A_i)$

la règle générale de modification des poids devient donc

$$\mathbf{W_{ij}(k) = W_{ij}(k-1) - \varepsilon(k)\, f_i \;\; x_j} \qquad (2)$$

Cet algorithme nous permet de réaliser l'apprentissage d'une association sur un réseau multicouches Il n'est pas restreint au cas ou les connexions vont uniquement d'une couche à la suivante les connexions peuvent par exemple "sauter" une ou plusieurs couches Une grave difficulte tient à ce qu'il n'y a pas garantie de la convergence et on n est donc pas assuré de trouver un minimum global de notre fonction de coût Toutefois, l'expérience montre qu'il permet en général de réaliser l'association, et il est préféré à des algorithmes possédant des propriétés mathématiques plus solides, mais extrêmement coûteux en temps de calcul

5- SIMULATIONS

5-1 présentation

La tâche étudiée consiste en l'association de formes binaires aléatoires dans le cas de l'auto-association et de l'hétéro-association Dans ce dernier cas les formes ont été associées à d'autres formes binaires aleatoires de même taille Les formes sont des vecteurs à 32 composantes qui ont pour valeur -1 ou +1 L'utilisation de formes aléatoires, donc complètement décorrélées ne permet pas aux réseaux d'utiliser d'éventuelles régularités de l'ensemble d'apprentissage et nous permet donc de tester uniquement les propriétes de mémorisation des réseaux étudiés Notons que pour cette même raison la tâche fixée est plus difficile et les taux de reconnaissance trouvés constituent donc une borne inferieure des performances que l'on peut attendre de mémoires associatives sur réseaux

Certains apprentissages ont été effectués en introduisant un "modèle" du bruit dans l'ensemble des formes présentées, en fait ce "modèle" est fourni par des exemples de formes bruitées Le bruit étudié ici consiste en une inversion aléatoire de bits Appliquer un bruit de taille i à une forme donnée correspond donc à inverser exactement i bits de cette forme

L'apprentissage d'une association consiste alors à repeter plusieurs fois la séquence suivante

1- réaliser d'abord un apprentissage pour un bruit de taille i, i prenant une certaine suite de valeurs, par exemple 6 puis 4, 2 et 1 Un apprentissage a bruit i consiste à réaliser, pour chaque forme d'entrée,par exemple 10 associations de versions bruitées de cette forme à la sortie désirée

2- présenter la suite des formes pures à mémoriser

Cette séquence est répétée un certain nombre de fois pour chacune des expériences et les poids sont modifiés après la présentation de chaque forme On a ainsi fourni au réseau des données qui l'aident à retrouver des images erronees en créant des bassins d'attraction autour des points représentatifs des formes exactes mémorisées Pour chacune de ces formes, à chaque sequence on apprend à reconnaître une region de l'espace correspondant au bruit utilisé en associant quelques points de cette région à la réponse que l'on désire obtenir

La modification des poids est faite par une variante de la formule (2) qui est

$$W_{ij}(k) = (1-\gamma)W_{ij}(k-1) - \Delta\, W_{ij}(k)$$

avec $\Delta W_{ij}(k) = \alpha\varepsilon(k)\, f_i\, x_j + \beta\, \Delta\, W_{ij}(k-1)$

Cette règle permet de tempérer les modifications des poids en effectuant, par l'intermédiaire du terme $\Delta\, W_{ij}(k)$, une moyenne pondérée des gradients sur l'ensemble des présentations (1-γ) est un facteur qui fait décroître les poids proportionnellement à leur norme Cela permet de diminuer la norme moyenne de la solution obtenue et de réduire l'écart dans les normes des poids

Dans les expériences réalisées, $\alpha = \beta = 0\ 5$ ε et γ prennent des valeurs décroissantes au cours de l'apprentissage respectivement 0 5, 0 25, 0 2 et 0 01, 0 0025, 0).

Lors de la reconnaissance, on teste pour une forme et un bruit donnés 20 versions bruitées de cette forme, et pour l'auto-association on effectue 5 itérations de reconnaissance

Nous avons testé différentes architectures en faisant varier le nombre de couches et le nombre de cellules intermédiaires Dans chacun des cas les connexions sont complètes et vont d'une couche à la suivante Dans de nombreuses applications pratiques, une architecture liée à la nature de la tâche s'impose Ce n'est pas le cas ici, ce qui confère d'ailleurs à l'exemple sa portée générale au niveau de la comparaison Pour tous ces réseaux, on conserve un nombre de connexions constant afin de ne pas augmenter artificiellement leur capacité Les réseaux possèdent tous 32 cellules sur les couches d'entrée et de sortie, ceux à trois couches ont 16 cellules intermédiaires et ceux à quatre couches 2 fois 13 cellules intermédiaires Nous avons donc testé pour les deux tâches l'influence du nombre de couches sur les performances Pour chaque réseau on a testé des ensembles d'apprentissage de taille 8, 16, 32

Notons enfin qu'un point sur une courbe est la moyenne de 5 expériences

5-2 Résultats

Sur la figure 2 , nous avons fait figurer les performances de différentes architectures pour des tâches d'auto-association qui correspondent à différentes tailles de memoire Les notations sont les suivantes A16 (8) désigne une tâche d'auto association pour mémoriser 8 formes sur une architecture à trois couches possédant 16 cellules intermédiaires, H13_13(16) correspond à une mémoire de 16 formes en hétéro-association pour une architecture à deux couches intermédiaires de 13 cellules chacune

Sur la figure 3 nous avons les résultats de tests analogues pour des tâches d'hétéro-association L'apprentissage a été réalisé avec une séquence de bruits (6, 4, 2, 1), dix présentations de l'association (forme d'entree bruitée, réponse désirée) pour chaque forme d'entree et pour chaque taille de bruit dans une séquence et respectivement 10 et 8 séquences pour l'hétéro association et l'auto-association Dans ce dernier cas on a utilisé 5 itérations de reconnaissance. Pour une architecture donnée, les performances décroissent quand la taille de la mémoire augmente (saturation) et pour une taille mémoire donnée elles augmentent avec le nombre de couches intermédiaires, du moins pour les architectures testées Dans la procedure utilisée, le temps d'apprentissage alloué à une tâche est proportionnel au nombre de formes memorisees et pour une tâche donnée est le même pour toutes les architectures Ce dernier point est important pour interpréter les tests, car pour une tâche donnée, la stabilisation des performances demande un apprentissage d'autant plus long que le nombre de couches est important Les performances portées sur ces courbes ne sont donc pas optimales pour les réseaux à plusieurs couches. D'autre part l'hétéro-association apparaît plus difficile à réaliser que l'auto-association et nécessite plus d'itérations d'apprentissage pour que les taux de reconnaissance soient stabilisés

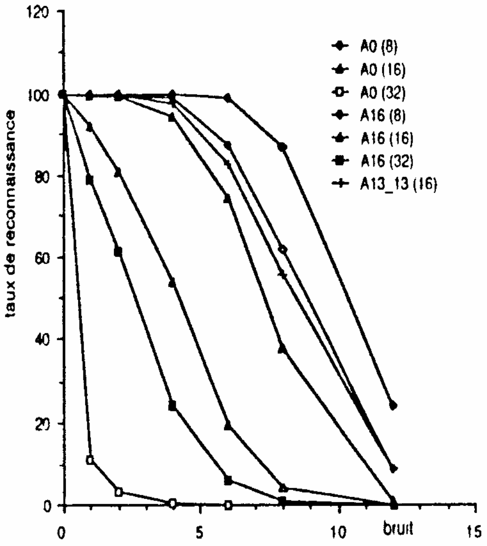

figure 2: capacité de différentes architectures pour des tâches d'auto-association correspondant à plusieurs tailles de mémoire

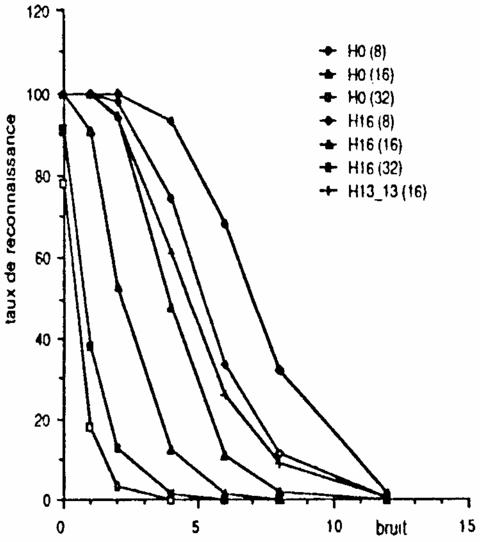

figure 3: capacité de différentes architectures pour des tâches d'hétéro-association.

Nous avons comparé les performances d'un réseau pour ces deux types de tâches, ces résultats apparaissent sur la figure 4 Pour cela, nous avons augmenté le nombre d'itérations d'apprentissage par rapport aux expériences précédentes en multipliant par 4 le nombre de présentations des différentes associations dans une séquence Bien que la stabilité n'ait pas été atteinte pour toutes les expériences, on peut ainsi avoir une idée de la difficulté respective des deux tâches L'expérience H13_13 (32) notamment n'a pas été continuée

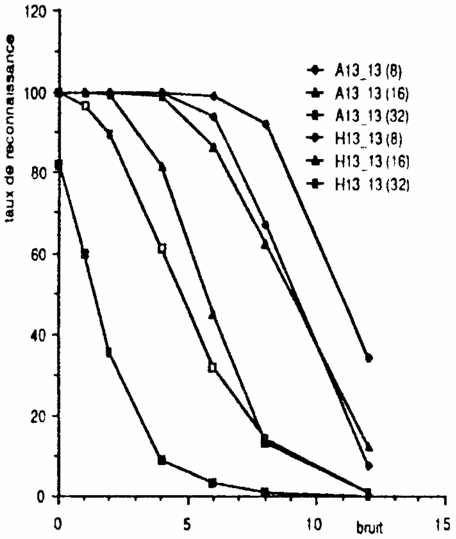

figure 4 comparaison des performances de differents réseaux pour des tâches d'auto-association et d'hétero-association Pour toutes les tâches, la durée de l'apprentissage a été augmentée pour tenter d'arriver à une stabilisation des performances

Ces expériences nous renseignent sur les performances de ces réseaux pour la restitution exacte (au seuil près) des associations mémorisees Qu en est-il des reponses considérées comme fausses? C'est à dire qui ont au moins un bit faux par rapport aux réponses attendues

Pour cela, nous avons calculé le rapport bruit en sortie / bruit à l'entrée pour les expériences precédentes Bien que l'évolution de ce rapport diffère selon la tâche demandée et l'architecture utilisée, dans chaque cas on observe un phénomène qui peut être illustre en regardant conjointement la figure 5 et les courbes correspondantes de la figure 4 A partir de ces résultats, on peut calculer pour chaque expérience le nombre moyen de bits faux en sortie pour les formes non reconnues Cette valeur reste proche du nombre de bits faux en entree, bien que variant selon l'architecture et la taille de la memoire Pour les formes non reconnues, le réseau ne diminue donc pas le bruit en entrée

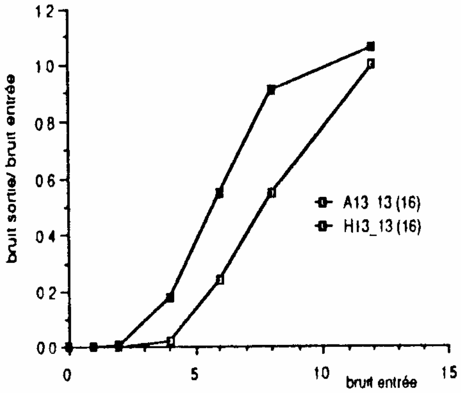

figure 5 évolution du rapport "bruit en sortie/ bruit à l'entrée" pour une architecture donnée (2 couches intermédiaires), une taille mémoire (16), une tâche d'auto-association et une tâche d'hétéro-association

Sur la figure 6, on a mis en évidence pour des apprentissages avec bruit (6, 4, 2, 1) l'évolution de la reconnaissance en fonction du temps et ce pour un certain nombre de bruits testés Les conditions de cette expérience sont semblables à celles de la figure 3 Les tâches qui sont réalisées le plus tôt, sont les moins "difficiles", c'est à dire celles qui nécessitent le moins de corrections L'ensemble des formes à distance i d'une forme mémorisée est une sphère centrée autour de cette forme Introduire du bruit dans l'apprentissage revient donc à apprendre une région de l'espace On remarque que cet apprentissage est réalisable a partir d'une présentation aléatoire de quelques éléments de cette région Une unité de temps sur cette figure correspond à une séquence d'apprentissage c'est à dire dans ce cas à 40 présentations par forme d'entree et par taille de bruit de l'association (forme d'entrée bruitée, réponse désirée)

MEMOIRES ASSOCIATIVES DISTRIBUEES: UNE COMPARAISON

DISTRIBUTED ASSOCIATIVE MEMORIES: A COMPARISON

F. Fogelman Soulié *, P. Gallinari**, Y. Le Cun ***, S. Thiria **

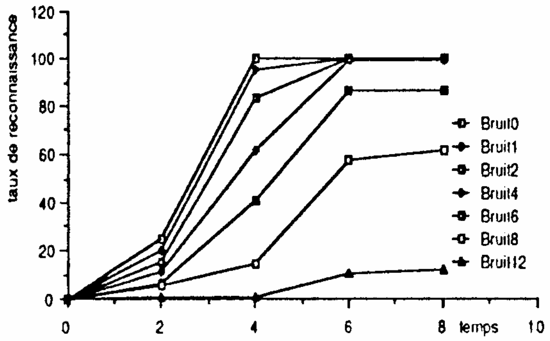

figure 6 évolution en fonction du temps des performances d une tâche A_13_13(16) On a tracé une courbe pour chaque taille de bruit imposée sur les formes en entree

Un facteur important dans une telle tâche est l'introduction de bruit pendant l'apprentissage. Les reseaux ont naturellement des propriétés de résistance au bruit, mais introduire une certaine image du bruit permet d'améliorer les performances Ce phénomène est mis en évidence sur la figure 7 ou figurent les performances d'un même réseau pour une même tâche mais pour deux types d'apprentissage L'apprentissage sans introduire de bruit, i e ou seules les associations pures ont été apprises, présente de moins bonnes propriétés de résistance au bruit De plus, après avoir atteint un optimum à un temps t'1 (au bout de t'1 passes de la base d'exemples), les performances du réseau, quand on lui presente des formes bruitées, décroissent si on continue l'apprentissage C'est ce que montre la courbe t'2, qui donne les performances à un temps t'2 ultérieur à t'1 L'autre apprentissage a été fait en introduisant des bruits de taille 6, 4, 2, 1 selon la procédure décrite précédemment. Dans ce cas l'apprentissage nécessite plus d'itérations pour obtenir une stabilisation des performances et celles-ci ne vont pas décroître si on continue la procédure Des essais pour introduire plus de bruit dans l'apprentissage n'ont pas conduit à de meilleures performances La procédure est dans ce cas beaucoup plus longue puisque le nombre d'associations est plus important D'autre part les régions de l'espace correspondant à l'apprentissage de formes trés bruitées sont beaucoup plus grandes donc plus difficiles à apprendre

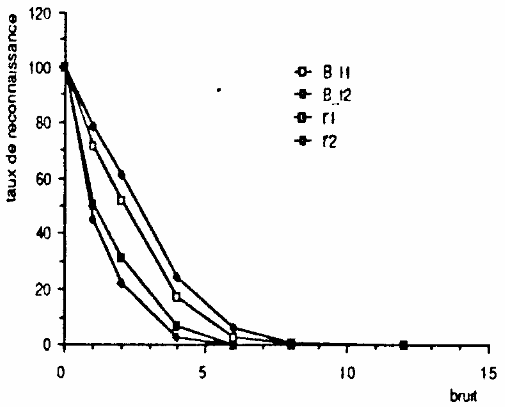

figure 7 comparaison des performances d'un réseau sur une tâche donnée pour un apprentissage sans bruit (courbes t'1 et t'2) et un apprentissage avec bruit (courbes B_t1 et B_t2) Dans chaque cas, on a relevé les performances a deux instants (t2 >t1 et t'2 >t'1) pendant l'apprentissage

6- CONCLUSION

Nous avons présenté plusieurs expériences portant sur la réalisation de mémoires associatives par des réseaux d'automates Ces tests nous renseignent sur le comportement encore peu connu de differents modèles et permettent de comparer leurs capacites pour une tâche de pure mémorisation Nous pensons que les propriétes mises en évidence s'étendent à un grand nombre de tâches d'association et donc que ce type d'etude permet de guider le choix entre différentes procédures utilisables Dans de nombreux cas bien sur on aura intérêt à adapter l'architecture au problème On doit évidemment, en fonction de la tâche demandée, trouver un compromis entre durée de l'apprentissage et performances A cet égard, l'algorithme GPB ne permet d'ailleurs pas, dans l'état actuel, l'apprentissage avec un nombre de couches trop élevé

7- BIBLIOGRAPHIE

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

[2] J A ANDERSON Cognitive capabilities of a parallel system dans [5], pp 109 226

[3] D H. BALLARD, G E HINTON T J SEJNOWSKI Parallel visual computation Nature, 306, pp 21 26, 1983

[4] D H BALLARD Parameter nets, Artificial Intelligence, 22 pp 235 267, 1984

[5] E. BIENENSTOCK F FOGELMAN SOULIE, G WEISBUCH Eds Disordered systems and biological organization, Springer Verlag, NATO Asi Series in Systems and Computer Science, F20 1986
[6] R O. DUDA, P.E HART Pattern Classification and Scene Analysis, J Wiley 1973
[7] F FOGELMAN SOULIE, G. WEISBUCH Random iterations of threshold networks and associative memory, à paraitre dans SIAM J. on Computing
[8] F FOGELMAN SOULIE, P. GALLINARI, S THIRIA Learning and associative memory à paraître dans Pattern Recognition, Theory and Applications P A. Devijver Ed., Nato Asi, Springer Verlag
[9] F FOGELMAN SOULIE, P. GALLINARI, Y LE CUN, S THIRIA Automata networks and artificial intelligence Dans «Computation on automata networks», F Fogelman Soulié Y Robert, M Tchuente Ed Manchester Univ Press, à paraitre
[10] F FOGELMAN SOULIE Le Connexionnisme, support de cours, MARI 87 Cesta Ed 1987
[11] T.N.E GREVILLE Some applications of the pseudo inverse of a matrix SIAM Rev 2, pp 15 22 1960
[12] G.E HINTON J A ANDERSON Eds Parallel Models of Associative Memory L.Erlbaum, 1981
[13] J J HOPFIELD: Neural Networks and Physical Systems with emergent collective computional abilities Proc Nat Acad Sc USA pp 2554-2558, 1982
[14] T KOHONEN· Self Organization and Associative Memory, Springer Verlag 1984.
[15] Y LE CUN A learning scheme for asymmetric threshold network dans «Cognitiva 85», Cesta Afcet Ed, t 2 pp 599 604 1985
[16] Y LE CUN· Learning Process in an Asymmetric Threshold Network dans [5] pp 209-226,1986
[17] M MINSKY, S PAPERT: Perceptrons, MIT Press, 1969
[18] D B PARKER Learning Logic Center for Computational Research in Economics and Management Science, MIT, TR 47, 1985
[19] C R. RAO, S K MITRA Generalized Inverses of Matrices and its Applications, Wiley 1971
[20] D E RUMELHART G E HINTON R J WILLIAMS Learning internal representations by error propagation Dans «Parallel Distributed Processing Explorations in the microstructure of cognition», Vol 1:Foundations, D E Rumelhart, J L McClelland Eds. MIT Press, 1986
[21] T DE SAINT PIERRE Codification et apprentissage de caractères multi polices MARI 87 Cesta Ed, 1987
[22] T J SEJNOWSKI, P K KIENKER, G E HINTON Learning symmetry groups with hidden units beyond the perceptron Physica 22 D 1986
[23] T J. SEJNOWSKI G.E. HINTON Separating figures from ground with a Boltzmann machine, dans «Vision, Brain and Cooperative Computation», M A Arbib, A R Hanson Eds, MIT Press, 1985
[24] T J SEJNOWSKI, C H ROSENBERG NETtalk a parallel network that learns to read aloud Johns Hopkins Technical report JHU/EECS 86/01
[25] B WIDROW H E HOFF Adaptive switching circuits 60 IRE WESCON Conv Record, Part 4, pp 96 104, Aug 1960