Réseaux multicouches de neurones artificiels : algorithmes d'apprentissage, implantations sur hypercube : applications

Shengrui Wang

Thèse

présentée par **Shengrui WANG**

pour obtenir le titre de **DOCTEUR**

de l'**INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE**
(arrêté ministériel du 5 juillet 1984)

Spécialité : **Informatique**

Réseaux Multicouches de Neurones Artificiels

Algorithmes d'apprentissage
Implantations sur Hypercube
Applications

Thèse soutenue le 26 septembre 1989

Composition du jury,

Président : J.P. VERJUS

Examinateurs : M. COSNARD
F. FOGELMAN-SOULIE
G. MAZARE
F. ROBERT
G. TIBERGHIEN

Thèse préparée au sein du laboratoire TIM3 de l'IMAG, U.A. au CNRS n°397.

A Zhanhong
A Ma grand mere et
mon grand pere
A Mes parents et mes amis

献给展红

献给我的祖母祖父

献给我的亲友

REMERCIEMENTS

Je tiens à remercier Monsieur J.P. VERJUS, Professeur à l'INP de Grenoble et Directeur de l'IMAG pour l'honneur qu'il me fait en présidant le jury de cette thèse.

J'adresse ma profonde gratitude à Monsieur F. ROBERT, Professeur à l'ENSIMAG, qui m'a dirigé pendant mes trois ans de thèse. Ses conseils et la confiance qu'il m'a témoignée m'ont permis de mener à bien ce travail. Mes quatre années de séjours à Grenoble m'ont permis d'apprécier ses qualités scientifiques et humaines.

Je voudrais remercier Madame F. FOGELMAN-SOULIE, Professeur à l'Université de Paris-Sud, pour sa gentillesse et le temps qu'elle m'a consacré, et pour avoir accepté d'être rapporteur de cette thèse.

Je dois ma reconnaissance à Monsieur M. COSNARD, Professeur à l'ENS de Lyon, pour sa contribution à la création, au sein de l'Equipe d'Algorithmique Parallèle et de Calcul Formel, de l'ambiance scientifique et humaine dont j'ai pu bénéficier. Je le remercie également pour l'intérêt avec lequel il a suivi ce travail et pour son acceptation d'être rapporteur de cette thèse.

Je remercie vivement Monsieur G. TIBERGHIEN, Professeur à l'Université des Sciences Sociales de Grenoble, pour la collaboration dont j'ai pu bénéficier avec son équipe et pour l'honneur qu'il me fait en participant au jury de cette thèse.

Mes remerciements vont également à Monsieur G. MAZARE, Professeur à l'ENSIMAG, pour l'intérêt qu'il porte à mes travaux et pour l'honneur qu'il me fait de participer à ce jury.

Je ne peux pas oublier les personnes avec qui j'ai eu beaucoup de plaisir à travailler. Que Gérard BAILLY, Thierry BESSON, Tao PHAM DINH, Stéphane ROUSSET, Anne-Caroline SCHREIBER, Adnan YASSINE et Haiyen YE trouvent ici mes sincères remerciements. Je dois remercier en particulier Anne-Caroline SCHREIBER et Stéphane ROUSSET qui ont eu la gentillesse et le courage de lire la première version de cette thèse et de corriger les nombreuses fautes de français.

Que tous les membres de l'Equipe d'Algorithmique Parallèle et de Calcul Formel reçoivent mes profonds remerciements pour leur sympathie et leur aide dans la recherche et dans la vie quotidienne.

Je dois remercier ma chère Zhanhong qui, tout au long de la préparation de cette thèse, m'a apporté encouragements, soins et aides techniques. Elle a fait autant d'efforts et de sacrifices que moi pour assurer le bon déroulement de ces travaux.

Enfin, je remercie Madame Brigitte VIVION et tous les membres du service de la scolarité de l'INPG et du service de reprographie de l'IMAG pour la réalisation finale de cette thèse.

Table des matières

Introduction

Un réseau de neurones artificiels est un système de calcul distribué dont les composants élémentaires sont des unités de calcul qui reçoivent en permanence des entrées et qui produisent des sorties. On utilise le terme "neurone" parce que le fonctionnement de ces unités est inspiré de certains effets neurophysiologiques observés dans le cerveau humain. Généralement, une telle unité calcule la somme pondérée de ses entrées; sa sortie est une fonction de cette somme calculée, fonction qui varie selon les modèles de réseau. Un réseau est constitué par l'interconnexion de ces unités (que l'on appelle souvent "cellules" ou "neurones artificiels"). Les entrées de chaque cellule peuvent être celles du réseau ou des sorties issues d'autres cellules. La sortie de chaque cellule peut être dirigée vers d'autres cellules ou simplement constituer une partie de la sortie du réseau.

Il est assez difficile de donner une description générale des modèles de réseaux en quelques phrases, car il en existe de nombreuses variétés. Du point de vue structurel, il existe certains modèles, comme le modèle de réseau multicouches (Fogelman-Soulie *et al.*1987, Rumelahart *et al.* 1986) et la machine de Boltzmann (Hinton, Sejnowsky & Ackley 1984), où l'on distingue cellules *exposées* (en interaction avec le système extérieur) et cellules *cachées* (n'ayant de contact qu'avec d'autres cellules). Dans d'autres modèles, il n'existe pas de distinction entre les cellules d'entrée, de sortie et les cellules cachées, comme dans celui de Kohonen (Kohonen 1984) et de Hopfield (Hopfield 1982). Quant à leur fonctionnement, la différence est encore plus marquée.

Premièrement, le fonctionnement de chaque cellule est variable. Dans les réseaux de Kohonen, la fonction de la décision est linéaire; dans la machine de Boltzmann elle est stochastique; tandis que dans des réseaux de Hopfield, le Perceptron (Minsky & Papert, 1969) et les modèles multi-couches, la fonction de décision est à seuil discret ou continu (sigmoïde).

Deuxièmement, le calcul de la sortie du réseau est aussi très variable. Dans les réseaux de Hopfield, la sortie est l'état stable d'un processus itératif. Dans la machine de Boltzmann, la sortie est le résultat d'un processus stochastique (recuit simulé) qui minimise une certaine fonction de critère. Alors

que dans le Perceptron et les réseaux multi-couches, la sortie est simplement constituée par l'état des cellules en sortie, qui est la conséquence de la progression des états des cellules d'entrée à travers les couches. Tous les réseaux mentionnés jusqu'ici travaillent en temps discret, mais il existe aussi des réseaux qui travaillent en temps continu, tels les Avanches (Grossberg, 1967, 1988) de Grossberg qui sont modèlisés par des équations différentielles. Dans ce dernier cas, les sorties sont les états stables du réseau ou l'ensemble limite des solutions (des équations différentielles).

Finalement ces modèles se distinguent entre eux par la façon dont les paramètres définissant le fonctionnement d'un réseau se construisent. On utilise souvent le terme "apprentissage" (par le réseau), parce qu'un tel ensemble de paramètres devrait être trouvé, souvent de manière adaptative, de façons à ce que le réseau réalise une fonction désirée. Dans certains modèles, comme celui de Hopfield, l'apprentissage n'est pas un problème parce qu'il existe un algorithme direct qui permet de calculer tous les paramètres nécessaires en un seul coup. Mais dans certains autres, l'apprentissage consiste en un processus d'adaptation (très lent) des paramètres pour satisfaire les contraintes imposées par un "superviseur". Ce genre d'apprentissage, souvent appellé "apprentissage dirigé" est utilisé par la machine de Boltzmann et les réseaux multi-couches. Récemment des recherches ont été entreprises sur l'apprentissage non-dirigé, qui consiste à laisser un réseau évoluer tout seul selon une loi préconstruite, sans imposer aucune contrainte. Si la loi d'évolution est bien choisie (par exemple. celle de Hebb, (Hebb 1949)), on peut observer des configurations très intéressantes de l'ensemble des paramètres, obéissant à des lois biologiques et physiques.

"A quoi servent les réseaux de neurones ? "

Les réseaux de neurones constituent une approche importante du traitement de l'information. Il y a deux caractéristiques communes aux réseaux neuronaux qui les rendent particulièrement intéressants. Ils sont *adaptatifs* et *massivement parallèles*. L'application des réseaux de neurones recouvre plusieurs domaines tels que:

La mémoire : qui consiste à compléter l'information à partir d'indices partiels et/ou bruités. L'information obtenue sert d'identification de l'entrée;

Le traitement des informations sensorielles : qui concerne spécialement les prétraitements de bas niveau des informations auditives ou visuelles comme dans

le cerveau humain ;

La catégorisation : qui consiste à apprendre des *patterns* (nous choisissons d'utiliser le mot "pattern" plutôt que "forme" ou "motif") ayant des étiquettes prédéfinies, de telle manière qu'un pattern d'entrée quelconque pourrait être "identifié" soit comme un des patterns appris de par l'étiquette qui lui sera accordée par le réseau, soit comme un pattern jamais recontré, "je ne sais pas à quoi il correspond";

L'auto-organisation et la formation de catégories : qui permettent de partitionner les patterns d'entrée en groupes ou "clusters" en utilisant des données sans étiquettes ;

Les problèmes d'optimisation combinatoire : l'exemple le plus cité est la résolution du problème du voyageur de commerce par le modèle de Hopfield.

Les applications concrètes sont déjà très nombreuses. Parmi elles, on peut citer les travaux de Sejnowski & Rosenberg (1985) sur un système texte-à-parole (NETtalk); de Le Cun sur le diagnostic médical (1987); de Rousset, Schreiber & Wang (1988) sur la mémoire de visages; de Watrous (1988) et Weibel (1988) sur la reconnaissance de la parole; de Grossberg & Marshall (1989) sur la modélisation du cortex, etc. Nous avons l'impression que les réseaux de neurones peuvent être appliqués dans tous les problèmes d'association et de classification ayant besoin d'une représentation distribuée des connaissances et un traitement parallèle des informations.

Quel est l'avantage des réseaux sur les méthodes classiques ? L'apprentissage automatique (par des exemples et non par des règles) nous a permis de modéliser les effets cognitifs qui n'ont pas pu être réalisés par des systèmes computo-symboliques (Schreiber, *et al.*1989, voir le Chapitre III). Mais si l'on raisonne sur les indices physiques, tels le temps pour l'apprentissage, le taux de reconnaissance, etc, il semble que les réseaux de neurones ne peuvent pas encore atteindre une performance comparable avec celle des méthodes classiques. Bien que les réseaux de neurones soient capables de résoudre pas mal de problèmes difficiles, ils continuent de subir des critiques qui viennent en général des checheurs de l'I.A. (Intelligence Artificielle) traditionelle, de biologistes et de neurophysiologistes.

Relation avec l'I.A. :

Il faut noter que l'I.A. et les réseaux de neurones s'adressent à un même type de problèmes -établir des systèmes artificiels pour simuler l'intelligence humaine. Malgré des travaux considérables de l'I.A. durant ces vingt dernières années, ces idées restent encore dans le domaine de recherche. En fait, l'I.A. ne présente qu'une faible structure (ou schéma) théorique pour diriger la résolution de problèmes (*problem solving*). Ses succès sont largement basés sur des techniques heuristiques. Les projets d'I.A. se traduisent souvent par des programmes informatiques constituant une approche heuristique du problème posé, comme par exemple la plupart des systèmes experts le font. Cependant, les réseaux de neurones, eux aussi, ont peu de structure théorique; et la plupart des travaux actuels consistent à établir un modèle de réseaux permettant de résoudre un problème posé. On comprend souvent mal comment les résultats obtenus pourraient s'intégrer dans un cadre théorique.

La différence principale entre l'I.A. et l'approche réseaux de neurones réside dans la façon dont les connaissances sont acquises. Dans l'I.A., en particulier les systèmes experts, il faut une collection importante de connaissances des experts pour encoder "l'expertise" humaine de résolution des problèmes. Les systèmes basés sur les réseaux de neurones ressemblent plus aux systèmes biologiques du point de vue de l'acquisition des connaissances : Ils apprennent par des "expériences" répétées.

En fait, des notions telles que l'apprentissage et le traitement des informations sont difficiles à préciser, car elles concernent des réalités extrêmement diverses que l'on sait mal formaliser. L'approche "bas niveau" que constituent les réseaux de neurones et celle de "haut niveau" de l'I.A. nécessitent, toutes les deux, des études plus approfondies. Leur réconciliation sera indispensable et fructeuse (Amy & Tiberghien, 1988).

Deux grandes lignes de recherche dans les réseaux de neurones :

Les réseaux de neurones formels les plus efficaces sont des systèmes non linéaires. La non-linéarité, qui est à l'origine de la puissance de ces systèmes, constitue en revanche l'obstacle principal pour l'apprentissage et pour

l'interprètation des résultats. Pour réussir des applications, il faut avoir une bonne connaissance pratique du comportement dynamique des réseaux, et une bonne maitrise des algorithmes d'apprentissage. Les techniques que l'on pourrait emprunter à l'algèbre, à l'analyse mathématique et/ou à l'analyse statistique ne permettent pas encore de résoudre les problèmes posés. Les approches actuelles s'avèrent donc très expérimentales et empiriques.

Un autre problème critique dans la recherche en réseaux de neurones concerne les outils et les techniques de simulation. Dans des laboratoires, on utilise en général des machines conventionelles telles VAX, PC, SUN, etc. Sur ces machines, on peut développer des logiciels sophistiqués qui permettent de simuler n'importe quelle architecture de réseaux. Or la puissance de calcul de ces machines est trop faible pour supporter des applications de taille importante. L'implantation directe des réseaux sur des circuits (VLSI) n'est intéressante que pour quelques applications limitées, parce que les circuits, une fois fabriqués, ne peuvent généralement plus être reconfigurés. Un bon compromis est fourni par l'utilisation des machines massivement parallèles telles la Connection Machine, les machines d'hypercube, le Butterfly, T-node etc. L'implantation des réseaux de neurones sur des machines parallèles constitue un problème intéressant mais encore mal résolu..

Les applications effectuées à ce jour sont tellement variées que l'on ne peut plus douter du potentiel des réseaux de neurones. Cependant, il faudra faire encore beaucoup d'efforts tant en études théoriques qu'en implantations pratiques avant de prétendre maitriser réellement le domaine.

Plan de la thèse :

Les travaux de cette thèse s'inscrivent dans un cadre d'expérimentation du modèle de réseaux multicouches. Ils concernent l'implantation du modèle sur la machine hypercube FPS, les applications en mémoire des visages et en reconnaisance des mots parlés (parole), l'étude du comportement des réseaux et l'amélioration de l'algorithme d'apprentissage. Ces travaux, concernant différents aspects de la recherche en réseaux de neurones, sont présentés en cinq chapitres comme suit :

Le Chapitre I est consacré à l'introduction des réseaux de neurones

multicouches. Nous allons présenter l'algorithme de rétro-propagation de gradient (noté GBP pour Gradient Back-Propagation, familièrement appelé "Backprop") dans le contexte de la minimisation d'une fonction d'erreur; puis indiquer les applications qui peuvent être envisagées directement. Une grande partie du chapitre est utilisée pour présenter une étude du comportement dynamique des réseaux fonctionnant en GBP. Cette étude, inspirée de notre modélisation de la mémoire des visages, fournit des indices plus clairs sur la dynamique des fonctions réalisées par réseaux (après apprentissage), plus particulièrement en ce qui concerne la création d'attracteurs. L'extension de l'algorithme GBP en réseaux cycliques ouvre une nouvelle perspective au problème du traitement de séquences temporelles (dont l'importance est maintenant bien reconnue). Le chapitre se termine par la présentation d'un modèle de réseaux multicouches assez différent de ceux basés sur l'algorithme de GBP. Notre intention est de montrer l'idée astucieuse de construire des réseaux puissants par la combinaison de réseaux relativement simples.

Le Chapitre II est entièrement consacré à l'implantation de réseaux de neurones multicouches (GBP) sur la machine hypercube FPS. L'intérêt de simuler des réseaux de neurones sur une telle machine (parallèle) réside dans le compromis entre la vitesse de calcul et la flexibilité d'utilisation (reconfigurabilité du réseau implanté). L'implantation de réseaux de neurones sur des machines parallèles (qui constitue en soi un sujet intéressant) permet d'évaluer les performances de différentes architectures pour cette classe d'applications, et de mieux comprendre la nécessité architecturale dans la conception de machines parallèles spécialisées en réseaux de neurones.

Le Chapitre III présente une tentative d'utilisation du modèle connexionniste comme nouveau paradigme de simulation de phénomènes cognitifs. Nous exposons une simulation de l'identification des visages en contexte, qui intègre le rôle dynamique des informations contextuelles dans l'élaboration de l'identité. Les résultats présentés dans ce chapitre montrent que le modèle connexionniste, en tant qu'outil informatique, permet de simuler plusieurs effets dans la reconnaissance des visages qui n'ont pas été, jusqu'à présent, très bien pris en compte par les systèmes de type computo-symbolique. Cette application a été développée en collaboration avec le Laboratoire de Psychologie Expérimentale de l'Université de Grenoble II (Tiberghien, Schreiber et Rousset).

Le Chapitre IV concerne une application des réseaux multicouches à la reconnaissance de mots isolés. L'objectif du chapitre est d'étudier le besoin architectural des réseaux multicouches pour des applications en traitement de la parole. Nous nous intéresserons spécialement au problème du traitement d'informations temporelles. L'architecture à connexions partielles et superposées, que nous proposerons, permet de détecter de manière efficace la régularité dans les variations temporelles. Cette application a été développée en collaboration avec l'Institut de la Communication Parlée de l'ICP/INPG (H.Yé et G. Bailly).

Le Chapitre V est consacré à l'amélioration de l'algorithme de GBP. C'est un sujet difficile car l'apprentissage consiste à minimiser une fonction non convexe. Pour ce genre de problème, il est pratiquement impossible d'obtenir des informations sur la forme globale de la fonction. Il existe deux approches principales : L'une plus heuristique consiste à adapter de manière dynamique certains paramètres d'apprentissage; l'autre, plus systématique, consiste à utiliser des méthodes d'optimisation du second ordre. Nous exposerons d'abord les idées générales de la première approche, et présenterons ensuite une adaptation de la méthode des Régions de Confiance (RC) au problème de l'apprentissage dans un réseau multicouches. L'application de la méthode RC a été développée en collaboration avec Pham Dinh Tao et A.Yassine de l'équipe d'Optimisation du Laboratoire TIM3.

Chapitre I

Une introduction aux réseaux multicouches

L'algorithme GBP (Gradient Back-Propagation) pour le modèle des réseaux multicouches n'a commencé à devenir célèbre qu'en 1985, grâce aux travaux de Le Cun et de Rumelhart et Hinton. Actuellement, c'est l'un des algorithmes les plus importants pour les réseaux de neurones. Les expériences principales rapportées dans cette thèse sont des applications de ce modèle.

Jusqu'au début des années 80, la plupart des modèles connexionnistes consistaient en des réseaux monocouches, qui ne contenaient donc qu'une seule couche de cellules (de traitement) dont la fonction est modifiable par apprentissage. On peut distinguer, par exemple, le Madaline de Widrow & Hoff, (1960), le Perceptron de Rosenblatt (1958), le Modèle linéaire de Kohonen (1984) pour la mémoire associative, etc. Ces modèles sont munis d'algorithmes d'apprentissage basés sur la théorie de la minimisation (pour le Madaline et le Perceptron) ou sur la théorie de la corrélation (notamment les matrices pseudo-inverses pour le modèle de Kohonen). Les travaux de Nakano (1972), Amari (1972), Kohonen (1984), Hopfield (1982, 1984 & 1985) et Fogelman (1985a) sur les modèles de mémoires associatives et sur les réseaux d'automates permettent, dans une grande mesure, de clarifier le comportement dynamique de ces réseaux.

Bien que les réseaux monocouches soient déjà largement utilisés dans beaucoup d'applications comme la reconnaissance de visages et le traitement de signal, leur efficacité pour traiter les problèmes compliqués est très limitée. Les théorèmes de limitation du Perceptron, mis au point par Minsky et Papert (1969), mettent en évidence la relation entre la complexité de certains problèmes (fonctions de parité) et le rôle de la fonction réalisée pas des unités d'association (pour un prétraitement des entrées). Il est facile d'en tirer la conclusion suivante : un Perceptron ne peut apprendre des fonctions comme XOR et des fonctions de parité, sauf s'il est conçu spécialement pour ça. Ceci signifie qu'il faut construire la couche d'unités d'association tel que l'ensemble des entrées devienne un ensemble de patterns qui soient linéairement séparables, car chaque cellule de sortie ne peut réaliser qu'une séparation linéaire. Cette couche d'association revêt donc une importance capitale pour que le Perceptron puisse réaliser la fonction

demandée. Cette limitation est en fait générale pour tous les réseaux monocouches.

L'idée de base des réseaux multicouches est tout simplement de décomposer une tâche en plusieurs étages, dont chacun est censé être plus simple à résoudre. En fait, le Perceptron est un des premiers qui emploient cette idée, la couche d'association étant une couche de traitement intermédiaire. La différence principale entre le Perceptron et un réseau multicouches, qui nous concerne actuellement, réside dans la façon dont la fonction des cellules intermédiaires est générée. Les cellules d'association dans le Perceptron sont codées à la main, tandis que la fonction des toutes les cellules intermédiaires dans un réseau multicouche pourrait être générée automatiquement par un algorithme d'apprentissage.

Bien que ce ne soit pas le premier algorithme d'apprentissage pour les réseaux multicouches, l'algorithme de GBP est pourtant le premier qui résolve ce problème de génération de façon relativement satisfaisante. Avant son apparition, on connaissait déjà le principe de la Machine de Boltzmann qui repose sur un algorithme stochastique. Ce dernier, découvert par Hinton, Sejnowsky et Ackley (1984), a été largement utilisé dans le traitement de la parole. Mais à cause de sa faible efficacité, il a fait progressivement place à celui de GBP. Ces dernières années ont vu apparaître plusieurs variantes de GBP, et des modèles différents, qui sont tous présentés comme étant améliorations du GBP. Le nombre des publications est explosif, et la recherche évolue très vite. C'est la raison pour laquelle les travaux de cette thèse seront présentés directement à partir de l'algorithme de GBP en dispense d'un rappel historique du développement de réseaux de neurones. La these de Le Cun (1987) et celle de Thiria (1989) donnent une excellente synthese des traveaux précédents.

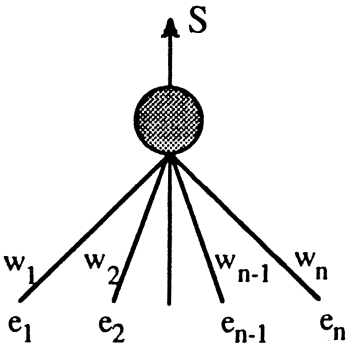
Dans ce chapitre, nous commencerons d'emblée par une introduction de l'algorithme de GBP (paragraphe I.1), suivie par une discussion de quelques sujets généraux concernant le modèle (paragraphe I.2). Dans le paragraphe I.3, nous présenterons une étude du comportement dynamique des réseaux basés sur l'algorithme GBP. Les résultats théoriques dans ce paragraphe sont démontrés pour les réseaux monocouches, mais nous montrerons des résultats expérimentaux concernant les réseaux multicouches. Nous présenterons dans le paragraphe I.4 une variante importante du modèle initial : les réseaux

séquentiels, toujours basés sur l'algorithme de GBP. Ils sont conçus spécialement pour apprendre des séquences et sont donc adaptés aux problèmes du traitement de la parole continue. Les résultats expérimentaux seront exposés plus tard, dans le Chapitre IV. Le Chapitre I se terminera par la présentation d'un modèle de réseaux multicouches assez différent de ceux basés sur l'algorithme de GBP. Ces réseaux utilisent des cellules dont le "champ de réception" est local (ou limité). Intuitivement, ce modèle est plus adapté aux problèmes de traitement d'images, mais le point le plus important de cette approche est l'idée de combiner deux modèles simples pour en construire un plus "puissant".

I.1 L'algorithme de rétro-propagation du gradient (GBP)

I.1.1 Formalisation de réseaux à cellules sigmoïdales

Un réseau multicouches est un réseau de cellules interconnectées et réparties en couches.



$S = f(A)$ où
$A = \sum_i w_i * e_i - b$: l'entrée totale
$f(x) = (1-\exp(-\alpha x)/(1+\exp(-\alpha x)$

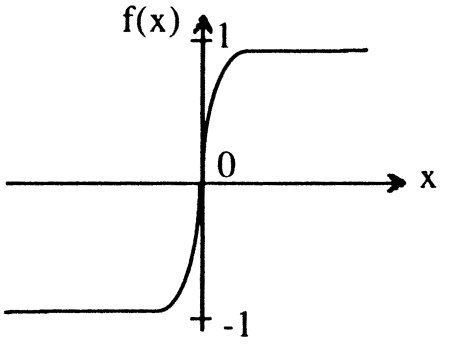


Figure 1.1 : La cellule élémentaire pour construire des réseaux. e_i représente une entrée de la cellule. Elle pourrait être un composant de l'entrée au système ou la sortie d'une autre cellule. w_i est le poids "synaptique" du $i^{ème}$ lien. A_i, somme pondérée des entrées, est appellé entrée totale. La fonction f est une fonction sigmoïde caractérisée par le paramètre α qui définit la pente de la fonction.

Chaque cellule est une unité de calcul et peut être assimilée à un dipositif ayant plusieurs liens d'entrée et une ou plusieurs sorties. Le fonctionnement d'une cellule sigmoïdale est détaillé dans la Figure 1.1. La e_i représente l'entrée à travers le lien i et le w_i est le poids du liens (ou de la connection). L'entrée totale

d'une telle cellule est calculée comme la somme pondérée de toutes les entrées (produit scalaire entre le vecteur de poids et le vecteur des valeurs d'entrée), soustraite par un seuil b qui est propre à la cellule. L'état ou la sortie de la cellule, deux notions dont la différence est souvent négligée, est calculée comme une fonction sigmoïde de son entrée totale

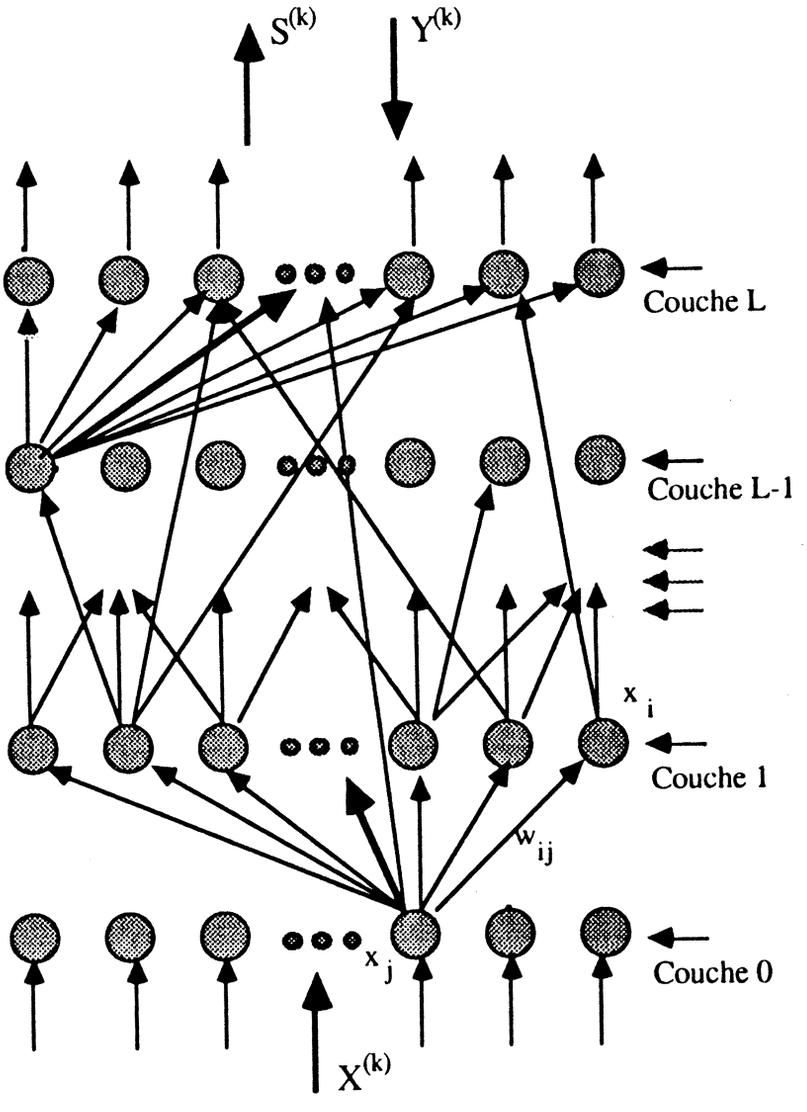


Figure 1.2 : Schéma général d'un réseau multicouches avec L+1 couches de cellules. Les flèches représentent la direction de la propagation des états lors de calcule la sortie du réseaux $S^{(k)}$ pour l'entrée $X^{(k)}$. $Y^{(k)}$ est la sortie désirée correspondant à $X^{(k)}$. L'erreur exprimée par $S^{(k)}$-$Y^{(k)}$ sera rétro-propagée, c'est-à-dire, propagée dans le sens inverse des flèches.

Le réseau est construit par interconnection des cellules, en les structurant en couches (Figure 1.2). On suppose, pour le moment, que les cellules des couches du haut ne peuvent recevoir leurs entrées qu'à partir des cellules des

couches du bas. Le poids de la connexion de la cellule j à la cellule i est représenté par w_{ij}. En général, $w_{ij}=0$ signifie que la cellule i ne reçoit pas d'information de la cellule j, mais le fait que la cellule i ne reçoit pas d'information de la cellule j peut signifier aussi qu'il n'y pas de lien entre ces deux cellules. Donc ici on distingue le cas où $w_{ij}=0$ et le cas où les deux cellules sont déconnectées, puisque dans le premier cas w_{ij} est susceptible d'être modifié alors que dans le dernier il ne peut pas l'être.

Un réseau multicouches peut encore être considéré comme un graphe orienté, dont chaque arête est pondérée par une valeur réelle. Les sommets du graphe correspondent aux cellules (les unités de traitement) et les arêtes pondérées correspondent aux connexions entre les cellules. Dans la version originale de GBP, on suppose qu'il n'y a pas de circuit dans le graphe, ce qui veut dire comme dans le paragraphe ci-dessus, que les informations se propagent dans un sens unique pendant une phase de calcul. L'algorithme de GBP sera déduit sous cette contrainte architecturale. C'est dans le paragraphe I.3 que l'on présentera une extension de l'algorithme à une structure cyclique des réseaux, que l'on appelle les réseaux séquentiels (Jordan, 1986).

Le terme "multicouches" provient du fait que les cellules peuvent être regroupées dans une structure en couche, où deux cellules entrent dans une même couche si et seulement s'il n'y a pas de chemin dans le graphe qui les relie. Par convention, on suppose qu'il y a une couche de cellules qui reçoit des entrées de l'extérieur du réseau (couche 0), et il y a aussi une couche de cellules qui envoie leurs sorties à l'extérieur (couche L). Les autres couches de cellules, n'ayant aucune interaction directe avec l'extérieur du réseau, sont appelées "couches cachées". Les cellules de ces couches sont des "cellules cachées" ou "unités cachées".

Le principe de fonctionnement du réseau est le suivant : Etant donné un pattern d'entrée sous forme d'un vecteur (qui pourrait représenter une unité d'informations réelles), les cellules dans la couche 0 prennent les composants du vecteur d'entrée comme leurs états; puis chaque cellule dans les couches suivantes calcule son état dès que toutes les cellules qui sont connectées en dessous ont fini la mise à jour de leurs états. On pourrait imaginer une propagation successive des états : d'abord ce sont les cellules de la couche 1 qui calculent leurs états, ensuite les cellules de la couche 2, ainsi de suite... A la fin de cette propagation, le vecteur

récupéré, constitué des états des cellules dans la dernière couche, est considéré comme la sortie du réseau. Il est évident que l'objectif de la construction du réseau est de rendre la sortie (qui dépend de l'entrée) significative.

I.1.2 Enoncés des problèmes

Le problème de l'apprentissage peut être d'abord énoncé comme suit.

Mémorisation des "patterns":

Supposons que l'on ait K associations de patterns (les couples d'entrée-sortie) à réaliser : $(X^{(k)}, Y^{(k)})$ pour k=1,...,K. Ici $X^{(k)}$ appartenant à $[-1, 1]^n$, $Y^{(k)}$ à $[-1, 1]^m$, représentent respectivement les informations à classifier et les "étiquettes" représentant ces informations. $X^{(k)}$ est souvent choisi comme un des exemples les plus "représentatifs" d'un ensemble d'informations. Bien sûr ils peuvent être plusieurs, et dans ce cas, les $Y^{(k)}$ correspondants doivent être les mêmes. Etablir un réseau qui réalise cet ensemble d'associations signifie :

1° Choisir une architecture du réseau, ce qui comprend le nombre de couches, le nombre de cellules dans chaque couche, le mode de connexion entre les cellules (existe-t-il une connexion d'une cellule à une autre ?).

2° Trouver les poids des connexions.

Pour le premier problème, il n'y a pas de théorie qui nous permette de faire un choix dit optimal. La solution est souvent très empirique. On choisit généralement beaucoup trop de cellules pour augmenter la dimension de l'espace de décision afin de pouvoir réaliser facilement les associations désirées. Dans les chapitres suivants, nous montrerons, à l'aide des applications, les principes de choix de l'architecture en référence aux fonctionnalités dont on a besoin.

Pour le deuxième problème, il existe certaines méthodes dérivées de celles conçues pour les problèmes d'optimisation. La plus connue est fondée sur la méthode du gradient.

Généralisation :

La performance de généralisation est considérée comme une des mesures les plus importantes de "l'INTELLIGENCE" acquise par un réseau à

travers l'apprentissage. En pratique, il s'agit de tester les patterns qui ne sont pas dans le corpus d'apprentissage. En fonction des besoins de l'application, on peut obtenir des critères pour juger de la capacité du réseau. Par exemple, pour un problème de classification d'images, on pourrait espérer que le réseau puisse associer chaque image d'entrée à la classe à laquelle il appartient, même s'il n'a pas été réellement "vu" par le réseau au cours de l'apprentissage. En général, la notion de généralisation concerne un processus de recherche et de prise de décision qui permet d'aboutir à une sortie qui correspond à la réalité. C'est une notion de base pour l'intelligence des systèmes construits à l'aide des réseaux de neurones.

L'objectif principal de l'apprentissage est donc de rendre un réseau capable de généralisation. Or la notion de généralisation est très difficile à formaliser car elle dépend des applications réelles, où il existe rarement des critères stricts qui nous permettraient de mesurer mathématiquement la pertinence des informations d'output. Pourtant, il y a des méthodes empiriques pour tester la capacité de généralisation, par exemple celle qui calcule le pourcentage des "bonnes" réponses pour des "patterns jamais appris" sans trop se soucier de la pertinence de ces "bonnes réponses", ni de la représentativité des patterns testés.

Cependant, on se pose souvent la question de la relation entre la capacité de généralisation et la performance d'apprentissage, qui est caractérisée par le degré avec lequel le réseau a réalisé les associations désirées. Dans les applications, on fait beaucoup d'efforts pour construire un réseau qui exhibe la généralisation que l'on espére. On sait que l'architecture du réseau et la représentation des données jouent des rôles très importants.

Quels que soient les algorithmes, l'apprentissage consiste à chercher dans l'espace des paramètres, la (ou les) solution(s) qui permet au réseau de satisfaire le mieux possible les contraintes imposées par les exemples d'apprentissage. La dimension de cet espace est en relation étroite avec la vitesse d'apprentissage et la capacité de généralisation. Normalement, si la dimension est relativement petite et que les exemples sont encore "apprenables" par le réseau, alors la vitesse d'apprentissage devrait être relativement lente, mais la capacité de généralisation sera bonne car les paramètres ainsi trouvés devraient pouvoir mieux exploiter la régularité de l'ensemble des exemples. Ceci, pourtant, varie

beaucoup selon l'architecture du réseau, dépendant en général du nombre de cellules dans chaque couche, la façon dont les cellules sont connectées entre elles. Deux réseaux ayant la même dimension pour leurs espaces des paramètres pourraient manifester des comportements très différents si leurs architectures sont différentes. Pour concervoir une architecture "intelligente", il faut, d'une part avoir une bonne maitrîse du fonctionnement, de la capacité et du comportement dynamique du réseau et d'autre part essayer d'obtenir des connaissances les plus détaillées possibles de l'étendue dans l'espace et/ou dans le temps de la fornction à réaliser. L'application des réseaux de neurones devient donc un véritable "casse tête" pour trouver l'architecture adéquate. La recherche de bonne architecture constitue une des plus importantes préoccupations dans nos applications du modèle de réseaux neuronaux.

Le problème de l'apprentissage comme un problème d'optimisation :

L'algorithme de GBP, peut être obtenu en représentant le problème d'apprentissage comme un problème d'optimisation, que l'on résoudra par application de l'algorithme du gradient.

Dans un réseau multicouche comme celui présenté dans la Figure 1.2, l'état d'une cellule est déterminée par

$$x_i=f_i(A_i) \qquad A_i=\sum_j w_{ij}*x_j-b_i \qquad (1.1)$$

ici f_i est une fonction sigmoïde dont la forme est (Figure 1.1)

$$f_i(x)=\frac{1-e^{-\alpha x}}{1+e^{-\alpha x}} \qquad (1.2)$$

où $\alpha>0$ est un paramètre qui caractérise la forme sigmoïdale de la fontion f_i. Il peut être différent pour des cellules différentes. b_i est le seuil de la cellule i. On suppose, par convention, qu'il y a une cellule dont l'état est égal à la constante 1, et qui est connectée à toutes les autres cellules. b_i sera considéré comme le poids de la connexion entre cette cellule et la cellule i. Les formules dans (1.1) s'écriront donc :

$$x_i=f_i(A_i) \qquad A_i=\sum_j w_{ij}*x_j \qquad (1.1')$$

La fonction de coût, qui dépend de l'ensemble des poids de connexions,

est définie par

$$C(W)=\sum_{k=1}^{K} \left\| S^{(k)}-Y^{(k)} \right\|_2^2 \qquad (1.3)$$

avec $S^{(k)}$ qui représente la sortie effective du réseau pour l'entrée $X^{(k)}$.

I.1.3 L'algorithme de rétro-propagation du gradient (GBP) (Rumelhart, Hinton & Williams 1986; Le Cun, 1987)

Appliquer la méthode du gradient consiste à calculer les dérivées partielles de la fonction C par rapport à tous les poids w_{ij}, et à modifier w_{ij} selon la formule :

$$w_{ij}(t+1)=w_{ij}(t)-\gamma \frac{\partial C}{\partial w_{ij}} \qquad (1.4)$$

où t représente tout simplement l'instant et γ doit être calculé afin de minimiser la fonction C dans la direction du gradient. Le procédé doit être répété plusieurs fois pour que C(W(t)) atteigne un minimum (local ou global). C'est un apprentissage global puisque l'amélioration des poids des connexions s'effectue suivant les "erreurs" que le réseau "commet" pour l'ensemble des patterns exemplaires.

On voudrait que le réseau puisse adapter ses poids petit à petit pour chaque association. Si on reprend C(W) sous la forme :

$$C(W)=\sum_{k=1}^{K} C^{(k)}(W) \qquad \text{avec } C^{(k)}(W)=\left\| S^{(k)}-Y^{(k)} \right\|_2^2 \qquad (1.5)$$

et qu'on applique la méthode d'optimisation sur $C^{(k)}$ une fois seulement à chaque étape, on calcule en fait :

$$w_{ij}(t+1)=w_{ij}(t)-\gamma_t \frac{\partial C_t}{\partial w_{ij}} \qquad (1.6)$$

C_t représente en fait la fonction $C^{(k)}(W(t))$ qui dépend à la fois de l'ensemble des poids W(t) à l'instant t et des patterns $X^{(k)}$ et $Y^{(k)}$ présentés à cet instant. Pour la simplicité, γ_t ici est choisi comme un petit réel positif, ou une suite décroissante de réels positifs. En terme de la méthode d'optimisation, c'est la méthode du gradient décomposé. On obtient ainsi l'ensemble des poids W(t+1) disponibles pour l'instant t+1.

Le problème crucial de ce procédé est la convergence! Il n'y a pas

d'étude théorique qui l'affirme, mais l'experience montre que l'on pourrait effectivement conduire C(W(t)) vers un minimum (souvent zéro) par le choix approprié de l'architecture du réseau, de l'ordre de présentation des patterns et du taux d'apprentissage γ. Ces choix sont très empiriques.

Un problème pratique se pose aussi pour le calcul de $\frac{\partial C_t}{\partial w_{ij}}$, puisque l'architecture du réseau peut être très variée et surtout très irrégulière. Ce problème a été résolu en introduisant très astucieusement la notion du gradient de C_t par rapport aux entrées totales,

$$y_i=\frac{\partial C_t}{\partial A_i} \tag{1.7}$$

Puisque

$$\frac{\partial C_t}{\partial w_{ij}}=\frac{\partial C_t}{\partial A_i}*\frac{\partial A_i}{\partial w_{ij}}=y_i*x_j \tag{1.8}$$

Le calcul de $\frac{\partial C_t}{\partial w_{ij}}$ est donc largement simplifié car il suffit de connaître le y_i associé à chaque cellule (de traitement). De plus, tous les y_i peuvent être calculés successivement comme suit : pour les cellule i en couche L (de sortie) :

$$y_i=\frac{\partial C_t}{\partial A_i}=\frac{\partial}{\partial A_i}\left(\sum_h (S_h^{(k)}-Y_h^{(k)})^2\right)=2(S_i^{(k)}-Y_i^{(k)})*f'(A_i) \tag{1.9}$$

où la sommation est effectuée pour tous les indices des cellules de la dernière couche. Si tous les y_i en couche L, L-1, ..., n+1 sont calculés, on peut alors calculer y_i en couche n. Notons $y_q=\partial C_t/\partial A_q$ pour toutes les cellules q qui reçoivent la sortie de la cellule i comme une de leurs entrées, et en utilisant la règle de dérivation de fonction composée (chain rule), nous avons :

$$y_i=\frac{\partial C_t}{\partial A_i}=\sum_q \frac{\partial C_t}{\partial A_q}\frac{\partial A_q}{\partial A_i}=\sum_q \frac{\partial C_t}{\partial A_q}\frac{\partial}{\partial A_i}\left(\sum_j w_{qj}*x_j\right)$$

$$=\sum_q y_q*w_{qi}*\frac{\partial f_i(A_i)}{\partial A_i}=f_i'(A_i)\sum_q y_q*w_{qi} \tag{1.10}$$

Donc, chaque y_i peut être calculé, si tous les y_q correspondant aux cellules qui recoivent leurs entrées de la cellule i sont calculés.

L'algorithme d'apprentissage GBP

En résumé, l'algorithme d'apprentissage peut être présenté par les 3 phases suivantes.

1° Calcul de la sortie : Pour une entrée X=X(t) donnée à l'instant t, calculer la sortie S=S(t) du réseau par les formules :

si la cellule i est en couche 0 $x_i=X_i$

sinon $x_i=f_i(A_i)$ avec

$A_i=\sum_j w_{ij}(t)*x_j$

2°) Calcul du gradient : Pour la sortie desirée Y=Y(t)

si la cellule s est en couche L : $y_s=2(S_s-Y_s)*f_s'(A_s)$

de la couche L-1 à la couche 1 : $y_i= f_i'(A_i)*\sum_j w_{ji}(t)y_j$

3°) Amélioration de poids : $w_{ij}(t+1)=w_{ij}(t)-\gamma_t\, y_i*x_j$

γ_t est une suite de réels positifs (convergant vers zéro ou vers un très petit réel positif) qui définissent le taux de progression des poids dans la direction du gradient.

La variation de w_{ij} pour chaque pattern (X(t),Y(t)) présenté au réseau à l'instant t s'écrit $\Delta w_{ij}(t)$. Il est clair que dans l'algorithme GBP présenté ci-dessus, on a :

$$\Delta w_{ij}(t)=w_{ij}(t+1)-w_{ij}(t)=-\gamma_t \frac{\partial C_t}{\partial w_{ij}} =-\gamma_t\, y_i*x_j. \qquad (1.11)$$

Cette variation dépend de l'ensemble des poids actuels et du pattern (X,Y). Les nouveaux poids $w_{ij}(t+1)$ ainsi obtenus vont servir de poids actuels pour le pattern suivant.

Un problème, issu de la façon dont on calcule $\Delta w_{ij}(t)$, est que certains patterns imposent des variations trop grandes, et par conséquent les poids oscillent au cours de l'apprentissage. En fait, la variation calculée pour chaque pattern ne favorise en principe que la mémorisation de ce pattern particulier. Si la variation pour chaque pattern est trop grande, les poids vont évoluer dans un sens qui pénalise lourdement l'acquisition des autres patterns. On espère donc que l'apprentissage de toutes les associations se déroule d'une manière équilibrée. Dans le cas contraire, le processus d'apprentissage serait très long.

Le fait que chaque pattern soit founi au réseau une fois par cycle est une précaution pour éviter la variation brutale. Une autre technique souvent utilisée

consiste à calculer $\Delta w_{ij}(t)$ selon le gradient actuel et la variation précédente $\Delta w_{ij}(t-1)$. Celle-ci nous donne une variante importante de l'algorithme GBP (voir Rumelhart, Hinton & Williams 1986).

L'algorithme d'apprentissage GBP avec le momentum

Il s'agit d'un prenant en compte de la variation précédente de w_{ij} lors qu'on calcule le changement (la variation) actuel(le). La formule (1.11) devient donc,

$$\Delta w_{ij}(t)=-\gamma_t\, y_i*x_j+\alpha\Delta w_{ij}(t-1) \qquad (1.11')$$

avec $1>\alpha\geq 0$ et γ_t défini comme dans l'algorithme précédent. Le terme de momentum $\alpha\Delta w_{ij}(t-1)$ permet d'agrandir la variation $\Delta w_{ij}(t)$ lors que $-y_i*x_j$ et $\Delta w_{ij}(t-1)$ partagent le même signe et de la diminuer lors que les deux termes ont des signes opposées. Le premier cas correspond souvent à la situation où la surface d'erreur contient une décente longue et lisse dans la dimension de l'espace, il est donc interressant d'avancer plus vite le poids w_{ij}; le deuxième cas correspond souvent à la situation d'oscilliation dans laquelle il est souhaitable de diminuer la glandeur du changement (de poids) dans cette direction. Le paragraphe V.2 du chapitre V donne une discussion plus détaillée de ce sujet.

Cet algorithme d'apprentissage GBP avec le momentum β est utilisé dans la plupart des expériences présentées dans cette thèse. Dans le Chapitre V, qui est consacré à l'amélioration de l'algorithme GBP, on trouvera que la technique du momentum est une application des principes heuristiques utilisés pour accélérer le processus de recherche du minimum de la fonction d'erreur.

L'algorithme GBP adapte, de manière progressive, la réponse d'un réseau à l'ensemble des patterns d'association entrée-sortie. L'ordre de présentation des patterns, souvent appelé ordre d'apprentissage, doit être précisé. Deux stratégies sont fréquemment utilisées : la stratégie séquentielle et la stratégie aléatoire. Dans le mode séquentiel, on présente les patterns au réseau selon un ordre fixé a priori. Dans le mode aléatoire, on tire au hasard un entier entre 1 et K (nombre de patterns), et on fait alors l'apprentissage sur le pattern qui correspond à l'entier tiré. Le terme "une itération d'apprentissage" correspond dans le cas séquentiel à un parcours de tous les patterns, et dans le cas aléatoire à K tirages. Il faut ajouter aussi que dans certaines applications, on désire que certains patterns soient appris plus souvent que d'autres; pour cela il

faut "dupliquer" les indices de ces patterns et les placer dans des endroits différents de la liste d'apprentissage. Enfin, pour chaque pattern pris dans l'ordre ou au hasard, on n'effectue qu'une seule fois l'apprentisage pour équilibrer l'influence de chaque pattern. Il est déconseillé de répéter plusieurs fois de suite l'apprentissage sur le même pattern.

Pourquoi "rétro-propagation"? En fait le calcul du gradient y dans l'algorithme d'apprentisage commence par la dernière couche, et progresse couche par couche dans le sens contraire à celui de la mise à jour des sorties : c'est pourquoi on utilise le terme rétro-propagation du gradient. On remarque également que y_s à la dernière couche dépend de l'erreur de la réponse du réseau par rapport à la sortie désirée. Il en est de même pour tout les autres y_i du fait même que y_i est une combinaison linéaire des y_s. C'est la raison pour laquelle l'algorithme possède un autre nom "rétro-propagation des erreurs".

I.2. Une approche du traitement parallèle et distribué des connaissances

Un réseau de neurones est, avant tout, un système de calcul qui est adaptatif. Mathématiquement, le réseau transforme les vecteurs d'un espace (ou ensemble) en vecteurs d'un autre espace (ou ensemble). En vertu des informations représentées par ces vecteurs et de la manière dont le réseau est construit, il pourrait être utilisé pour réaliser des fonctions d'association et de classification des informations, de satisfaction de contraintes, si les informations dans les configurations des vecteurs sont ainsi interprétées, et de compression d'informations (image ou parole).

Les réseaux de neurones peuvent être donc utilisés comme support de systèmes intelligent. La différence principale entre les réseaux de neurones et les techniques de l'I.A. traditionelle réside dans la manière dont les informations sont représentées et traitées. Le principle d'I.A. est de représenter chaque entité d'information par une unité de calcul. On utilise souvent le terme de "*représentation locale*". Les relations entre les entités d'informations sont spécifiées par les "*lois*" selon lesquelles les unités interagissent entre elles. Dans les réseaux de neurones, par contre, les informations sont représentées par l'activation d'un ou de plusieurs groupes de cellules (unités de calcul) et chaque cellule ne peut représenter qu'une caractéristique microscopique (pour "micro-feature", terme dû à Hinton) d'une entité d'information. Chaque cellule participe à la représentation de plusieurs entités d'information. On dit que les réseaux de neurones utilisent le mode de "*représentation distribuée*". Par rapport à la représentation locale, la représentation distribuée est beaucoup moins populaire et difficile à imaginer. Hinton *et al.* (1986) présentent une discussion intérèssante de la représentation distribuée.

Les représentations distribuées fournissent une manière naturelle de traiter la similarité entre des informations et d'utiliser efficacement des architectures parallèles. Par exemple dans une CAM (Content-Adressable Memory), la recherche d'une entité d'information s'effectue, à partir d'une représentation clé souvent incomplète, en faisant le "matching" de la clé avec toutes les entités d'information. L'entité d'information trouvée dans la CAM doit être celle qui "colle" le mieux à la clé. Les techniques de l'I.A. pour résoudre ce genre de problèmes sont très artificielles, car la comparaison entre deux entités

d'information représentées par deux unités différentes semble aberrante. Cette difficulté méthodologique provient du fait que la représentation locale ne conserve que la caractéristique macroscopique des informations et n'a pas accès à leur caractéristique microscopique. La représentation distribuée, grâce à sa nature microscopique, permet que la similarité entre les entités d'informations soit capturée via des méthodes plus naturelles. De plus, la comparaison peut être, en principe, plus facilement réalisée sur des architectures parallèles pour les représentations distribuées car elles concernent seulement des descriptions partielles des entités d'information.

Un autre avantage des représentations distribuées réside dans la création des nouveaux concepts une fois que le système est construit. Dans les systèmes à base de représentation locale, il faut normalement ajouter une nouvelle unité et il faut spécifier les interactions entre l'unité ajoutée et le reste du système. C'est un processus très délicat quand le système devient grand. Dans les systèmes à base de représentation distribuée par contre, il suffit de trouver une représentation pour le nouveau concept, cette représentation étant intégrée automatiquement dans le système (au moins jusqu'à une certaine limite). Le problème avec la représentation distribuée est dans le choix approprié du pattern pour le nouveau concept. Il faut que ce pattern soit proche de ceux qui sont utilisés pour représenter les concepts proches.

La représentation distribuée a aussi des inconvénients. Le défaut le plus remarquable est dans l'interprétation des patterns d'activation. Il est difficile d'expliquer le role exact de chaque cellule, ce qui nous donne souvent l'impression comme "il marche mais on ne sais pas comment ça marche". Il faut prendre la représentation locale et la représentation distribuée comme des approches complémentaires pour la représentation des connaissances. La conciliation de l'approche de l'I.A. traditionnelle et de l'approche connexionniste pourrait beaucoup apporter à notre connaissance de l'intelligence. C'est un sujet qui attire l'intérêt des chercheurs de plusieurs domaines comme l'I.A.et Sciences Cognitives, dont le nombre est en train de croître actuellement.

I.2.1 Association, Classification et Satisfaction de contraintes

En fonction de la signification des vecteurs d'entrée $X^{(k)}$ et de celle des vecteurs de sortie $Y^{(k)}$, la transformation $X^{(k)} \rightarrow Y^{(k)}$ (pour k=1 ... K) réalisée par

un réseaux multicouches a des interprétations très variées :

Si les $X^{(k)}$ représentent des "faits", cette transformation, qui est une association entre les faits et leurs "étiquettes" ou les "conséquences" $Y^{(k)}$, réalise donc des concepts ou des relations de type "Is-A";

Si les $X^{(k)}$ sont des signaux de parole, la transformation $X^{(k)} \rightarrow Y^{(k)}$ pourrait être interprétée comme une classification qui produit des représentations permettant des traitements plus aisés de niveau plus élévé;

Dans des applications robotiques ou dans des systèmes de reconnaissance, les $X^{(k)}$ peuvent correspondre aux contraintes (ou conditions) qui doivent être satisfaites pour dégager des commandes d'action ou pour produire des identifications (If ... Then ...).

La manière dont le réseau acquiert sa capacité de traitement d'information est un problème très délicat et technique. Elle est essentiellement façonnée par l'algorithme utilisé pour l'apprentissage, l'architecture du réseau, la représentation et la répartition des données. Ces facteurs influencent intéractivement le processus de l'acquisition. On ne sait pas encore formaliser rigoureusement cette interaction. C'est pourquoi on fait appel sans arrêt à des heuristiques et des intuitions pour concevoir et justifier les réseaux utilisés dans les applications.

Il est nécessaire de souligner que tous les réseaux qui réalisent la transformation $X^{(k)} \rightarrow Y^{(k)}$ (pour k=1 ... K) n'ont pas forcément la capacité attendue de traitement d'information. Comme souligné dans le paragraphe précédent I.1, plus grand est le réseau, plus il y a de chance qu'il apprenne facilement, mais moins il y a de chance qu'il effectue la "bonne" généralisation. D'ailleurs, les applications ont toutes leurs particularités, dont il faut tenir compte. Si certaines exigent que le réseau fasse le "best mapping"[1], comme dans le traitement d'images ou certaines simulations de mémoire, d'autres insistent beaucoup plus sur le fait que le réseau soit capable de produire des outputs qui traduisent les variations intrinsèques des signaux d'entrée, comme dans le traitement du signal (parole).

---

(1) le fait de produire Y^{k0}, une des sorties apprises correspondant à X^{k0}, qui est "le plus proche" de l'entrée à tester.

I.2.2 Compression des informations

Une application importante des réseaux multicouches est la compression des informations. Pour exposer cette idée, nous choisirons le cas le plus simple. La Figure 1.2 montre un réseau de trois couches avec 8 cellules d'entrée, 8 cellules de sortie et 3 cellules intermédiaires. Les connexions entre deux couches successives sont complètes et il n'y a pas de connexions qui traversent la couche intermédiaire. Ce qu'il faut souligner en ce qui concerne ce réseau, c'est que l'on a le même nombre de cellules d'entrée que de cellules de sortie, et que le nombre de cellules intermédiaires est très inférieur à celui des cellules de sortie.

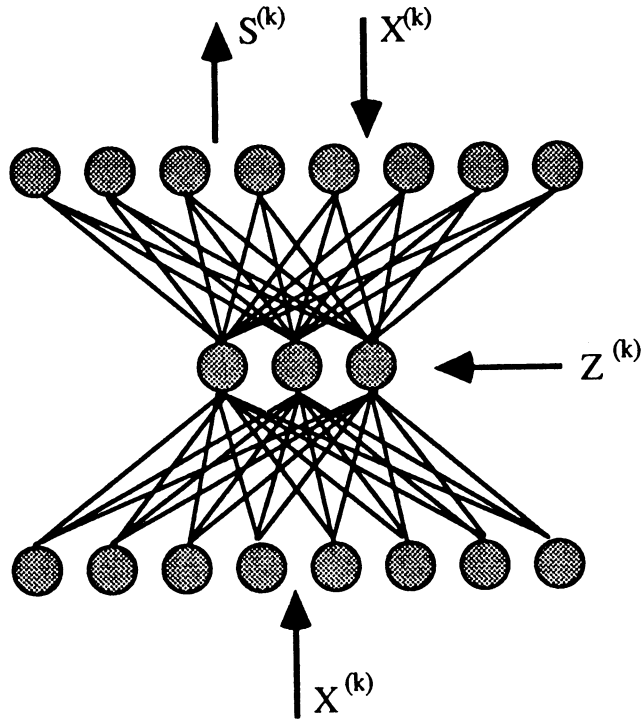


Figure 1.3 : Une "imagette" : réseau pour la compression des informations.

Ici on suppose que $Y^{(k)}=X^{(k)}$: la sortie désirée pour chaque entrée exemplaire $X^{(k)}$ est elle-même. L'algorithme de GBP, ou n'importe quel autre algorithme pour les réseaux multicouches, est appliqué pour que le réseau réalise la transformation $X^{(k)}\rightarrow X^{(k)}$ pour k=1, ... ,K. Si l'apprentissage réussit, on dit que les cellules intermédiaires réalisent une compression des informations, car l'état $Z^{(k)}$ de ces cellules, qui est un vecteur de dimension très inférieur et qui permet aux cellules de sortie de reproduire $X^{(k)}$, contient en principe toutes les

informations de $X^{(k)}$. $Z^{(k)}$ pourrait être utilisé comme entrée d'un autre système. De par le fait que les $Z^{(k)}$ soit les représentations compressées des $X^{(k)}$, les informations des $X^{(k)}$ pourraient donc être traitées de manière plus efficace.

Dans les applications réelles, ce type de "réseaux compresseurs" est appellé imagette. Il faut souvent construire plusieurs imagettes "partiellement superposées" qui recouvrent des grandes images ou des signaux longs. Pour des applications spéciales, l'algorithme qui est basé sur la minimisation de la fonction d'erreur calculée par la distance eulidienne peut ne pas être satisfaisant. Il y a encore d'autres problèmes pratiques et théoriques à résoudre pour mieux exploiter le potentiel des réseaux multicouches dans la compression des informations. Les rapports (Foldiak, 1989; Cottrell, G.W., Munro, P. & Zipser 1987 et Sanger 1988) fournissent des discussions plus approfondies sur le sujet.

I.3 Comportement dynamique des réseaux de neurones fonctionnant en GBP

Dans ce paragraphe, nous nous pencherons sur le comportement dynamique des réseaux de neurones dont l'algorithme d'apprentissage est le GBP. Plus précisément, nous démontrerons que, sous certaines conditions d'apprentissage "idéales" et si les patterns sont codés en 1 et -1, les réseaux tendent à réaliser des fonctions qui sont contractantes dans les domaines où se situent les patterns d'entrée exemplaires. En particulier, si les patterns d'entrée et les patterns de sortie sont identiques, les réseaux ont tendance à créer des attracteurs qui sont très proches de ces patterns exemplaires ; ceci signifie que, partant d'un pattern "proche" d'un des patterns appris et en réinjectant la sortie du réseau à son entrée, le réseau génère une suite de patterns qui convergera vers l'attracteur correspondant au pattern appris. On obtient ainsi une mémoire auto associative. Les résultats expérimentaux seront donnés à la fin du paragraphe.

Il faut souligner que la mémoire réalisée par les réseaux (de l'algorithme GBP) a une nature différente de celle réalisée par le modèle de Hopfield. Les réseaux de Hopfield fonctionnent très bien quand les patterns à mémoriser sont quasi-orthogonaux entre eux. En absence de cette condition, il faut recourir à des algorithmes adapdatifs en introduisant des "bruits" dans les patterns exemplaires (Amit, Gutfreund & Sompolinsky (1985); Peretto 1984), tandis que le modèle de réseaux multicouches se montre beaucoup plus efficace (Gallinari, Le Cun, Thiria & Fogelman-Soulie 1987) pour traiter ce cas difficile. Les expériences effectuées ici ont pour objectif d'illustrer la variation de la capacité de mémorisation et le comportement dynamique au cours des itérations (recherche) en fonction des architectures des réseaux.

Le fait que les réseaux multicouches soient capables de créer des attracteurs présente un intérêt bien au-delà de la mémoire associative. Pour beaucoup d'applications, les problèmes de généralisation sont en relation étroite avec l'attraction de certains patterns (ou plutôt la contraction de la fonction réalisée par le réseau). Enfin l'architecture souple dans le modèle des réseaux multicouches permet d'intégrer plusieurs fonctionnalités dans une même architecture. Les expériences présentées dans le Chapitre III montrent comment on a réussi à exploiter l'interaction entre l'aspect de mémoire associative et celui du système de classification, pour modéliser et simuler une mémoire des visages

en contexte. Les résultats présentés dans ce paragraphe servent de base théorique à l'application de reconnaissance des visages.

I.3.1 Conventions et définitions

On suppose, tout au long du paragraphe, que les vecteurs d'entrée et ceux de sortie désirée sont toujours codés en -1 et 1. La dimension des patterns d'entrée et celle des patterns de sortie ne seront spécifiées que si ceci est vraiment nécessaire.

Une cellule, qui peut être considérée comme le plus petit réseau, est caractérisée par un vecteur **w** des poids de ses liens d'entrée, une valeur de seuil et une fonction sigmoïde f. Pour simplifier les notations, on utilise toujours un élément, le dernier dans le vecteur des poids, pour représenter le seuil en supposant qu'il y ait une entrée correspondante dont l'état est toujours 1. Cette convention générale permet de représenter l'entrée totale de la cellule par un produit scalaire **w*x**. Le α, qui détermine la pente de la fonction sigmoïde f, est supposé être constant pour toutes les cellules, sans pour autant perdre aucune généralité.

Supposons qu'un ensemble $S \subseteq \{-1, 1\}^m$ ait une partition S^+ et S^- fixé pour un certain objectif. Nous savons qu'en pratique, beaucoup de partitions peuvent être réalisées par une seule cellule à condition que les deux ensembles S^+ et S^- soient linéairement séparables. On introduit ici la notion de qualité de la séparation effectuée par une cellule.

Définition I.1: Etant donné δ, $0<\delta<1$: si $\forall \mathbf{x} \in S^+$, $f(\mathbf{w}*\mathbf{x})>1-\delta$ et $\forall \mathbf{x} \in S^-$, $f(\mathbf{w}*\mathbf{x})< -(1-\delta)$, on dit que la cellule réalise une δ-*séparation* de **S**.

Il est clair que l'existence d'une telle séparation est conditionnée, en grande partie, par la séparabilité de **S** en S^+ et S^-, tandis que la valeur δ est liée à la sortie de la cellule pour chacune des entrées prises dans l'ensemble **S** : c'est-à-dire dans quelle mesure la valeur $f(\mathbf{w}*\mathbf{x})$ est proche de 1 $\forall \mathbf{x} \in S^+$, et proche de -1 $\forall \mathbf{x} \in S^-$.

Par la suite, le terme "un schéma d'apprentissage basé sur le GBP" signifie qu'à chaque étape d'appentissage, la formule utilisée pour modifier les poids de connection est essentiellement celle de l'algorithme de GBP, ou des variantes comme celle avec le momentum.

On s'intéresse à l'effet asymptotique d'un schéma d'apprentissage sur les propriétés des patterns appris (par le réseau). Un des effets les plus simples que l'on puisse étudier est le problème présenté ci-dessus de séparation par une seule cellule. On a besoin de la définition suivante pour simplifier la terminologie.

Définition I.2 : On dit qu'un schéma effectue un *apprentissage équilibré* sur une cellule, si $\exists\varepsilon>0$, $\forall\delta>0$ il existe un entier N_δ tel que le vecteur des poids $\mathbf{w}^{(n)}$ de l'étape n ($>N_\delta$) satisfasse $\left|\frac{\mathbf{w}^{(n)*}\mathbf{x}}{\|\mathbf{w}^{(n)}\|_2}\right|>\varepsilon\ \forall\mathbf{x}\in\mathbf{S}$, et la cellule réalise une δ-*séparation* de **S**.

Nous avons une hypothèse sous-jacente ici pour la notion d'*apprentissage équilibré* : **S** est linéairement séparable en $\mathbf{S}^+$ and $\mathbf{S}^-$. On se pose naturellement des questions sur l'existence d'un tel apprentissage. Il faut peut-être clarifier certaines notions employées fréquemment dans les communications pour mieux répondre à cette question :

Premièrement, une séparation linéaire de **S** en $\mathbf{S}^+$ et $\mathbf{S}^-$ signifie $\mathbf{w}^*\mathbf{x}>0$ pour $\mathbf{x}\in\mathbf{S}^+$ et $\mathbf{w}^*\mathbf{x}<0$ pour $\mathbf{x}\in\mathbf{S}^-$, où **w** est un vecteur semblable à celui des poids d'entrée à une cellule. Comme **w** détermine un hyperplan, la séparation de **S** peut être considérée comme le fait que l'hyperplan "coupe" l'hyperespace en deux parties avec $\mathbf{S}^+$ qui se trouve dans le côté (supérieur) de l'hyperplan et $\mathbf{S}^-$ qui se trouve dans l'autre côté (inférieur) ;

Deuxièmement, Le critère pratique pour mesurer la séparation de **S** par une cellule est le suivant : $f(\mathbf{w}^*\mathbf{x})$ doit être proche de 1 pour $\mathbf{x}\in\mathbf{S}^+$ et proche de -1 pour $\mathbf{x}\in\mathbf{S}^-$. Si la fonction de décision f est discrète avec les valeurs -1 et 1, le critère est satisfait une fois que la séparation linéaire de **S** est réalisée par le vecteur des poids de la cellule. Cependant, si f est une fonction (continue) sigmoïde, une séparation linéaire de **S** par **w** n'est plus suffisante car, bien que le signe soit correct, $\mathbf{w}^*\mathbf{x}$ pour un $\mathbf{x}\in\mathbf{S}$ peut être trop petit en valeur absolue pour rendre la sortie $f(\mathbf{w}^*\mathbf{x})$ proche de 1 ou -1. Autrement dit, la fonction continue (de décision) impose une condition plus stricte pour que la cellule soit capable de séparer **S** : $\forall\mathbf{x}\in\mathbf{S}$, $\mathbf{w}^*\mathbf{x}$ doit non seulement avoir le signe correct (dépendant de la

classe S^+ ou S^- à qui **x** appartient) mais doit aussi être grand en valeur absolue (en relation avec paramètre α, pente de f) ;

Avec l'*apprentissage équilibré*, on s'intéresse ici spécialement à la manière dont l'algorithme d'apprentissage se comporte lorsque le vecteur des poids **w** "commence" à séparer **S**, la réponse de la cellule f(**w***x) n'étant pas encore suffisamment proche de 1 ou -1.

Un tel apprentissage peut être obtenu à partir de n'importe quel algorithme capable d'apprendre à une cellule à séparer **S**. Par exemple, si un algorithme *A1* peut apprendre à une cellule *C*, en N_0 étapes, à réaliser une séparation linéaire de **S**, alors si on pose $\mathbf{w}^{(n)}=2\mathbf{w}^{(n-1)}$ ($\mathbf{w}^{(n)}$ représentant le vecteur des poids de l'étape n) pour tout $n>N_0$ on obtient un nouvel algorithme *A2* qui réalise un *apprentissage équilibré* sur la cellule *C*. Pour prouver ceci, il suffit de prendre :

$$\varepsilon=\min\{|\mathbf{w}^{(N_0)}*\mathbf{x}| \mid \mathbf{x}\in \mathbf{S}\}/(2*\|\mathbf{w}^{(N_0)}\|_2)$$

car $\forall \mathbf{x}\in \mathbf{S}$ on a, $\forall n>N_0$ $\left|\frac{\mathbf{w}^{(n)}*\mathbf{x}}{\|\mathbf{w}^{(n)}\|_2}\right|=\left|\frac{\mathbf{w}^{(N_0)}*\mathbf{x}}{\|\mathbf{w}^{(N_0)}\|_2}\right|>\varepsilon$ et

$$\lim_{n\to\infty}(1-|f(\mathbf{w}^{(n)}*\mathbf{x})|)=0.$$

En effet, si $\mathbf{x}\in S^+$ alors

$$1-|f(\mathbf{w}^{(n)}*\mathbf{x})|=1-f(\mathbf{w}^{(n)}*\mathbf{x})=2e^{-\alpha(\mathbf{w}^{(n)}*\mathbf{x})}/(1+e^{-\alpha(\mathbf{w}^{(n)}*\mathbf{x})})$$
$$=[e^{-\alpha(\mathbf{w}^{(n-1)}*\mathbf{x})}(1+e^{-\alpha(\mathbf{w}^{(n-1)}*\mathbf{x})})/(1+e^{-\alpha(\mathbf{w}^{(n)}*\mathbf{x})})]*(1-f(\mathbf{w}^{(n-1)}*\mathbf{x}))$$
$$=\ldots=[e^{-\alpha((\mathbf{w}^{(n-1)}+\mathbf{w}^{(n-2)}+\ldots+\mathbf{w}^{(N_0)})*\mathbf{x})}(1+e^{-\alpha(\mathbf{w}^{(N_0)}*\mathbf{x})})/(1+e^{-\alpha(\mathbf{w}^{(n)}*\mathbf{x})})]*$$
$$(1-f(\mathbf{w}^{(N_0)}*\mathbf{x}))$$
$$<C_0*e^{-C(n)} \qquad (1.12)$$

où $C_0=2*(1-f(\mathbf{w}^{(N_0)}*\mathbf{x}))$ et $C(n)=\alpha*(n-N_0)^2*(\mathbf{w}^{(N_0)}*\mathbf{x})/2$. On voit donc que $1-|f(\mathbf{w}^{(n)}*\mathbf{x})|$ tend vers zéro exponentiellement par rapport à $(n-N_0)^2$. Le résultat est aussi valide pour $\mathbf{x}\in S^-$.

L'exemple ci-dessus montre un cas radical de l'*apprentissage equilibré*. Plus généralement, si les paramètres d'apprentissage dans l'algortihme de GBP sont choisis suffisamment petits, à partir du moment où le vecteur des poids de la cellule sépare **S**, alors on peut obtenir l'effet d'*apprentissage equilibré*. Comme $\left|\frac{\mathbf{w}^{(n)}*\mathbf{x}}{\|\mathbf{w}^{(n)}\|_2}\right|$ est la longeur de la projection **x** sur **w**, un tel apprentissage signifie que durant les "dernières étapes", chaque hyperplan déterminé par **w** reste dans une zone de position où chacun des $\mathbf{x}\in \mathbf{S}$ est

écarté au moins d'une distance fixée de l'hyperplan (Figure 1.4). Aucun point particulier de S ne peut être considéré comme "mieux séparé" que les autres.

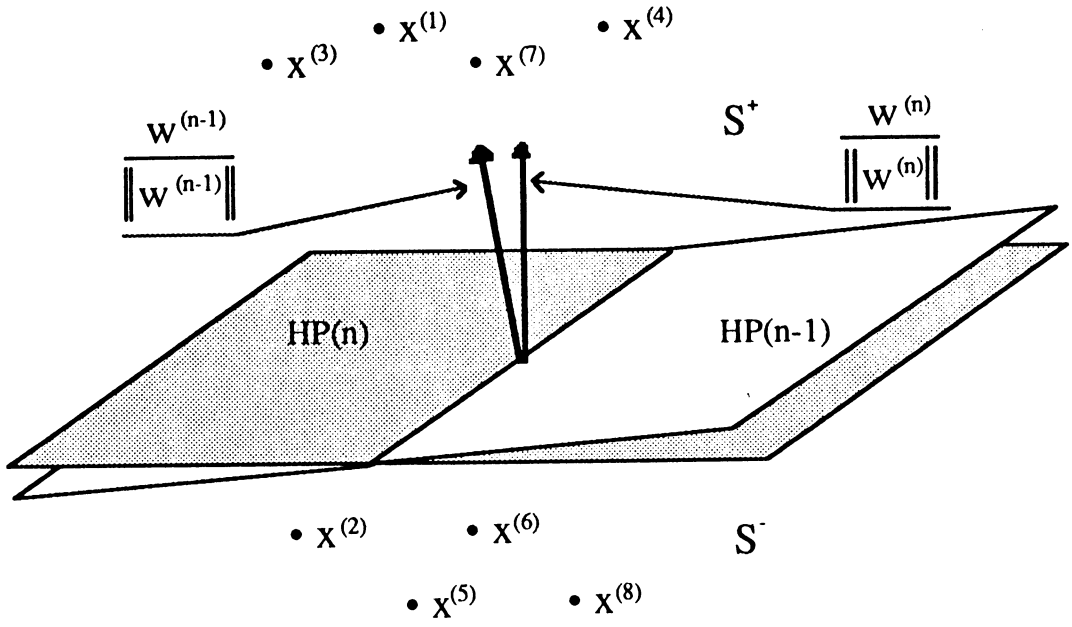


Figure 1.4 : Une illustration de l'*apprentissage équilibré* sur une cellule pour séparer l'ensemble S en S^+ et S^-. w ici correspond au vecteur des poids de connexions d'entrée. •••

Il est pourtant important de remarquer que la notion d'*apprentissage équilibré* concerne ici seulement un effet obtenu par application d'un algorithme à une cellule, pour séparer un ensemble S. Un algorithme qui réalise un *apprentissage équilibré* sur une cellule n'est pas censé avoir la même performance sur une autre cellule ou pour un autre ensemble S' à séparer.

I.3.2 Contraction de la fonction réalisée par des réseaux : Généralisation

Quelles sont les impacts de l'*apprentissage équilibré*? Les résultats théoriques suivants essayent de répondre à cette question.

Lemme 1.1 : Pour le problème de séparation ci-dessus, s'il existe un algorithme qui réalise un *apprentissage équilibré* sur une cellule, alors, après un nombre fini d'étapes d'apprentissage, la fonction f(w*•) de la cellule satisfait : $\forall x \in S$, $\exists$ un voisinage $V(x) \subseteq [-1,1]^m$ tel que

$$\|f(w*y)-f(w*z)\|_2 < \mathfrak{L} * \|y-z\|_2 \qquad \forall\, y,z \in V(x) \qquad (1.13)$$

où $\mathfrak{L}$ est une constante inférieure à 0.5. #

Démonstration : On prouvera, sans perdre la généralité, le lemme seulement pour $\mathbf{x}\in S^+$. Supposons que l'on ait $\varepsilon>0$ tel que $\forall\delta>0$ il existe un entier N_δ tel que le vecteur des poids $\mathbf{w}^{(n)}$ de l'étape n ($>N_\delta$) satisfasse $\left|\frac{\mathbf{w}^{(n)}*\mathbf{x}}{\|\mathbf{w}^{(n)}\|_2}\right|>\varepsilon$, et $f(\mathbf{w}*\mathbf{x})>1-\delta$ $\forall\mathbf{x}\in S^+$. Définissons :

$$V(\mathbf{x})=\{\mathbf{u}|\ \mathbf{u}\in[-1,1]^m \text{ et } \|\mathbf{x}-\mathbf{u}\|_2<\varepsilon/2\}$$

$V(\mathbf{x})\subseteq[-1,1]^m$ et

$$\left|\frac{\mathbf{w}^{(n)}*\mathbf{u}}{\|\mathbf{w}^{(n)}\|_2}\right|>\varepsilon/2 \qquad \forall\mathbf{u}\in V(\mathbf{x}) \text{ et } n>N_\delta$$

Si δ a été choisi tel que

$$\delta<1 \text{ et } 2*\varepsilon^{-2}*\sqrt{m}*\frac{1}{-\ln(\delta)}<\mathfrak{L}$$

alors, on aurait

$$\alpha*\|\mathbf{w}^{(n)}\|*\|\mathbf{x}\|\geq\alpha*\mathbf{w}^{(n)}*\mathbf{x}=\ln(e^{(\alpha*\mathbf{w}^{(n)}*\mathbf{x})})\geq\ln(\frac{1+e^{-\alpha\mathbf{w}*\mathbf{x}}}{2\,e^{-\alpha\mathbf{w}*\mathbf{x}}})$$

$$=\ln(\frac{1}{1-f(\mathbf{w}*\mathbf{x})})\geq\ln(\frac{1}{\delta})$$

Donc

$$\alpha*\|\mathbf{w}^{(n)}\|\geq\frac{1}{\sqrt{m}}\ln(\frac{1}{\delta}) \qquad (1.14)$$

$$2\alpha*e^{(-2\alpha*|\mathbf{w}^{(n)}*\mathbf{u}|)}*\|\mathbf{w}^{(n)}\|<2\alpha*e^{(-\alpha\varepsilon*\|\mathbf{w}^{(n)}\|)}*\|\mathbf{w}^{(n)}\|$$

$$<2\alpha*(-\alpha\varepsilon*\|\mathbf{w}^{(n)}\|)^{-2}*\|\mathbf{w}^{(n)}\|=2\alpha^{-1}*\varepsilon^{-2}*\|\mathbf{w}^{(n)}\|^{-1}$$

$$<2*\varepsilon^{-2}*\sqrt{m}*\frac{1}{-\ln(\delta)}<\mathfrak{L} \qquad (1.15)$$

Pour $\mathbf{y},\mathbf{z}\in V(\mathbf{x})$, il existe un $0<\theta<1$ tel que $\mathbf{u}'=\theta*\mathbf{y}+(1-\theta)*\mathbf{z}\in V(\mathbf{x})$ et

$$f(\mathbf{w}^{(n)}*\mathbf{y})-f(\mathbf{w}^{(n)}*\mathbf{z})=\text{grad } f(\mathbf{w}^{(n)}*\bullet)\big|_{\mathbf{u}'}*(\mathbf{y}-\mathbf{z})=\frac{2\alpha\,e^{-2\alpha\mathbf{w}^{(n)}*\mathbf{u}}}{(1+e^{-\alpha\mathbf{w}^{(n)}*\mathbf{u}})^2}[\mathbf{w}^{(n)}*(\mathbf{y}-\mathbf{z})]$$

D'après l'expression (1.15), on a

$$\|f(\mathbf{w}^{(n)}*\mathbf{y})-f(\mathbf{w}^{(n)}*\mathbf{z})\|=\frac{2\alpha\,e^{-2\alpha\mathbf{w}^{(n)}*\mathbf{u}}}{(1+e^{-\alpha\mathbf{w}^{(n)}*\mathbf{u}})^2}*\|\mathbf{w}^{(n)}\|*\|\mathbf{y}-\mathbf{z}\|<\mathfrak{L}*\|\mathbf{y}-\mathbf{z}\| \qquad \#$$

Le lemme peut être généralisé comme suit. Supposons que R soit une application d'un ensemble $S\subseteq\{-1,1\}^m$ à $\{-1,1\}^{m'}$ qui doit être réalisée par un

réseau à deux couches (une couche d'entrée et une couche de sortie). L'architecture du réseau implique que chaque cellule de sortie est censée réaliser une séparation de **S** ou (pour être plus précis) une séparation de la projection S' de S dans une sous-espace de $\{-1,1\}^m$. Appliquer un algorithme d'apprentissage au réseau signifie, en fait, d'appliquer m fois le même algorithme (éventuellemnt different pour certains paramètres) aux m cellules de sortie respectivement.

Théorème 1.2 : Si un algorithme basé sur le GBP, et appliqué à un réseau à deux couches, réalise un *apprentissage équilibré* sur toutes les cellules de sortie, alors après un nombre fini d'étapes d'apprentissage, la fonction F résultante du réseau satisfait : $\forall \mathbf{x} \in \mathbf{S}$, $\exists$ un voisinage $V(\mathbf{x}) \subseteq [-1,1]^m$ tel que

$$\|F(\mathbf{y})-F(\mathbf{z})\|_2 < \mathfrak{L} * \|\mathbf{y}-\mathbf{z}\|_2 \qquad \forall \mathbf{y}, \mathbf{z} \in V(\mathbf{x}) \tag{1.16}$$

où $\mathfrak{L}$ est une constante inférieure à 0.5. #

La démonstration de ce théorème est simple. Il suffit d'appliquer le **Lemme 1.1** à toutes les cellules de sortie, de définir $V(\mathbf{x})$ comme l'intersection de $V_c(\mathbf{x})$, et de prendre $\mathfrak{L}$ comme le minimum de tous les $\mathfrak{L}_c$ obtenus pour toutes les cellules i de sortie.

On peut remarquer, de par la démonstration du **Lemme 1.1**, que le paramètre $\mathfrak{L}$ peut devenir infiniment petit à force de prolonger l'apprentissage. Ceci signifie que la fonction F de transformation, réalisée par le réseau, est très contractante au moins dans le voisinage de chaque $\mathbf{x} \in \mathbf{S}$. L'ampleur de ce voisinage dépend de l'ensemble **S** et de l'application R. Pour le cas le plus simple où R correspond à la séparation de **S**, le voisinage $V(\mathbf{x})$ recouvre toute la région qui est à une distance fixe de l'hyperplan déterminé par **w**. Il n'est pas difficile à comprendre que, pour le cas général traité dans le **Théorème 1.2**, $V(\mathbf{x})$ est au moins l'intersection des voisinages $V_c(\mathbf{x})$ pour chaque cellule c de sortie : c'est-à-dire $V(\mathbf{x})=\cap_c V_c(\mathbf{x})$.

Le problème de généralisation (des connaissances) est en relation étroite avec la propriété de contraction. Un pattern d'entrée sera facilement reconnu s'il se trouve dans une zone où la fonction F est fortement contractante. Si le pattern se trouve dans une zone où F n'est pas très contractante, le système a encore une chance de donner une "bonne identification". Dans ce dernier cas ou lorsque le pattern tombe dans une zone ambiguë, il y a toujours une partie des cellules qui est très active. La réponse du réseau donne des informations sur les "conditions"

qui ont été satisfaites ou non par le pattern d'entrée. Ceci est, en effet, la satisfaction des contraintes. On peut imaginer que si le réseau était muni d'un mécanisme de recherche dynamique, il pourrait manifester des capacités encore plus riches de traitement d'informations. Dans le paragraphe suivant, on étudiera un cas particulier de comportement dynamique des réseaux dans le but d'approfondir cet aspect de connaissance, ce qui servira de guide théorique à la modélisation présentée plus tard.

La généralisation des résultats théoriques présentés dans ce paragraphe aux réseaux ayant plus de deux couches est assez problématique. Il faut imposer des conditions plus restrictives sur les paramètres d'apprentissage pour avoir des résultats semblables. Les réseaux multicouches réalisent des fonctions dont la variabilité est beaucoup plus riche.

I.3.3 Attractivité des patterns appris

Dans le cas particulier où l'application R est identique ($R(\mathbf{x})=\mathbf{x}$), le voisinage $V(\mathbf{x})$ (supposé fermé) pour chaque $\mathbf{x}$ peut être choisi tel que l'image de $V(\mathbf{x})$ après l'application de la fonction F reste dans $V(\mathbf{x})$. Ceci constitue, avec la propriété de contraction, la fameuse condition suffisante pour que F admette un attracteur (unique) dans $V(\mathbf{x})$.

Théorème I.3 : Si $m=m'$ et $R(\mathbf{x})=\mathbf{x}$, sous les même conditions que dans le **Théorème I.2**, on a $\forall \mathbf{x}\in \mathbf{S}$, $\exists$ un voisinage $V(\mathbf{x})\subseteq[-1,1]^m$ fermé tel qu'il existe un unique $\mathbf{x}^*\in V(\mathbf{x})$ satisfaisant $F(\mathbf{x}^*)=\mathbf{x}^*$, et $\mathbf{x}^*$ étant attractif au moins dans $V(\mathbf{x})$.#

Démonstration : Il suffit de démontrer qu'il y ait un voisinage $V(\mathbf{x})$ fermé (dans ce cas il est compact) sur lequel la fonction F est contractante et injective. En fait, d'après le **Théorème I.2**, il existe un voisinage $V'(\mathbf{x})$ de $\mathbf{x}$ tel que :

$$\|F(\mathbf{y})-F(\mathbf{z})\|<\mathfrak{L}*\|\mathbf{y}-\mathbf{z}\| \qquad \forall \mathbf{y}, \mathbf{z}\in V'(\mathbf{x})$$

où $\mathfrak{L}$ est une constante inférieure à 0.5. Supposons $d>0$ tel que :

$$V(\mathbf{x})\equiv\{\mathbf{y} \mid \|\mathbf{y}-\mathbf{x}\|\leq d\}\subseteq V'(\mathbf{x}).$$

De par l'hypothèse d'*apprentissage équilibré*, la fonction F satisfait, après un nombre suffisamment grand d'étapes d'apprentissage, la condition suivante :

$$\|F(\mathbf{x})-\mathbf{x}\|<d/2$$

On a donc, $\forall \mathbf{y}\in V(\mathbf{x})$

$$\|F(y)-x\| \leq \|F(y)-F(x)\| + \|F(x)-x\| < \mathfrak{B} * \|y-x\| + d/2 < d,$$

c'est-à-dire $F(y) \in V(x)$. D'ailleurs, il est évident que F est contractante sur $V(x)$. #

Ce théorème implique que, sous la condition d'*apprentissage équilibré*, l'algorithme d'apprentissage crée des attracteurs dynamiques de la fonction réalisée par le réseau à deux couches. Le champ d'attraction pour chacun des attracteurs, bien que caractérisé dans le théorème comme un (petit) voisinage d'un pattern exemplaire(2), pourrait recouvrir un espace beaucoup plus large. Il dépend de la distribution des patterns exemplaires. En pratique, Il pourrait y avoir aussi des attracteurs parasites qui sont créés en même temps que ceux qui sont très proches des patterns exemplaires. Leurs relations ne sont pourtant pas très claires.

Comparée avec des résultats semblables obtenus sur le modèle de Hopfield (Hopfield 1982), l'importance des résultats ici est que la matrice des poids de connexion est obtenue par l'algorithme GBP, et qu'elle est en général non symétrique. La généralisation des résultats présentés ici aux cas des réseaux ayant plus de deux couches nécessite une étude mathématique plus profonde. Dans le paragraphe suivant, nous donnerons des résultats expérimentaux pour illustrer certains comportements de création d'attracteurs par des réseaux multicouches qui apprennent dans des conditions tout à fait normales. Il est évident que cette propriété de création d'attracteurs joue un rôle très important dans la simulation de la mémoire (cf. Chapitre V).

I.3.4 Quelques résultats expérimentaux sur la création des attracteurs

Les resultats, qui seront présentés dans ce paragraphe, sont issus des travaux pratiques menés en commun avec T. BESSON. Il s'agit d'une série d'expériences pour tester la dynamique de création des attracteurs par des réseaux fonctionnant en GBP. Nous avons choisi trois réseaux dont l'un est un réseau à deux couches 8-8, et les deux autres sont des réseaux à trois couches 8-3-8 et 8-4-8. Nous avons générés trois ensembles de patterns (de huit bits) pour l'apprentissage. Le nombre des patterns dans chaque ensemble est égal à 5, 10 et 15 respectivement. La seule précaution qui a été prise pendant la génération de

(2) Théoriquement, un pattern exemplaire ne peut jamais devenir lui-même un attracteur parce que la valeur de sortie de chaque cellule n'atteint jamais à 1 ou à -1.

ces patterns est que, deux patterns dans un ensemble doivent être différents au moins de deux bits. Ces ensembles sont :

EP1={
P1 = (1 1 1 1 1 1 1 1)
P2 = (-1 -1 1 1 1 1 1 1)
P3 = (1 1 -1 -1 1 1 1 1)
P4 = (1 1 1 1 -1 -1 1 1)
P5 = (1 1 1 1 1 1 -1 -1)
}
EP2=EP1+{
P6 = (-1 -1 -1 -1 -1 -1 -1 -1)
P7 = (1 1 -1 -1 -1 -1 -1 -1)
P8 = (-1 -1 1 1 -1 -1 -1 -1)
P9 = (-1 -1 -1 -1 1 1 -1 -1)
P10= (-1 -1 -1 -1 -1 -1 1 1)
}
EP3=EP2+{
P11= (1 -1 -1 -1 -1 -1 -1 1)
P12= (-1 1 1 1 1 1 1 -1)
P13= (-1 1 -1 -1 -1 -1 1 -1)
P14= (1 -1 1 1 1 1 -1 1)
P15= (-1 -1 1 -1 -1 1 -1 -1)
}

Chaque ensemble a été présenté à tous les réseaux pour faire de l'apprentissage. Dans nos expériences, nous avons choisi le taux d'apprentissage comme 0.15 et le momentum comme 0.1 tout au long des apprentissages. Et les poids de connexions sont toujours initialisés entre -0.5 et 0.5. Quant aux tests d'itération (correspondant à la phase de reconnaissance) partant d'un pattern quelconque, nous avons des critères suivants pour mesurer la convergence : si le vecteur de sortie égale au vecteur d'entrée ou si la somme de cinq erreurs consécutives, dont chacun est en fait la puissance au carré de la distance euclidienne entre le vecteur d'entrée et le vecteur de sortie, est inférieure à 0.05, alors le réseau est considéré avoir convergé vers un attracteur (= le vecteur de sortie actuel). Dans les statistiques de résultat, deux attracteurs ainsi obtenus sont considérés comme le même si leur distance est inférieure à 0.1, et un attracteur est considéré comme celui associé à un pattern exemplaire, s'il est le plus proche (du pattern) parmi tous les attracteurs ayant le même signe à tous les composantes avec le pattern. Il faut remarquer que les patterns exemplaires de chaque ensemble ne sont pas orthogonaux entre eux, ils ne sont même pas bien distribués : par exemple, les patterns dans **EP1** sont très proches du pattern P1 et les

patterns dans **EP2** sont très proches du P1 ou du P6. Il est évident que ce genre de distributions ne favorisent pas du tout la création des attracteurs.

Dans nos expériences pour tester les bassins d'attraction, nous nous somme intéressés plutôt aux points loins des attracteurs. En fait, nous avons testé les 256 patterns encodées en huit bits de 1 et -1. Les statisiques ont été faites sur le nombre des patterns qui sont finalement "attirés" par chaque attracteur. Le tableau 1.1 et le tableau 1.2 montrent les résultats concernant le cas où l'ensemble des patterns exemplaires est **EP1** et le cas où l'ensemble des patterns exemplaires est **EP2**. Il faut souligner que, tous les pattrens exemplaires sont bien appris au sens que, le vecteur de sortie est très proche du vecteur d'entrée pris dans l'ensemble des patterns exemplaires, mais il n'y a pas toujours un attracteur associé à chaque pattern appris. Sur ces deux tableaux, on a des remarques suivantes :

1° Le réseau 8-8 arrive toujours, sans trop de difficultés, à créer un attracteur pour chaque pattern exemplaire, donc, le **Théorème 1.3** est vérifié. Le nombre des patterns qui sont attirés par les attracteurs parasites augmente d'une manière monotone.

2° Le réseau 8-3-8 est le réseau qui a le plus du mal à créer des attracteurs. Il n'a pas pu créer un attracteur pour chaque pattern exemplaire. Au bout de 500 itérations d'apprentissage, il y a 4 attracteurs contre 5 patterns pour **EP1** et 6 attracteurs contre 10 patterns pour **EP2**. De plus, ce sont toujours des patterns "mal placés" qui n'ont pas d'attracteur associé, par exemple P1 dans **EP1** et P1 P2 dans **EP2**.

3° Le réseau 8-4-8 semble avoir la même performance pour la création des attracteurs que le réseau 8-8, mais la variation du nombre des patterns attirés par des attracteurs parasites est difficilement interprètable, ce qui est aussi le cas pour le réseau 8-3-8.

	50	100	200	300	400	500
88	1 | 256 | [0]	1 | 256 | [0]	1 | 256 | [0]	5 | 94 45 27 54 36 | [0]	5 | 84 31 33 27 27 | [54]	5 | 81 27 27 27 34 | [60]
838	1 | 256 | [0]	5 | 121 35 51 6 43 | [0]	4 | 44 50 46 78 | [38]	4 | 42 40 32 90 | [52]	4 | 74 49 60 73 | [0]	4 | 44 26 20 90 | [76]
848	1 | 256 | [0]	2 | 216 40 | [0]	5 | 51 60 31 56 15 | [43]	5 | 68 33 29 29 34 | [63]	5 | 33 90 21 101 11 | [0]	5 | 27 61 46 61 61 | [0]

Tableau 1.1 : Statistique des nombres d'attracteurs associés aux patterns appris, des nombres de patterns attirés par ces attracteurs et des nombres de patterns attirés par des attracteurs parasites pour l'ensemble de patterns exemplaires **EP1**. En haut, les nombres 50, 100, ..., 500 correspondent aux étapes d'itération d'apprentissage. 88, 838, 848 correspondent aux trois réseaux. Dans chaque case, le nombre à côté gauche du vertical représente le nombre des attracteurs associés aux patterns exemplaires, le nombre dans le carré à côté droit du vertical représente le nombre des patterns qui sont attirés par des attracteurs parasites (dont le nombre n'est pas affiché) et le reste des nombres à côté droit du vertical représente la distribution du nombre des patterns qui sont attirés par chaque attracteur associé au pattern exemplaire.

	50	100	200	300	400	500
88	2 | 157 93 | [6]	10 | 77 18 18 8 19 8 65 7 7 16 | [13]	10 | 73 14 14 9 15 8 66 10 15 11 | [21]	10 | 70 14 15 9 13 13 65 12 11 13 | [21]	10 | 70 13 13 10 12 12 67 11 9 13 | [26]	10 | 67 14 10 8 10 8 14 65 8 12 | [40]
838	2 | 61 38 25 64 25 42 | [1]	4 | 64 35 68 34 | [55]	6 | 37 34 40 38 40 37 | [30]	6 | 57 30 23 63 32 20 | [31]	8 | 86 18 28 20 19 18 46 11 | [0]	6 | 63 45 38 34 39 37 | [0]
848	4 | 85 31 105 34 | [1]	10 | 22 14 22 21 22 22 21 23 29 16 | [44]	8 | 46 30 20 20 20 19 48 30 | [23]	10 | 10 23 18 22 14 10 23 14 20 22 | [80]	10 | 12 17 18 28 22 29 22 18 15 19 | [56]	10 | 32 16 6 11 15 19 18 44 8 12 | [75]

Tableau 1.2 : Même statistique que montre le **Tableau 1.1** mais pour l'ensemble de patterns exemplaires **EP2**.

Il y a une coincidence entre le 8-8 et 8-4-8 : ils ont prèsque le même nombre de connexions (une différence de 4 connexions pour représenter les seuils). Cela ne serait pas une explication pertinente de leur performance semblable. En fait, la même expérience faite pour l'ensemble **EP3** qui contient 15 patterns exemplaires montre que, le réseau 8-8 crée, sans problème, 15 attracteurs associés à chaque pattern, mais le réseau 8-4-8 ne crée que 12 même pour un apprentissage prolongé jusqu'à 1000 itérations. Par contre, une comparaison entre la performance du réseau 8-3-8 et du réseau 8-4-8 pourrait donner des révélations intéressantes. On sait que ce sont toujours des patterns mal placés comme P1 et P2 qui posent des problèmes pour le réseau 8-3-8, on a tout à fait raison de dire que, le fait d'avoir une cellule cachée de plus rend le réseau 8-4-8 capable de créer plus de représentations internes "stables" ou "attractives" pour les patterns appris.

Les Figures 1.5 à 1.22 donnent plus de détails sur l'évolution des attracteurs créés par chaque réseau. Nous montrerons ici seulement les attracteurs associés aux patterns exemplaires de l'ensemble **EP1**. Sur les Figures 1.5-1.10, nous voyons que, jusqu'à 200 itérations d'apprentissage, il y a toujours un seul attracteur associé à P1. Puis il y a un saut au bout de 300 itérations où l'on voit 5 attracteurs qui apparaissent associés à tous les patterns de **EP1**. Ces attracteurs continuent à être "renforcés" au cours des 200 itérations d'apprentissage qui suivent. On dirait que cette présentation manque un peu de finesse, car entre 200 et 300 itérations, l'évolution se fait d'une manière progressive comme l'on a pu constater par un test spéciale sur 230, 250 et 280 itérations d'apprentissage.

Les Figures 1.11-1.16 illustrent des phénomènes plus complexes mais intéressants. Jusqu'à 300 itérations, il y a toujours un attracteur associé à P1 dont la force de chaque composante varie suivant le nombre d'itérations. Puis au bout de 400 itérations, nous ne voyons plus cet attracteur. En même temps, 4 attracteurs associés respectivement à P2, P3, P4 et P5 se sont "confortablement installés". Il semble que ces 4 patterns sont plus compétitifs vis à vis de P1 grâce à la différence entre eux (4 bits de différence). La compétitivité est d'autant plus manifestée que les attracteurs associés à ces patterns sont apparus et ont disparu au cours des 300 premières itérations : à 100 itérations, il y a des attracteurs associés à P2, P3 et P4; à 200 itérations, il n'y a plus d'attracteur associé à P4 mais un attracteur associé à P5; à 300 itérations, l'attracteur associé à P4

"réapparait" mais celui associé à P2 a disparu.

Les Figures 1.17-1.22 illustrent l'évolution des attracteurs créés par le réseau 8-4-8. Il y a beaucoup moins de variation par rapport à celles manifestées par le réseau 8-3-8. Le comportement de ce réseau est proche de celui de 8-8.

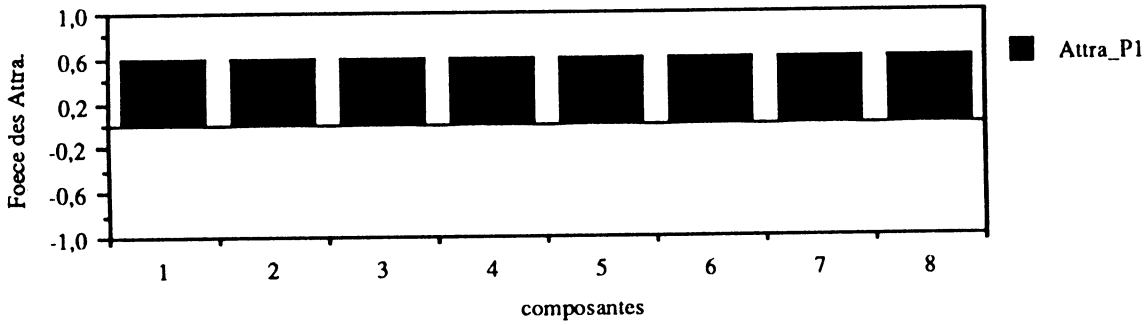


Figure 1.5 : Attracteur, associéé à l'**EP1**, créé par le réseau 8-8 au bout de 50 itérations d'apprentissage.

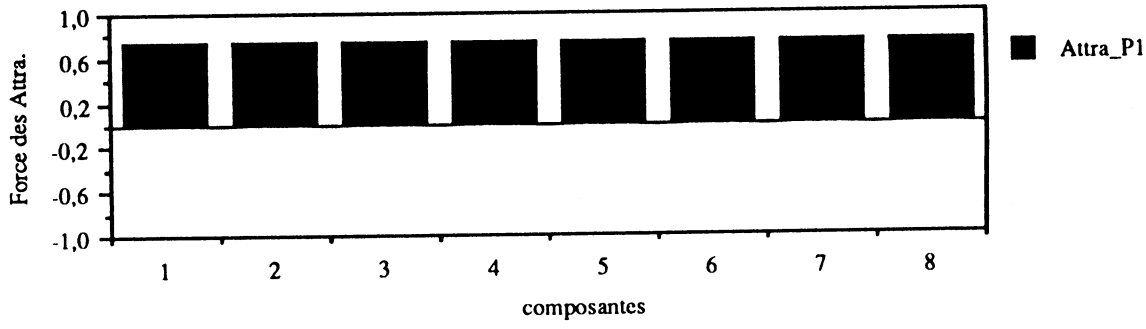


Figure 1.6 : Attracteur, associé à l'**EP1**, créé par le réseau 8-8 au bout de 100 itérations d'apprentissage.

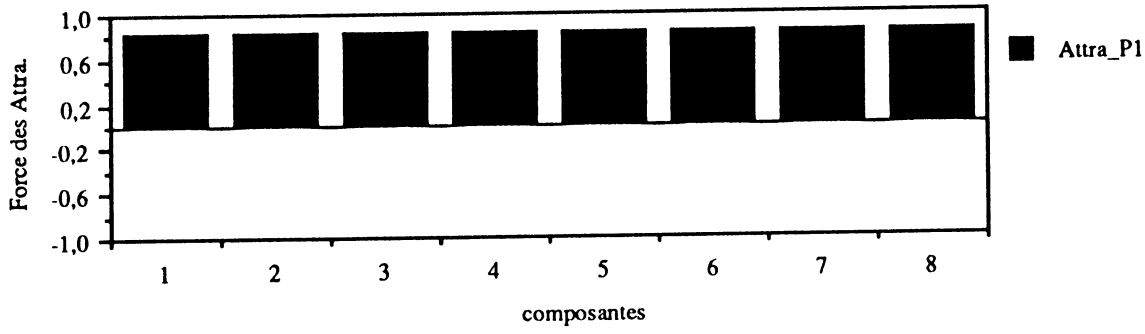


Figure 1.7 : Attracteur, associé à l'**EP1**, créé par le réseau 8-8 au bout de 200 itérations d'apprentissage.

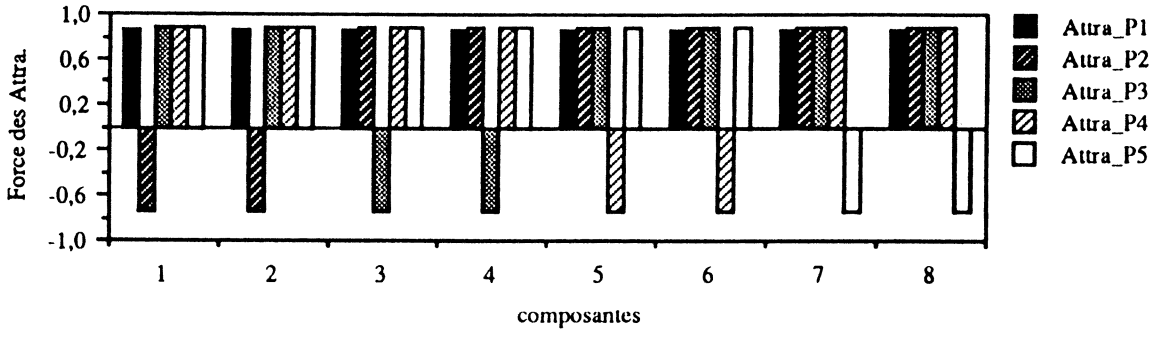


Figure 1.8 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-8 au bout de 300 itérations d'apprentissage.

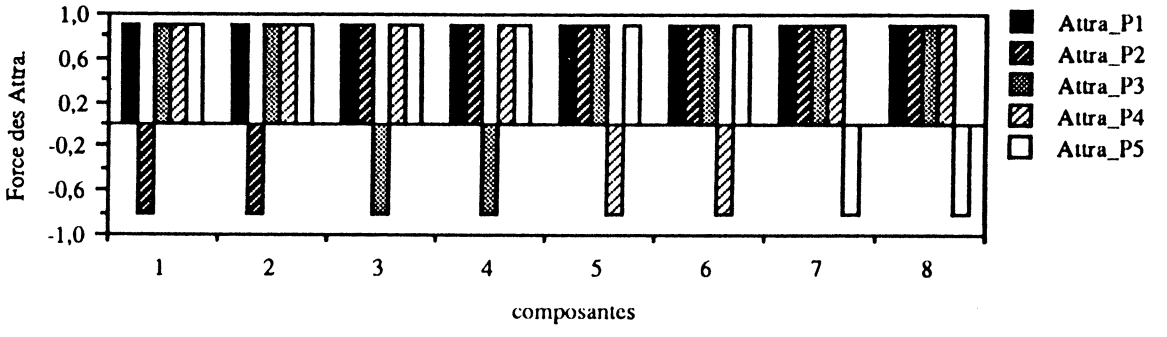


Figure 1.9 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-8 au bout de 400 itérations d'apprentissage.

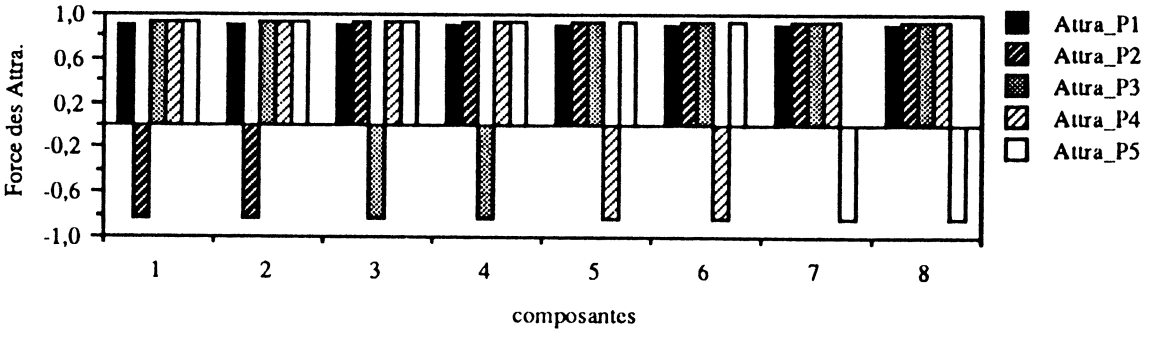


Figure 1.10 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-8 au bout de 500 itérations d'apprentissage.

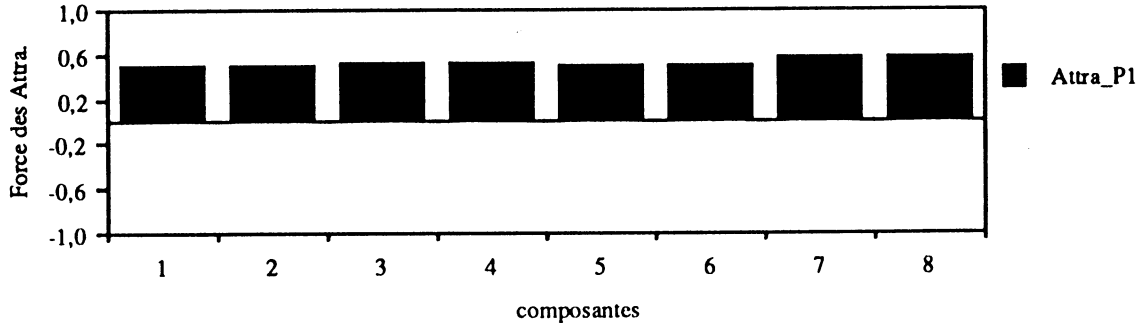


Figure 1.11 : Attracteur, associé à l'**EP1**, créés par le réseau 8-3-8 au bout de 50 itérations d'apprentissage.

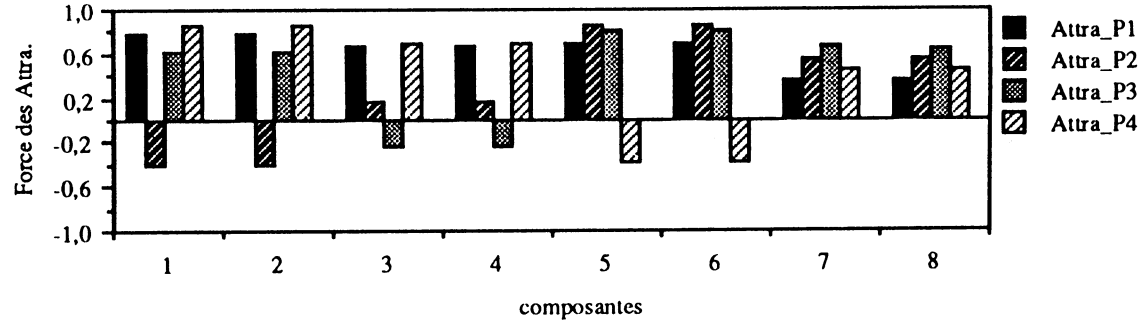


Figure 1.12 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-3-8 au bout de 100 itérations d'apprentissage.

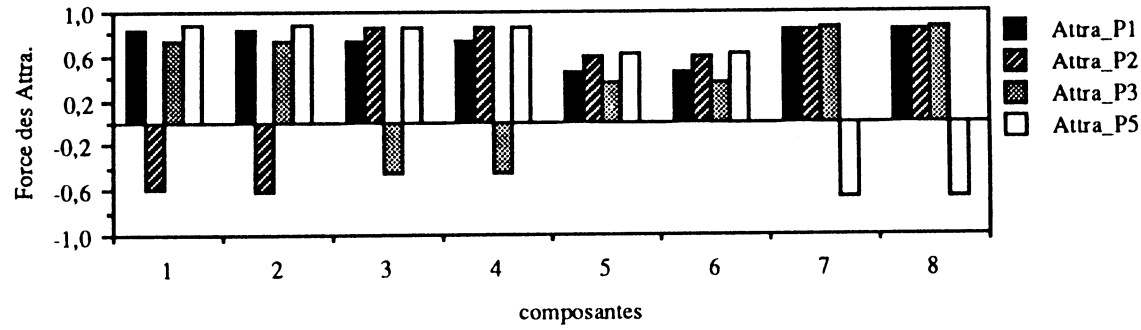


Figure 1.13 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-3-8 au bout de 200 itérations d'apprentissage.

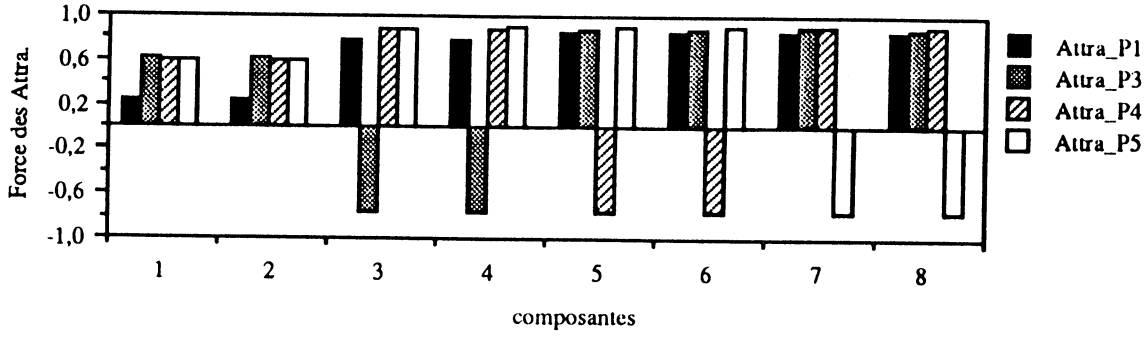


Figure 1.14 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-3-8 au bout de 300 itérations d'apprentissage.

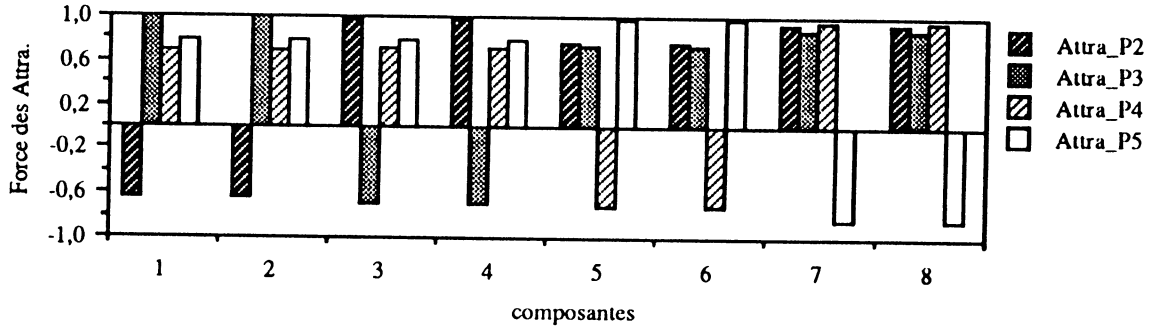


Figure 1.15 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-3-8 au bout de 400 itérations d'apprentissage.

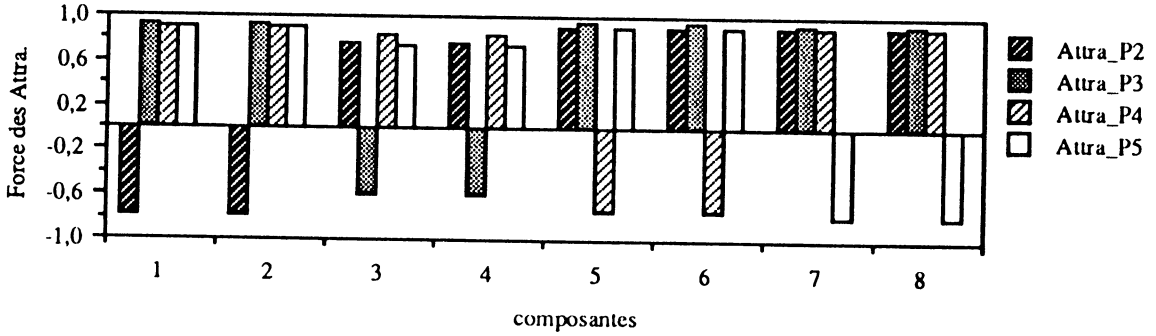


Figure 1.16 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-3-8 au bout de 500 itérations d'apprentissage.

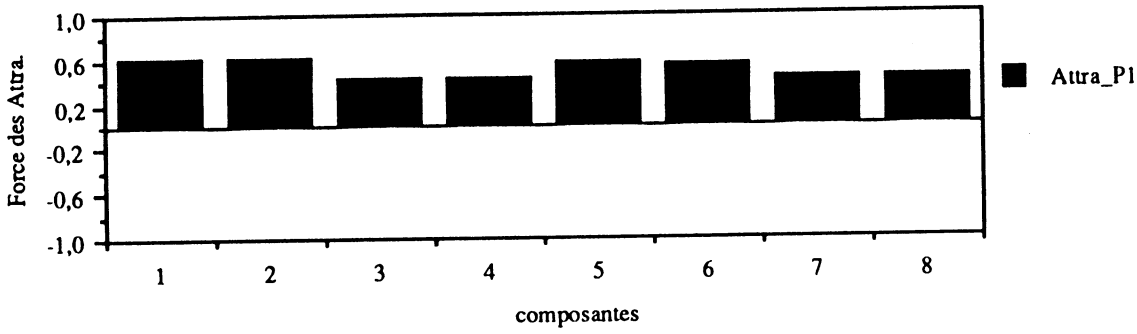


Figure 1.17 : Attracteur, associé à l'**EP1**, créé par le réseau 8-4-8 au bout de 50 itérations d'apprentissage.

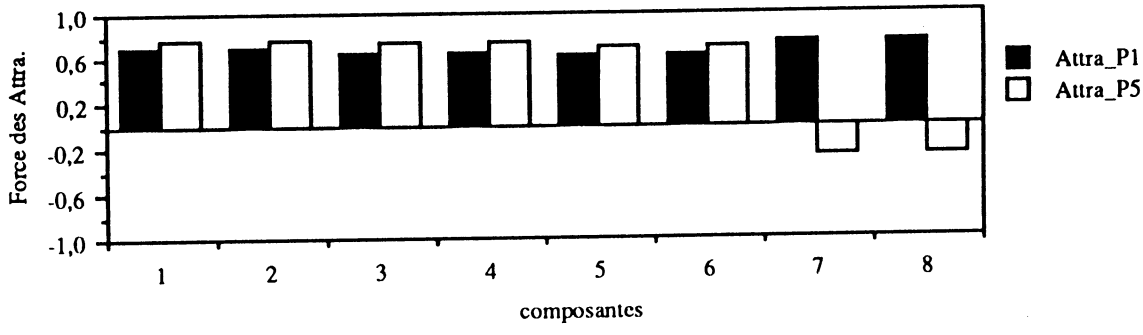


Figure 1.18 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-4-8 au bout de 100 itérations d'apprentissage.

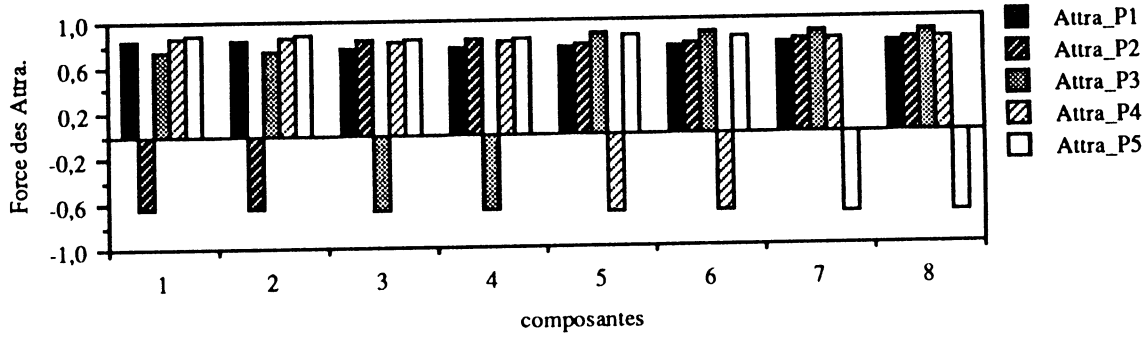


Figure 1.19 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-4-8 au bout de 200 itérations d'apprentissage.

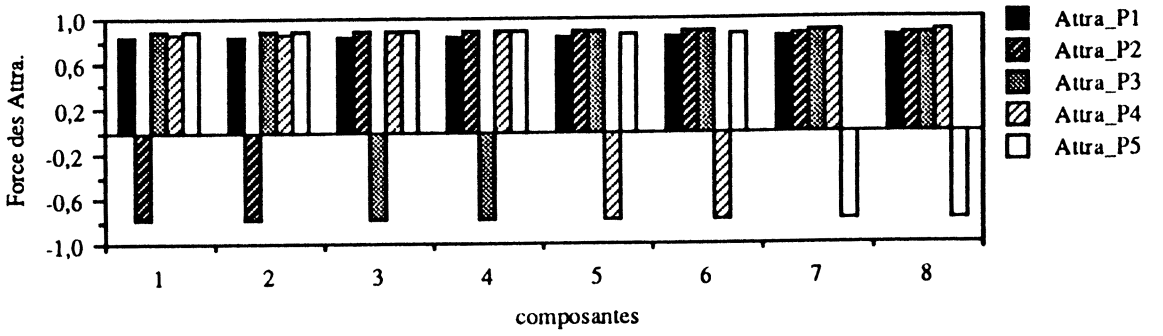


Figure 1.20 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-4-8 au bout de 300 itérations d'apprentissage.

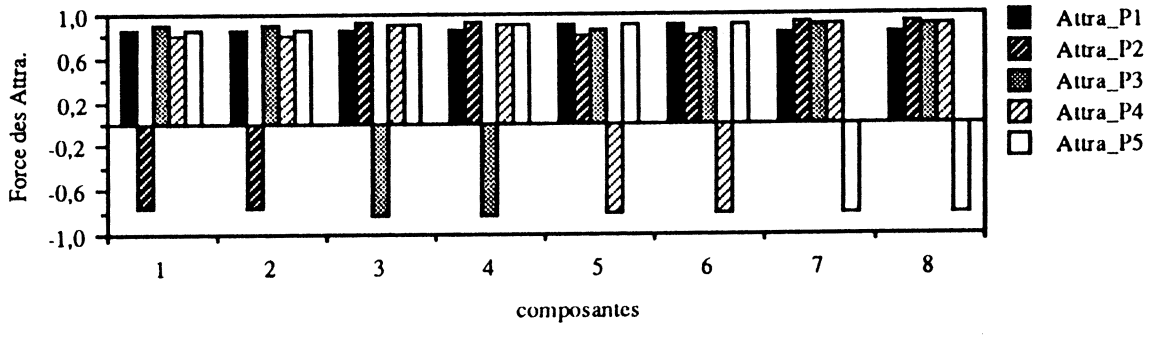


Figure 1.21 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-4-8 au bout de 400 itérations d'apprentissage.

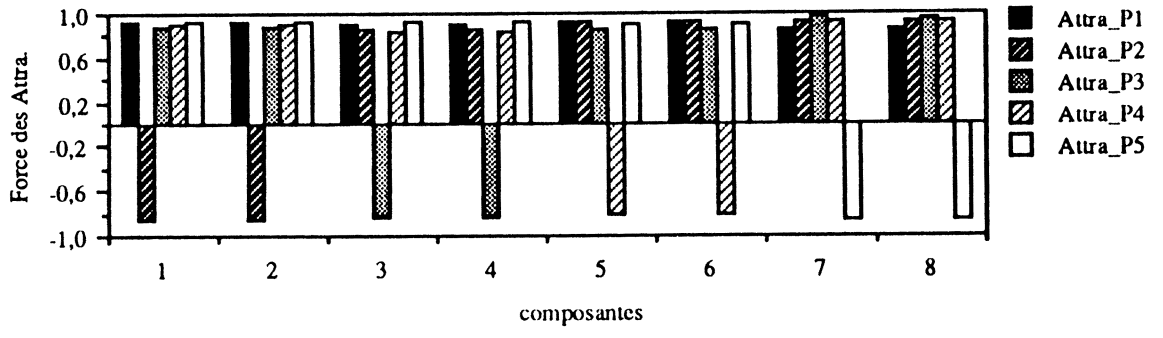


Figure 1.22 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-4-8 au bout de 500 itérations d'apprentissage.

Nous nous sommes également intéressés au problème de la vitesse de convergence d'un réseau vers un attracteur, c'est-à-dire que l'on initialise l'entrée du réseau par un pattern et regarde en combien d'itérations le réseau peut converger vers un attracteur ainsi que les vecteurs sur le trajet. Ce pattern initial n'est pas pris au hasard puisque nous avons voulu cibler les tests sur des attracteurs spécifiques. Pour cette raison, nous avons récupéré, pour chaque

attracteur y compris l'attracteur parasite, l'ensemble de tous les patterns permettant d'initialiser le réseau pour qu'il converge vers l'attracteur.

On a observé que si un réseau peut créer un attracteur pour chaque pattern exemplaire, comme ceci est le cas pour le réseau 8-8, le réseau 8-4-8 dont l'ensemble de patterns exemplaires est **EP1** ou **EP2**, la convergence est toujours assez rapide : en général le nombre d'itérations est inférieur à 10. Si ce n'est pas le cas, comme pour le réseau 8-4-8 avec l'ensemble **EP3** ou le réseau 8-3-8, la convergence vers certains attracteurs peut être très lente. La Figure 1.23 illustre un cas du réseau 8-4-8 dont l'ensemble de patterns exemplaires est **EP3**. Le réseau, partant du pattern

(-1, -1, -1, 1, 1, 1, 1, -1)

converge vers l'attracteur associé à P1. On voit sur la figure que la sortie du réseau a beaucoup de mal à "quitter" le zone proche de la configuration

(-1, -1, 1, 1, 1, 1, 1, 1)=P2

pour converger vers l'attracteur associé à P1. En fait, P2 est dans l'ensemble des patterns exemplaires. Il a été très bien "appris" par le réseau, mais il n'y pas d'attracteur créé associé à ce pattern.

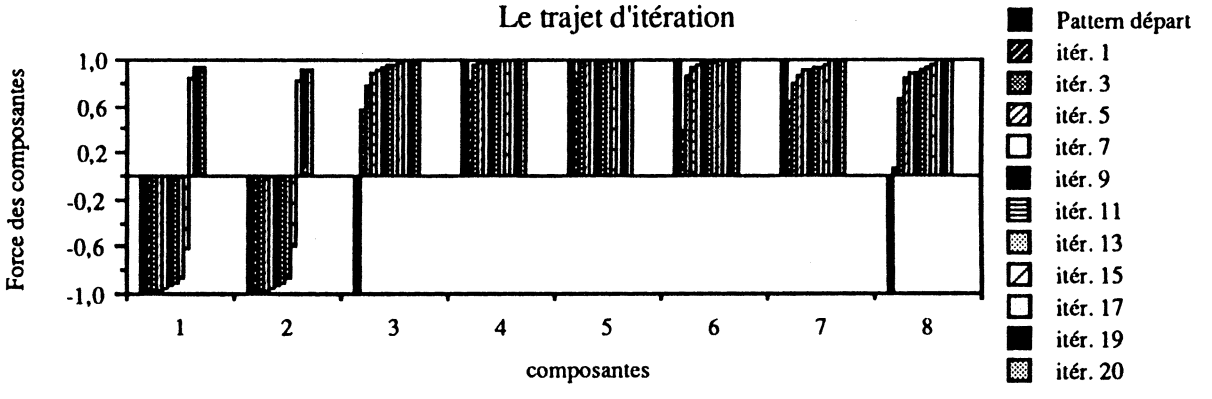


Figure 1.23 : Attracteurs, associés à l'**EP1**, créés par le réseau 8-4-8 au bout de 500 itérations d'apprentissage.

Dans ce paragraphe, nous avons démontré que, sous la condition que les patterns exemplaires soint codés en 1 et -1, les réseaux fonctionnant en GBP tentent de réaliser des fonctions contractantes. Dans le cas particulier, où l'objectif d'apprentissage est de réaliser des autoassociations, les réseaux ont tendance à créer des attracteurs associés aux patterns appris. Ces propriétés sont prouvées pour des réseaux à 2 couches. La création des attracteurs est aussi

vérifiée par des résultats expérimentaux.

Nous avons aussi effectué, sur des réseaux à trois couches, des expériences sur la création des attracteurs. Ces réseaux multicouches ont des comportements assez complexes. Le fait d'ajouter une seule cellule intermédiaire à un réseau 8-3-8 (pour former un réseau 8-4-8) résulte en une augmentation très importante de capacité de stockage. La conclusion que l'on peut tirer de ces expériences est que, les réseaux multicouches tentent de créer des attracteurs d'une manière plus sélective. Dans les modélisations plus avancées, comme celle présentée au Chapitre III, ces propriétés de réseaux multicouches seront exploitées.

I.4 Réseaux séquentiels : une extension importante de l'algorithme de rétro-propagation du gradient

L'objectif de ce paragraphe est de présenter des réseaux multicouches possèdant des connexions récurrentes. Les sorties des réseaux à chaque instant sont influencées par les sorties aux instants précédents. Cette influence sera présente non seulement pendant la reconnaissance mais aussi pendant l'apprentissage. En fait, ces réseaux sont conçus spécialement pour le traitement des séquences, qui est un problème très compliqué dans les simulations des actions humaines notamment dans la reconnaissance de la parole. Notre intention ici n'est bien sûr pas de proposer une solution complète à ce problème, mais plutôt de montrer comment on peut obtenir un nouvel outil, qui est basé sur un modèle (connexioniste) de calcul parallèle et distribué.

I.4.1 Problème d'apprentissage et de traitement des séquences

Les études sur le comportement humain montrent que les mouvements du corps, la parole et même la pensée semblent impliquer l'apprentissage de séquences d'événements qui se succèdent l'un après l'autre dans le temps. On peut effecturer énormément de séquences ou répéter une même séquence dans des contextes très variés. De plus, la plupart des séquences sont apprises au travers d'exemples.

La difficulté principale avec le traitement des séquences par un système informatique est l'aspect parallèle de l'ordre séquentiel. Un exemple typique est la *coarticulation* dans la parole. Ce phénomène illustre deux formes principales de parallelisme des actions (les théoriciens les distinguent ainsi) : l'une donne l'illusion d'une exécution "superposée" des actions qui sont dans la séquence; l'autre apparaît comme l'exécution simultanée de plus de deux actions. Les efforts pour caractériser les deux formes de parallélisme dans les processus séquentiels ont largement contribué à l'instauration de la théorie de l'ordre séquentiel (serial order theory) (Lashley, 1951; Shaffer, 1976; Sternberg, Monsell, Knoll, & Wright, 1978; Jordan 1986)

Un exemple d'ordre séquentiel : la coarticulation :

Le phénomène de coarticulation, bien connu dans le traitement de la parole, est un très bon exemple pour montrer l'aspect parallèle de l'ordre séquentiel.

Moll et Daniloff (1971) montrent que, dans la prononciation du mot anglais *freon*, l'ouverture vélaire du nasal [*n*] pourrait débuter aussi tôt que la première voyelle est prononcée, ce qui par conséquent nasalise les voyelles. Benguerel et Cowan (1974) ont étudié des phrases telles *une sinistre structure*, où il y a une chaine de six consonnes [*strstr*] suivis par la voyelle courte [*y*]. Ils montrent que l'arrondissement des lèvres pour [*y*] pourrait commencer dès la première consonne [*s*]. Ce sont des exemples de la *coarticulation anticipée*. On peut aussi observer les phénomènes de persévération appelées, *coarticulation retardée*.

Un système de traitement de la parole doit tenir suffisamment compte des articulateurs libres et les utiliser dans l'anticipation des actions futures. Mais le problème est compliqué si l'on doit prendre soin des contraintes exceptionnelles d'articulation. Par exemple, dans le cas de [*strstry*], seul l'arrondissement du [*y*] peut être anticipé. Le voisement du [*y*], entrainant, lui aussi, un articulateur qui n'est pas utilisé par les consonnes, ne peut pas être anticipé parce que cela va changer l'identité des consonnes ([*s*] deviendrait un [*z*]). La parole présente donc un problème difficile de contrôle distribué, où des contraintes sont imposées sur les parallelismes et la séquentialité qui peuvent être obtenues dans une prononciation.

La théorie de l'ordre séquentiel :

Il existe deux approches classiques en ce qui concerne le traitement des séquences : celle dite de *chaine associative* et la *technique de "buffer"*. Dans l'approche en chaine associative, on suppose que l'ordre séquentiel est codé par des liens orientés entre les éléments de contrôle qui représentent les actions à effectuer, et que l'exécution d'une séquence implique le suivi d'un chemin des éléments de contrôle. L'approche de la technique de "buffer" est basée sur la métaphore informatique. Le "buffer" est chargé avec des actions à exécuter et il y a un programme compteur qui compte les étapes dans le "buffer".

Les deux approches ont des inconvénients très graves. Lashley (1951) argumente contre l'approche en chaine associative à cause de son incapacité de spécifier le choix des liens issus d'un élément lorsqu'il y en a plusieurs, ce qui ne lui permet pas de traiter des ordres différents des mêmes actions. L'extension proposée par Wickelgren (1969) consiste à utiliser différents éléments pour représenter la même action dans des chaines différentes ($_{A}B_{C}$ pour B dans ABC et $_{C}B_{A}$ pour B dans CBA). Ceci résoud le problème de spécification mais pose plusieurs autres problèmes, comme la necessité d'un nombre surabondant d'éléments, aussi bien pour représenter des mots isolés comme *barbarement*, que pour tenir compte des effects contextuels. Cette méthode rend également très difficile la manipulation des types. Quant à l'approche par la technique du "buffer", bien qu'elle soit très générale, elle semble trop simple pour permettre de manipuler la coarticulation et d'introduire les interactions entre les différentes positions du "buffer" (Shaffer et Hardwick, 1970).

Une approche parallèle a été proposée par Fowler, (1980); et Rumelhart & Norman, (1982) qui supposent que les actions soient produites par l'influence simultanée de plusieurs éléments de contrôle. Pour produire l'ordre temporel, on utilise des connexions d'inhibition latérale qui permettent de "supprimer" des éléments "inutiles" (qui ne devraient plus participer à la décision de sortie). Cette approche permet, dans une certaine mesure, de tenir compte des effets coarticulatoires, mais elle tombe aussi sous la critique de Lashley, car elle manque de mécanismes pour choisir les connexions inhibitrices utilisées dans l'exécution d'une séquence particulière.

Récemment, Jordan (1986) a proposé une approche sous forme de réseau connexionniste. L'idée est de représenter les actions des séquences par des patterns d'activation d'un ensemble d'éléments ("représentation distribuée") et de faire une claire distinction entre l'état et la sortie du système de production. L'ordre séquentiel est encodé ici dans la fonction de sortie et dans les connexions récurrentes qui affectent les cellules représentant l'état du réseau, au lieu d'être encodé dans des connexions directes entre les cellules de sortie (d'action). C'est ce qu'on appelle les *réseaux séquentiels*. Le paragraphe suivant sera consacré à la présentation de ces réseaux.

I.4.2 Introduction de feedback dans des réseaux multicouches

L'idée des réseaux séquentiels que l'on va aborder consiste à adapter le modèle de réseaux multicouches aux problèmes de traitement d'information, où la notion du temps intervient obligatoirement. Le principal changement par rapport au modèle original tient dans la participation active des états précédents du réseau à la production de la sortie actuelle.

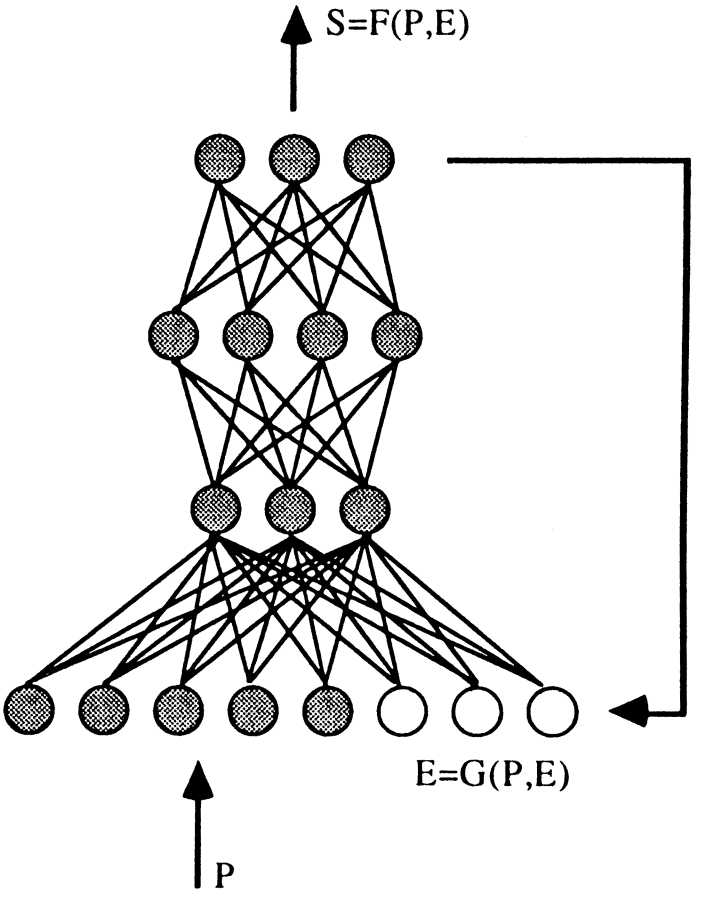


Figure 1.24 : illustration d'un réseau séquentiel.

Représentons la sortie d'un réseau par S et l'état du réseau, qui est le pattern d'activation d'un ensemble de cellules, par E. La fonction de sortie F dépend de l'état du réseau E et de l'entrée (au réseau) P, qui est interprétée ici comme un *plan* : S=F(P,E). L'état du réseau est déterminé par une fonction G qui prend l'état précédent et le plan comme ses variables : E=G(P,E). En utilisant la récurrence, une séquence est produite par une alternance de la mise au point des états et des sorties : S_n=F(P, E_n) et E_{n+1}=G(P,E_n). Le plan P joue le rôle de

clé qui, en principe, caractérise la séquence particulière. La Figure 1.24 donne une illustration des réseaux séquentiels avec des feedbacks.

La fonction de sortie F est apprise par un algorithme d'apprentissage, alors que la fonction de transition d'état G peut être une fonction prédéfinie. La seule contrainte sur cette fonction G est la continuité. La fonction F, elle, doit pouvoir assurer une généralisation telle que pour le plan et les états similaires, la sortie doit également être similaire.

L'apprentissage consiste en un processus d'adaptation pour que le réseau puisse produire des séquences exemplaires. Notons les séquences à apprendre par $X_1^{(k)}$, $X_2^{(k)}$, ..., $X_n^{(k)}$, ..., et le plan associé à chaque séquence (k) par $P^{(k)}$. Pendant l'apprentissage de la séquence (k), la sortie désirée du réseau sera successivement $X_i^{(k)}$, pour l'entrée qui est constituée de deux parties : l'une, correspondant au plan, est toujours $P^{(k)}$; l'autre, correspondant à $E_i^{(k)}$, état du réseau à l'instant i, est choisie soit comme $G(P^{(k)}, X_{i-1}^{(k)})$ soit comme $G(P^{(k)}, S_{i-1}^{(k)})$. Le premier choix est basé sur l'hypothèse que le réseau devrait finalement produire les sorties exactes pour la séquence exemplaire, tandis que le second choix représente une volonté de laisser le réseau évoluer avec plus de "liberté". Le principe de l'algorithme GBP est appliqué, une seule fois par étape, pour réaliser la transformation $(P^{(k)},E_i^{(k)})\rightarrow X_i^{(k)}$.

Le choix de la fonction G dépend beaucoup de l'application envisagée. Jordan (1986) propose une solution :

$$G(P,E)=\mu E+S$$

où $0\leq\mu<1$. Cette fonction permet de garder une influence exponentiellement décroissante des états précédents du réseau sur son état actuel et donc rend possible, théoriquement, le traitement des séquences même s'il y a des sous-séquences répétitives. Jordan a pu montrer la capacité de traitement à l'aide d'exemples, comme l'apprentissage des trajets planaires et des coarticulations dans la prononciation de *sinistre structure* (Jordan, 1986;1988).

En résumé, le problème de l'ordre séquentiel est un problème très important du point de vue des applications et difficile à résoudre à cause de l'occurrence à la fois du parallèlisme et de la séquentialité. Les réseaux

séquentiels fournissent une nouvelle architecture parallèle qui permet de tenir compte des deux aspects de l'ordre séquentiel.

Il faut souligner que l'intérêt du modèle de réseaux séquentiels réside dans son idée d'utiliser les feedbacks temporaires pendant l'apprentissage et la reconnaissance. Cette idée nous a suggéré d'adapter les réseaux multicouches au problème de la détection automatique des formants, une application qui est amenée à prendre de l'ampleur (Bailly, Wang & Yé 1989, en cours de préparation).

I.5 Réseaux de cellules à champ de réception limité (CRL)

Il semble délicat de demander à un seul modèle de réseau d'avoir toutes les capacités nécessaires de traitement d'information. La combinaison de différents modèles pourrait donner des réseaux plus puissants. Le modèle des réseaux qui sera présenté dans ce paragraphe a été proposé pour résoudre le problème de l'apprentissage trop lent de l'algorithme GBP (Moody *et al* 1988). Pour certaines applications (e.g. la reconnaissance de forme), il pourrait être plus efficace. Mais le plus important est la façon dont ces réseaux sont construits qui est assez novatrice.

I.5.1 Présentation des réseaux de cellules quasi-gaussiennes

Les réseaux présentés ici utilisent des cellules qui sont très "activées" dans une zone sphérique et qui sont très "calmes" en dehors de cette zone. Elles ne sont plus celles qui modèlisaient les vrais neurones physiologiques : cellules de McCulloch & Pitts (1943) et cellules sigmoïdales (paragraphe I.1).

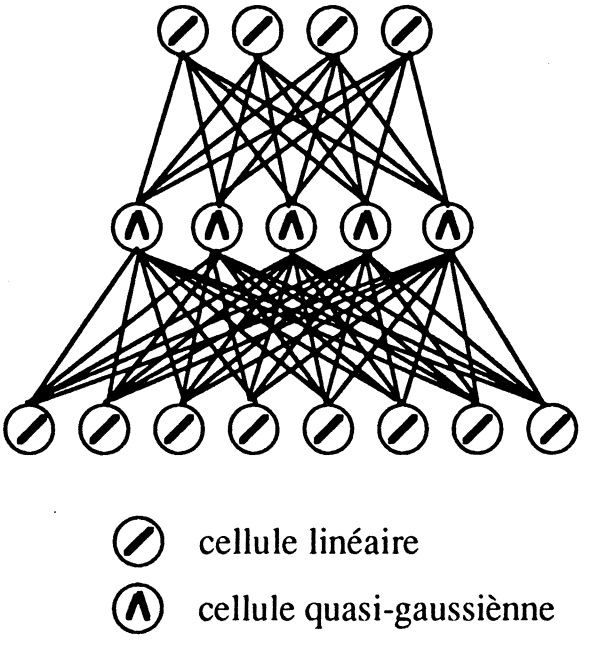


Figure 25 : Un réseau de cellules CRL. Il n'a qu'une seule couche de cellules non-linéaires.

Le modèle consiste pour le moment en des réseaux à trois couches (Figure 25). Les cellules d'entrée et les cellules de sortie sont des cellules linéaires. Les cellules dans la couche intermédiaire sont des cellules de type CRL

(Champ de Réception Limité). Les cellules d'entrée font seulement une combinaison linéaire de données d'entrée. Leurs fonctions sont fixées par l'utilisateur. Si les données sont considérées comme déjà bien préparées, les cellules d'entrée ont à passer les données aux cellules intermédiaires. L'état d'une cellule intermédiaire est déterminée par la formule suivante :

$$R_i(X)=R(\frac{\|X-X^{(i)}\|}{\sigma^{(i)}}) \qquad (1.17)$$

Ici, X est un vecteur de valeurs réelles dans l'espace d'entrée, R_i est la réponse de la $i^{ème}$ cellule CRL, R est une fonction radialement symétrique avec un seul maximum à l'origine et qui tend vers zéro rapidement quand le rayon devient grand, $X^{(i)}$ et $\sigma^{(i)}$ correspondent respectivement au centre et la largeur sensible (pour ne pas dire le rayon) du $i^{ème}$ champ de réception. En pratique, la fonction R peut être choisie, par exemple, comme une gaussienne :

$$R(x)=\exp(-x^2). \qquad (1.18)$$

La sortie du réseau est calculée par :

$$F(X)=\sum_{i=1}^{M} F^{(i)} R_i(X) \qquad (1.19)$$

où $F^{(i)}$ est le vecteur (ou la fonction) associé à chaque champ et M est le nombre total de champs (cellules intermédiaires). X appartenant à R^n et $F^{(i)}$ appartenant à R^p, F réalise donc une transformation de R^n vers R^p. La formule utilisée en pratique pour F(X) s'écrit souvent comme suit :

$$F(X)=\frac{\sum_{i=1}^{M} F^{(i)} R_i(X)}{\sum_{i=1}^{M} R_i(X)} \qquad (1.20).$$

C'est-à-dire que, la sortie F(X) du réseau est normalisée par un facteur qui est la somme des états de toutes les cellules CRL.

Il faut remarquer que $F^{(i)}$, en terme de poids synaptiques, correspond aux connexions de la $i^{ème}$ cellule CRL aux cellules de sortie. Les $F^{(i)}$ constituent une partie de l'ensemble des paramètres modifiables, tandis que l'autre partie consiste en des centres $X^{(i)}$, les largeurs $\sigma^{(i)}$ des champs de réception, ainsi que leur nombre.

I.5.2 L'algorithme d'apprentissage

L'apprentissage est formalisé comme un processus hybride en deux étages. Les centres et les largeurs des champs de réception sont déterminés de manière "bottom-up" (auto-organisation), tandis que les poids $F^{(i)}$ sont obtenus de manière "top-down" par la règle LMS (l'algorithme de Adaline, voir Widrow & Hoff 1960). L'objectif est d'allouer de manière efficace les ressources du réseau en plaçant les centres des champs de réception dans des régions de l'espace d'entrée où les données sont présentes.

Pour trouver les centres, on utilise l'algorithme du k-moyen pour le "clustering", qui consiste à chercher k points dans R^n : $X^{(i)}$ pour i=1, à k qui minimisent une fonction

$$E(X^{(i)} \mid i=1, 2,... k)= \sum_{\substack{i=1,k \\ j=1,N}} M_{ij} \left\| X^{(i)}-x^{(j)} \right\|^2 \qquad (1.21)$$

où $\{x^{(j)} | j=1, ... N\}$ est l'ensemble des patterns d'entrée exemplaires, M_{ij} est la fonction du membre (=1 si $x^{(j)}$ appartient au groupe centré autour de $X^{(i)}$, =0 sinon). La matrice $M=(M_{ij})_{kxN}$ contient donc des 0 et des 1 avec exactement un seul 1 par colonne. Le processus de minimisation est un processus itératif. A chaque étape, on fait varier les candidats (pour les centres) $X^{(i)}$ puis on fait une évaluation des fonctions du membre M_{ij} où chacun des patterns exemplaires $x^{(j)}$ doit être dans le groupe centré autour du $X^{(i)}$ qui est le plus proche (de $x^{(j)}$) parmi tous les candidats. L'itération s'arrête quand on rencontre un minimum (local).

Une version pratique de l'algorithme itératif pour la minimisation est donnée. Elle n'a pas besoin de stocker les exemplaires. Si x(t) représente le pattern exemplaire fourni à l'instant t et $X^{(i0)}$ représente le candidat le plus proche, alors ce candidat sera modifié selon la formule :

$$\Delta X^{(i0)}=\gamma(x(t)-X^{(i0)}) \qquad (1.22)$$

où le taux d'apprentissage γ est choisi grand au début et il sera diminué lentement vers zéro (selon un schéma de "recuit" par exemple).

Après avoir trouvé l'ensemble des centres, il suffit de minimiser une autre fonction :

$$E(\sigma^{(i)} \mid i=1, 2,... k)=\frac{1}{2}\sum_{i=1,k}\left[\sum_{j=1,k} R_i(X^{(j)})\left\|\frac{X^{(i)}-X^{(j)}}{\sigma^{(i)}}\right\|^2-P\right]^2 \qquad (1.23)$$

où P est un paramètre de superposition. La fonction ne dépend pas explicitement des patterns exemplaires. La minimisation de cette fonction permet d'assurer que les cellules CRL réalisent une interpolation lisse et contiguë des régions dans l'espace d'entrée qu'elles représentent.

La recherche des $F^{(i)}$ fait appel à un algorithme d'apprentissage supervisé. Si la sortie désirée pour l'entrée $x^{(j)}$ est représentée par $F^*(x^{(j)})$, alors la fonction :

$$E(F^{(i)} \mid i=1, 2,... k)=\frac{1}{2}\sum_{j=1,N}\left\|F^*(x^{(j)})-F(x^{(j)})\right\|^2 \qquad (1.24)$$

doit être minimisée en utilisant l'algorithme de Adaline ou LMS. Une formule adaptative pour trouver les $F^{(i)}$ est :

$$\Delta F^{(i)}(t)=\gamma\,(F^*(x(t))-F(x(t)))\;R_i(x(t))$$

où x(t) représente évidemment le pattern exemplaire présenté à l'instant t.

Le modèle peut être utilisé pour réaliser des "mappings" aléatoires d'un espace à l'autre et pourrait donc être efficace pour des problèmes de classification, plus particulièrement pour la reconnaissance de formes. L'application astucieuse des techniques d'auto-organisation, qui accomplissent une transformation des données (d'entrée) d'un espace à un autre, permet aux cellules de sortie, pourtant linéaires, d'effectuer un traitement très efficace. Cette idée de combiner deux modèles en un seul constitue une approche très intéressante dans le domaine des modèles connexionistes.

Le modèle a été appliqué à un problème de prédiction des séquences générées par l'équation différentielle de Mackey-Glass avec le délai du temps (Moody *et al* 1988). Il donne des résultats meilleurs que celui de GBP aussi bien en vitesse d'apprentissage qu'en généralisation. Bien que les raisonnemnts avancés dans cet article ne soient pas incontestables (car on a comparé les temps d'apprentissage entre deux réseaux implantés sur deux machines de puissance incomparable; de plus le nombre de paramètres modifiables dans les deux réseaux n'a pas été précisé), on peut néanmoins avancer que l'algorithme GBP semble moins performant que le modèle présenté ici vis-à-vis des problèmes d'apprentissage de séquences. L'algorithme GBP exige souvent un certain

prétraitement des données pour détecter la régularité et il n'est pas très efficace dans sa version originale pour traiter les séquences, comme nous l'avons déjà souligné dans le paragraphe I.4 ci-dessus.

Pour certaines tâches, comme les "mappings" aléatoires, ce modèle semble plus efficace que le modèle GBP. Mais il est moins souple en architecture et pas assez général pour pouvoir y intégrer plusieurs fonctionnalités. L'apprentissage nécessite une collection abondante de patterns exemplaires pour que l'auto-organisation dans la recherche des centres soit réussie.

I.6. Conclusion du chapitre

Le modèle de réseaux multicouches avec l'algorithme GBP a été présenté au début du chapitre. Il pourrait être utilisé pour résoudre des problèmes dans plusieurs domaines tels l'I.A., le traitement de la parole, le traitement d'images, la robotique, etc. Il constitue le noyau autour duquel les travaux de cette thèse sont organisés.

Nous avons présenté également une étude sur le comportement dynamique des réseaux fonctionnant en GBP. Cette étude, qui a été inspirée de notre modélisation de la mémoire des visages, porte sur les propriétés qu'un réseau pourrait exhiber après apprentissage. La contractivité de la fonction réalisée par un réseau est directement liée au problème de généralisation et se présente, dans le cas particulier où le réseau doit réaliser des auto-associations, comme une clé pour la création des attracteurs associés aux patterns exemplaires. Cette étude, bien qu'elle ne soit pas encore très approfondie, sert de base théorique à notre application de reconnaissance des visages.

Le traitement parallèle et distribué dans les modèles connexionnistes apporte de nouvelles perspectives à des problèmes qui n'ont pas été très bien résolus par les modèles classiques. Le problème de l'ordre séquentiel, ainsi qu'une théorie basée sur le modèle de réseaux multicouches ont été présentés. Les réseaux multicouches munis de feedbacks au cours de l'apprentissage constituent une extension très importante du modèle, qui lui permet de traiter une classe de problèmes difficiles, notamment la parole et la robotique.

Un modèle des réseaux multicouches très différent du modèle GBP a été présenté. Ce modèle utilise des cellules à champ de réception limité. Il résulte d'une idée astucieuse de combinaison : les modèles simples pourraient être combinés pour former des modèles plus compliqués ayant, bien sûr, des capacités supérieures. Ce serait sûrement une voie à suivre pour l'amélioration de l'algorithme de GBP.

Chapitre II

Implantation des réseaux multicouches sur une machine Hypercube

II.1. Introduction.

Les progrès technologiques obtenus dans le domaine du calcul scientifique ces dernières années ont fortement contribué au regain d'intérêt pour les réseaux de neurones. De nombreuses machines parallèles ont vu le jour, tels le CRAY, l'IBM 3090, l'Alliant, l'Hypercube (Intel, FPS, n-cube), la Connection machine etc... Ces machines non spécialisées ont permis de résoudre beaucoup de problèmes de très grande taille dans le domaine des prévisions météorologiques, de la réaction nucléaire, de la médecine, de l'Intelligence Artificielle, etc, problèmes qui n'ont pas pu être bien résolus par des machines monoprocesseurs, dites traditionnelles (ou conventionnelles). Par ailleurs, les progrès dans les technologies de circuit VLSI et WSI, voire en optique, permettent maintenant de réaliser des applications spécialisées et atteignent des vitesses de calcul très importantes : plusieurs dizaines de millards d'opérations flottantes par seconde.

Quels sont les impacts de ces technologies sur les modèles de réseaux de neurones ?

En 1982, 1984 et 1985, après avoir formulé mathématiquement le comportement dynamique des réseaux monocouches avec des connexions complètes et symétriques, Hopfield et ses collègues ont proposé une architecture de circuit VLSI qui pourrait être utilisée pour résoudre le problème du voyageur de commerce. Un prototype a été réalisé qui leur permet de trouver, en une durée très brève, de bonnes solutions (pas forcément optimales) du problème, pour un nombre maximum de dix villes. Ceci a grandement stimulé la recherche en réseaux de neurones et semblait très prometteur à l'époque.

Depuis plusieurs années, le modèle de Hopfield et celui de Kohonen (tous les deux utilisent la principe de corrélation) font l'objet d'études de réalisation "hard". Il faut citer plusieurs personnes : Graf *et al* (1988), Thakoor *et al* (1986), Weinfeld (1988) (sans pretendre être exhaustif). Ces auteurs

emploient plusieurs technologies, digitales ou analogiques, qui dépendent de leurs moyens pour concevoir des circuits du même modèle. Jusqu'à prèsent, la plupart des circuits réalisés ne contiennent qu' une dizaine à une centaine de cellules.

Parmi les travaux de réalisation, il faut noter de très importantes contributions des chercheurs grenoblois. Peretto et Niez du LETI-CENG ont réalisé une machine spécialisée. A. Guérin du LTIRF a construit une machine à l'architecture systolique qui permet de simuler plusieurs types de réseaux neuromimétiques. Hurat et Blayo (Hurat, 1989) du LGI-IMAG, ont réalisé un circuit systolique qui implante le réseau de Hopfield pour la reconnaissance. La performance peut atteindre 0.6 milliard d'opérations par seconde.

Le modèle de Hopfield, bien qu'à l'origine de la renaissance des réseaux de neurones, n'a pourtant que des domaines d'application très limités. Il est utilisé essentiellement pour réaliser des mémoires associatives. Jusqu'à prèsent la plupart des applications importantes ont été réalisées par des modèles beaucoup plus sophistiqués et complexes : e.g. le modèle de réseaux multicouches avec l'algorithme de GBP, la Boltzmann Machine, etc... La réalisation "hard" de ces modèles complexes s'avère très difficile. Ceci est dû au fait que, l'architecture des réseaux n'étant pas régulière, et l'ensemble des poids de connexions ne pouvant pas être représenté de manière efficace par une ou plusieurs matrices, la nécessité de la reconfigurabilité n'est plus réductible à une simple manipulation d'ajout ou de suppression de quelques cellules, mais il s'agit de pouvoir de plus, redistribuer (concevoir librement) les connexions entre les cellules. La présence des cellules cachées, qui est indispensable pour obtenir une capacité supérieure, rend les algorithmes d'apprentissage beaucoup plus compliqués que ceux des réseaux monocouches.

Dans ces types de réseaux, les informations circulent dans les deux sens d'une connexion, ce qui se traduit en termes d'opération matricielle (le cas le plus simple) comme la multiplication d'une matrice par un vecteur et celle de la matrice transposée par un autre vecteur. Pour une réalisation VLSI, la transposition d'une matrice nécessite une permutation des données, ce qui est en pratique très coûteux. Or les méthodes qui permettent de résoudre ces problèmes de multiplication sans effectuer la transposition de matrice exigent, de leur part, des contrôles et des communications tellement sophistiqués qu'ils leur font

perdre beaucoup des caractéristiques des circuits spécialisés.

Il semble que les machines spécialisées en architecture parallèle soient très prometteuses pour les applications des réseaux de neurones. Actuellement, il existe déjà beaucoup de machines ou de cartes commerciales qui peuvent simuler plusieurs modèles de réseaux de neurones, comme par exemple les Mark III et Mark IV de la compagnie HNC, Delta floating point processor de Int'l Corp (réf à ajouter). Mais ces machines, munies d'unités spécialisées pour effectuer les calculs matriciels, imposent encore des contraintes très fortes sur l'architecture des réseaux, et de plus elles ne sont pas vraiment des machines parallèles.

Alors, quel est le rôle joué par les machines parallèles dans le développement des réseaux de neurones ?

Simuler des réseaux de neurones nécessite non seulement une grande puissance en termes de vitesse de calcul mais aussi une souplesse du système. Une machine conventionnelle monoprocesseur peut offrir une grande souplesse pour manipuler n'importe quel genre de réseaux, mais la puissance est loin d'être suffisante du fait qu'elle ne permet pas d'exploiter le parallélisme intrinsèque des réseaux. Les circuits VLSI, comme soulignés ci-dessus, offrent une puissance de calcul surprenante, mais ils ne peuvent pas être utilisés pour simuler des réseaux dont l'architecture n'est pas régulière. Une fois fabriqué, un circuit ne peut plus être reconfiguré. Seul les machines parallèles peuvent réaliser un compris satisfaisant entre puissance et souplesse.

Les implantations des réseaux de neurones sur des machines parallèles ont permis aux chercheurs de mener plusieurs expériences difficilement supportables par des machines conventionnelles. Le programme NETtalk de Terrence Sejnowski tourne 200 fois plus vite sur la Connection Machine de Thinking Machine Corporation que sur un Vax 780. Avec cette machine, il n'est plus nécessaire d'attendre 15 jours pour finir cette expérience. Un autre exemple est donné dans le chapitre IV. Avec une implantation de réseaux multicouches sur la machine hypercube FPS, nous avons pu effectuer une expérience d'apprentissage (pour la reconnaissance de mots isolés) en 14 heures, alors qu'il faudrait au moins six jours de calculs sur un micro-Vax.

Outre son intérêt direct pour la simulation, l'implantation des réseaux

de neurones sur des machines parallèles (ce qui constitue en soi un sujet intéressant) permet d'évaluer la performance de différentes architectures pour cette classe d'applications importante et de mieux comprendre la nécessité architecturale dans la conception des machines parallèles spécialisées aux réseaux de neurones. C'est dans ce triple objectif que nous avons effectué cette étude expérimentale d'implantation sur la machine hypercube FPS.

La machine hypercube FPS est une machine de type MIMD sans mémoire partagée. Un problème d'application doit être, en général, décomposé en plusieurs parties dont chacune est exécutée dans un processeur. Cette exécution est souvent accompagné de communications pour échanger des informations entre les processeurs. La topologie hypercube permet d'inclure plusieurs sous-architectures (l'anneau, l'arbre, les tores, etc), qui pourraient être choisies pour s'adapter à chaque application particulière. La machine est facilement extensible, grâce à son architecture régulière, pour accueillir plus de processeurs afin de pouvoir traiter des problèmes de taille plus grande. A part ces propriétés générales de l'architecture hypercube, la machine hypercube FPS possède, dans chaque noeud une unité vectorielle (VPU) très puissante pour les opérations en virgule flottante. L'utilisation du VPU constitue un facteur très important pour aboutir à une bonne performance de cette machine.

Simuler un modèle de réseaux de neurones sur une telle machine consiste à élaborer un logiciel qui permette d'exécuter l'algorithme du modèle. Une approche standard comporte trois étapes : D'abord il faut déterminer une structure des données pour la représentation des réseaux, ce qui délimite en grande partie l'architecture des réseaux qui peuvent être éventuellement simulés; Deuxièmement, il faut trouver une méthode pour distribuer le réseau sur les différents noeuds de la machine. Enfin troisièmement, il faut trouver des schémas qui parallèlisent l'algorithme du modèle. A cette étape, ce sont les calculs ayant besoin des informations provenant de plusieurs processeurs qui posent vraiment problème. Sa résolution oblige à des communications entre les processeurs, ce qui peut coûter très cher en temps de calcul. Un bon schéma de parallélisation doit pouvoir réduire au maximum le coût de communication. Il faut souligner que la parallélisation de l'algorithme est très influencée par la façon dont les réseaux sont distribués.

La solution des ces problèmes dépend de la configuration des

processeurs et de l'architecture des réseaux. *Nous supposons qu'il y a toujours beaucoup plus de cellules dans le réseau à implanter que de processeurs.* Dans notre implantation, l'algorithme de GBP sera réalisé tel quel. Nous allons présenter quelques stratégies d'implantation des réseaux multicouches sur deux configurations de processeurs : l'anneau et l'hypercube.

II.2. Présentation de la machine hypercube FPS

Nous allons présenter ici brièvement une série de machines parallèles dont l'architecture est hypercube. Ces machines, fabriquées par la compagnie FPS, sont appellées T-série machines.

Les T-série machines, comme d'autres supercalculateurs, ont été conçues pour exploiter une nouvelle architecture des machines parallèles. L'architecture des T-série machines a été soigneusement étudiée de telle façon que l'arrangement des composants du système soit optimal.

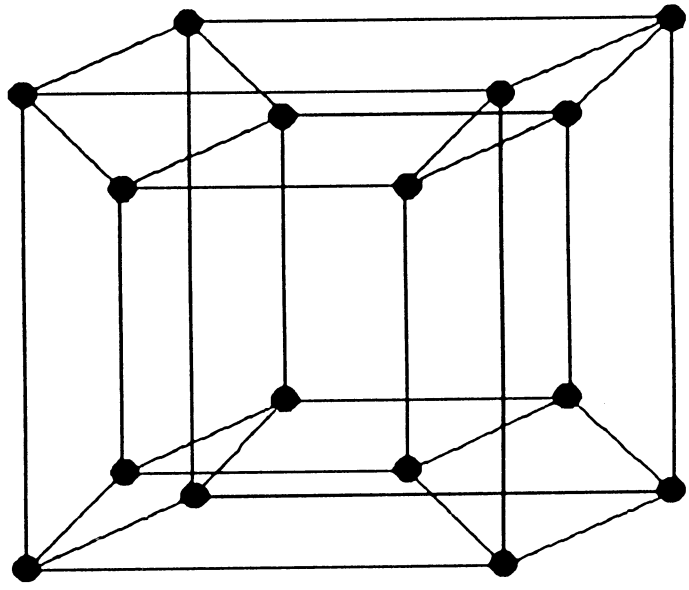


Figure 2.1 : L'architecture d'un hypercube de dimension 4.

La configuration des processeurs est un hypercube dont la dimension varie selon les modèles. Autrement dit, l'architecture d'une T-série machine peut être considérée comme un n-cube $\{0,1\}^n$, où chaque sommet (noeud) correspond à un processeur. Il y a, au total, $P=2^n$ noeuds (ou processeurs) dans une machine de dimension n. Chaque processeur a un numéro p ($0 \leq p \leq P-1$) qui correspond à la valeur décimale de la représentation binaire de chaque noeud du n-cube. Les liens entre les différents processeurs sont caractérisés par les arêtes du n-cube : c'est-à-dire que chaque noeud ne peut directement communiquer qu'avec ses n voisins, alors que la communication entre les noeuds non voisins se fait au travers d'autres noeuds intermédiaires. La Figure 2.1 illustre un 4-cube dont le nombre

de processeurs est égal à 16. Ceci correspond à une machine T20. Dans les T-série machines, le nombre de processeurs peut varier de $8=2^3$ jusqu'à $16384=2^{14}$.

Les T-série machines sont de type MIMD (réf) à mémoire distribuée, car chaque processeur dans une machine exécute ses propres instructions indépendamment des autres processeurs et possède une mémoire qui n'est accessible que par lui même. Il n'y a pas de mémoire partagée. L'avantage de cet arrangement est sa structure modulaire : il suffit de "connecter" deux machines de même dimension pour construire une machine de dimension plus élevée; les codes peuvent être facilement transportés sur des machines de la même série, de n'importe quelle dimension.

Comme il n'y a pas de mémoire partagée, dans la plupart des cas les processeurs doivent communiquer entre eux. L'architecture hypercube permet à chaque processeur d'envoyer un message à un autre processeur à travers un maximum n ($n=\log_2 P$: P est le nombre de processeurs dans une T-série machine) liens. Sur ce point, l'architecture hypercube réalise un de bon compromis entre la faisabilité et l'extensibilité du système. Il faut ajouter que, dans la T-série, il n'y a pas de contrôle global et que les processeurs travaillent de manière asynchrone.

Le système de base dans les T-série machines est un module qui comprend un cube de 8 processeurs reliés à un noeud système (qui n'est pas un noeud de l'hypercube) par un bus. Le noeud système gère, d'une part l'échange des données entre les processeurs du même module et le seul disque associé au module, d'autre part la communication entre chaque processeur et la machine frontale (qui est un Micro-VAX dans notre cas). Les noeuds systèmes sont reliés entre eux suivant d'une structure d'anneau. Un T20 contient deux modules, un T40 contient 4 modules. Ces noeuds systèmes sont transparents pour les utilisateurs.

II.2.1 Les principaux composants

Chaque noeud contient les éléments spécialisés suivants :

-Un Transputer qui est un microprocesseur T414 à 32 bits fabriqué par la société INMOS. Ce microprocesseur spécialisé pour des communications parallèles sert, dans chaque noeud, d'unité de contrôle chargée d'assurer le bon déroulement du programme sur l'unité arithmétique, la gestion de la mémoire et

les communications avec les noeuds voisins ou avec son noeud système;

-Une unité vectorielle arithmétique (VPU pour Vector Processing Unit) pour effectuer des calculs en virgule flottante. Elle est composée d'un additionneur et d'un multiplieur pipelinés, ainsi que de registres (Figure 2.2), et réalise des calculs très rapides sur des réels codés en 32 bits et en 64 bits (double précision). C'est l'ensemble des VPU qui fait que la puissance théorique de T-série machines est de l'ordre de 128 MFlops à plusieurs centaines de GFlops;

-Une mémoire rapide de type vidéo à deux accès, aléatoire du côté de l'unité de contrôle (Transputer) et série-parallèle du côté du VPU (par les registres). La taille de la mémoire est d'un MB. Elle est organisée en 1024 tranches (slices) d'un KB;

-Un multiplexeur pour augmenter la capacité de communication du Transputer qui n'a que quatre canaux d'entrée-sortie.

Toutes ces unités qui sont réalisées en technologie VLSI, sont regroupées et rangées sur une seule carte.

Un noeud système est composé d'un disque dur de 85 MOctets et d'un transputer de contrôle. Il sert d'interface entre le micro VAX et les processeurs du module et est chargé de transférer les codes du programme sur les processeurs, de stocker et d'échanger des données. Le noeud système est aussi réalisé sur une carte et est placé à côté du disque.

II.2.2 Les principes de fonctionnement

Une T-série machine est accessible via un micro VAX sous le système d'exploitation Ultrix ou VMS. Les languages de programation sont Occam, C et Fortran. Un programme est tout d'abord compilé par le micro VAX pour créer des codes exécutables. Les mêmes codes sont chargés, par la suite, sur tous les noeuds de l'hypercube pour être exécutés. Ceci suppose que les instructions de chaque processeur soient explicitement spécifiées dans le programme. Chaque noeud identifie, via certains paramètres du système (numéro absolu ou relatif du noeud, sa position dans une configuration, etc.), les instructions qui lui sont propres. Tant qu'il n'y a pas de communication entre un processeur et ses voisins, le processeur exécute ses instructions de calcul indépendamment des autres processeurs.

Une communication entre deux processeurs voisins se fait par l'échange

de message au travers un canal. Les deux processeurs se synchronisent sur la communication où l'un envoie et l'autre reçoit. La communication ne peut être réalisée que si les deux processus sont prêts. Chaque processeur peut communiquer avec un maximum de 14 voisins (s'il y a autant) à l'aide de quatre ports de communication sur le Transputer et de 14 liens multiplexés. A chaque instant, il y a au plus 4 canaux qui sont actifs du fait que le Transputer ne supporte que quatre communications parallèles. La perfomance de communication dépend beaucoup de l'implantation physique du système et du language utilisé. C'est Occam qui permet de réaliser les communications les plus rapides car il est spécialement conçu pour programmer le Transputer. Les tests réalisés par G. Villard (1988) montrent qu'en Occam la vitesse de communication sur un seul canal peut atteindre 0.937 Mbytes/s lorsque la taille du message dépasse 1 Kbytes, la vitesse normale étant de l'ordre de 0.66 Mbytes/s. Avec 4 canaux actifs, les communications en parallèle seraient un peu ralenties sur chaque canal.

La formule proposée par Y. Saad (Saad et Schultz, 1985) pour calculer le temps de transfert d'un message sur un canal est :

$$\beta+m\tau$$

où β est le temps pour initialiser une communication, τ est le temps d'unité pour envoyer un mot (de 32 bits) et m est la longueur du message (le nombre de mots). Sur le T20 (version release C00 Février 1988), τ=1.44 et β=700 microsecondes.

Bien que le Transputer (T414) soit capable de simuler les calculs flottants, il est vivement conseillé d'utiliser le VPU pour effectuer des opérations flottantes sur des vecteurs ou des matrices. Le VPU est spécialement conçu pour ce genre d'opérations. Il a une puissance théorique de 12 MFlops et peut en pratique offrir une puissance réelle de 10 MFlops si on respecte certaines conditions (à préciser).

L'additioneur et le multiplieur du VPU sont des unités pipelinées de Weitek, de six et sept étages respectivement. Ils ont chacun deux entrées couplées sur les entrées des registres A et B. L'enchaînement des opérations est possible. Le VPU possède deux entrées : l'une provenant de l'un des trois registres B et l'autre du registre A. L'unique sortie (unique) est dirigée vers les quatre registres (Figure 2.2). Chaque registre a un accès bidirectionel à une partie de mémoire centrale qui lui est réservée.

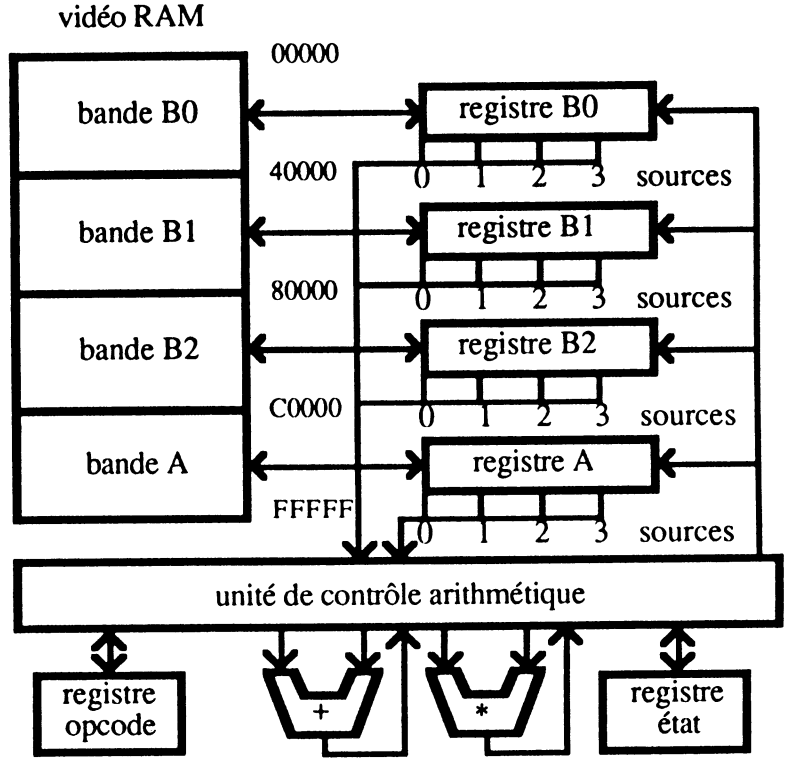


Figure 2.2 : L'organisation des composants du VPU et de la mémoire.

Les instructions du VPU sont exécutées en plus haute priorité afin d'assurer la gestion d'interruptions (l'opération pipelinée sur le VPU ne doit pas être interrompue), le chargement et le déchargement des registres. Plusieurs niveaux de programmation sont possibles. Les bas niveaux, tel Vector Forms ou Parameters Block Routines, permettent aux utilisateurs de manipuler les composants du VPU. A ce niveau, le language de programmation est très proche de la machine et nécessite une parfaite maîtrise du fonctionnement du VPU et une connaissance profonde et très détaillée des calculs à effectuer. L'avantage est évidemment la possibilité d'obtenir une meilleure performance du VPU. Les niveaux plus élevés, comme "Generic Routines" et "Single Node Routines", permettent aux utilisateurs d'avoir une utilisation plus aisée du VPU.

La bonne utilisation du VPU est conditionnée par l'emplacement des données en mémoire. Deux vecteurs d'opérande dans une opération doivent être placés l'un sur le banc A et l'autre sur un des bancs B. Le premier élément (du chaque vecteur) doit être à la première position d'une tranche de mémoire qui contient 128 réels. Pour les vecteurs de taille supérieure à 128 le découpage est

fait pour qu'ils soient alignés et occupent le moins possible de tranches. Les "Generic Routines" et "Single Nodes Routines" assurent la gestion de la mémoire si les données ne sont pas dans les "bons" endroits, mais les "Parameters Block Routines" exigent, au contraire, que les programmeurs arrangent leurs données pour assurer le bon déroulement du VPU. Bien que la gestion de la mémoire ne soit pas obligatoire si l'on n'utilise que les "Generic Routines" et "Single Nodes", la différence de performance que cela peut engendrer n'est pas du tout négligeable.

Enfin la communication entre chaque processeur et la machine frontale (le micro VAX) se fait au travers un bus. A un instant donné, il y a un et un processeur seulement qui peut utiliser le bus pour communiquer avec le frontal.

II.2.3 Les critères pour optimiser l'utilisation de l'Hypercube

La résolution d'un problème d'application sur l'hypercube consiste à étudier la nature du problème, à développer les algorithmes parallèles sur une ou plusieurs configurations de processeurs, et à les mettre en oeuvre. Voici quelques grands principes à respecter afin d'obtenir une bonne performance de la machine :

-L'algorithme développé doit pouvoir exploiter au mieux les parallélismes intrinsèques du problème d'application .

-La topologie de la configuration des processeurs doit s'adapter à l'application et les données doivent être distribuées de façon à équilibrer les charges dans chaque processeur.

-Il faut utiliser au mieux la capacité de chaque processeur et minimiser le temps de communication entre eux.

Malheureusement, ces principes sont un peu flous, voire même contradictoires. Prouver, d'une manière rigoureuse, qu'une configuration de processeurs s'adapte mieux à une application que d'autres est très problématique. S'il faut distribuer les données pour équilibrer les calculs, des communications d'informations ne peuvent pas être évitées, sauf dans les rares cas où le problème d'application peut être "disséqué" en plusieurs morçeaux indépendants. Il n'est donc pas trivial d'atteindre une augmentation de vitesse, via le parallèlisme, sur une telle architecture. C'est pourquoi ce sujet a fait couler beaucoup d'encre ces dernières années.

Néanmoins dans des contextes précis, on pourrait concevoir des algorithmes *pratiques* qui respectent, dans certaines mesures, ces principes. En général, sur des machines à haute latence de communication (comme les hypercubes), l'algorithme doit être structuré de manière à ce qu'une grande quantité de calcul soit effectuée entre les communications, et que les calculs et les communications se chevauchent (overlapping).

En ce qui concerne les détails de la programmation des T-série machines, on peut consulter le manuel (FPS T20 Manuel Utilisateur). Désormais, nous nous intéresserons aux stratégies générales d'implantation des réseaux de neurones multicouches sur les machines hypercubes.

II.3 Les problèmes généraux de l'implantation

Nous allons discuter, dans ce paragraphe, des sujets concernant les problèmes de représentation, de distribution des réseaux, et de parallélisation de l'algorithme d'apprentissage. Ces problèmes peuvent être étudiés dans un cadre mathématiquement rigoureux, ce qui est une démarche très difficile. Il est également possible de traiter ces problèmes de manière heuristique. Dans notre étude, nous avons choisi cette attitude pragmatique, bien que nous cherchions aussi l'optimalité de certaines solutions.

II.3.1 Problème de représentation

La représentation du réseau constitue un problème général quelle que soit la machine utilisée pour l'implantation. La représentation d'un réseau est constituée d'un ensemble de paramètres et de spécifications des relations (dépendances) entre ces paramètres. Elle varie selon les modèles, et influence directement la stratégie de parallélisation d'algorithme.

Les unités élémentaires dans un réseau sont les cellules et les poids de connexion. Pour des modèles simples, comme celui de Hopfield ou de Kohonen, il suffit d'employer une structure matricielle pour représenter ces paramètres et d'autres paramètres intermédiaires (les entrées totales par ex.). L'implantation de ces types de modèles se réalise facilement puisque leur architecture est régulière, et que l'application des techniques d'opérations matricielles suffit.

Pour les modèles complexes comme les réseaux multicouches avec des connexions non régulières (connexions partielles ou/et localement quasi-aléatoires), le problème de la représentation est beaucoup plus délicat, car les connexions sont creuses et irrégulières. De plus, les fonctions de décision des cellules différentes peuvent varier, variation allant de quelques paramètres (caractérisant une même type de fonction) jusqu'à des fonctions de nature complètement différentes. Pour manipuler ces matrices creuses et grandes, il faut déjà utiliser des techniques sophistiquées. Au delà pour traiter différentes fonctions de décision, le problème semble être inabordable uniquement avec des tableaux.

Il faut donc trouver une structure de données plus souple pour la

représentation. On a choisi des variables et des listes de variables de type "enregistrement" (record en PASCAL) pour représenter les cellules et les liens. Ceci devrait permettre à chaque cellule d'accéder facilement à ces variables pour évaluer les fonctions spécifiques de décision et pour modifier les poids de connexion. La Figure 2.3 donne un exemple d'une telle représentation pour l'implantation sur une machine monoprocesseur.

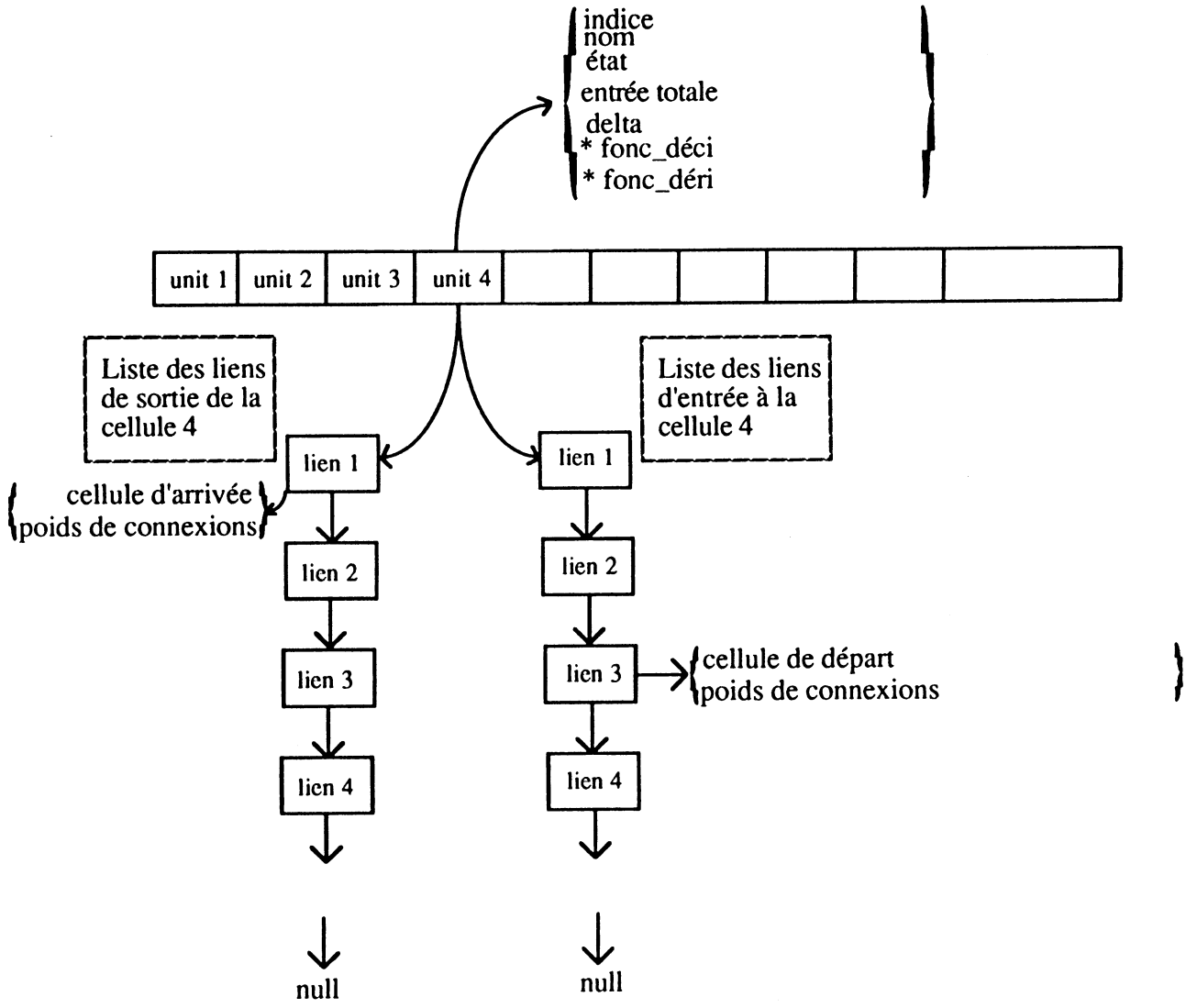


Figure 2.3 : Représentation d'une cellule et de ses liens.

Une cellule est représentée ici par une structure *unit*, qui comprend un ensemble de variables représentant l'indice, le label (nom), l'état, l'entrée totale, delta (dérivée de la fonction d'erreur par rapport à l'entrée totale), le seuil, un pointeur qui pointe sur la fonction de décision pour calculer l'état de cellule, un

pointeur qui pointe sur une autre fonction pour calculer la dérivée, un pointeur qui pointe sur la liste de liens d'entrée et un pointeur qui pointe sur la liste de liens de sortie. Chaque lien est lui-même une structure de variables représentant l'indice de la cellule d'où le lien est issu, l'indice de la cellule à laquelle le lien arrive, le label (nom pour les regroupements) et le poids du lien. Les pointeurs sur les fonctions permettent aux utilisateurs de modifier ces fonctions selon leur besoin. Ainsi, le programme d'implantation peut être adapté à plusieurs modèles sans trop de difficultés.

La technique de représentation ci-dessus pour une machine monoprocesseur s'étend facilement pour des machines parallèles à mémoire distribuée, comme la machine hypercube. La différence principale entre une représentation pour une machine parallèle et pour une machine séquentielle concerne les indices nécessaires pour repérer les cellules. Il faut donc ajouter aux spécifications de chaque cellule le numéro du processeur où la cellule se trouve.

Il faut souligner que la maniabilité s'obtient au prix d'un surcoût de contrôle car la gestion des listes de variables et des multiples accès indirects à la mémoire pénalisent de manière considérable la vitesse de calcul. L'intérêt d'une telle représentation réside dans son efficacité vis-à-vis de la plupart des réseaux utilisés dans des applications réelles où les connexions sont irregulières et difficiles à manipuler avec les méthodes habituelles.

II.3.2 Problème de distribution

La distribution d'un réseau de neurones consiste à répartir l'ensemble des paramètres représentant le réseau dans les différents processeurs de manière à ce que les calculs dans le réseau puissent être effectués en parallèle. La description d'un réseau multicouches comme un graphe orienté, prèsentée au paragraphe I.1, permet d'exprimer le problème de distribution comme "le mapping" du graphe sur une topologie de configuration des processeurs. Le graphe doit être "divisé" en plusieurs morceaux dont chacun est attribué à un processeur. La répartition doit permettre à chaque processeur d'avoir presque la même quantité de calculs, et de plus, la quantité des données à tranférer doit être limitée au minimum. Or l'optimalité d'une répartition est très difficile à atteindre; elle n'est même pas facile à formaliser. En fait, le problème du "mapping" nécessite une étude théorique très approfondie.

Nous nous contenterons, pour le moment, d'utiliser une méthode heuristique qui consiste à répartir tout simplement les cellules de manière équilibrée. Cette répartition ne tient pas compte de la façon dont les cellules sont connectées entre elles.

La méthode de la distribution en moyenne :

Cette méthode est basée sur le principe suivant : Si P représente le nombre de processeurs utilisé pour l'implantation, L, le nombre de couches, et C_k, le nombre de cellules dans la couche k, alors

$$m_k=(\frac{C_k}{P})+1$$

est défini comme le nombre moyen de cellules de la couche k qui devraient être distribuées à chaque processeur.

La distribution se fait en décomposant l'ensemble des cellules de la couche k en P sous-ensembles, dans l'ordre de l'indice des cellules (classique!). Chacun de ces sous-ensembles contient m_k ou m_k-1 cellules. Elles seront distribuées à un des P processeurs. La distribution des poids de connexion est déterminée par la distribution des cellules. On a, en général, deux choix possibles : soit distribuer les connexions d'entrée de chaque cellule au processeur qui contient la cellule; soit distribuer les connexions issues de chaque cellule au même processeur. Les deux choix ne présentent pas vraiment une différence importante pour la parallélisation de l'algorithme, puisque les informations se propagent dans les deux sens de chaque lien. Nous avons choisi la seconde solution pour l'implantation.

Le Tableau 2.1, avec la Figure 2.4, illustrent une distribution, sur une configuration de 16 processeurs, d'un réseau ayant 54 cellules d'entrée, 10 cellules intermédiaires et 21 cellules de sortie, avec des connexions complètes entre les couches consécutives. On voit, dans ce tableau, que les quatre premières cellules (1,2,3,4) sont distribuées au processeur 0, les cellules suivantes (5,6,7,8) sont distribuées au processeur 1, ainsi de suite. Il faut aussi remarquer que la cellule 55 (première cellule de la couche 1) et les cellules 65 et 66 (premières cellules de la couche 2) sont également distribuées au processeur 0, etc.

couche / position	0	1	2
0	1, 2, 3, 4	55	65,66
1	5, 6, 7, 8	56	67,68
2	9,10,11,12	57	69,70
3	13,14,15,16	58	71,72
4	17,18,19,20	59	73,74
5	21,22,23,24	60	75
6	25,26,27	61	76
7	28,29,30	63	77
8	31,32,33	64	78
9	34,35,36		79
10	37,38,39		80
11	40,41,42		81
12	43,44,45		82
13	46,47,48		83
14	49,50,51		84
15	52,53,54		85

Tableau 2.1 : Distribution des cellules d'un réseau (**Figure 2.4**) à un ensemble de 16 processeurs.

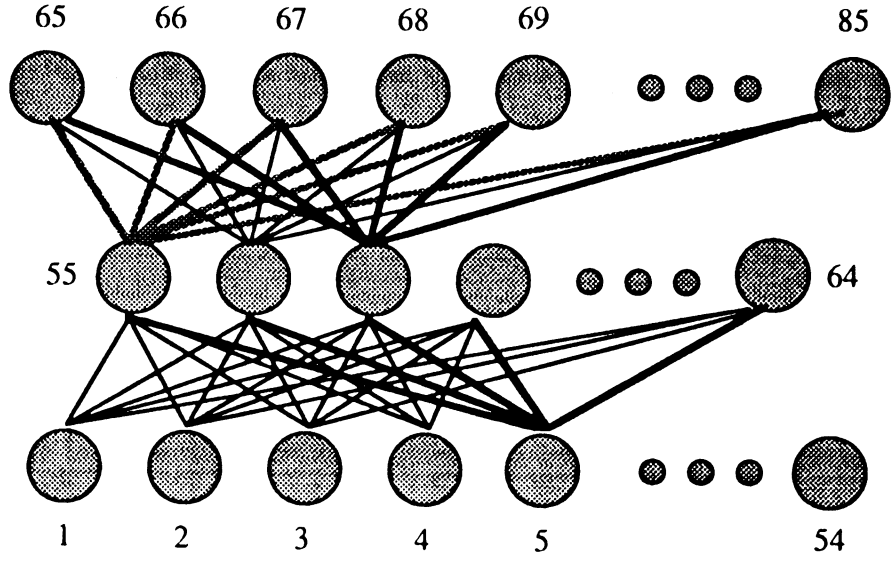


Figure 2.4 : La distribution des poids de connexion selon celle des cellules : les connexions issues d'une cellule sont distribuées au même processeur où réside la cellule

L'avantage de cette méthode est sa simplicité. Lorsque les connexions ne sont pas creuses, ou même si elles sont creuses mais irrégulières, cette méthode est efficace. Mais le défaut de cette méthode est qu'elle ne tient pas du tout compte de la forme de la répartition des connexions entre les cellules, ce qui peut engendrer des distributions qui ne s'accordent pas avec la configuration des processeurs. Certes, on peut trouver des compromis pour des cas particuliers où l'on connaît parfaitement le réseau. Mais dans le cas général, le problème de distribution optimale reste ouvert.

Une autre démarche a été entreprise par Beynon et Nodd (1987). Ils utilisent des techniques de la théorie des graphes pour tranformer le problème de distribution en un problème de minimisation d'une fonctionnelle. Ensuite, ils appliquent la technique du "recuit simulé" en faisant varier le "mapping" afin de trouver un minimum. Le "mapping" qui permet à la fonctionelle d'atteindre son minimum sera donc pris comme une solution optimale. La méthode a été appliquée à un cas simple où il y a 9 cellules et 9 Transputers dont la configuration est un tore de dimension 2. Dans ce cas, la méthode peut donner la solution optimale au bout d'une centaine d'itérations de "recuit". Cette approche intéressante devrait être améliorée, car elle ne peut pas encore traiter des problèmes de taille réelle.

Une approche heuristique est présentée dans mon rapport (Wang 1989). Elle permet de distribuer, de manière systématique, les cellules selon des critères heuristiques qui permettent de réduire la possibilité de distribution des ensembles de cellules "fortement connectées" dans des processeurs "éloignés". Ces critères comprennent les distances entre les processeurs, la connectivité entre les cellules à distribuer et les cellules déjà distribuées, etc. La méthode est très pratique, bien qu'elle ne donne pas en général des solutions optimales. Elle peut aussi être adaptée à différents types d'architecture sans faire beaucoup de modifications.

Pour terminer cette partie, il faut citer les travaux de Joydeep, Ghosh et Kai Hwang (1988). Dans leur étude de l'architecture nécessaire à un système multiprocesseur à mémoire distribuée pour simuler les réseaux de neurones, ils avancent une hypothèse sur l'architecture des réseaux, qui consiste à supposer que les cellules peuvent être groupées en plusieurs "cores" à l'intérieur desquels la densité de connexions entre les cellules est très haute; les "cores" peuvent encore être regroupés en différentes "régions" où la densité des connections entre

les cellules d'une région (mais pas d'un même core) est encore nettement plus haute que celle entre les cellules de régions différentes. Ces caratéristiques structurales permettent de distribuer un réseau sur une configuration de processeurs de manière plus efficace. L'intérêt de cette approche est qu'elle permet de dégager des estimations sur les performances de différentes configurations des processeurs.

II.3.3 Problèmes de communication dans la parallélisation de l'algorithme GBP

La méthode de distribution des réseaux détermine, dans une certaine mesure, la façon dont on peut paralléliser l'algorithme d'apprentissage. Sous l'hypothèse de la distribution en moyenne, nous présenterons, dans ce paragraphe, un schéma général de parallélisation et deux problèmes de communication associés. Les solutions, qui dépendent de la configuration des processeurs, seront données dans les paragraphes qui suivent.

Calculs Locaux et Calculs Globaux :

Pour faciliter la lecture, on rappellerons ici très brièvement l'algorithme GBP qui a été présenté au Chapitre I (paragraphe I.1.3) sous une forme un peu différente. Dans ce chapitre, w_{ij} est toujours utilisé pour représenter le poids de connexion de la cellule j à la cellule i. Pour une paire de vecteurs (X, Y) quelconque, X et Y dans $[-1, 1]^{dn}$ et $[-1, 1]^{dm}$ où dn et dm sont égaux au nombre de cellules dans la couche 0 et la couche L, le procédé d'apprentissage s'écrit comme suit:

1° Calculer la sortie du réseau via la propagation de l'état des cellules d'entrée : pour l'indice de la couche k variant de 0 à L

si la cellule i est dans la couche 0 : $x_i=X_i$

sinon $A_i=\sum_j w_{ij} x_j$ (2.1)

$x_i=f(A_i)$ (2.1)'

2° Calculer le gradient et améliorer les poids :

I) Calculer le gradient y_i via la rétro-propagation :

si la cellule i est dans la couche L : $y_i=2(S_i-Y_i)*f'(A_i)$ (2.2)

De la couche L-1 à 1 : $B_j=\sum_j w_{ij}*y_i$ (2.3)

$y_j=f'(A_j)*B_j$ (2.3)'

II) Améliorer les poids : $\Delta w_{ij}(t)=\beta\Delta w_{ij}(t-1)-\alpha\ \gamma_t\ y_i*x_j$ (2.4)

avec $\alpha,\beta>=0$, $\alpha+\beta=1$ et γ_t parametrès d'apprentissage qui sont des constantes ou

des variables bien déterminées.

La parallélisation de l'algorithme GBP s'effectue couche par couche dans le sens (en avant ou en arrière) correspondant à celui de la propagation des informations. Il est à remarquer que la propagation des informations dans un réseau multicouches est synchrone. En général, lors de la propagation des informations, toutes les cellules d'une couche doivent être mises à jour avant que les calculs dans la couche suivante puissent être commencés. En effet, les cellules de cette couche pourraient avoir besoin des informations issues de presque toutes les cellules des couches précédentes.

Pour mieux cibler les problèmes, il est nécessaire de distinguer deux types de calcul : les calculs locaux et les calculs globaux en rapport avec chaque cellule. Ici, les calculs locaux sont ceux qui nécessitent seulement des paramètres locaux propres à la cellule, les calculs globaux faisant intervenir des informations provenant d'autres cellules. Par exemple, le calcul $f(A_i)$ dans (2.1)' est local à la cellule i; les calculs de (2.1) et (2.3) sont globaux parce qu'ils ont besoin des informations de plusieurs cellules des couches précédentes. Les calculs effectués dans (2.4) seront aussi considérés comme globaux, car ils concernent des informations relatives à des paires de cellules (i et j).

Une distinction encore plus pratique pour l'implantation doit être posée entre calculs locaux par rapport à un processeur et calculs globaux par rapport à l'ensemble des processeurs (système tout entier). Les calculs locaux par rapport à un processeur sont des calculs qui n'utilisent que des paramètres stockés dans le processeur. Les calculs globaux par rapport au système sont, bien sûr, ceux qui nécessitent l'échange d'information entre les processeurs. Ici, on constate que, les calculs totaux d'une cellule se décomposent en deux parties, dont l'une concerne les calculs locaux par rapport au processeur à qui la cellule appartient et l'autre concerne les calculs globaux par rapport au système.

Ces notions nous permettent de focaliser l'attention sur les principaux problèmes de parallélisation. Ce sont les problèmes de communication entre les processeurs pour effectuer les calculs globaux par rapport au système. L'un des objectifs dans nos implantations a été de minimiser la quantité d'informations à transférer, et d'exploiter au mieux la capacité de communication des configurations de processeurs : anneau puis hypercube.

Schéma général de la parallélisation de GBP :

Si on définit U(p)={j | cellule j est connectée à la cellule i et distribuée au processeur p}. (2.1) peut être réécrite sous la forme :

$$A_i=\sum_j w_{ij} x_j = \sum_{p=0}^{P-1} \sum_{j\in U(p)} w_{ij}*x_j = \sum_{p=0}^{P-1} A_{i,p} \qquad (2.5)$$

où $A_{i,p}= \sum_{j\in U(p)} w_{ij}*x_j$ est une somme partielle pour A_i calculée à partir des états des cellules et des poids de connexion dans le processeur p. Les calculs sont locaux par rapport au processeur p. Pour simplifier l'expression, $A_{i,p}$ sera appelée la contribution du processeur p à la cellule i ou simplement à A_i. L'ensemble des contributions (du processeur p) à toutes les cellules dans la couche k est réparti en P sous-ensembles représentés par $T_{dp,p}$, pour p=0, 1, ..., P-1.

$$T_{dp,p}=\{A_{i,p}| \text{ cellule } i\in \text{processeur } dp\}. \qquad (2.6)$$

Autrement dit, $T_{dp,p}$ est un "paquet" qui contient toutes les sommes partielles $A_{i,p}$ calculées dans le processeur p, et qui sont parmi les informations nécessaires pour la mise à jour des états de cellule dans le processeur dp. Nous allons considerer le processeur dp comme le processeur de destination du $T_{dp,p}$.

De même, nous allons utiliser GT_p pour représenter le "paquet" du gradient y correspondant aux cellules de couche k et distribuées au processeur p.

$$GT_p=\{y_i| \text{ la cellule } i\in \text{processeur } p\} \qquad (2.7)$$

Ici le processeur p est considéré comme l'origine de GT_p.

$T_{dp,p}$ et GT_p sont des vecteurs de longueur m_k+1 (m_k est le nombre moyen des cellules de couche k qui sont distribuées à chaque processeur). Le premier élément de $T_{dp,p}$ est égal à dp, numéro du processeur de destination et le premier élément de GT_p est égal à p, numéro de son origine. $T_{dp,p}$ et GT_p ne portent pas explicitement le k, indice de couche, car la parallélisation de l'algorithme est presque identique dans chaque couche. Il s'agit donc tout simplement de "boucler" le même procédé. Voici une version de l'algorithme de GBP parallélisé.

Algorithme Parallèle de GBP :

1° Propagation d'état

Initialisation ;

Pour k=1 à L

(Mise à jour des états des cellules dans la couche k)

1) Calculer les contributions $T_{dp,p}$ à la couche k dans chacun des processeurs ;

2) Calculer la somme des $T_{dp,p}$ pour chaque dp et pour p=0, 1, ... P-1, et mettre à la disposition de chaque processeur dp la somme correspondante ;

3) Calculer les états des cellules distribuées dans chaque processeur

Fin du Pour

2° Rétro-propagation du gradient

Pour k=L à 1

1) Calculer le paquet GT_p composé de tous les y_i dans chaque processeur p par la formule (2.2) si k=L ou par (2.3)' sinon;

2) Diffuser GT_p de chaque processeur à tous les autres, puis calculer les produits par la formule (2.3) si k>1;

3) Modifier les poids w_{ij} par la formule (2.4) pour toutes les cellules j dans la couche k-1 et les cellules i qui leur sont connectées.

Fin du Pour

Problème d'accumulation pour la propagation d'état :

Le premier problème de communication résulte de la mise à jour des états des cellules dans chaque couche. La première partie du schéma ci-dessus contient trois phases. Les phases 1) et 3) ne concernent que des calculs locaux par rapport à chaque processeur, qui n'ont pas besoin de communication entre les processeurs. Dans la phase 2) par contre, chaque processeur apporte une "contribution" à tous les autres et chaque processeur doit avoir la somme[3] de toutes les contributions qui lui sont destinées. Donc un problème d'accumulation se pose:

---

[3]La somme de 2 "tokens" $T_{dp,p}$ et $T_{dp,p'}$ ayant la même destination, dp signifiant la somme des contributions correspondant a $A_{i,p}$ et $A_{i,p'}$ à la même cellule i.

($\mathcal{P}1$) : Pour une configuration donnée de processeurs, trouver un schéma adéquat de communication et de calcul qui permette à chaque processeur dp de calculer la somme de tous les paquets $T_{dp,p}$ (de même taille) fournis par l'ensemble des processeurs dans un temps le plus court possible.

La résolution du problème dépend de la topologie de la configuration des processeurs. Une idée triviale et naïve est d'envoyer, d'abord, tous les $T_{dp,p}$, ayant même destination dp, à ce processeur avant de calculer la somme. Cela coûte cher et surtout n'est pas nécessaire. Les algorithmes proposés dans les paragraphes qui suivent, consistent à envoyer successivement chaque $T_{dp,p}$ du processeur de son origine vers les plus proches et à "fusionner" les $T_{dp,p}$ de chaque processeur par addition, dès qu'il y a plus de deux paquets ayant la même destination. Une combinaison des communications et des calculs intermédiaires permet de réduire considérablement le volume de transfert des données.

Problème de diffusion pour la rétro-propagation du gradient :

La formule (2.3) est un produit scalaire entre les poids de connexion et les composantes y_i du gradient dans une couche. Comme ces composantes sont calculées par plusieurs processeurs, il existe forcément une phase de diffusion dans la deuxième partie du schéma ci-dessus. Les communications sont donc nécessaires pour la phase 2) où s'effectue la véritable "rétro-propagation".

($\mathcal{P}2$) Pour une configuration donnée de processeurs, trouver un schéma adéquat de communication qui permette à chaque processeur d'envoyer un GT_p (de même taille) à chaque processeur en un temps le plus court possible.

La diffusion successive (de la couche L jusqu'à la couche 1) permet à chaque processeur d'être au "courant" de toutes les informations du gradient dans les couches précédentes. La mémorisation de ces informations dépend de l'architecture des réseaux à simuler. Par exemple, pour des réseaux qui n'admettent que des connexions entre couches consécutives, il suffit de garder les informations du gradient qui viennent être diffusées. Dans tous les cas, l'allocation d'un peu de mémoire spéciale pour le gradient permet d'éviter beaucoup de communications.

Le problème ($\mathcal{P}2$) est typiquement un problème de communication

"all-to-all". Il est en relation étroite avec le problème ($\mathcal{P}$ 1). Nous montrerons comment les schémas et les paramètres de contrôle qui ont servi pour la résolution du problème ($\mathcal{P}$ 1) pourraient éventuellement être utilisés pour résoudre ($\mathcal{P}$ 2).

II.4. Une implantation sur l'anneau (un modèle de structure régulière)

La machine hypercube est fabriquée pour des opérations vectorielles. Pour arriver à une bonne performance, il faut que les données soient placées consécutivement dans la mémoire et bien alignées relativement à leur participation aux opérations. Pour être plus précis, les vecteurs représentant les états des cellules, le gradient, et les poids de connexion doivent être distribués et placés selons certaines règles (FPS T20 Manuel Utilisateur). Il est clair alors que l'on est obligé d'imposer certaines contraintes sur l'architecture des réseaux à simuler, surtout en ce qui concerne les connexions entre les cellules.

II.4.1 Quelques contraintes et possibilités

Reconfigurabilité du réseau :

Les réseaux de cette implantation peuvent avoir autant de couches et de cellules que l'on souhaite, mais le mode de connexions entre les cellules est limité. On suppose que les connexions existent seulement entre les cellules de deux couches consécutives et sont complètes. Dans la pratique, on peut, en fait, simuler facilement les réseaux avec des connexions partielles entre les couches consécutives, ce qui est très important dans le traitement de la parole. La parallélisation de l'algorithme GBP est effectué sous l'hypothèse de connectivité complète entre les couches.

La régularité du modèle simplifié permet d'utiliser une représentation matricielle. Les calculs en virgule flottante sont effectués sur les VPU. Les poids de connexion sont stockés dans les bancs B et les vecteurs des états de cellule et du gradient sont stockés dans la banc A pour respecter la règle d'emplacement de mémoire, car les principaux calculs dans un réseau sont le produit scalaire entre les vecteurs de poids et les vecteurs d'états de cellule ou de gradient. Par contre, la règle d'alignement est difficilement respectée, car on n'a pas voulu fixer à priori le nombre de couches, et surtout pas le nombre de cellules dans chaque couche.

Une interface entre les réseaux et les problèmes d'application a été créée. Elle consiste en un ensemble de programmes qui facilitent la spécification du réseau à construire, la statistique de la performance du réseau et de la

machine, la conversion des données entre le format pour l'hypercube et la machine frontale (microVAX sous VMS), la communication entre le terminal et les noeuds processeurs, etc. Du travail reste encore à faire pour rendre l'environnement d'utilisation extrêmement agréable. Nous nous sommes plutôt efforcés de construire un simulateur relativement puissant pour nos problèmes d'application, et de mieux comprendre la nécessité architecturale et algorithmique des machines (à mémoire locale) pour les réseaux de neurones à travers la performance réalisée par la machine hypercube.

Configuration de l'hypercube en anneau :

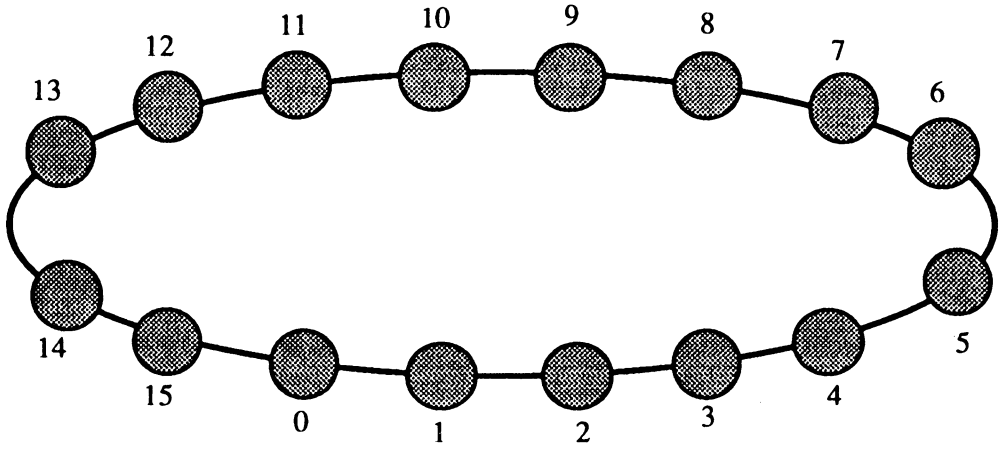


Figure 2.5 : Un anneau de seize processeurs. Les numéros marquent tout simplement la position de chaque processeur sur l'anneau.

L'implantation est réalisée sur un anneau de processeurs. Sur cette configuration, chaque processeur ne peut communiquer qu'avec deux voisins. Les primitives fournies par FPS pour générer un anneau exigent que le nombre de processeurs dans un anneau soit égal à une puissance de 2. La Figure 2.5 illustre un anneau de seize processeurs sur le T20.

Voici quelques notations utiles pour la description de nos algorithmes. Sur un anneau, on fixe un ordre tel qu'il est montré par la Figure 2.5. Chaque processeur p a donc un numéro de position pos(p). On utilise p_p pour représenter le numéro absolu du processeurs placé avant le processeur p, c'est à dire que $p\_p=pos^{-1}(pos(p)-1)$; et de même on utilise $p\_s=pos^{-1}(pos(p)+1)$ pour représenter le numéro absolu du processeur placé après le processeur p. On suppose que les valeurs de p_p et de p_s soient connues par chaque processeur.

Dans les présentations suivantes, nous allons employer deux mots clés du language OCCAM : SEQ et PAR. SEQ suivi par une suite des instructions indentées signifie que les processus correspondant à l'ensemble d'instructions seront exécutés séquentiellement, tandis que PAR suivi par une suite des instructions indentées signifie que les processus seront exécutés en parallèle.

II.4.2 Algorithme d'accumulation sur l'anneau

La propagation de l'état des cellules commence par une initialisation des cellules d'entrée par les données, données déjà bien distribuées en correspondance avec la distribution du réseau. Ensuite, les calculs de l'état des cellules dans une couche k (k>0) peuvent être effectués selon l'algorithme parallèle donné au paragraphe II.3.3. Nous proposons ici un algorithme d'accumulation sur l'anneau qui résoud le problème ($\mathcal{P}$1).

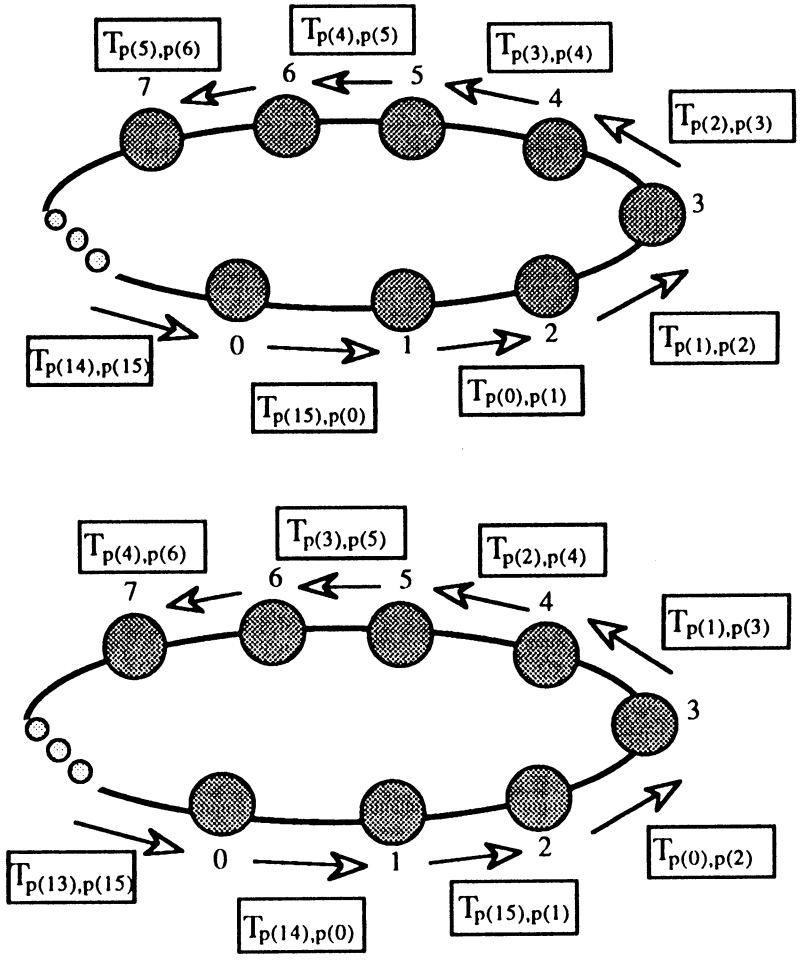
Algorithme II.1 (pour le problème d'accumulation ($\mathcal{P}$1)) :

```
Faire dans chaque processeur p
SEQ
   dp=p_p ;
   PAR
      Envoyer T_dp,p au processeur p_s ;
      Recevoir un paquet T_dp',p' du processeur p_p ;
   T_dp',p ← T_dp',p' + T_dp',p. (4) ;
   SEQ (pour i=2 à P-1)
      dp=dp';
      PAR
         Envoyer T_dp,p au processeur p_s ;
         Recevoir un paquet T_dp',p' du processeur p_p ;
      T_dp',p ← T_dp',p' + T_dp',p ;
   Fin pour
Fin
```

L'algorithme se termine en P-1 étapes et chaque processeur aura exactement la somme de tous les paquets qui lui sont destinés. La Figure 2.6 illustre deux étapes de cet algorithme. Les communications se réalisent en

(4) voir la note de la page 84.

cascade et les paquets sont successivement "ramassés". Il faut remarquer que les communications et les calculs sont bien équilibrés c'est-à-dire qu'à chaque étape, la quantité de données à transférer sur chaque canal est la même, ainsi que celle des vecteurs de l'addition intermédiaire dans chaque processeur. Le temps de communication dans l'algorithme est de $(P-1)(\beta+m_k*\tau)$ où $m_k=(\frac{C_k}{P})+1$ et C_k est le nombre des cellules de la couche k.



p(pos) donne le numéro du processeur à la position pos.

$T_{p(15),p(0)}$: le paquet dans le processeur p(0) destiné au processeur p(15)

Figure 2.6 : Deux étapes de L'**Algorithme 1** pour la résolution du problème d'accumulation ($\mathcal{P}$ 1).

Nota : L'Algorithme II.1 est un algorithme efficace bien qu'il ne soit pas optimal. Son efficacité sera illustrée par les résultats expérimentaux donnés au paragraphe II.4.5. On pourrait remarquer que l'avantage de cet algorithme est

sa simplicité et l'équilibrage entre la communication et le calcul. A aucun moment un processeur quelconque n'est mis en attente.

II.4.3 Algorithme de diffusion sur anneau

Le problème de diffusion ($\mathcal{P}$ 2) semble moins difficile que celui d'accumulation, surtout après avoir résolu ($\mathcal{P}$ 1). Dans l'**Algorithme II.1**, le paquet $T_{dp,p}$ du processeur p au processeur dp de son voisin "de gauche" fait un parcours "contre-la-montre" pour ramasser tous les paquets destinés au même processeur dp. Si $T_{dp,p}$ est remplacé par GT_p, on obtient un algorithme de diffusion sur l'anneau en enlevant la partie d'addition intermédiaire.

Algoritme II.2 (le problème de diffusion ($\mathcal{P}$ 2) sur l'anneau)

```
Faire dans chaque processeur p
SEQ
   dp=p_p ;
   PAR
      Envoyer GT_dp,p au processeur p_s ;
      Recevoir un paquet GT_dp',p' du processeur p_p ;
   SEQ (pour i=2 à P-1)
      dp=dp';
      PAR
         Envoyer GT_dp,p au processeur p_s ;
         Recevoir un paquet GT_dp',p' du processeur p_p ;
   Fin pour
Fin
```

L'algorithme se termine en P-1 étapes et chaque processeur aura tous les morceaux du gradient calculé à la couche k. La Figure 2.7 illustre deux étapes de cette algorithme. Comme dans l'algorithme d'accumulation, les communications s'effectuent aussi en cascade. Le temps de communication est $(P-1)(\beta+m_k*\tau)$.

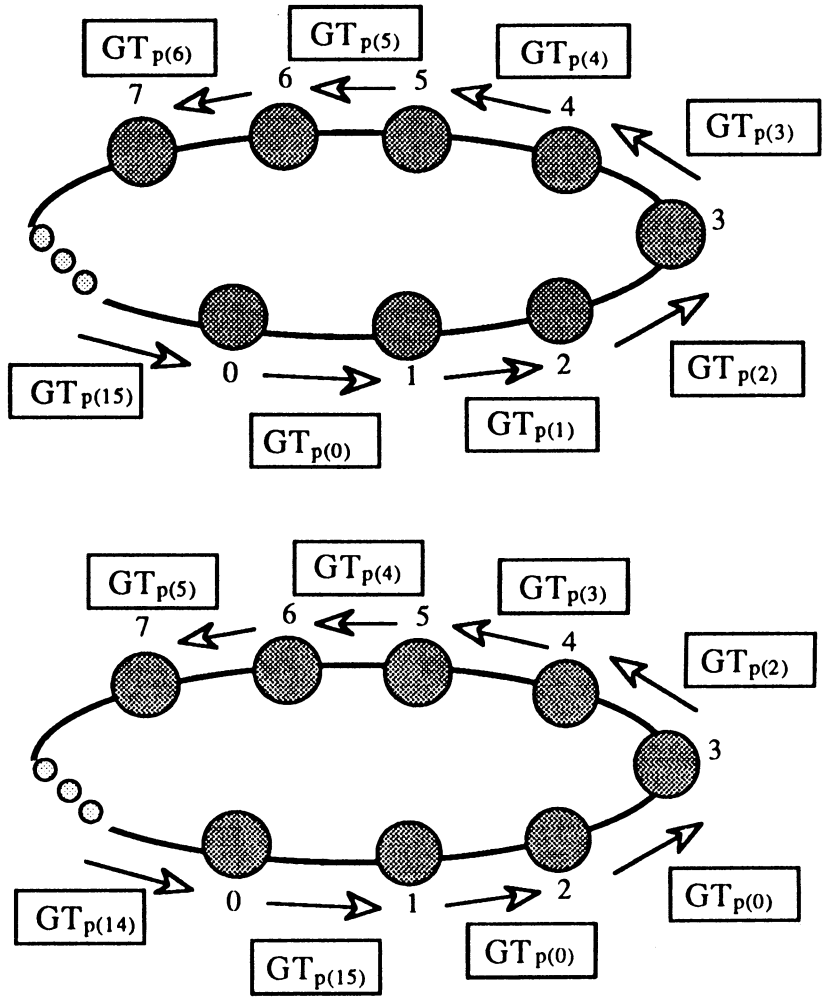


Figure 2.7 : Deux étapes de L'**Algorithme 2** pour la résolution du problème de diffusion ($\mathcal{P}$ 1).

II.4.4 Utilisation de l'unité de calcul vectoriel (VPU)

Dans cette première implantation, on a utilisé les VPU pour effectuer les calculs flottants grâce à la structure régulière du modèle. On peut principalement distinguer sept parties :

1°) Calculer les paquets $T_{dp,p}$ de chaque processeur définis par (2.6) : ceci comprend essentiellement les sommes de produits entre les états de cellule et les vecteurs de poids ;

2°) Calculer les sommes partielles des paquets lors de la phase d'accumulation.

3°) Calculer les fonctions sigmoïdes. Ceci nécessite d'abord une évaluation de la fonction exponentielle.

4°) Calculer les dérivées des fonctions sigmoïdes. On utilise la formule suivante pour éviter la réévaluation de la fonction exponentielle :

$$f'(A_i)=\frac{\alpha}{2}(1-f(A_i)^2)$$

où $f(A_i)$ correspond, dans le réseau, à l'état de la cellule i.

5°) Calculer les paquets GT_p du gradient dans la dernière couche.

6°) Calculer les produits scalaires entre le vecteur du gradient et les vecteurs des poids de connexion. Mettre à jour les gradients y.

7°) Modifier les poids.

On trouvera dans mon rapport (Wang, 1988) plus de détails concernant ces différentes parties. Les VPU ont été programmés au niveau de "Generic et Singlenode Routines".

II.4.5 Expérimentations

Dans ce paragraphe, nous allons donner quelques résultats expérimentaux effectués pour tester l'efficacité de l'implantation. Le terme "un apprentissage" désigne une application de l'algorithme GBP à un réseau pour une paire de patterns d'entrée-sortie (désirée). Le temps d'un apprentissage sera considéré comme un indice de la performance de l'implantation. Ce temps est, en effet, la somme des temps d'initialisation du réseau, de propagation des états de cellule, de rétro-propagation du gradient et de modification des poids de connexion.

Le premier exemple est destiné à illustrer l'influence de la structure des réseaux sur le temps d'apprentissage. Trois types de réseaux ont été choisis. Ils ont tous les mêmes nombres de cellules d'entrée et de sortie, mais le nombre de couches intermédiaires varie de 1 à 3. Pour rendre les choses comparables, les nombres de cellules dans chaque couche intermédiaire ont été générés de telle manière que les réseaux de chaque type puissent avoir approximativement la même séquence du nombre total des connexions. Puisque l'apprentissage consiste à améliorer les poids de connexion, nous devons donc tester des réseaux ayant le même nombre de connexions, pour comparer l'influence de l'architecture du réseau sur le temps.

Exemple 1 :

Dans cet exemple, net1, net2 et net3 représentent trois ensembles de réseaux dont le nombre de cellules d'entrée et de sortie est toujours 25. Les tableaux suivants montrent la variation du nombre de cellules intermédiaires. La Figure 2.8 donne une comparaison de l'évolution du temps de calcul (d'un apprentissage) pour chaque ensemble.

NET1

N° de réseau	nb. cellu couche 1
1	60
2	120
3	180
4	240
5	300
6	360
7	420
8	480
9	540
10	600
11	660
12	720
13	780
14	840
15	900
16	960

NET2

N° de réseau	nb. cellu couche 1	nb. cellu couche 2
1	47	25
2	95	30
3	135	35
4	169	40
5	198	45
6	233	50
7	245	55
8	264	60
9	281	65
10	297	70
11	311	75
12	323	80
13	335	85
14	345	90
15	355	95
16	364	100

NET3

N° de réseau	nb. cellu couche 1	nb. cellu couche 2	nb. cellu couche 3
1	25	35	25
2	30	75	30
3	35	104	35
4	40	125	40
5	45	142	45
6	50	155	50
7	55	166	55
8	60	175	60
9	65	183	65
10	70	190	70
11	75	195	75
12	80	200	80
13	85	205	85
14	90	209	90
15	95	212	95
16	100	215	100

Tableau 2.2 : Trois ensembles de réseaux qui ont été testés pour évaluer la performance de l'implantation pour différentes architectures de réseau. Ces réseaux ont tous le même nombre de cellules d'entrée et le même nombre de cellules de sortie (25). Le tableaux montre le nombre de cellules dans les couches intermédiaires.

On peut remarquer que :

(1) Le taux d'augmentation du temps de calcul est très inférieur à celui du nombre total de poids. Pour le cas extrême, par exemple, le nombre des poids dans les trois ensembles augmente presque de 48000/3000=16 fois tandis que l'augmentation du temps d'un apprentissage est seulement de 4 fois pour Net1, 2.44 pour Net2 et 2.07 pour Net3. Ce sont les réseaux de taille plus grande qui

permettent d'atteindre une meilleure performance de la machine.

(2) La taille de réseau n'est pas le seul facteur qui influence la performance de la machine, car on constate que les trois courbes ne croissent pas au même rythme. L'architecture du réseau joue un rôle très important. Par exemple, le temps pour les réseaux de l'ensemble Net1 continue à augmenter très rapidement et finit par dépasser celui de Net2 et Net3.

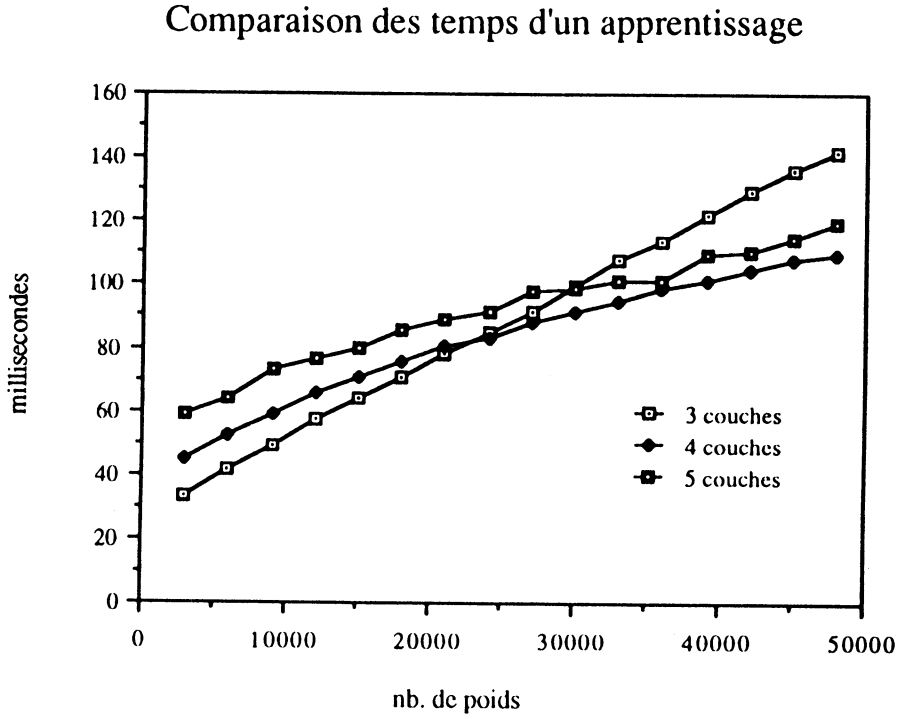


Figure 2.8 : La comparaison du temps d'un apprentissage des réseaux dans chaque ensemble montré par le Tableau 2.2.

Il est difficile de mesurer avec précision cette influence de l'architecture, à cause des nombreuses fractions de calcul et de communication dans un réseau. Une analyse plus approfondie du programme nous a fourni certaines indices plus clairs. Bien que le coût de communication soit relativement faible pour les réseaux de l'ensemble Net1, l'appel des VPU est pourtant plus fréquent pour de petits calculs. L'architecture des réseaux de l'ensemble Net2 et Net3 permet, par contre, une meilleure utilisation des VPU quand le nombre de cellules dans les couches intermédiaires devient plus grand.

Exemple 2 :

Il existe une autre façon simple de montrer l'efficacité d'une implantation : on peut comparer le temps d'un apprentissage, pour des réseaux

qui sont "simulés" sur un anneau des processeurs ou sur un seul processeur. Dans ce but, on a généré un ensemble de réseaux ayant toujours deux couches intermédiaires (Tableau 2.3). Ici le nombre de cellules d'entrée et de cellules de sortie est toujours 25, et le nombre des poids de connexion varie de 3025 à 48125. Ces réseaux ont été testés sur un processeur et un anneau de seize processeurs.

N° de rés	nb. cellu couche 1	nb. cellu couche 2	nb. de poids	N° de rés	nb. cellu couche 1	nb. cellu couche 2	nb. de poids
1	48	25	3025	9	282	65	27005
2	96	30	6030	10	298	70	30060
3	136	35	9035	11	312	75	33075
4	170	40	12050	12	324	80	36020
5	199	45	15055	13	336	85	39085
6	224	50	18050	14	346	90	42040
7	246	55	21055	15	356	95	45095
8	265	60	24025	16	365	100	48125

Tableau 2.3 : L'ensemble des réseaux présentés ici est testé pour évaluer la performance de la parallélisation : c'est-à-dire, comparer le temps d'un apprentissage pour ces réseaux sur un anneau de 16 processeurs et sur un seul processeur. Ces réseaux ont tous le même nombre de cellules d'entrée et le même nombre de cellules de sortie (25).

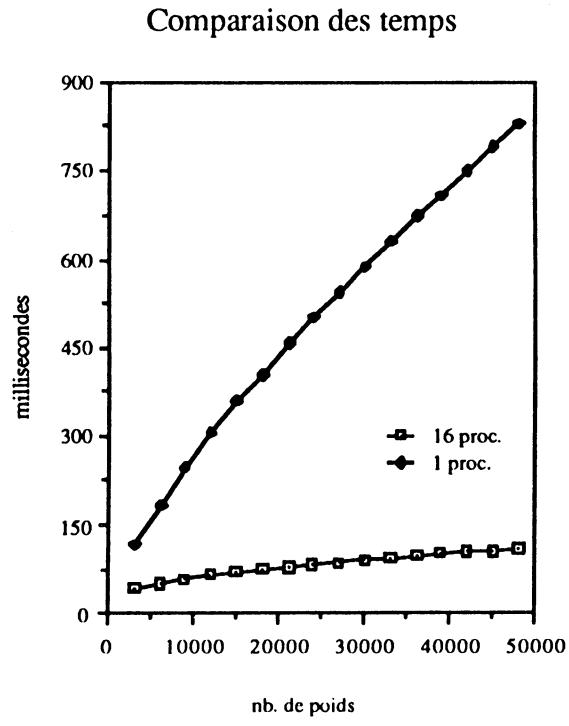


Figure 2.9 : Un comparaison du temps d'un apprentissage pour les réseaux donnés par le Tableau 2.3 sur un anneau de 16 processeurs et sur un processeur.

La Figure 2.9 illustre une comparaison entre la performance de seize processeurs et la performance d'un seul processeur. En interprétant ces résultats, on doit tenir compte de deux facteurs qui donnent des avantages aux réseaux simulés sur un seul processeur. D'abord, il n'y a pas de communication à faire; puis, comme il n'y a pas de distribution, les vecteurs de données qui participent aux calculs sont beaucoup plus longs, ce qui permet d'atteindre une meilleure performance des VPU. Donc, on ne peut pas espérer que le facteur d'accélération avec seize processeurs puisse atteindre seize.

Sur la Figure 2.9, on remarque que le temps d'un apprentissage sur un seul processeur augmente très vite. Le rapport entre les augmentations totales est égal à

$$\frac{T_1(48125)-T_1(3025)}{T_{16}(48125)-T_{16}(3025)} \approx 700/65=10,77$$

tandis que le rapport entre les augmentations en deuxième partie des courbes est égal à

$$\frac{T_1(48125)-T_1(27005)}{T_{16}(48125)-T_{16}(27005)} \approx 13.5.$$

Sur la figure, on peut remarquer une forte croissance de l'écart entre le temps pour un processeur et celui pour seize processeurs. Le meilleur gain obtenu par l'utilisation de 16 processeurs dans cet exemple est égal à

$$\frac{T_1(48125)}{T_{16}(48125)} \approx 8.5.$$

Il faut ajouter qu'un réseau de cette taille permet le (seul) processeur de simulation d'atteindre une bonne performance tandis que le même réseau distribué sur 16 processeurs ne permet pas à chacun des processeurs d'atteindre une performance équivalente. Dans ce dernier cas, chaque processeur ne simule, en gros, que 3000 poids sur un nombre total de 48125. On a atteint une performance de 10 mégaflops avec seize processeurs. Cette performance est jugée bonne si on la compare avec les performances obtenues pour des opérations comme la multiplication matrice-vecteur dans la condition la plus favorable (SAXPY avec une matrice 128x128 et un vecteur 128, les deux étant bien alignés), cette dernière est d'environ quarante megaflops. Dans notre cas, ces conditions ne peuvent pas être satisfaites compte tenu des variations d'architecture, et de l'évaluation de plusieurs exponentielles.

Voici quelques détails de l'implantation. Dans chaque processeur, on

réserve un maximum de deux bancs dans la mémoire (B1 et B2) pour stoker les poids et un minimum d'un banc pour stoker les états des cellules, le gradient, les données pour l'apprentissage et certaines variables intermédiaires. Le nombre de poids dans un processeur pourrait atteindre jusqu'à 2*256*128=65536 à condition que la mémoire dans le banc A soit suffisante pour stocker les données d'apprentissage.Sur un anneau de 16 processeurs, le nombre maximal des poids d'un réseau à simuler pourrait atteindre un million (1048576 exactement). Pour des réseaux de taille grande (par ex. plus de 100000 poids) avec une architecture relativement régulière, la machine atteint facilement une performance de 20 MFlops.

En résumé, on constate que si l'architecture des réseaux à simuler est très régulière (notamment pour des réseaux dont les connexions entre les couches sont complètes), les algorithmes d'accumulation et de diffusion donnés ci-dessus sont assez efficaces pour atteindre une bonne performance d'utilisation des multiprocesseurs dont la configuration n'est qu'un anneau.

Quand le nombre total de processeurs est faible, le coût de communication (pour l'accumulation et pour la diffusion) n'a pas une influence très importante si les tâches de calcul sont bien réparties sur les différents processeurs. Mais quand le nombre de processeurs augmente, ce qui signifie en général une baisse de la quantité des calculs dans chaque processeur, la réduction du coût de communication devient de plus en plus importante. Dans ce cas, les algorithmes de communication joueront un rôle plus important pour diminuer le temps d'un apprentissage. L'anneau semble trop restrictif (simple) pour la conception des algorithmes de communication, surtoût lors qu'il s'agit d'une sous-architecture de l'hypercube. Comment utiliser au mieux la capacité de communication de l'architecture hypercube? Ce sera le sujet du paragraphe suivant.

II.5. Algorithmes pour résoudre les problèmes de communication sur l'hypercube

L' argument important pour travailler sur l'hypercube au lieu de travailler sur un anneau est le suivant : sur un hypercube de dimension n, il y a P=2n processeurs et $\frac{Pn}{2}=n2^{n-1}$ canaux de communication; si l'on travaille sur la sous-architecture anneau, on utilise seulement $P=2^n$ canaux et quand n est supérieur ou égal à 4, la plupart des canaux restent "inactifs"; ce qui résulte en une sous-exploitation du potentiel de l'hypercube. Bien que cet argument tout seul ne justifie pas complètement la nécessité d'utilisation de la configuration hypercube (il existe aussi d'autres facteurs à considérer comme l'architecture des réseaux à simuler, la puissance du coprocesseur arithmétique, etc), il est suffisamment motivant pour mener une recherche d'algorithmes pour l'hypercube, ce qui n'est pas aussi facile que pour l'anneau.

Les algorithmes présentés dans ce paragraphe, décrivent deux schémas de communication sur une configuration d'hypercube des processeurs. Tous les canaux seront utilisés afin d'obtenir une meilleure utilisation du potentiel de communication. Ces deux algorithmes, dont l'un est asymptotiquement optimal, peuvent être réalisés d'une manière systématique. Les résultats expérimentaux présentés dans le paragraphe II.3 en justifient l'efficacité.

Pour l'étude de la complexité des algorithmes, nous avons besoin de deux hypothèses supplémentaires sur le temps de communication parallèle à travers plusieurs canaux. L'hypothèse de base a été donnée au paragraphe II.2.2. Il s'agit de calculer le temps de transfert d'un message sur un canal par la formule $\beta+m\tau$, β étant appellé temps de "start-up" et τ correspondant au temps unitaire de communication.

(Hypo 1) *Communications simultanées : le temps nécessaire à un processeur pour envoyer n morceaux de données de même taille à ses n processeurs voisins est égal approximativement au temps nécessaire pour envoyer un morceau à un seul processeur voisin.*
(Hypo 2) *Deux communications sur un même canal ne peuvent pas être pipelinées. C'est à dire que, si un processeur doit envoyer, à travers un canal, deux morceaux de données qui ne sont pas stockées l'un après l'autre dans la*

mémoire, il sera obligé d'initialiser deux fois la communication et le temps de communication contiendra donc deux β.

L'hypothèse (Hypo 1) peut sembler un peu idéaliste, car aucune technique actuellement connue ne permet de réaliser un système qui atteigne un tel niveau de parallélisme, bien qu'on pourrait en être très proche(iPSC 2 par ex.). Cette hypothèse est nécessaire pour l'analyse de la complexité des algorithmes. Dans le paragraphe II.5.4. de ce chapitre, on donnera un aperçu de la performance de FPS T40 vis-à-vis de cette hypothèse. L'hypothèse (Hypo 2), qui est à rattacher à la propriété de la machine hypercube FPS T40 sur laquelle les tests ont été effectués, n'est en fait pas indispensable pour l'étude de la complexité. Nous indiquerons les différences entre les résultats obtenus sous l'hypothèse (Hypo 2) et les résultats obtenus sous l'hypothèse contraire à (Hypo 2).

Pour deux nombres quelconques p_1 et p_2 $0 \leq p_1, p_2 < P$, on utilise $d_H(p_1,p_2)$ pour représenter la distance de Hamming entre le processeur p_1 et le processeur p_2, qui est égale au nombre des bits différents dans les représentations binaires de p_1 et de p_2. Un paquet $T_{dp,p}$ sera occasionnellement appelé "x-paquet" et un morceau du vecteur de gradient GT_p "y-paquet".

II.5.2 Algorithme d'accumulation sur hypercube

Un x-paquet $T_{dp,p}$ est identifié dans chaque noeud par l'indice dp du noeud de sa destination et l'indice p du noeud de départ. $d_H(dp,p)$ sera interprété comme la distance entre $T_{dp,p}$ dans son noeud de départ et sa destination. Il y a $C_d^n = \binom{n}{d}$ x-paquets dans chaque noeud à distance d de leurs C_d^n destinations différentes. Nous avons un schéma général de communication qui est spécifié sous forme de processus exécutés dans chaque noeud.

Schéma général pour l'accumulation :

1° Envoyer l'unique x-paquet de distance n à un voisin, recevoir un x-paquet provenant d'un autre voisin; Calculer la somme du paquet reçu avec le paquet ayant la même destination.

2° Pour la distance d variant de n-1 à 1, envoyer les C_d^n x-paquets de distance d à leur n voisins de manière que chaque voisin soit à distance d-1 de la destination de chaque paquet reçu; Calculer la somme des paquets reçus avec le

paquet (du processeur) ayant la même destination.

L'idée principale du schéma est que chaque noeud envoie successivement ses paquets aux voisins qui sont les plus proches de leurs destinations. Et lorsqu'un noeud possède plusieurs paquets qui partagent la même destination, ce noeud doit calculer la somme de ses paquets afin de réduire la quantité des données à transférer. A l'étape d, chaque noeud envoie et reçoit des paquets à distance d de leur destination. Chacun de ces paquets étant une somme partielle des paquets de distance plus élevée, sera "fusionné avec le paquet de distance d-1 pour former un nouveau paquet (de distance d-1)".

Problème de répartition :

Dans le schéma ci-dessus, la première étape représente un cas particulier, car chaque noeud possède un seul x-paquet de distance n. Ce paquet peut être envoyé à n'importe quel voisin parce qu'ils sont tous des noeuds de distance n-1 de leur destination. La façon la plus efficace est de permettre à chaque noeud de faire l'envoi et la réception en parallèle. Ceci se réalise facilement sur un anneau.

Pour les étapes de la deuxième phase, les communications seront effectuées sur la configuration hypercube. Dans chaque noeud, les C_d^n paquets de distance d devraient être répartis en n groupes, après quoi les noeuds vont envoyer les paquets de chaque groupe à un voisin. Une contrainte évidente pour cette répartition est que les paquets de chaque groupe partagent un voisin commun, qui est à distance d-1 de leurs destinations. On espère aussi que chaque groupe a $\frac{1}{n}C_d^n$ (au moins) ou $\frac{1}{n}C_d^n+1$ (au plus) paquets de ce genre pour équilibrer les tâches de communication sur l'ensemble des canaux et les calculs intermédiaires dans chaque noeud. Autrement dit, on a un problème de répartition qui se pose comme suit :

Problème de répartition ($\mathcal{P}$ $\mathcal{P}$) :

Pour un noeud p quelconque, $0 \le p \le P-1$, comment répartir l'ensemble

$V^{(d)}=\{T_{dp,p} \mid 0 \le dp \le P-1,\ d_H(dp,p)=d\}$

en n groupes : $V_1^{(d)}, V_2^{(d)}, \ldots, V_n^{(d)}$ tel que

$$(\tfrac{1}{n}C_d^n) \le |V_j^{(d)}| \le (\tfrac{1}{n}C_d^n+1)$$

et $\forall T_{dp,p} \in V_j^{(d)}$, j=1, 2, ..., n,

$$d_H(dp, L^j(p)) = d-1$$

où $|V_j^{(d)}|$ donne le nombre d'éléments dans $V_j^{(d)}$ et $L^j(p)$ est l'indice du voisin qui ne diffère de p qu'au $j^{ième}$ bit.

On peut démontrer que le problème ($\mathcal{P}$ $\mathcal{P}$) admet des solutions qui ne sont pas uniques. La recherche de cette répartition est pourtant très problématique. Nous allons proposer un algorithme qui permet de trouver une des solutions possibles. Il peut être considéré comme une preuve constructive de l'existence de telles répartitions.

Algorithme II.3 (pour résoudre le problème ($\mathcal{P}$ $\mathcal{P}$)) :

1°)$\forall$ j=1, 2, ..., n, trouver tous les $T_{dp,p} \in V^{(d)}$ tels que $d_H(dp, L^j(p))=d-1$. Noter

$$S_j=\{T_{dp,p} \mid T_{dp,p} \in V^{(d)}, d_H(dp, L^j(p))=d-1\}$$

2°) Prenons $F(S_j)$ comme le nombre de paquets $T_{dp,p}$ qui se trouvent uniquement dans S_j, alors on peut effectuer un processus d'élimination selon les stratégies suivantes :

Tant qu'il existe un $T_{dp,p}$, j, j' et $j \neq j'$ tel que $T_{dp,p} \in S_j$ et $T_{dp,p} \in S_{j'}$ faire
si ($F(S_j)>F(S_{j'})$), alors enlever le $T_{dp,p}$ de S_j ;
sinon si ($F(S_{j'})>F(S_j)$), alors enlever le $T_{dp,p}$ de $S_{j'}$.
sinon si ($|S_j|>|S_{j'}|$) alors enlever le $T_{dp,p}$ de S_j;
sinon si ($|S_j|<|S_{j'}|$) alors enlever le $T_{dp,p}$ de $S_{j'}$;
sinon enlever le $T_{dp,p}$ de S_j ou de $S_{j'}$.

L'ensemble $\{S_j | j=1, 2, ..., n\}$ ainsi obtenu sera une solution à ($\mathcal{P}$ $\mathcal{P}$).

Arrangement de la mémoire :

La recherche de la répartition $\{V_1^{(d)}, V_2^{(d)}, ..., V_n^{(d)}\}$ devrait s'accorder avec la distribution du réseau. Cette répartition suggère une façon d'arranger les données dans la mémoire. Pour éviter d'initialiser une communication pour chaque paquet (voir l'hypothèse (Hypo 2) ci-dessus), on pourrait stocker les paquets qui sont dans le même ensemble $V_j^{(d)}$ dans les locations consécutives car on sait que ces paquets devraient être envoyés au même noeud voisin. Cette

répartition doit faire partie des paramètres permanents de contrôle.

Il suffit donc d'une seule initialisation de communication pour envoyer au moins $\frac{1}{n}C_d^n$ paquets dans un même groupe. Ainsi, un temps de $\frac{1}{n}C_d^n$-1 "start-ups" pourrait être épargné.

L'algorithme d'accumulation et sa complexité :

Nous allons présenter l'algorithme d'accumulation et un théorème sur la complexité de cet algorithme. Il sera démontré que l'algorithme proposé ici est asymptotiquement optimal.

Algorithme II.4 :

Supposons que l'on ait un anneau qui est inscrit dans l'hypercube. L'accumulation s'effectue en deux phases d'opérations dans chaque noeud p :

1° Pour d=n, faire en parallèle (sur l'anneau)

envoyer le paquet $T_{dp,p}$, pour dp qui satisfait $d_H(dp,p)=n$, au noeud suivant ;

recevoir un paquet $T_{dp',p'}$ du noeud précédent p' ;

Puis calculer $T_{dp',p} \leftarrow T_{dp',p'} + T_{dp',p}$ [5]

2° Pour d=n-1 à 1 faire

(1) Si le nombre de 1 dans la représentation binaire du numéro p est impair, alors

envoyer en parallèle à chaque noeud $L^j(p)$, pour $1 \le j \le n$, les paquets de $V_j^{(d)}$.

sinon recevoir en parallèle les paquets envoyés du noeud voisin $L^j(p)$;

(2) Si le nombre de 1 dans la représentation binaire du numéro p est pair, alors

envoyer en parallèle à chaque noeud $L^j(p)$, pour $1 \le j \le n$, les paquets de $V_j^{(d)}$.

sinon recevoir en parallèle les paquets envoyés du noeud voisin $L^j(p)$;

(3) Pour tous les paquets reçus $T_{dp',p'}$ avec $p'=L^j(p)$ $1 \le j \le n$, calculer

$T_{dp',p} \leftarrow T_{dp',p'} + T_{dp',p}$

où dp' est tel que $d_H(dp',p)=d-1$.#

---

[5] voir la note de la page 84.

Pour donner une idée concrète de cet algorithme, nous allons montrer une application de l'algorithme sur un hypercube de dimension 4, qui est numéroté comme dans la Figure 2.10. Le Tableau 2.4 donne, pour chaque processeur, l'ensemble des paquets à communiquer, à chaque étape d de 4 à 1 et sur chaque canal d'entrée-sortie du processeur.

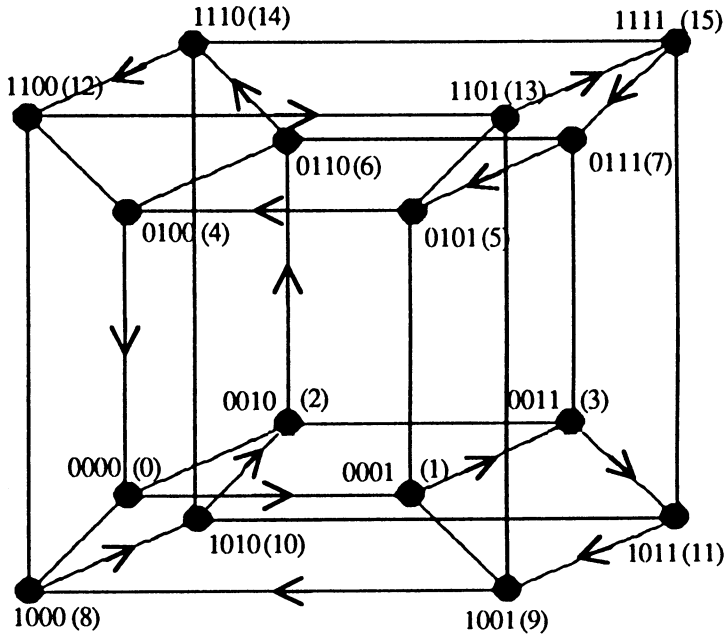


Figure 2.10 : Un hypercube numéroté de dimension 4, avec un anneau qui lui est inscrit.

N° de proc. p	Etape d=4 à 1	Paquets à comm. entre p et $L^1(p)$	Paquets à comm. entre p et $L^2(p)$	Paquets à comm. entre p et $L^3(p)$	Paquets à comm. entre p et $L^4(p)$
proc 0	d=4 :	$V_1^{(4)}=\{T_{15,0}\}$			
	d=3 :	$V_1^{(3)}=\{T_{13,0}\}$	$V_2^{(3)}=\{T_{11,0}\}$	$V_3^{(3)}=\{T_{7,0}\}$	$V_4^{(3)}=\{T_{14,0}\}$
	d=2 :	$V_1^{(2)}=\{T_{5,0}\}$	$V_2^{(2)}=\{T_{3,0}, T_{10,0}\}$	$V_3^{(2)}=\{T_{6,0}\}$	$V_4^{(2)}=\{T_{9,0}, T_{12,0}\}$
	d=1 :	$V_1^{(1)}=\{T_{1,0}\}$	$V_2^{(1)}=\{T_{2,0}\}$	$V_3^{(1)}=\{T_{4,0}\}$	$V_4^{(1)}=\{T_{8,0}\}$
proc 1	d=4 :		$V_2^{(4)}=\{T_{14,1}\}$		
	d=3 :	$V_1^{(3)}=\{T_{12,1}\}$	$V_2^{(3)}=\{T_{10,1}\}$	$V_3^{(3)}=\{T_{6,1}\}$	$V_4^{(3)}=\{T_{15,1}\}$
	d=2 :	$V_1^{(2)}=\{T_{4,1}\}$	$V_2^{(2)}=\{T_{2,1}, T_{11,1}\}$	$V_3^{(2)}=\{T_{7,1}\}$	$V_4^{(2)}=\{T_{8,1}, T_{13,1}\}$
	d=1 :	$V_1^{(1)}=\{T_{0,1}\}$	$V_2^{(1)}=\{T_{3,1}\}$	$V_3^{(1)}=\{T_{5,1}\}$	$V_4^{(1)}=\{T_{9,1}\}$
proc 2	d=4 :			$V_3^{(4)}=\{T_{13,2}\}$	
	d=3 :	$V_1^{(3)}=\{T_{15,2}\}$	$V_2^{(3)}=\{T_{9,2}\}$	$V_3^{(3)}=\{T_{5,2}\}$	$V_4^{(3)}V_4^{(3)}=\{T_{12,2}\}$

d=2 : $V_1^{(2)}=\{T_{7,2}, T_{11,2}\}$ $V_2^{(2)}=\{T_{1,2}\}$ $V_3^{(2)}=\{T_{4,2}\}$ $V_4^{(2)}=\{T_{8,2}, T_{14,2}\}$
d=1 : $V_1^{(1)}=\{T_{3,2}\}$ $V_2^{(1)}=\{T_{0,2}\}$ $V_3^{(1)}=\{T_{6,2}\}$ $V_4^{(1)}=\{T_{10,2}\}$

proc 3 : d=4 : $V_4^{(4)}=\{T_{12,3}\}$
d=3 : $V_1^{(3)}=\{T_{14,3}\}$ $V_2^{(3)}=\{T_{8,3}\}$ $V_3^{(3)}=\{T_{4,3}\}$ $V_4^{(3)}=\{T_{13,3}\}$
d=2 : $V_1^{(2)}=\{T_{6,3}, T_{10,3}\}$ $V_2^{(2)}=\{T_{0,3}\}$ $V_3^{(2)}=\{T_{5,3}\}$ $V_4^{(2)}=\{T_{9,3}, T_{15,3}\}$
d=1 : $V_1^{(1)}=\{T_{2,3}\}$ $V_2^{(1)}=\{T_{1,3}\}$ $V_3^{(1)}=\{T_{7,3}\}$ $V_4^{(1)}=\{T_{11,6}\}$

proc 4 : d=4 : $V_3^{(4)}=\{T_{11,4}\}$
d=3 : $V_1^{(3)}=\{T_{9,4}\}$ $V_2^{(3)}=\{T_{15,4}\}$ $V_3^{(3)}=\{T_{3,4}\}$ $V_4^{(3)}=\{T_{10,4}\}$
d=2 : $V_1^{(2)}=\{T_{7,4}, T_{13,4}\}$ $V_2^{(2)}=\{T_{2,4}\}$ $V_3^{(2)}=\{T_{1,4}\}$ $V_4^{(2)}=\{T_{8,4}, T_{14,4}\}$
d=1 : $V_1^{(1)}=\{T_{5,4}\}$ $V_2^{(1)}=\{T_{6,4}\}$ $V_3^{(1)}=\{T_{0,4}\}$ $V_4^{(1)}=\{T_{12,4}\}$

proc 5 : d=4 : $V_1^{(4)}=\{T_{10,5}\}$
d=3 : $V_1^{(3)}=\{T_{8,5}\}$ $V_2^{(3)}=\{T_{14,5}\}$ $V_3^{(3)}=\{T_{2,5}\}$ $V_4^{(3)}=\{T_{11,5}\}$
d=2 : $V_1^{(2)}=\{T_{6,5}, T_{12,5}\}$ $V_2^{(2)}=\{T_{3,5}\}$ $V_3^{(2)}=\{T_{0,5}\}$ $V_4^{(2)}=\{T_{9,5}, T_{15,5}\}$
d=1 : $V_1^{(1)}=\{T_{4,5}\}$ $V_2^{(1)}=\{T_{7,5}\}$ $V_3^{(1)}=\{T_{1,5}\}$ $V_4^{(1)}=\{T_{13,5}\}$

proc 6 : d=4 : $V_4^{(4)}=\{T_{9,6}\}$
d=3 : $V_1^{(3)}=\{T_{13,6}\}$ $V_2^{(3)}=\{T_{8,6}\}$ $V_3^{(3)}=\{T_{1,6}\}$ $V_4^{(3)}=\{T_{11,6}\}$
d=2 : $V_1^{(2)}=\{T_{3,6}\}$ $V_2^{(2)}=\{T_{5,6}, T_{12,6}\}$ $V_3^{(2)}=\{T_{0,6}\}$ $V_4^{(2)}=\{T_{10,6}, T_{15,6}\}$
d=1 : $V_1^{(1)}=\{T_{7,6}\}$ $V_2^{(1)}=\{T_{4,6}\}$ $V_3^{(1)}=\{T_{2,6}\}$ $V_4^{(1)}=\{T_{14,6}\}$

proc 7 : d=4 : $V_2^{(4)}=\{T_{8,7}\}$
d=3 : $V_1^{(3)}=\{T_{12,7}\}$ $V_2^{(3)}=\{T_{9,7}\}$ $V_3^{(3)}=\{T_{0,7}\}$ $V_4^{(3)}=\{T_{10,7}\}$
d=2 : $V_1^{(2)}=\{T_{2,7}\}$ $V_2^{(2)}=\{T_{4,7}, T_{13,7}\}$ $V_3^{(2)}=\{T_{1,7}\}$ $V_4^{(2)}=\{T_{11,7}, T_{14,7}\}$
d=1 : $V_1^{(1)}=\{T_{6,7}\}$ $V_2^{(1)}=\{T_{5,7}\}$ $V_3^{(1)}=\{T_{3,7}\}$ $V_4^{(1)}=\{T_{15,7}\}$

proc 8 : d=4 : $V_2^{(4)}=\{T_{7,8}\}$
d=3 : $V_1^{(3)}=\{T_{5,8}\}$ $V_2^{(3)}=\{T_{15,8}\}$ $V_3^{(3)}=\{T_{6,8}\}$ $V_4^{(3)}=\{T_{3,8}\}$
d=2 : $V_1^{(2)}=\{T_{11,8}, T_{13,8}\}$ $V_2^{(2)}=\{T_{2,8}\}$ $V_3^{(2)}=\{T_{4,8}, T_{14,8}\}$ $V_4^{(2)}=\{T_{1,8}\}$
d=1 : $V_1^{(1)}=\{T_{9,8}\}$ $V_2^{(1)}=\{T_{10,8}\}$ $V_3^{(1)}=\{T_{12,8}\}$ $V_4^{(1)}=\{T_{0,8}\}$

proc 9 : d=4 : $V_1^{(4)}=\{T_{6,9}\}$
d=3 : $V_1^{(3)}=\{T_{4,9}\}$ $V_2^{(3)}=\{T_{14,9}\}$ $V_3^{(3)}=\{T_{7,9}\}$ $V_4^{(3)}=\{T_{2,9}\}$
d=2 : $V_1^{(2)}=\{T_{10,9}, T_{12,9}\}$ $V_2^{(2)}=\{T_{3,9}\}$ $V_3^{(2)}=\{T_{5,9}, T_{15,9}\}$ $V_4^{(2)}=\{T_{0,9}\}$
d=1 : $V_1^{(1)}=\{T_{8,9}\}$ $V_2^{(1)}=\{T_{11,9}\}$ $V_3^{(1)}=\{T_{13,9}\}$ $V_4^{(1)}=\{T_{1,9}\}$

proc 10 : d=4 :				$V_4^{(4)}=\{T_{5,10}\}$
d=3 : $V_1^{(3)}=\{T_{13,10}\}$	$V_2^{(3)}=\{T_{4,10}\}$	$V_3^{(3)}=\{T_{7,10}\}$	$V_4^{(3)}=\{T_{1,10}\}$	
d=2 : $V_1^{(2)}=\{T_{3,10}\}$	$V_2^{(2)}=\{T_{9,10}, T_{12,10}\}$	$V_3^{(2)}=\{T_{6,10}, T_{15,10}\}$	$V_4^{(2)}=\{T_{0,10}\}$	
d=1 : $V_1^{(1)}=\{T_{11,10}\}$	$V_2^{(1)}=\{T_{8,10}\}$	$V_3^{(1)}=\{T_{14,10}\}$	$V_4^{(1)}=\{T_{2,10}\}$	
proc 11 : d=4 :	$V_2^{(4)}=\{T_{4,11}\}$			
d=3 : $V_1^{(3)}=\{T_{12,11}\}$	$V_2^{(3)}=\{T_{5,11}\}$	$V_3^{(3)}=\{T_{6,11}\}$	$V_4^{(3)}=\{T_{0,11}\}$	
d=2 : $V_1^{(2)}=\{T_{2,11}\}$	$V_2^{(2)}=\{T_{8,11}, T_{13,11}\}$	$V_3^{(2)}=\{T_{7,11}, T_{14,11}\}$	$V_4^{(2)}=\{T_{1,11}\}$	
d=1 : $V_1^{(1)}=\{T_{10,11}\}$	$V_2^{(1)}=\{T_{9,11}\}$	$V_3^{(1)}=\{T_{15,11}\}$	$V_4^{(1)}=\{T_{3,11}\}$	
proc 12 : d=4 : $V_1^{(4)}=\{T_{3,12}\}$				
d=3 : $V_1^{(3)}=\{T_{11,12}\}$	$V_2^{(3)}=\{T_{7,12}\}$	$V_3^{(3)}=\{T_{2,12}\}$	$V_4^{(3)}=\{T_{1,12}\}$	
d=2 : $V_1^{(2)}=\{T_{5,12}\}$	$V_2^{(2)}=\{T_{6,12}, T_{15,12}\}$	$V_3^{(2)}=\{T_{9,12}, T_{10,12}\}$	$V_4^{(2)}=\{T_{0,12}\}$	
d=1 : $V_1^{(1)}=\{T_{13,12}\}$	$V_2^{(1)}=\{T_{14,12}\}$	$V_3^{(1)}=\{T_{8,12}\}$	$V_4^{(1)}=\{T_{4,12}\}$	
proc 13 : d=4 :	$V_2^{(4)}=\{T_{2,13}\}$			
d=3 : $V_1^{(3)}=\{T_{10,13}\}$	$V_2^{(3)}=\{T_{6,13}\}$	$V_3^{(3)}=\{T_{3,13}\}$	$V_4^{(3)}=\{T_{0,13}\}$	
d=2 : $V_1^{(2)}=\{T_{4,13}\}$	$V_2^{(2)}=\{T_{7,13}, T_{14,13}\}$	$V_3^{(2)}=\{T_{8,13}, T_{11,13}\}$	$V_4^{(2)}=\{T_{1,13}\}$	
d=1 : $V_1^{(1)}=\{T_{12,13}\}$	$V_2^{(1)}=\{T_{15,13}\}$	$V_3^{(1)}=\{T_{8,13}\}$	$V_4^{(1)}=\{T_{5,13}\}$	
proc 14 : d=4 :	$V_2^{(4)}=\{T_{1,14}\}$			
d=3 : $V_1^{(3)}=\{T_{9,14}\}$	$V_2^{(3)}=\{T_{5,14}\}$	$V_3^{(3)}=\{T_{3,14}\}$	$V_4^{(3)}=\{T_{0,14}\}$	
d=2 : $V_1^{(2)}=\{T_{7,14}\}$	$V_2^{(2)}=\{T_{4,14}, T_{13,14}\}$	$V_3^{(2)}=\{T_{8,14}, T_{11,14}\}$	$V_4^{(2)}=\{T_{2,14}\}$	
d=1 : $V_1^{(1)}=\{T_{15,14}\}$	$V_2^{(1)}=\{T_{12,14}\}$	$V_3^{(1)}=\{T_{10,14}\}$	$V_4^{(1)}=\{T_{6,14}\}$	
proc 15 : d=4 :				$V_4^{(4)}=\{T_{0,15}\}$
d=3 : $V_1^{(3)}=\{T_{8,15}\}$	$V_2^{(3)}=\{T_{4,15}\}$	$V_3^{(3)}=\{T_{2,15}\}$	$V_4^{(3)}=\{T_{1,15}\}$	
d=2 : $V_1^{(2)}=\{T_{6,15}\}$	$V_2^{(2)}=\{T_{5,15}, T_{12,15}\}$	$V_3^{(2)}=\{T_{9,15}, T_{10,15}\}$	$V_4^{(2)}=\{T_{3,15}\}$	
d=1 : $V_1^{(1)}=\{T_{14,15}\}$	$V_2^{(1)}=\{T_{13,15}\}$	$V_3^{(1)}=\{T_{11,15}\}$	$V_4^{(1)}=\{T_{7,15}\}$	

Tableau 2.4 : Une illustration de la répartition des paquets dans chaque processseur et des numéros des canaux à travers lesquels ces paquets doivent être envoyés : par exemple, $V_2^{(2)}=\{T_{5,15}, T_{12,15}\}$ dans le processeur 15 doit être envoyé au processeur $L^2(15)=13$ à l'étape correspondant à d=2. Ici, $T_{5,15}, T_{12,15}$ sont supposés avoir été stockés dans deux locations consécutives de la mémoire. Il ne faut pas oublier que, dans l'**Algorithme II.4**, quand la moitié des processeurs dont le numéro est pair envoient, l'autre moitié recoivent et vice versa.

Le temps de communication dans la première phase est $\beta+(m_k+1)\tau$,

tandis que le temps de communication à chaque étape d de la deuxième phase est

$$2*[\beta+(\frac{1}{n}C_d^n+1)*(m_k+1)\tau].$$

Donc le temps total de communication est :

$$\beta+ (m_k+1)\tau + 2*\sum_{d=1}^{n-1} [\beta+(\frac{1}{n}C_d^n+1)*(m_k+1)\tau]$$

$$= -\beta-(m_k+1)\tau + 2*[n*\beta+\sum_{d=1}^{n-1} (\frac{1}{n}C_d^n+1)*(m_k+1)\tau]$$

$$\leq -\beta-(m_k+1)\tau + 2*[n*\beta+\sum_{d=1}^{n-1} (\frac{1}{n}C_d^n+1)*(m_k+1)\tau]$$

$$\leq -\beta-(m_k+1)\tau + 2*[n*\beta+(\frac{1}{n}*\sum_{d=1}^{n-1} C_d^n+n)*(m_k+1)\tau]$$

$$= (2n-1)*\beta+2*[\frac{2^n}{n}+(n-1)]*(m_k+1)\tau$$

$$= (2*\log_2(P)-1)*\beta+2*[\frac{P}{\log_2(P)}+(\log_2(P)-1)]*(m_k+1)\tau$$

Dans l'expression ci-dessus, $(2*\log_2(P)-1)*\beta$ représente le temp total d'initialisation de toutes les communications. Il est assez difficile de faire mieux sans faire appel à des stratégies de contrôle très compliquées, y compris la duplication multiple des paquets. En fait, pour réduire davantage ce temps d'initialisation, il faut regrouper les paquets de telle manière que les paquets à transférer pendant chaque communication initialisée soient plus nombreux, ce qui signifie naturellement un "mélange", dans les groupes de répartition, des paquets de distance de destination différente.

A condition que le problème d'accumulation soit résolu en n étapes avec chaque étape effectuant la communication des paquets à même distance de leurs destinations, nous avons le théorème d'optimalité suivant :

Théorème 2.1

La complexité C_p de l'algorithme d'accumulation proposé ci-dessus pour la mise à jour des entrées totales aux cellules dans la couche k est donnée par:

$$C_p<(2*\log_2(P)-1)*\beta+2*[\frac{P}{\log_2(P)}+(\log_2(P)-1)]*(m_k+1)\tau \qquad (2.8)$$

tandis qu'une borne inférieure est

$$C_{p0} \geq \log_2(P)*\beta+2*\frac{P}{\log_2(P)}*(m_k+1)\tau. \qquad (2.9)$$

où C_{p0} est supposée être la complexité d'un algorithme optimal. L'algorithme est asymptotiquement optimal au sens que $\lim_{P\to\infty} \frac{C_p}{C_{p0}} = 1$ pour $m_k \geq \delta$, où $\delta>0$ est une constante.

$m_k \geq \delta$ signifie que la longueur de paquet doit être supérieure à une constante lorsque le nombre de processeur P s'accoît. C'est une hypothèse raisonnable, parce que l'on n'aurait jamais besoin d'un hypercube de 1024 processeurs pour simuler un réseau de 128 neurones.

Démonstration du **Théorème 2.1** : Il suffit de trouver la borne inférieure pour C_{p0}. En effet, comme l'algorithme d'accumulation doit être accompli en n étapes, il y a au moins n initialisations de communication. A l'intérieur d'une étape d de communication, il y a au moins $P*C_d^n$ paquets à transférer sur un nombre total de $\frac{Pn}{2}$ canaux de l'hypercube. Donc, quel que soit l'algorithme, il y a au moins un canal sur lequel le temps de communication à l'étape d est supérieur à $\beta+(\frac{PC_d^n}{Pn/2})(m_k+1)\tau=\beta+2\frac{C_d^n}{n}(m_k+1)\tau$. La somme des bornes pour d varient de n à 1 donne la borne (2.9). #

Remarque : Le temps d'initialisation des communications est au moins de $\frac{3}{2}*\log_2(P)*\beta$, parce que pour $d<(\frac{n}{2}+1)$ deux initialisations sont nécessaires quel que soit l'algorithme. En fait, pour établir des chemins de longueur d, qui permettent de relier un noeud p quelconque à tous les noeuds à distance d du noeud p, il faut que ces chemins contiennent, au moins, n+1-d voisins du noeud p, sinon il y aurait toujours un noeud à distance d du noeud p qui ne peut pas être atteint. Autrement dit, si l'on veut envoyer les C_d^n paquets du noeud p à ses voisins tel que chaque paquet soit plus proche de sa destination, il faut au moins n+1-d voisins pour la réception. Il y a donc au moins n+1-d canaux issus de chaque noeud qui doivent être utilisés pour envoyer les paquets. Lorsque $d<(\frac{n}{2}+1)$, si le transfert des paquets doit encore être accompli en une seule initialisation de communication sur chaque canal, alors P(n+1-d), nombre maximal des canaux nécessaires pour envoyer les paquets, est supérieur à P*(d-1), nombre maximal de canaux disponibles pour recevoir des paquets dans tous les noeuds, ce qui n'est pas possible. Ceci prouve que le nombre total des initialisations est, au moins,

$\frac{3}{2}*\log_2(P)$. Mais on ne peut pas dire que $\frac{3}{2}\log_2(P)*\beta+2*\frac{P}{\log_2(P)}*(m_k+1)\tau$ est une borne inférieure de C_{p0}, parce que le canal sur lequel la communication est initialisée deux fois dans une étape pourrait ne pas transférer $2*(\frac{C_d^n}{n})$ paquets.

Il faut noter aussi que l'algorithme est asymptotiquement optimal au sens habituel, c'est-à-dire que quand la longueur de paquet tend vers l'infini $m_k \to \infty$, on a $\lim_{m_k\to\infty} \frac{C_p}{C_{p0}} = 1$.

II.5.3 Algorithme de diffusion sur hypercube

Un y-paquet GT_p est identifié seulement par le numero p de son noeud d'origine, parce qu'il doit être envoyé à tous les autres. Il n'y a pas de calcul à faire pendant les communications. Envoyer un paquet GT_p à partir d'un noeud p à tous les autres peut être considéré comme, jusque dans certaine mesure, l'opération inverse des communications étudiées ci-dessus dans le problème ($\mathcal{P}$ 1). A chaque étape d de l'**Algorithme II.4**, les communications doivent joindre les paquets de distance d à ceux qui sont à distance d-1. L'inverse de cette opération permet d'envoyer les y-paquets de distance d-1 de leur origine aux noeuds où ils deviennent des paquets de distance d.

Pour un d quelconque : un noeud p1 reçoit d'un voisin p2 un y-paquet GT_p satisfaisant $d_H(p2, p)=d-1$, si et seulement si p1 a envoyé, au noeud p2 dans le schéma précédent, le x-paquet $T_{p,p1}$ satisfaisant $d_H(p1,p)=d$. Autrement dit, le noeud p2 diffuse GT_p à tous les voisins lui ayant envoyé les x-paquets (de distance d) qui ont le même noeud de destination p.

Supposons que $V_j^{(d)}(p)$ soit la répartition $V_j^{(d)}$ de x-paquets de distance d dans le noeud p. Pour $0\le d<n$ et $1\le j\le n$, notons :

$$GV_j^{(d)}(p)=\{GT_{p'} | T_{p',L^j(p)} \in V_j^{(d+1)}(L^j(p))\}$$

$GV_j^{(d)}(p)$ contient donc les y-paquets de distance d de leur origine; ils doivent être envoyés au noeud $L^j(p)$ à travers un canal reliant p et $L^j(p)$. Voici l'algorithme pour le problème de diffusion ($\mathcal{P}$ 2).

Algorithme II.5 :

La diffusion s'effectue, comme l'accumulation, en deux phases d'opérations dans chaque noeud p, mais contrairement à l'**Algorithme II.4**, ici la communication sur l'anneau aura lieu à la fin (deuxième phase) :

1° Pour d=0 à n-2 faire

(1) Si le nombre de 1 dans la représentation binaire du numéro p est pair,
alors envoyer en parallèle à chaque noeud $L^j(p)$, pour $1 \le j \le n$, les paquets de $GV_j^{(d)}(p)$.
sinon recevoir en parallèle les y-paquets envoyés du noeud voisin $L^j(p)$;

(2) Si le nombre de 1 dans la représentation binaire du numéro p est impair,
alors envoyer en parallèle à chaque noeud $L^j(p)$, pour $1 \le j \le n$, les paquets de $GV_j^{(d)}(p)$.
sinon recevoir en parallèle les paquets envoyés du noeud voisin $L^j(p)$;

2°. Pour d=n-1, faire en parallèle (sur l'anneau)
envoyer l'unique y-paquet de $GV_j^{(d)}(p)$ au noeud précédent ;
recevoir un y-paquet envoyé à partir du noeud suivant;

Théorème 2.2

La complexité de l'algorithme, proposé ci-dessus pour diffuser les informations du gradient de la couche k, est :

$$GC_p \le 2[\frac{P}{\log_2(P)} + (\log_2(P)-1)]*[\beta+(m_k+1)\tau]. \qquad (2.10)$$

tandis qu'une borne inférieure est

$$GC_{p0} \ge \frac{3}{2}\frac{P}{\log_2(P)}\beta + 2\frac{P}{\log_2(P)}(m_k+1)\tau \qquad (2.11)$$

où GC_{p0} est supposée être la complexité d'un algorithme optimal.

La démonstration est similaire à celle du **théorème 2.1**. On peut remarquer que le temps total pour les initialisations de communication est $2*\frac{P}{\log_2(P)} + (\log_2(P)-1)*\beta$ au lieu d'être $(2*\log_2(P)-1)*\beta$ dans le **théorème 2.1**. C'est parce que pour la diffusion, chaque noeud doit envoyer un paquet à plusieurs voisins et qu'il n'est pas possible de placer, dans des emplacements consécutifs de mémoire, chaque groupe de paquets à transférer à un voisin sans effectuer la duplication des données. Donc à l'étape d, chaque noeud doit initialiser $2(\frac{1}{n}C_d^n)$ fois les communications pour envoyer et recevoir, tandis que

dans l'algorithme d'accumulation, chaque noeud initialise seulement deux fois la communication à chaque étape.

L'algorithme de diffusion n'est pas théoriquement optimal. Mais il est facile à implanter et efficace parce que les communication sont bien équilibrées sur chaque canal, et on peut utiliser les même paramètres de contrôle pour l'accumulation. De plus, la différence principale entre les deux bornes ci-dessus, $\frac{P}{2*\log_2(P)}\beta$, représente une proportion de $\frac{1}{4}$ dans le temps total si $(m_k+1)\tau<<\beta$, $\frac{1}{8}$ si $(m_k+1)\tau\approx\beta$ et bien sûr presque rien si $(m_k+1)\tau>>\beta$.

II.5.4. Performance des algorithmes de communication testés sur le T40

Les résultats expérimentaux présentés dans ce paragraphe ont été obtenus sur la machine FPS T40. C'est un hypercube de dimension 5 avec 32 noeuds. Voici quelques caractéristiques importantes de cette machine qui peuvent nous aider à mieux comprendre les résultats.

Premièrement, la communication sur un canal est réalisée à travers un multiplexeur, contrôlé par Transputer dans chaque noeud. Comme un Transputer ne possède que quatre canaux, il peut y avoir au maximum quatre communications parallèles issues d'un noeud. Par conséquent, s'il y a cinq communications qui utilisent les cinq canaux d'un noeud du T40, le temps sera au moins doublé par rapport au temps de communication sur un seul canal (on suppose naturellement que la longueur des messages est la même) ;

Deuxièmement, même si le nombre de communications parallèles (issues d'un noeud) est inférieur à 4, il y a toujours un délai, comparé avec la communication sur un seul canal. Ce délai, dû à la réalisation hardware du système, ne peut être limité au maximum que si l'on programme en Occam, langage dédié au Transputer. La différence causée par l'utilisation d'un autre langage de programmation n'est pas négligable. Dans les interprétations suivantes, cet effet de délai sera dénommé *effet de parallélisme limité*. Il a été montré pour la nouvelle version du système pour le T40 (Release C01) que le temps unitaire de communication τ=1.44μs (microseconde) et le temps de start-up β varie de 1200μs, pour la communication sur un seul canal, à 3000μs pour les communications sur quatre canaux (*effet de parallèlisme limité*).

Nous avons testé les deux algorithmes sur quatre sous-cubes du T40 : cube de dimension deux, trois, quatre (T20) et cinq (T40). La longueur des

paquets varie de 10 à 200 valeurs de virgule flottante de huit octets. Pour mesurer exactement le temps de communication, les opérations d'addition dans l'algorithme d'accumulation ont été enlevées. Pour des raisons pratiques, les programmes ont été écrits en C. Nous donnerons aussi des comparaisons avec les algorithmes sur l'anneau qui sont proposés dans le paragraphe II.4 et servent de "benchmarks" ici.

Temps de communication pour l'accumulation :

Lorsque la dimension d'un hypercube est incrémentée de n à n+1, le nombre de processeurs et le nombre de paquets dans chaque processeur sont doublés. Pour un hypercube de dimension n, le nombre moyen de paquets à communiquer à travers chaque canal est supérieur ou égal à $2\frac{P-1}{n}$ ($P=2^n$). Donc, si la dimension devient n+1, le nombre moyen devient $2\frac{2P-1}{n+1}$. Nous pouvons obtenir un indice important qui caractérise la croissance de la complexité de communication en évaluant le rapport entre les deux valeurs moyennes :

$$\sigma(n)=\frac{2\frac{2P-1}{n+1}}{2\frac{P-1}{n}}=(2+\frac{1}{P-1})\frac{n}{n+1}.$$

Cet indice est d'autant plus exact comme mesure de l'efficacité de l'algorithme que la machine supporte des communications parallèles à travers des canaux.

La Figure 2.11 ci-dessous montre une comparaison de la performance de l'**algorithme II.4,** sur quatre hypercubes dont la dimension varie de deux à cinq. La croissance du temps réel est presque linéaire par rapport à celle de la longueur de paquet et les taux de croissance sont très proches. Le tableau ci-dessus donne une autre illustration de la performance de l'algorithme et de l'*effet de parallélisme limité.* Ici $\sigma(n)$ est évalué pour les valeurs n=2,3, et 4. $\pi(n)$ est défini comme la moyenne des rapports entre les temps correspondant à l'hypercube de dimention n+1 et à l'hypercube de dimension n (la moyenne calculée sur la longueur de paquet varie de 10 à 200). $\pi(n)$ peut donc être considéré comme le taux de croissance du temps de communication quand la dimension passe de n à n+1.

On peut constater qu'il y a une cohérence entre le taux de croissance réel

et le taux d'augmentation de la complexité. La croissance réelle semble respecter l'augmentation de complexité du problème. Cette cohérence ne peut être atteinte que si l'algorithme de communication exploite au mieux la capacité de l'hypercube. Pour mieux interpréter les différences entre le taux d'aumentation de la complexité et le taux de croissance réelle, il faut tenir compte de deux caractéristiques de la machine, présentées au début de ce paragraphe : le ralentissement des communications sur cinq canaux issus d'un même noeud, l'*effet de parallélisme limité*, et le coût de contrôle lorsque la dimension n'est pas grande et la distribution des paquets à communiquer sur chaque canal n'est pas très équilibrée. Ces facteurs jouent des rôles différents pour les hypercubes de dimension différente. Néanmoins, les courbes de croissance du temps et la cohérence observée ici montrent bien la valeur pratique de notre algorithme d'accumulation.

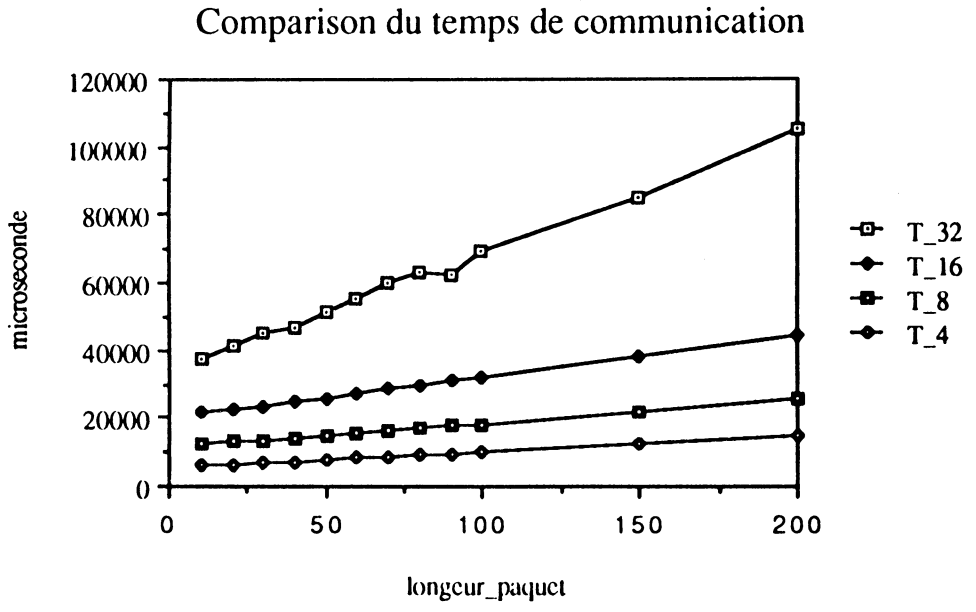


Figure 2.11 : Comparaison du temps de communication pour l'**Algorithme II.4** sur les hypercubes de dimension 2, 3 et 4.

	n=2	n=3	n=4
$\sigma(n)$	1.56	1.61	1.65
$\pi(n)$	1.75	1.73	2.03

Tableau 2.5 : Comparaison entre le taux d'augmentation de la complexité et le taux de croissance réelle du temps de communication pour l'accumulation, lorsque la dimension passe de n à n+1.

Temps de communication pour la diffusion :

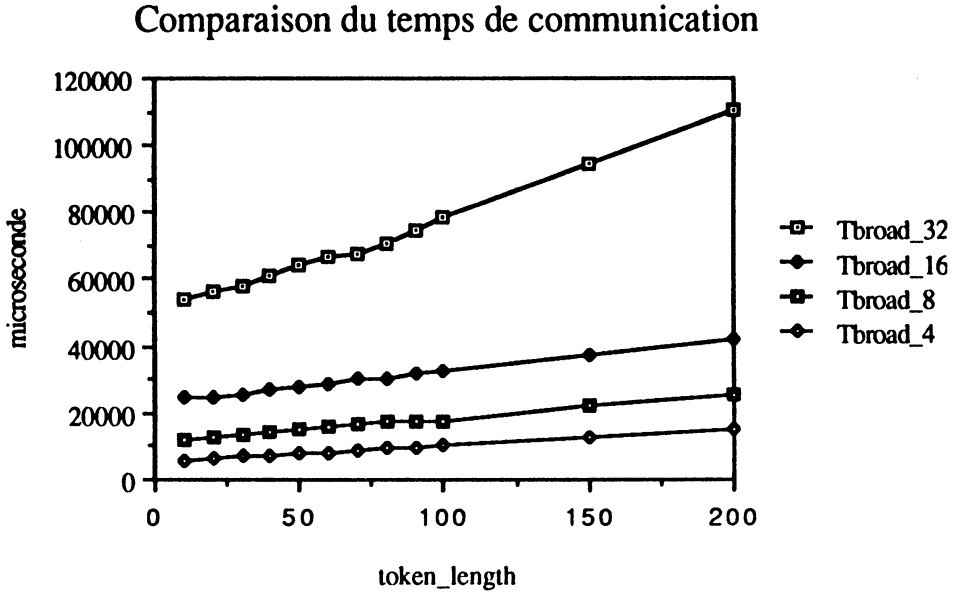


Figure 2.12 : Comparaison du temps de communication pour l'**Algorithme II.5** sur les hypercubes de dimension 2, 3, 4 et 5.

	n=2	n=3	n=4
$\alpha(n)$	1.56	1.61	1.65
$\eta(n)$	1.76	1.82	2.4

Tableau 2.6 : Comparaison entre le taux d'augmentation de la complexité et le taux de croissance réel du temps de communication pour la diffusion, lorsque la dimension passe de n à n+1.

La Figure 2.12 illustre une comparaison du temps de diffusion (all-to-all) sur les hypercubes de dimension différente. A cause des initialisations supplémentaires de communication expliquées dans le **théorème 2.2**, le temps de communication est plus élevé que dans l'algorithme d'accumulation. Mais le taux de croissance de la dimension n à n+1 est encore proportionel au taux de complexité. Si $\eta(n)$ est défini d'une façon similaire à $\pi(n)$ ci-dessus, $\alpha(n)$ est le même que pour l'algorithme d'accumulation. Comme les deux problèmes ont la même complexité en nombre de paquets à transférer sur chaque canal, on peut obtenir un tableau (Tableau 2.6) similaire à celui d'en haut (Tableau 2.5). La croissance du temps de communication correspond bien à celle du taux de complexité.

Comparaisons avec les "benchmarks" :

Nous allons prendre l'**Algorithme II.1** et l'**Algorithme II.2** sur l'anneau comme algorithmes de référence pour l'**Algorithme II.4** et l'**Algorithme II.5**. Si on enlève la partie de calcul flottant dans l'**Algorithme II.1** pour tester seulement le temps de communication, les deux références deviennent identiques. Nous espérons qu'une comparaison avec une référence simple et connue pourra donner des idées plus concrètes de nos algorithmes développés sur la configuration d'hypercube.

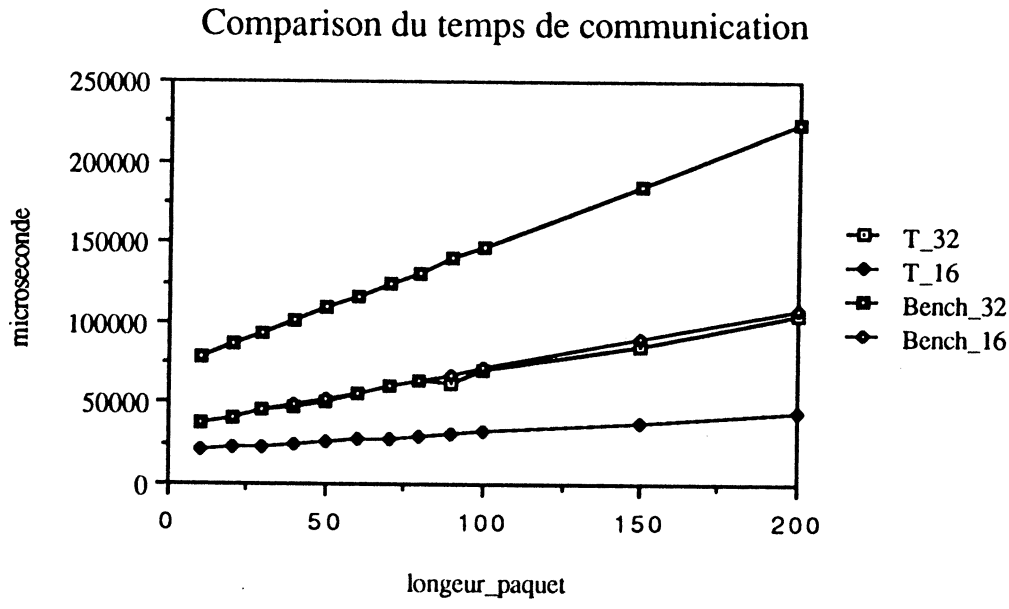


Figure 2.13 : Comparaison du temps de communication pour l'accumulation sur les hypercubes avec le temps de référence.

La Figure 2.13 montre une comparaison entre le temps de communication pour l'algorithme de référence et le temps d'accumulation sur des hypercubes de dimension 4 et 5. Le taux moyen entre le temps de référence et le temps de l'**Algorithme II.4** est de 2.22 pour 16 noeuds (l'anneau de 16 noeuds contre le T20) et de 2.11 pour 32 noeuds (l'anneau de 32 noeuds contre leT40). Il faut noter que le rapport entre le nombre de canaux de l'hypercube et le nombre de canaux utilisés sur un anneau est égal à 2 pour 16 noeuds et 2.5 pour 32 noeuds. Ceci signifie que l'accélération moyenne obtenue par l'utilisation de tous les canaux d'un hypercube correspond à l'avantage du nombre des canaux d'un hypercube sur celui d'un anneau. Il est aussi intéressant de remarquer que le temps de communication sur l'hypercube T40 est très proche de celui de la

référence sur l'anneau de 16 noeuds.

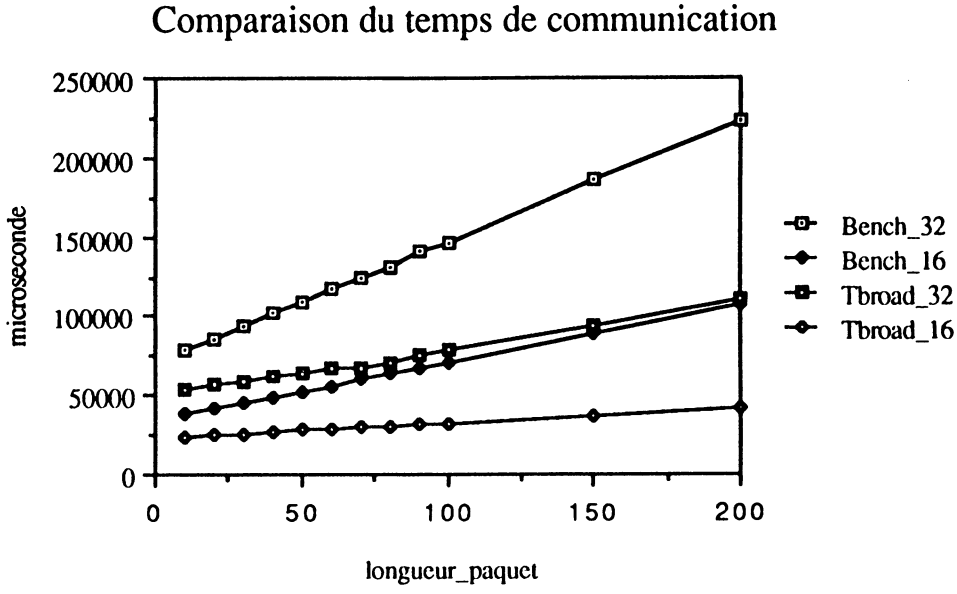


Figure 2.14 : Comparaison du temps de communication pour la diffusion sur les hypercubes avec le temps de référence.

L'avantage de l'**Algorithme II.5** sur l'algorithme de référence pour la diffusion sur l'anneau est aussi clairement manifeste dans la Figure 2.14 ci-dessus, bien qu'il ne soit pas aussi fort que pour l'accumulation, à cause des initialisations supplémentaires de communication.

Jusqu'à ici les comparaisons sont toujours faites entre les temps de communication parce que les études dans cette partie du chapitre sont concentrées à la minimisation du temps de communication. Il faut faire une comparaison entre les temps d'accumulation pour donner une idée plus concrète sur la performance de l'algorithme.

Les Figures 2.15 et 2.16 montrent des comparaisons du temps réels pour l'accumulation lors que les additions intermédiares sont effectuées sur les VPU et lors qu'elles sont effectuées sur les Transputers T414. La Figure 2.16 montre que si la puissance de calcul de chaque processeur est très faible par rapport à la vitesse de transfert des messages les algorithmes sur l'anneau ou sur l'hypercube ne font pas beaucoup de différence. Quant à la Figure 2.15, nous pouvons remarquer que l'avantage de l'algorithme sur l'hypercube, illustré par la Figure 2.13, est préservé grâce à la puissance du processeur arithmétique (ici

le VPU), ce qui est souvent le cas dans des machines à l'architecture hypercube.

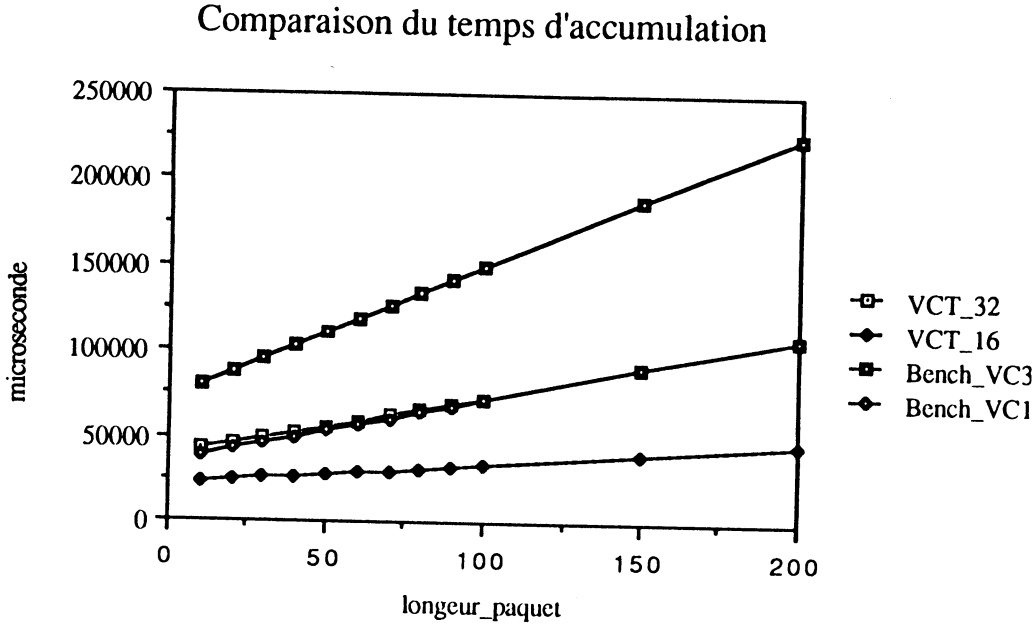


Figure 2.15 : Comparaison du temps d'accumulation sur les hypercubes avec le temps de référence lors que les additions sont effectuées sur les VPU.

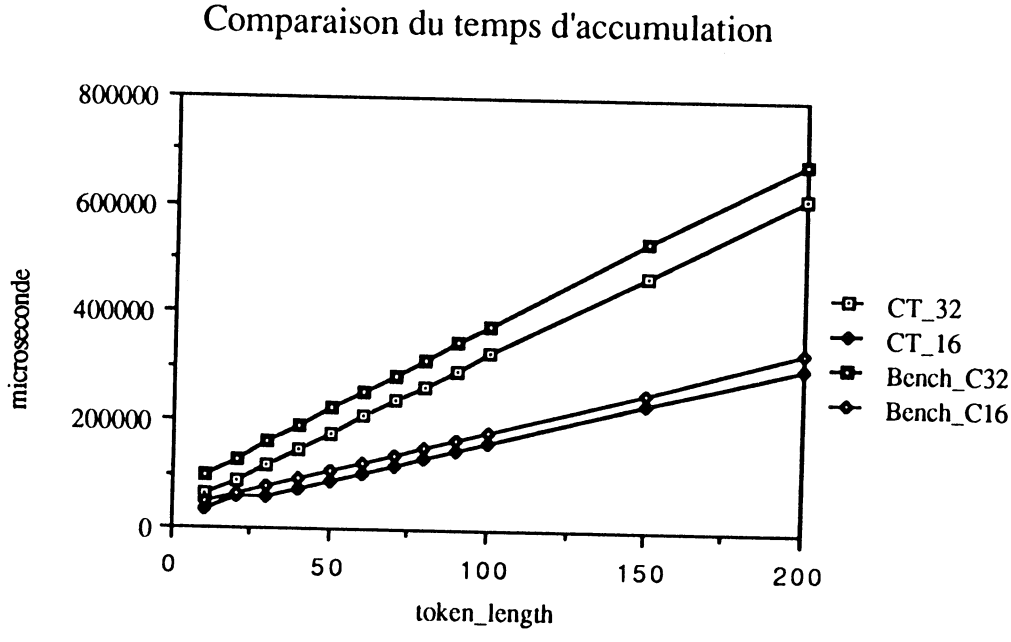


Figure 2.16 : Comparaison du temps d'accumulation sur les hypercubes avec le temps de référence lors que les additions sont effectuées sur les Transputer T414.

II.6. Discussion et Conclusion du chapitre

L'intérêt d'implanter des réseaux de neurones sur des machines parallèles réside dans la puissance de calcul de ces machines et la flexibilité de reconfiguration des réseaux. Un tel système est indispensable pour des expérimentations avec des réseaux de grande taille. Les machines hypercubes sont considérées comme un bon candidat pour l'implantation car l'architecture hypercube réalise un bon compromis entre la capacité de communication et l'extensibilité du système. Il faut bien sûr trouver des algorithmes qui permettent de mieux exploiter cette architecture de multi-processeurs.

Après avoir discuté quelques sujets généraux concernant la représentation et la distribution des réseaux, nous avons présenté une implantation, sur l'anneau, d'un modèle de réseaux dont la structure est régulière. Grâce à cette régularité, nous avons pu effectuer, dans chaque processeur, des calculs en virgule flottante sur le VPU, unité vectorielle arithmétique très puissante. La parallélisation de l'algorithme GBP sur l'anneau a été réalisée de manière très efficace. Les résultats expérimentaux montrent que le gain obtenu en utilisant 16 processeurs pourrait atteindre un rapport de13 pour des réseaux de taille importante.

Pour des réseaux d'architecture non-régulière, le problème devient beaucoup plus délicat. La distribution du réseau sur les différents processeurs joue ici un rôle particulièrement important dans la réduction du coût de la parallélisation, surtout en ce qui concerne la communication. Pour ce type de réseaux, les calculs flottants ne sont plus très adaptés aux VPU car les données peuvent être très dispersées. Cette réalité nous indique qu'une meilleure architecture, pour l'implantation de la plupart des réseaux de neurones, serait un hypercube de grande dimension (c'est-à-dire avec beaucoup de processeurs) avec, dans chaque noeud, un coprocesseur arithmétique qui devrait être facile à utiliser.

Le problème de la distribution optimale d'un réseau quelconque sur une architecture de processeurs est un problème très difficile et qui reste ouvert. Il mérite une étude approfondie. Pour le moment, nous utilisons encore la stratégie de distribution en moyenne pour le cas général. Un autre problème de parallélisation est de trouver les schémas de communication. Bien sûr, c'est un

problème qui est en relation étroite avec le problème de distribution. L'implantation sur l'anneau consiste déjà en une solution du problème pour le cas où l'architecture de processeurs est un anneau et où la distribution du réseau est faite selon la stratégie en moyenne. Cependant, l'anneau est une sous-architecture de l'hypercube et ne permet pas de profiter pleinement de sa capacité de communication lorsque la dimension de l'hypercube est grande. C'est une des raisons qui nous ont poussés à étudier le problème de parallélisation sur l'hypercube.

Nous avons proposé deux algorithmes efficaces, dont l'un est asymptotiquement optimal, pour l'accumulation ($\mathcal{P}1$) et la diffusion ($\mathcal{P}2$), problèmes associés à la parallélisation. Les résultats présentés au paragraphe II.5.4 montrent que ces algorithmes permettent d'utiliser au mieux l'architecture hypercube pour la communication. Les indices fournis dans ce paragraphe nous permettent de conclure que l'hypercube est une architecture rentable du point de vue de sa capacité de communication.

Cette étude sera prolongée dans les directions suivantes. Premièrement, il est intéressant d'introduire certains retards dans la rétropropagation du gradient et dans la modification des poids pour réduire le coût de communication. Pour cela, il faut modifier l'algorithme de GBP de manière à ce que la perturbation ainsi provoquée soit limitée. Cette approche pourrait être fructueuse pour certaines applications, comme par exemple, pour le traitement de la parole. Deuxièmement, nous espérons attaquer le problème de la distribution par des méthodes heuristiques qui permettraient de résoudre le problème pour des classes particulières de réseaux. Troisièmement, un étude sur l'équilibrage entre la puissance des coprocesseurs arithmétiques et la capacité de communication serait nécessaire pour compléter l'étude sur le besoin architectural des machines parallèles pour l'implantation des réseaux de neurones.

Chapitre III

Modélisation de la reconnaissance de visages en contexte

Ce chapitre présente une tentative d'utilisation du modèle connexionniste comme nouveau paradigme de simulation des phénomènes cognitifs. Les travaux ont été effectués en collaboration avec G. Tiberghien, A-C. Schreiber et S. Rousset du Laboratoire de Psychologie Expérimentale. On expose ici une simulation de l'identification des visages en contexte, qui intègre le rôle dynamique des informations contextuelles dans l'élaboration de l'identité. Un réseau multicouches basé sur l'algorithme GBP a été choisi comme outil de simulation.

Le modèle connexionniste peut être adapté au problème de la reconnaissance des visages : ses applications à la reconnaissance de formes laissent entrevoir les capacités des réseaux à traiter efficacement des stimuli visuels complexes. De plus, l'identification d'un visage est une expertise cognitive "émergeant" de la présentation répétée des exemplaires; l'apprentissage par l'exemple étant également une caractéristique fondamentale des modèles connexionnistes. Les résultats présentés dans ce chapitre montrent que le modèle connexionniste, en tant qu'outil informatique, permet de simuler plusieurs effets dans la reconnaissance des visages qui n'ont pas été, jusqu'à présent, très bien pris en compte par les systèmes de type computo-symbolique.

Le paragraphe III.1 constitue un aperçu bref des résultats expérimentaux relatifs à la reconnaissance des visages, et du modèle cognitif de l'identification des personnes dont les fonctionnalités constituent les éléments de base de notre simulation. Le paragraphe III.2 présente l'architecture du système FACENET, la structure des informations et la procédure qui gère le fonctionnement du réseau lors de la phase de reconnaissance.Les résultats de la simulation sont exposés dans le paragraphe III.3. On étudiera alors l'effet de la nature de la relation Visage-Contexte sur la structuration de l'identité à l'apprentissage, la dynamique du réseau pour les recherches inter-contextuelles, ainsi que le fonctionnement des fausses reconnaissances.

III.1. Introduction

Le modèle cognitif élaboré par Bruce et Young (1986) sera présenté, puisqu'il sert de cadre de référence dans notre démarche de spécification du système FACENET. Le rôle du contexte dans *l'identification* des personnes à partir de leur visage sera mis en évidence afin de justifier l'importance de l'intégration des informations contextuelles dans notre réseau.

III.1.1 Présentation d'un modèle cognitif de la mémoire de visages

Le visage revêt une importance sociale considérable de par sa fonction de communication et d'identificateur universel. Cette fonction d'identification est d'autant plus remarquable que les visages possèdent tous la même structure fondamentale (symétrie, mono-orientation, nombre limité de traits) : Il existe donc une grande similarité entre les exemplaires. Certains traits sont plus saillants pour la perception et la reconnaissance des visages (voir Shepherd *et al.*, 1981). Cependant, la perception d'un visage est largement fondée sur la configuration que forment les traits (Sergent, 1984; Young, Hellawel et Hay, 1987). Le stimulus visage est donc à la fois componentiel et configurationnel.

Le modèle proposé par Bruce et Young (1986) est issu de l'évolution et de l'intégration des différents modèles cognitifs de la reconnaissance des visages proposés ces sept dernières années (voir Bruyer, 1987). Ce modèle synthétise les résultats expérimentaux recueillis jusqu'en 1986 ainsi que les indications fournies par les cas cliniques de trouble de la reconnaissance des visages (prosopagnosie). Il est constitué d'un ensemble de modules en inter-relation, un module correspondant à une fonction qui doit pouvoir être isolable expérimentalement ou en fonction de cas cliniques.

Le modèle distingue sept codes qui sont en jeu dans la reconnaissance des visages. Chacun de ces codes correspond à une des informations qui peuvent permettre d'aboutir à l'identification de la personne. Ils sont :

(1) le code pictural et le code structural qui correspondent à la reproduction fidèle de "l'image" du visage au niveau perceptif;

(2) le code de l'expression et celui de la parole qui permettent de traiter des caractéristiques temporaires du visage; l'accès à ces deux codes s'établit en parallèle par rapport aux processus de reconnaissance proprement dits;

(3) trois derniers codes qui représentent plus précisément les aspects sémantiques du visage : le code dérivé visuellement, le code d'identité spécifique (formé à partir des informations contextuelles) et enfin celui du nom.

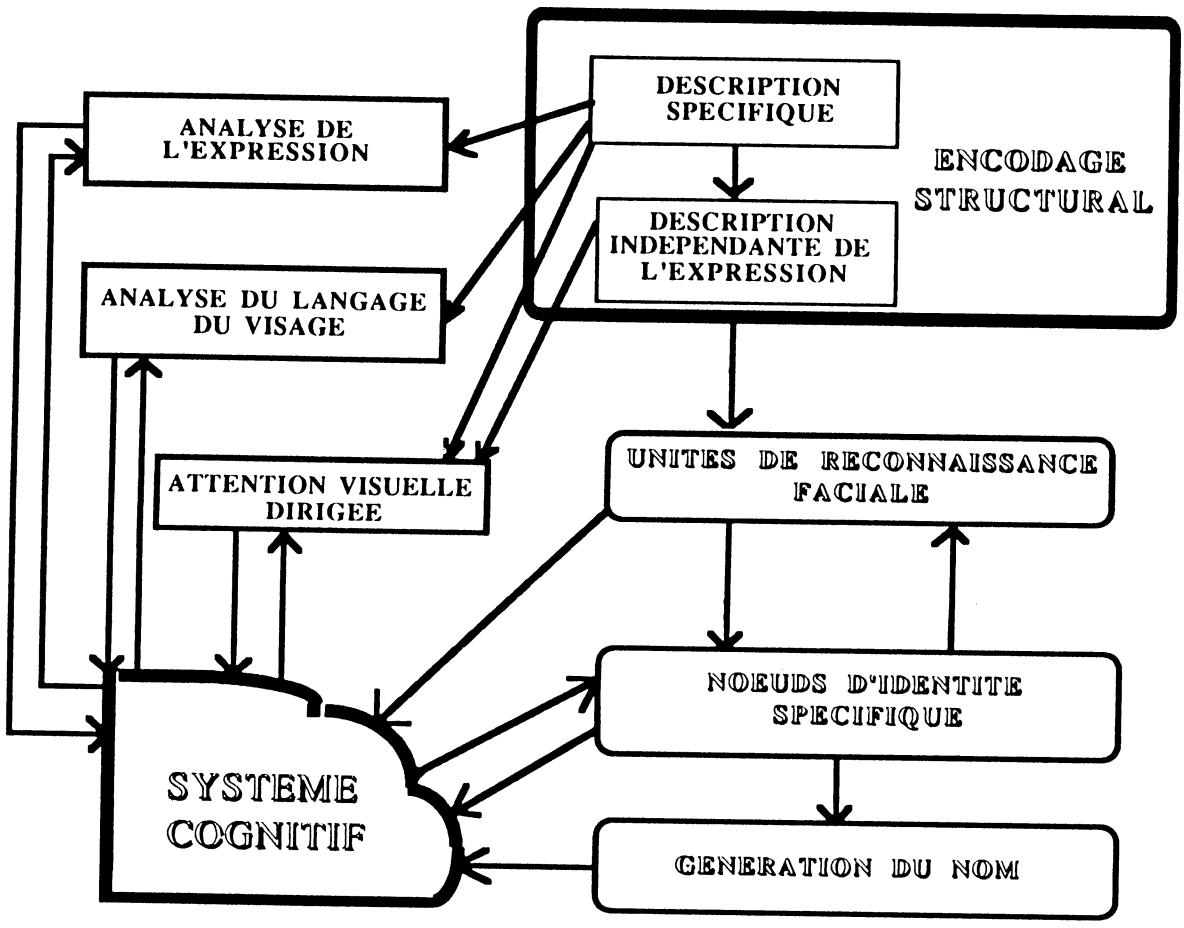


Figure 3.1 : schéma du modèle de Bruce et Young

Ces codes nécessaires à l'identification de la personne à partir de son visage sont produits et traités par cinq modules différents :

- **L'encodage structural** fournit une description spécifique du stimulus tel qu'il est vu (code pictural) et des descriptions plus élaborées de la configuration du visage, indépendamment de l'expression (code structural).

- Le **module d'attention visuelle dirigée** (par le système cognitif) permet d'obtenir les codes sémantiques dérivés visuellement et d'effectuer une analyse fine des caractéristiques physiques du visage perceptivement présent.

Les modèles cognitifs de la reconnaissance des visages proposés ces cinq dernières années reposent sur la notion d'**Unité de Reconnaissance des Visages** (URV) proposée par Hay et Young (1982). Les URV permettraient de passer d'une information perceptive complexe à une estimation de familiarité du visage.

Une URV correspond à la représentation mnésique de l'ensemble des informations perceptives sur un visage, celles-ci ayant pu être obtenues au cours des différentes occurrences du même visage. Ces représentations, ainsi regroupées, peuvent être activées dans leur totalité si une nouvelle information perceptive s'apparie avec l'une d'entre elles. L'activation globale de cette unité permet alors d'obtenir une estimation de familiarité.

- Les **noeuds d'identité spécifiques** contiennent les codes sémantiques d'identité qui sont censés représenter des informations telles que la profession de la personne, les endroits où elle est habituellement rencontrée...

Un des objectifs principaux de la simulation présentée ici sera précisément d'étudier l'influence de la relation visage-contexte dans la constitution et la structuration de l'identité. La hiérarchie introduite entre URV et Noeud d'identité est issue des études de latences (Young et coll., 1986) qui montrent qu'une décision de familiarité est plus rapide qu'une décision sémantique (catégorisation professionnelle).

- L'**unité de génération du nom** de la personne permet de produire celui-ci à partir des informations issues des noeuds d'identité. La hiérarchie introduite dans le modèle entre les noeuds d'identité et l'accès au nom est confirmée par le fait que l'on n'accède jamais au nom en l'absence d'informations d'identité (Young et coll., 1985).

- le **système cognitif** est censé prendre en compte tous les types de traitement non effectués par les autres composants du modèle. Plus précisément, trois fonctions différentes peuvent lui être attribuées :

Il contient toute la mémoire associative, c'est-à-dire l'ensemble des informations épisodiques et sémantiques non directement constitutives de l'identité et assure aussi la fonction de mécanisme de décision; Ainsi le système cognitif reçoit les informations issues de tous les autres modules (sauf de l'encodage structural) et décide d'une réponse, selon des critères non définis, en

fonction de la force d'activation des divers composants du système; La troisième fonctionnalité du système cognitif est illustrée par toutes les flèches qui en sont issues. Il s'agit d'une capacité générale de feedback, qui influence le traitement des autres modules (sauf celui du nom et des URV).

Hormis sa fonctionnalité de décision, nous avons vu que le système cognitif est responsable de processus *top-down*. C'est à travers lui que le contexte représentationnel et situationnel de la reconnaissance interagit avec le traitement spécifique du stimulus visage. Nous nous proposerons donc désormais d'étudier particulièrement l'influence des informations contextuelles dans l'identification des visages.

III.1.2 De l'importance du contexte

Dans les études sur la mémoire, on accepte généralement aujourd'hui que l'encodage et la récupération ne sont pas déterminés uniquement par les propriétés du stimulus lui-même, mais aussi par les caractéristiques de son contexte d'occurrence. Ainsi, le principe de l'encodage spécifique (Tulving et Thomson, 1973) intègre totalement les effets de contexte, en avançant que la probabilité de récupération de l'information mnésique dépend du degré de compatibilité entre les conditions d'encodage et de récupération. Tulving (1983) distingue la mémoire épisodique (faits personnellement vécus) très sensible aux effets de contexte, et la mémoire sémantique (concepts) beaucoup plus indépendante des fluctuations contextuelles.

Les différents types de contexte :

S'il existe différents types de contexte, tous n'ont pas les mêmes effets sur la performance mnésique. Godden et Baddeley (1980) proposent une distinction entre contexte intrinsèque et contexte extrinsèque. Baddeley et Woodhead (1982) préfèrent substituer à cette première distinction celle, plus dynamique, établie entre contexte interactif et contexte indépendant. Un contexte sera dit interactif s'il influence la façon dont le sujet encode le stimulus, tandis qu'un contexte indépendant ne modifiera pas l'encodage de la cible. Les deux distinctions sont reprises par Péris (1986) qui, en les croisant, obtient donc quatre catégories différentes, les contextes pouvant être : extrinsèque-indépendant, intrinsèque-indépendant, extrinsèque-interactif ou enfin

intrinsèque-interactif. D'après les résultats de toute une série d'expériences (rapportées dans Péris, 1986), et en référence aux théories de la reconnaissance médiatisée par un double processus, Péris propose l'existence de deux types différents d'effets de contexte. La dimension intrinsèque-extrinsèque du contexte jouerait un rôle au niveau du processus d'évaluation de la familiarité, alors que la dimension interactif-indépendant affecterait le processus de recherche conditionnelle en mémoire.

Des visages en contexte:

Si le contexte joue dans le processus d'évaluation de la familiarité, son rapport avec le visage est particulièrement intéressant en ce qui concerne le processus d'acquisition de la familiarité, et ses conséquences sur l'élaboration d'une représentation d'identité. La construction d'une représentation d'identité s'effectue probablement par l'intégration des informations contextuelles lors des présentations répétées du visage (ou de la personne) en contexte. Ainsi, le processus d'apprentissage d'un visage (i.e. le processus d'acquisistion de la familiarité) est certainement dépendant de la variabilité contextuelle des occurrences d'un visage.

Les résultats obtenus à ce jour (Davies, 1986, 1988; Memon et Bruce, 1985; Tiberghien, 1986,Young, Hay et Ellis, 1985), indiquent que le contexte influence la capacité à reconnaître les visages. Cependant, l'amplitude de ces effets dépend du type de relation visage-contexte (intrinsèque, interactif, variable, constant..). Tiberghien (1986, 1988) propose qu'il puisse y avoir plusieurs types de familiarités (i.e. de représentations mnésiques d'identité) selon que le visage est présenté à plusieurs reprises dans un contexte identique ou dans des contextes variables. Cette variabilité contextuelle pourrait être à l'origine du passage des représentations de type épisodique (très dépendantes du contexte) aux représentations sémantiques (beaucoup moins sensibles aux influences contextuelles).

Dans la modélisation de la reconnaissance des personnes à partir de leur visage, une question reste aujourd'hui ouverte : comment se constituent les représentations d'identité? Bruce et Young (1986), ne proposent pas d'explication de la dynamique du contexte dans la génèse des noeuds d'identité. Leur modèle doit encore être modifié et précisé, et constitue seulement une

première étape dont la principale vertu est d'être heuristique plutôt que réellement exhaustive.

Vers une modélisation connexionniste :

Le problème principal de ce modèle est qu'il ne postule aucun mécanisme d'apprentissage, c'est-à-dire aucune intégration de nouvelles informations. Ainsi, la dynamique du contexte dans la construction des représentations mnésiques n'est pas évoquée. De plus, son interaction avec le traitement du stimulus visage reste encore très peu définie. Ainsi, par exemple, aucune structure n'est réellement définie pour chacun des modules fonctionnels. Ce modèle cognitif a donné lieu à plusieurs études expérimentales nécessaires à sa validation et, comme le soulignent Bruce et Young, de nombreuses expériences doivent être actuellement engagées afin de déterminer plus précisément la nature des modules ainsi que leurs influences réciproques.

Une modélisation de la reconnaissance des visages en termes connexionnistes a récemment été proposée par Tiberghien (1988). Il s'agit d'une retranscription, sous forme de couches de cellules interconnectées, des divers modules fonctionnels mis en avant par le modèle cognitif présenté ci-dessus. Nous nous sommes inspirés de cette démarche pour créer le système FACENET, qui consiste donc en une modélisation et simulation connexionniste des modules fonctionnels décrits par Bruce et Young. Nous avons de ce fait choisi d'entrer "dans les boites opaques" avec comme préoccupation actuelle la prise en considération du rôle des informations contextuelles dans la création des représentations d'identité.

Nous avons pour l'instant simulé une partie du modèle cognitif, cette première étape s'inscrivant dans un objectif plus général de simulation de l'ensemble du modèle cognitif. Avant d'entrer directement au coeur du système FACENET, nous verrons brièvement pourquoi le modèle connexionniste semble être un outil de simulation particulièrement pertinent (par rapport au modèle computo-symbolique) dans la mémoire des visages.

III.1.3 Intérêt d'une approche connexionniste de la mémoire des visages

Pour qu'une modélisation de type computo-symbolique puisse être

appliquée au stimulus visage, il faudrait déterminer avant tout quels sont les symboles pertinents du visage à traiter. A l'heure actuelle, les caractéristiques qui ont été utilisées dans les systèmes de reconnaissance sont les traits du visage (bouche, yeux, nez, cheveux...). Dans cette optique, la reconnaissance des visages a été définie comme la mise en oeuvre de deux processus distincts : le premier consiste en une extraction des traits du visage et le second en une recherche en mémoire des traits les plus compatibles avec ceux extraits. Dans les simulations effectuées selon cette méthode (Laughery, Rhodes et Batten, 1981; Davies et coll., 1982; Shepherd, 1986), l'extraction des traits s'effectue par mesure géométrique d'une photo du visage et/ou par description verbale. La recherche mnésique, elle, est assurée par deux types d'algorithmes, un algorithme séquentiel qui examine l'ensemble de la base et calcule des mesures de similarités inter-traits, et un algorithme d'appariement qui parcourt l'arbre des possibilités en l'élaguant progressivement selon les valeurs des traits recherchés.

Ces systèmes sont très lents et non adaptatifs. Le rappel verbal des traits n'est pas fiable puisque le stimulus visage est grandement configurationnel, l'interaction entre traits invalidant toute tentative de rappel du visage trait par trait. De plus, il est très difficile (et très arbitraire) de définir de façon définitive quelles informations sont regroupées sous un même trait. Il en est de même pour trouver une base adéquate des traits de visage pour un traitement symbolique. Il faut donc trouver un modèle de traitement qui permette de considérer le visage comme un stimulus unitaire. Une telle démarche est rendue possible par les modèles connexionnistes qui proposent d'aller en deçà du niveau symbolique.

Le modèle connexionniste est pertinent si l'on considère la manière dont l'expertise de reconnaissance des visages est acquise. De plus, connaissant l'importance des informations contextuelles dans la reconnaissance des visages, une modélisation connexionniste est ici intéressante de par ses capacités de traitement parallèle et interactif. Ainsi, Amy et Tiberghien (1988) mettent en évidence les difficultés de l'I.A. traditionnelle à intégrer le contexte dans les systèmes séquentiels car "*un contexte n'attend pas la fin de telle ou telle étape d'un processus pour l'influencer*" (page 27). Rumelhart et McClelland (1986) considèrent également que l'intégration des effets de contexte est une propriété intrinsèque des modèles PDP (*Parallel Distributed Processing*).

Un exemple d'application du paradigme connexionniste à la mémoire

des visages est rapporté par O'Toole, Millward et Anderson (1988). Ces chercheurs s'appuient sur le modèle des mémoires auto-associatives de Kohonen (1982) pour construire un "artéfact connexionniste". Celui-ci sera soumis aux mêmes conditions expérimentales que les sujets humains. La tâche consiste à apprendre et reconnaître des visages préalablement filtrés sur les hautes ou basses fréquences ou normaux. O'Toole et coll. retrouvent globalement les mêmes formes de transfert (entre les diverses manipulations de fréquences effectuées) en ce qui concerne les résultats du réseau et les performances des sujets humains.

Le modèle connexionniste fournit donc des indications simples sur la manière dont les visages peuvent être codés, stockés et retrouvés en mémoire, à partir de l'information visuelle de base. Ces auteurs ne pensent pas avoir forcément émulé un mécanisme de reconnaissance des visages, mais font toutefois remarquer qu'il pourrait parfois être possible d'expliquer des résultats expérimentaux par un modèle simple d'encodage et de récupération distribués, sans avoir recours à des interprétations complexes en terme de traitement symbolique, de changement de stratégie ou de modules spécialisés de traitement. Comme le souligne McClelland (1988), la vertu principale d'une simulation est son caractère heuristique qui permet de stimuler la recherche. Ainsi, on peut proposer de nouvelles interprétations de résultats déjà obtenus sur des sujets humains, ou trouver une cohérence explicative à des faits expérimentaux qui paraissaient contradictoires. Qui plus est, une simulation connexionniste permet parfois de proposer des mécanismes simples et précis pour expliquer des faits apparemment complexes.Elle permet enfin de prolonger la méthode expérimentale traditionnelle en permettant une expérimentation virtuelle sur le modèle et très contrôlable.

Deux objectifs principaux peuvent être poursuivis lors d'une simulation (Gluck et Bower, 1988). On peut désirer évaluer les capacités d'un réseau à produire des comportements complexes (e.g. apprendre à lire : Sejnowski et Rosenberg, 1986), mais aussi partir d'un paradigme expérimental précis (ou d'un modèle), en essayant de trouver sur le réseau des résultats déjà établis ou nouveaux, qui permettraient d'envisager des études expérimentales par rapport au modèle simulé. Dans le domaine de la reconnaissance des visages, ce deuxième objectif a déjà suscité quelques expériences de simulation fondées sur une correspondance avec des résultats expérimentaux bien établis (O'Toole *et al.*,

1988; Valentine, 1988).

Massaro (1988) s'interroge cependant sur l'apport d'une simulation connexionniste dans la démarche de modélisation des phénomènes cognitifs. *Etant donné la puissance des systèmes PDP, une simulation qui fournit des résultats conformes au modèle que l'on voulait tester ne crédite pas pour autant la pertinence du modèle cognitif en question.* En effet, des architectures connexionnistes totalement différentes peuvent néanmoins produire des performances similaires, ce qui témoigne de leur puissance potentielle, mais invalide le retour direct au modèle de base, à partir des résultats d'une simple évaluation du système implanté. Massaro conclut donc que la simulation d'un modèle ne prend toute sa valeur théorique et psychologique que si la démarche de simulation s'accompagne de tests expérimentaux précis et répétés : le modèle simulé devra donc être soumis à plusieurs protocoles expérimentaux et testé à ses limites pour être réellement heuristique et utile dans ses rapports avec la modélisation cognitive.

L'élaboration du système FACENET renvoie donc aux intérêts inhérents à toute simulation. Plus précisément, ce système peut être considéré comme un sujet virtuel et permettre de nombreuses "expérimentations" fines et très contrôlables. En effet, on peut faire varier de manière systématique sa structure, ses paramètres d'apprentissage, et étudier alors son comportement dans des conditions "idéales" : par exemple, son histoire faciale est parfaitement maîtrisable et manipulable à souhait, ce qui n'est pas le cas dans les expériences avec des sujets réels. La validité d'une telle démarche n'est bien sûr envisageable qu'à travers une référence constante aux résultats obtenus dans des conditions méthodologiques plus traditionnelles, c'est-à-dire avec des sujets humains (Tiberghien, 1988a).

III.1.4 Pourquoi choisir un réseau multicouches

En plus des points généraux du modèle connexionniste développés dans le chapitre I, nous avons deux raisons particulières de choisir un réseau multi-couches comme support de notre modélisation.

1). Souplesse de l'architecture : Il est facile de construire un réseau multi-couche avec plusieurs parties réalisant chacune des fonctionalités différentes, tout en conservant des interactions entre elles. Cette facilité est très importante

dans les modélisations où l'on doit absolument manipuler des fonctionalités "unitaires" et interactives.

2). Système dynamique pour la recherche mnésique : C'est la raison la plus importante de notre choix. Comme on l'a présenté au paragraphe I.3, un réseau multicouche peut être vu comme un système dynamique si on lui adjoint un processus itératif de réinjection. Il peut identifier les entrées "bruitées" en respectant les conditions initiales imposées par ces entrées. Cette capacité de recherche dynamique lui permet de retrouver progressivement des informations apprises et de mettre en évidence l'influence, lors de la recherche mnésique, de la structure des concepts induite par les modalités d'apprentissage. Pour nous, un concept possède une étiquette, imposée lors de l'apprentissage, qui est associée à une structure de connaissance (contenu) construite de façon différente selon les expériences du réseau (Par exemple, la nature du concept de l'identité d'une personne X, s'élabore en fonction des modalités de rencontre de cet individu X.).

III.2. Présentation du système FACENET

Le système FACENET a été élaboré en tenant compte des fonctionnalités mises en avant par les modèles cognitifs, fonctionnalités qui ont permis de déterminer l'architecture du réseau multi-couches. Le choix d'une telle architecture spécifique et fonctionnelle tient compte des propositions de modélisation avancées par Tiberghien (1988).

Le modèle de Bruce et Young décompose le processus d'identification et de dénomination des visages en trois niveaux de traitements hiérarchisés :

- perceptif, qui part de l'image du visage jusqu'aux URV (matching du code structural et des Unités de Reconnaissance de Visage qui donne naissance au sentiment de familiarité).
- recherche mnésique de l'identité qui succède à l'estimation de la familiarité et aboutit aux noeuds d'identités spécifiques (intégration d'informations contextuelles et sémantiques).
- accès au nom.

Le premier réseau perceptif et l'accès au nom n'ont pas encore été modélisés en termes connexionnistes. Les nombreuses applications actuelles utilisant des réseaux multi-couches en reconnaissance de formes indiquent qu'ils sont adaptés à la modélisation de ce type de problèmes. Il faudra cependant tenir compte de la spécificité du traitement perceptif du stimulus visage.

Nous nous sommes pour l'instant intéressés au second niveau de traitement, où la construction d'une identité dépend de l'intégration de deux types d'informations : le visage et les contextes d'occurrence de celui-ci. Dans ce qui suit, nous présenterons donc l'architecture détaillée du processus mnésique d'identification qui, lui, a déjà été implanté et soumis à des tests d'évaluation.

III.2.1 L'architecture du réseau d'identification

Schéma du réseau :

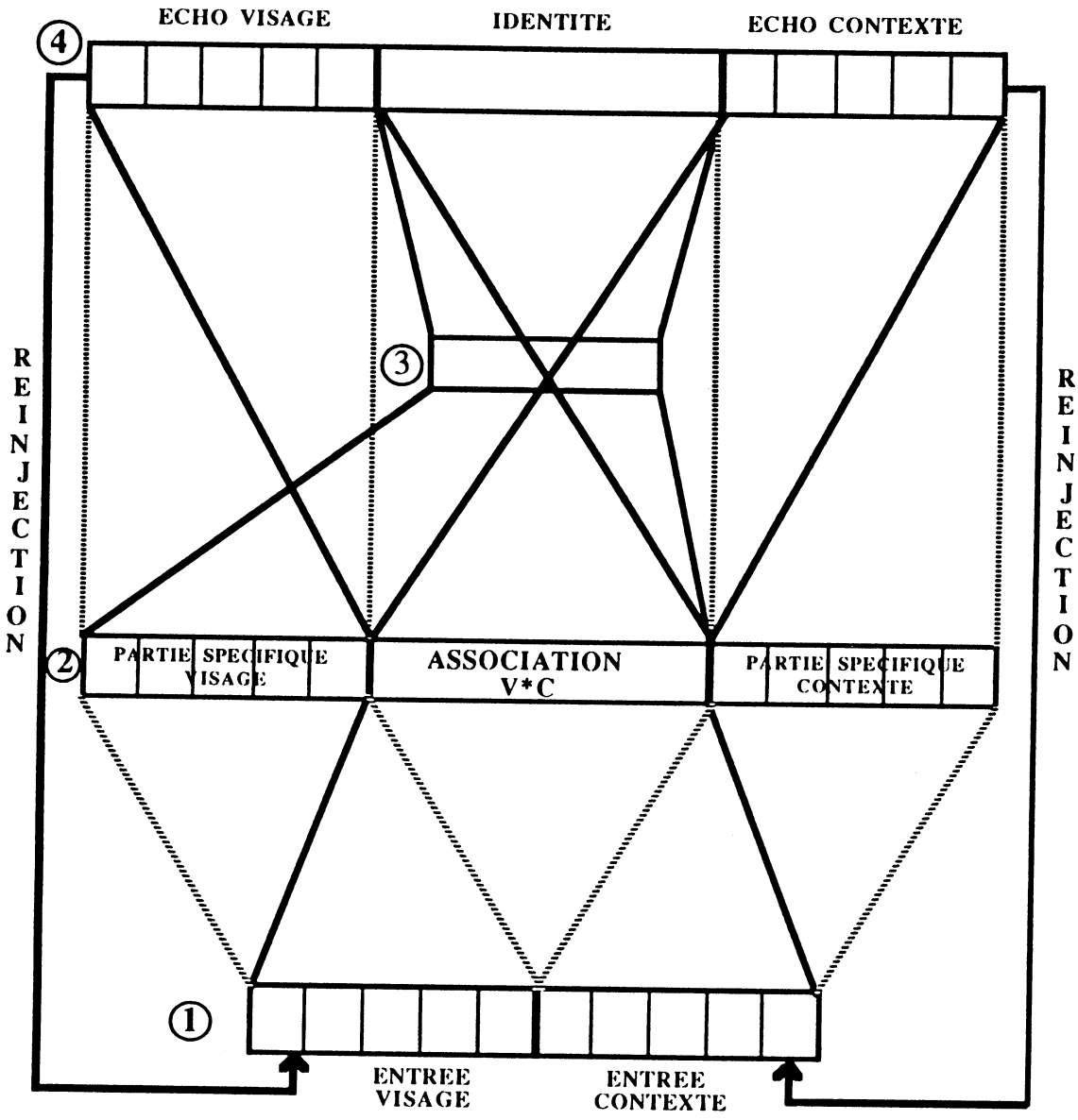


Figure 3.2 : architecture détaillée du réseau de recherche mnésique : il comporte 4 couches, 230 cellules et 3950 connexions. Les traits pleins délimitent les faisceaux de connexions totales inter-couches. Les traits en pointillés délimitent des faisceaux de connexions spécifiques, décrits en détail dans la partie 3.2.3. Les deux flèches latérales indiquent que les deux parties externes de la couche de sortie peuvent être réinjectées en entrée du réseau.

Description de l'architecture :

Dans cette application, les patterns, constitués d'association d'un visage et d'un contexte, sont non linéairement séparables. Une architecture multi-couches a été choisie en vertu de ses capacités à traiter efficacement ce genre d'informations (cf Chapitre I). La présence d'unités cachées, qui permet la

construction de représentations internes, va se montrer particulièrement intéressante pour notre application, concernée par l'élaboration de la représentation d'identité au cours de l'apprentissage. Enfin, cette architecture nous a permis de retranscrire facilement les fonctionnalités d'un modèle cognitif au travers des différentes couches et de leurs inter-connexions.

Nous donnerons tout d'abord un aperçu général des informations d'entrée / sortie de ce réseau mnésique d'identification : la couche d'entrée reçoit des informations de type visage et contexte. La couche de sortie renvoie un écho du visage et un écho du contexte après un cycle de traitement de reconnaissance par le réseau. La partie centrale de cette couche de sortie fournit une information sur l'identité de la personne. Une partie des sorties du réseau peut être renvoyée en entrée suivant une procédure de réinjection.

Les entrées du réseau actuel résultent du traitement perceptif. FACENET modélise le traitement mnésique qui conduit à l'estimation de familiarité et à l'identification. Ce traitement mnésique a lieu après le traitement perceptif, qui dans le modèle de Bruce et Young est supposé coder l'information perceptive selon les niveaux de traitement définis par Marr (1982). Si les entrées de Facenet correspondent à des informations perceptives, les trois couches suivantes participent donc au processus mnésique d'intégration des informations contextuelles qui aboutit à la reconnaissance. Cette identification est supposée précéder et permettre l'accès au nom dans une troisième étape.

Fonctionnalités de l'architecture :

* couche 1 (50 cellules) : sortie du réseau perceptif, utilisée comme entrée du réseau de recherche mnésique d'identification : elle est divisée en deux parties : une représente l'entrée visage et l'autre l'entrée contexte.

- Partie visage (25 cellules) : Nous avons pour l'instant choisi de définir cette information perceptive de manière simple, afin de pouvoir analyser point par point son traitement par le réseau. Nous avons donc divisé l'information visage en 5 blocs représentant des macro-composantes du visage. Chaque bloc comporte 5 unités, 1 unité codant une valeur particulière de ce macro-trait. L'aspect componentiel du visage est ainsi modélisé de façon simple puisque chaque bloc d'information est censé coder une macro-composante particulière du visage. Un traitement plus configurationnel et la hiérarchie entre les traits seront simulés

par l'intermédiaire des connexions spécifiques décrites en détail plus loin (tous les traits n'ont en effet pas le même poids, comme les expériences de rappel, de reconnaissance et de reconstruction ont pu le montrer).

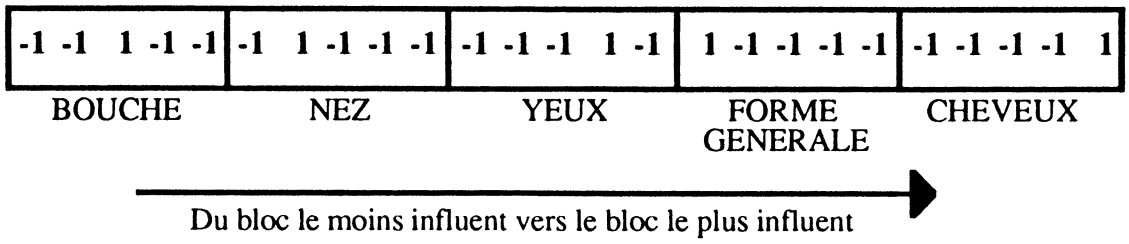


Figure 3.3 : description de la structure de l'information visage de la couche d'entrée (les noms de macro-traits ne sont donnés ici qu'à titre d'exemple).

- Partie contexte (25 cellules) : pour l'entrée contexte, nous avons choisi une représentation similaire à celle du visage. Ce choix est également réducteur étant donné l'étendue de ce qui peut être regroupé sous ce terme (contexte situationnel, contexte représentationnel).

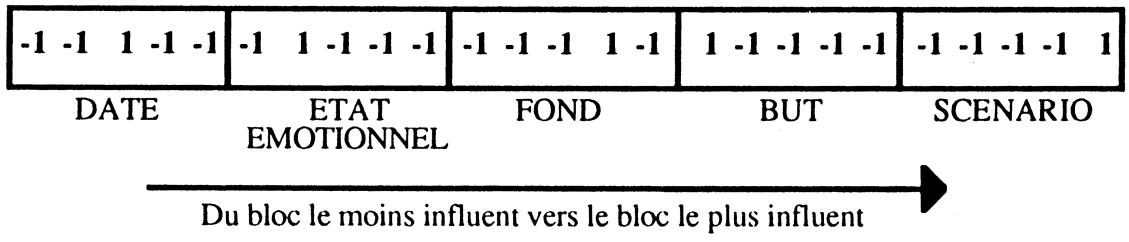


Figure 3.4 : description de la structure de l'information contexte de la couche d'entrée (les noms donnés ici ne prétendent pas épuiser l'ensemble des possibles).

Les contextes sont hiérarchisés de gauche à droite (tous n'ayant pas le même poids si l'on s'en réfère à la taxonomie des effets de contexte). Leur influence différentielle sera modélisée par l'intermédiaire des connexions spécifiques.

* couche 2 (80 cellules) : cette couche est subdivisée en 3 parties : partie spécifique visage (25 cellules), association Visage-Contexte (30 cellules), partie spécifique contexte (25 cellules).

- Association Visage-Contexte : ses 30 cellules sont connectées totalement à l'ensemble de la couche d'entrée. Par ces connexions s'effectue une première association entre visage et contexte, association qui psychologiquement est déterminante pour l'élaboration d'une identité. Elle modélise donc la

construction dynamique des représentations épisodiques contextualisées. Cette association permettra en outre de rappeler un contexte à partir d'un visage et inversement, grâce à son influence sur les parties latérales de sortie du réseau. Par la suite, cette association Visage-Contexte sera abrégée par le terme : "association V-C".

- Partie spécifique visage : elle possède la même structure que la couche d'entrée-visage et est connectée spécifiquement aux entrées visage seulement. La forme de connectivité entre ces deux couches est illustrée par une partie de la Figure 3.5.

Fonction des connexions spécifiques : les 5 cellules du bloc de gauche de la couche 2 sont influencées par l'ensemble des entrées du visage. Le deuxième bloc est celui influencé par l'ensemble des entrées visage sauf le premier bloc... et ainsi de suite en restreignant pour chaque bloc du haut l'amplitude des influences reçues de l'input visage. Donc, en suivant la dénomination donnée en exemple, le bloc d'entrée "cheveux" influence l'ensemble des blocs du visage de la couche 2, alors que le bloc d'entrée "bouche" ne se projette que sur la partie bouche de la couche 2. Cette partie spécifique visage de la couche 2 permet de passer d'une information componentielle en entrée à une information plus configurationnelle. Les connexions spécifiques modélisent ici essentiellement une pondération entre les blocs, lors de la recherche mnésique.

- Partie spécifique contexte : la même distribution des connexions a été effectuée sur les informations contextuelles afin de leur rendre leur caractère interactif. Ici, il s'agirait plutôt d'un caractère semi-interactif (les psychologues sociaux diraient simplement "rapport de domination") puisque l'influence potentielle est à sens unique, un bloc ne pouvant jamais influencer les blocs se trouvant sur sa droite.

* couche 3 (20 cellules) : couche de généralisation, en connexion totale avec l'association V-C et l'information visage de la couche précédente. On peut en effet attendre, de par la réduction du nombre des cellules entre les couches 2 et 3, un comportement de généralisation par le réseau. De plus, puisque c'est la seule couche connectée aux sorties d'identité, elle aura à abstraire des représentations d'identité pour satisfaire la contrainte imposée en couche 4. Cette couche 3 devrait modéliser, avec la couche 2, la mémoire sémantique et épisodique concernant l'identité de la personne. L'asymétrie, introduite dans le réseau par les connexions provenant de la partie spécifique-visage de la couche 2,

a pour fonction d'accentuer l'importance du visage (par rapport au contexte) dans le processus d'identification, et lui confère ainsi le statut de clé d'accès privilégiée à la représentation d'identité. Ceci devrait permettre de modéliser le statut focal du visage et le rôle important, quoique contextuel, du contexte !

* couche 4 (80 cellules) : cette couche de sortie du réseau est subdivisée en 3 parties : sortie d'identité (30 cellules), sortie écho-visage (25 cellules), sortie écho-contexte (25 cellules).

- Sortie d'identité : cette partie comporte 1 cellule par visage à apprendre. Chaque cellule correspond à un noeud d'identité spécifique en connexion totale avec la couche de généralisation. Les connexions entre les couches 2 et 3, et 3 et 4, représentent l'intégration des informations faciales et contextuelles qui conduit au noeud d'identité spécifique. Chaque noeud représente au fait l'endroit où les informations sémantiques et épisodiques convergent afin de spécifier l'identité du visage présenté en entrée. Ce sont ces faiseaux de connexions qui encodent la structure et le contenu des représentations d'identité en mémoire. La valeur d'un noeud d'identité est l'indicateur de l'activité de cette représentation d'identité particulière.

- Sortie écho-visage : elle est connectée totalement à la partie d'association V-C de la couche 2 et spécifiquement à la partie visage de cette même couche. Les deux échos (visage et contexte) représentent la réminiscence de la recherche, c'est-à-dire qu'ils contiennent l'image mentale (du visage et du contexte) émergeant de la recherche en mémoire. Le pattern d'activation des échos permet une estimation de familiarité par comparaison avec les entrées perceptives. Leur fonction est donc équivalente aux URV du modèle de Bruce et Young. Le faisceau de connexions totales permet de modéliser l'influence sur l'écho-visage des visages qui ont été activés par l'association V-C dans la couche 2. Ainsi le contexte, par le jeu des associations V-C, peut activer des visages dont l'évocation se retranscrira sur la sortie écho-visage. Le faisceau de connexions spécifiques prolonge ici le traitement configurationnel des visages effectué entre les couches 1 et 2, par le même type de connexions spécifiques. En effet, ceci permet d'affiner l'influence hiérarchique des traits en introduisant une pondération discriminative dans l'influence des traits les uns sur les autres : ainsi, le premier bloc de la sortie visage peut être influencé 5 fois par le cinquième bloc de l'entrée visage, 4 fois par le quatrième bloc... et 1 fois par le premier bloc.

Le schéma suivant représente l'architecture à deux étages de connexions

spécifiques, pour la partie "latérale spécifique visage" (l'autre partie "latérale spécifique contexte" ayant exactement la même structure).

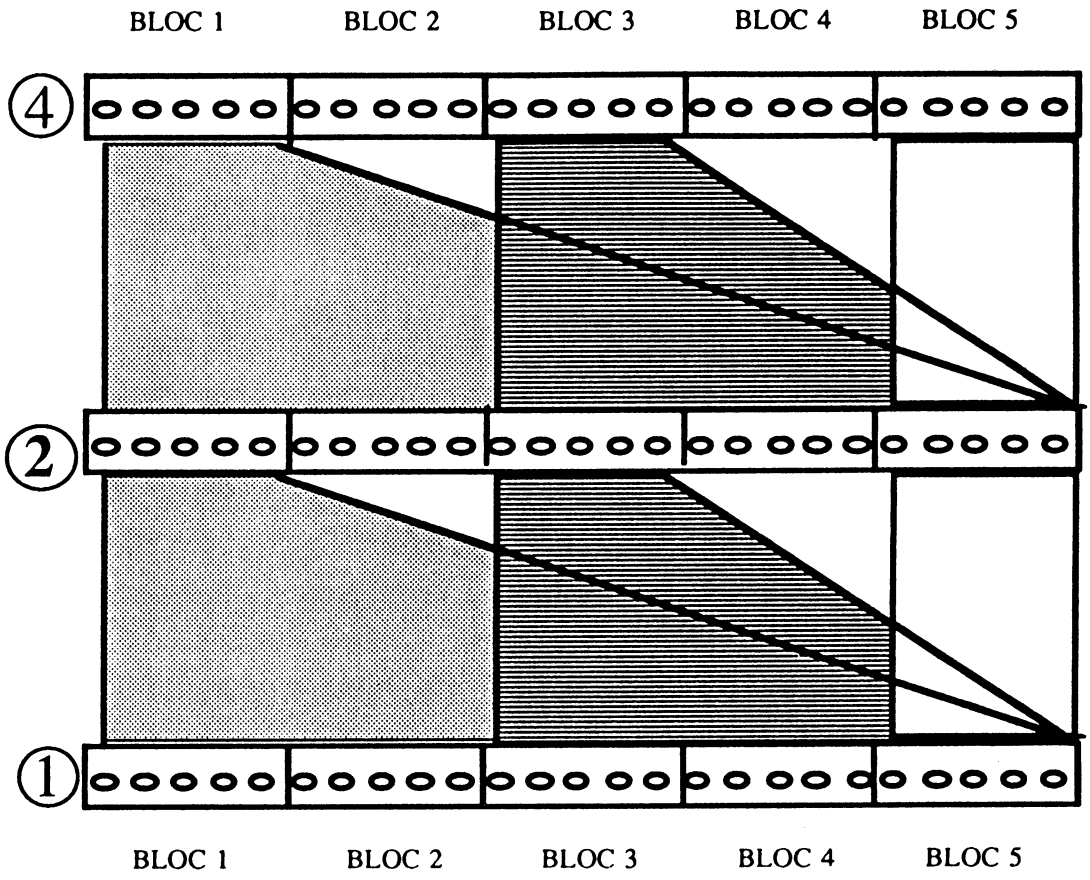


Figure 3.5 : la structure des connexions entre les couches 2 et 4 est strictement identique à celle existant entre les couches 1 et 2. Cette architecture à deux étages de connexions spécifiques permet de faire en sorte que chaque bloc ne soit pas influencé de manière équivalente par tous ceux qui se trouvent à sa droite. En effet, sur l'ensemble de la couche 2, le bloc d'entrée 5 est plus présent que le bloc 4, qui lui-même est plus présent que le bloc 3.... Ceci provient du fait que toutes les cellules de la couche 2 sont influencées par le bloc 5, alors que seulement quatre cinquième d'entre elles reçoivent des connexions du bloc 4, etc. Cette couche 2 servant de relais vers la couche 4, transmet donc les informations de la couche 1 mais en pondérant et hiérarchisant l'influence potentielle des blocs d'entrée dans ceux de sortie.

Cette modélisation de l'influence des traits est ici très systématique pour la simplicité de la simulation et demanderait à être affinée de façon plus discriminative en fonction d'une cartographie précise de l'influence hiérarchique (et réciproque) des traits dans la configuration visage.

- Sortie écho-contexte : le schéma de connexions est identique à celui effectué sur l'information du visage, à savoir un faisceau de connexions totales vers la partie V-C de la couche 2, et un faisceau de connexions spécifiques vers la partie

spécifique-contexte de cette même couche.

NB : l'écho-visage n'est jamais directement influencé par les entrées contextes et réciproquement. C'est toujours au travers des associations V-C, en activant d'autres visages, qu'un contexte peut avoir un rôle sur l'écho-visage (et vice versa).

III.2.2 Processus d'identification des visages en contexte

Le processus complet d'identification comporte deux mécanismes en cascade qui concourent à la décision finale d'identification de la personne à partir du visage : un mécanisme simple de gestion des cycles de réinjection et un mécanisme complexe de décision de reconnaissance. Le premier participe activement à la dynamique propre du réseau, c'est-à-dire à la recherche mnésique contextuelle, par un mécanisme de réinjection en entrée d'une partie de l'information obtenue en sortie. Le second connaît à tout moment l'ensemble des informations issues du processus de réinjection et possède en plus une estimation de la familiarité perceptive (par comparaison écho-entrée perceptive initiale). Il assure une fonction double : prendre la décision finale d'identification ou relancer le processus mnésique sur la base de nouvelles informations. Ce processus complexe d'intégration de nombreuses informations a été partiellement spécifié et n'a donc pas encore été implanté.

Procédure : cycle de réinjection

La procédure "cycle de réinjection" prend en compte 3 informations distinctes : celles concernant les cellules ayant répondues sur la couche d'identité et celles relatives aux sorties écho-visage et écho-contexte. Elle décide, sur la base d'informations calculées, si l'information de sortie doit être reinjectée telle quelle en entrée, ou si la partie contexte doit être atténuée.

* ALGORITHME DE LA PROCEDURE "CYCLE DE REINJECTION"

****

Cette procédure envoie au réseau l'information d'input initiale, décide en fonction des sorties ce qui doit être reinjecté et le représente en entrée. Les booléens qu'elle calcule ainsi que les sorties de la couche 4 sont disponibles à tout moment pour le processus complexe de

décision d'identification.
****

```
PRESENTATION(entrée : vecteur VISAGE-DEPART couche 1,
                      vecteur CONTEXTE-DEPART couche 1;
               sortie : vecteur IDENTITE couche 4,
                      vecteur ECHO-VISAGE couche 4,
                      vecteur ECHO-CONTEXTE couche 4)
        (* Fin de la première itération du réseau *)

TEST_IDENTITE(    entrée :  vecteur IDENTITE;
                sortie : booléen IDENTITE-TROUVEE)

tant que (non (IDENTITE-TROUVEE) )

    TEST_SI_BRUIT_VISAGE (    entrée :    vecteur ECHO-
VISAGE;
                                sortie :  booléen BRUIT-VISAGE )
    si ( non (BRUIT-VISAGE) ) alors
        COMPAR_V (entrée :  vecteur VISAGE-DEPART,
                            vecteur ECHO-VISAGE;
                  sortie :  booléen EGAL-OU-TEND-DEPART-
                            VISAGE )
    finsi
    COMPAR_C (entrée : vecteur CONTEXTE-DEPART,
                       vecteur ECHO-CONTEXTE;
              sortie : booléen EGAL-DEPART-CONTEXTE)

(*---- CALCUL DU CONTEXTE QUI DOIT ETRE REINJECTE ----*)
    si (EGAL-OU-TEND-DEPART-VISAGE et EGAL-DEPART-
        CONTEXTE) alors
        AFFAIBLIR ( entrée :  vecteur ECHO-CONTEXTE;
                    sortie :  vecteur CONTEXTE-CALCULE )
    sinon
        vecteur CONTEXTE-CALCULE=vecteur ECHO-CONTEXTE
    finsi
    (*---------- FIN DU CALCUL ----------*)
```

PRESENTATION (entrée en couche 1 : vecteur ECHO-VISAGE,
vecteur CONTEXTE-CALCULE;
sortie en couche 4 : vecteur IDENTITE,
vecteur ECHO-VISAGE ,
vecteur ECHO-CONTEXTE)
(* Fin de la nouvelle itération du réseau *)

TEST_IDENTITE(entrée : vecteur IDENTITE;
sortie : booléen IDENTITE-TROUVEE)
fin tant que.

* EXPLICATION DETAILLEE DES PROCEDURES APPELEES

- PRESENTATION : fournit un pattern Visage-Contexte en entrée du réseau et récupère les valeurs d'activation de la totalité de la couche de sortie dans 3 vecteurs, après une itération de reconnaissance du réseau.

- TEST-IDENTITE : cette procédure recherche dans le vecteur identité l'existence d'un maximum à la fois absolu et relatif; s'il en existe un, le booléen IDENTITE-TROUVEE prend la valeur vrai, le processus"cycle de réinjection" s'arrêtant alors. Pour cette simulation, le seuil absolu a été empiriquement choisi à 0,98 et le seuil relatif à 1. La valeur du seuil absolu est très élevée pour assurer que la recherche mnésique ne soit arrêtée spontanément que pour les associations Visage-Contexte très bien connues.

Le critère d'émergence d'une identité unique dépendant du nombre d'apprentissage total et de l'étendue de la base apprise, il ne peut être fixé a priori quelle que soit l'histoire du réseau. L'objectif à moyen terme est de trouver une fonction tenant compte de ces deux paramètres qui estime tous les seuils utilisés dans la recherche mnésique.

- TEST-SI-BRUIT-VISAGE : cette procédure teste la forme de l'écho-visage. Sur chaque bloc correspondant à un trait, elle recherche l'existence d'un maximum relatif. Si trois blocs au moins ne possèdent pas de maximum relatif, elle affecte au booléen BRUIT-VISAGE la valeur vrai, ce qui signifie que l'information de sortie n'a pas conservée la forme d'un visage. Comme précédemment, le critère de tolérance au bruit ne peut lui non plus être fixé a priori (choisi ici empiriquement à 0,1).

- COMPARE-VISAGE : elle calcule une distance euclidienne entre le visage d'entrée initial "maintenu en Mémoire à Court Terme" et l'écho-visage. Cette distance représente donc une estimation de familiarité du visage. Si la similitude ainsi calculée dépasse un seuil absolu (choisi ici à 0,60) le booléen EGAL-OU-TEND-DEPART-VISAGE prend la valeur vrai (forte familiarité). Si le visage obtenu est différent (faible familiarité) ou si l'information initiale s'est dégradée en mémoire à court terme, ce booléen prend donc la valeur faux (cette distance n'est pas calculée si BRUIT-VISAGE est vrai car elle n'aurait aucun sens).

- COMPARE-CONTEXTE : calcule aussi une mesure de similitude entrée / sortie, donc estime la familiarité du contexte. EGAL-DEPART-CONTEXTE prendra la valeur vrai si la similitude est supérieure à un seuil absolu (choisi empiriquement à 0,9).

- DECISION DE CE QUI DOIT ETRE REINJECTE EN ENTREE DU SYSTEME : La seule décision possible est une focalisation sur le visage durant la recherche mnésique; autrement, les échos visage et contexte sont reinjectés tels quels en entrée. Le processus de focalisation est simulé par une atténuation de l'information contextuelle : la procédure AFFAIBLIR-CONTEXTE divise chaque valeur du vecteur Echo-Contexte par un réel supérieur à 1.

Cette décision de focalisation n'a lieu que pour une seule combinaison de valeur des booléens : EGAL-OU-TEND-DEPART-VISAGE associé à EGAL-DEPART-CONTEXTE. En effet, dans ce cas, l'écho-visage indique que le visage a été appris (ou qu'il ressemble fortement a un visage appris) et que le contexte est très bien connu par le réseau. Or l'identité n'ayant pas émergée, c'est donc l'association Visage-Contexte qui semble erronée. Dans le cas d'un contexte fort et inadéquat, le processus de recherche inter-contextuelle mis en avant dans les théories de la reconnaissance, ne peut s'effectuer qu'en atténuant le poids du contexte permettant ainsi éventuellement, par le jeu des associations V-C, sa modification vers le contexte d'encodage du visage.

Pour toutes les autres combinaisons de valeurs des booléens, il n'y a pas d'atténuation du contexte, qui est alors reinjecté tel quel :

- premier cas : le visage obtenu ressemble au visage de départ et l'écho-contexte n'est pas égal au contexte d'input : le fait même que le traitement du réseau ait modifié le contexte d'entrée indique que son association avec le visage

était inadéquate, et que le réseau a commencé à le modifier par l'intermédiaire des connexions V-C. Contrairement au cas précédent, l'échec d'identification ne peut pas être attribué seulement au contexte puisqu'il n'est pas assez "familier" pour ressortir tel quel en écho; une focalisation sur le visage dans ce cas serait donc une manipulation purement ad hoc et d'autant plus artificielle que la recherche inter-contextuelle est déjà engagée.

- deuxième et troisième cas : le visage obtenu en sortie est différent du visage d'input : que ce soit par l'intermédiaire d'un jeu de ressemblances (non EGAL-DEPART-CONTEXTE) ou de par la forte influence du contexte (EGAL-DEPART- CONTEXTE), le traitement du réseau a fait converger le visage d'entrée sur un autre visage. Un atténuation du contexte semble aberrante puisque c'est d'abord le visage qui est inconnu. Dans ce cas, la recherche aurait tendance à réaliser une fausse reconnaissance ou à révéler un échec d'identification.

- quatrième cas : si BRUIT-VISAGE : ce cas n'a aucune raison d'être traité différemment des autres pour ce qui est de la procédure interne de réinjection . En effet, il peut représenter un état transitoire. Seul le fait de laisser boucler le système sur ses sorties pourra permettre d'éclaircir sa trajectoire. C'est une capacité caractéristique des réseaux connexionnistes de pouvoir générer et gérer "intelligemment" des patterns bruités, qui peuvent contenir des formes non perceptibles d'informations.

En résumé, cette procédure revêt réellement le caractère de simplicité qui lui est prêté depuis le début. Elle renvoie dans la plupart des cas l'information obtenue et dans un cas seulement elle focalise le visage durant la recherche mnésique. Ceci relève d'une volonté d'intervention minimale dans la dynamique propre du réseau. Cette focalisation reflèterait la capacité des sujets d'inhiber certaines informations contextuelles, qui sont très connues mais non pertinentes dans leur association au visage, lors du processus de recherche inter-contextuelle. Cependant, notre atténuation de la totalité de l'écho-contexte manque à l'évidence de finesse car toutes les composantes du contexte n'entretiennent pas la même relation d'indépendance par rapport à la cible : ainsi, les parties interactives du contexte devraient être moins fortement atténuées que les blocs indépendants.

Processus de décision d'identification :

Bien que le processus complexe de décision d'identification soit actuellement en cours d'élaboration, nous tenterons d'exposer brièvement ses fonctionnalités générales.

Les entrées de ce processus sont d'ores et déjà spécifiées pour ce qui est du traitement du visage en contexte. Il s'agit des trois booléens calculés par la procédure "cycle de réinjection" (BRUIT-VISAGE, EGAL-OU-TEND-DEPART-VISAGE, EGAL-DEPART-CONTEXTE) et des trois sorties du réseau mnésique. Ces six informations sont accessibles par le processus de décision à tout moment, dès qu'elles sont calculées pour chaque cycle de traitement du réseau. Cette disponibilité permet au mécanisme de décision de ne pas avoir à attendre l'arrêt de la procédure "cycle de réinjection" pour débuter son traitement.

Ce processus de décision, relevant de l'identification de la personne, devra aussi pouvoir prendre en compte des données issues d'autres clés d'accès aux noeuds d'identité spécifiques (analyse de la voix, analyse de l'allure générale...)

Les sorties éventuelles sont de deux types : décision explicite concernant l'identité de la personne; décision de relancer une autre recherche mnésique à partir d'une nouvelle prise d'informations perceptives ou représentationnelles. Ces deux sorties sont réellement éventuelles car la procédure de décision peut très bien lasser plusieurs cycles de réinjection si les informations qu'elle intègre ne sont momentanément pas à même de la conduire à fournir une des deux sorties ci-dessus.

* PRINCIPES DE FONCTIONNEMENT DE LA PROCEDURE DE DECISION :

- L'intégration de toutes ces entrées conduit à une réponse implicite concernant l'identité de la personne. Cette fonction d'intégration, de par le nombre d'informations à traiter, est particulièrement complexe à formaliser. Dans un premier temps, les modèles stochastiques de la reconnaissance (Tiberghien et Lecocq, 1983) devraient permettre de déterminer certaines relations entre paramètres.

- A l'issue de cette intégration, la réponse implicite peut être actualisée en réponse explicite d'identification ou de non identification, modulée par une certitude.

- Le fait qu'une réponse explicite ait été fournie ne constitue pas une condition d'arrêt de la recherche mnésique, qui peut continuer pour confirmer ou infirmer cette réponse, ou encore pour faire émerger de nouvelles informations sur la personne ("overshoot effect" : Williams, 1976 dans Linsday et Norman, 1980).

- Enfin, ce processus d'identification peut décider d'engager une recherche d'informations perceptives supplémentaires, ceci bien sûr à condition que le visage soit encore présent.

En résumé, les deux procédures qui concourent au processus d'identification fonctionnent en cascade et sans intervention de type top-down du mécanisme de décision sur les cycles de réinjection qui permettent l'évolution dynamique du réseau. En effet, la recherche mnésique en elle-même n'est jamais influencée par la procédure de décision, sauf indirectement par le fait que l'information d'entrée du réseau peut être modifiée lors d'une nouvelle prise d'information.

Intérêt des réinjections :

Du point de vue mathématique, l'existence du processus interne de réinjection est inspirée de la théorie des systèmes dynamiques. Ces réinjections nécessitent naturellement que l'entrée et la sortie du système aient le même type de représentations. Nous avons utilisé la stratégie des réinjections pour permettre au réseau de converger vers un état favorable en vue d'une décision d'identification. Or dans la plupart des cas, cet état favorable est en fait un état stable au sens de la théorie des systèmes dynamiques. Dans le cas présent, la pertinence des réinjections est étayée par les trois considérations suivantes :

1) Au cours de l'apprentissage, le réseau a appris à faire des correspondances à l'identique entre ses entrées et une partie de ses sorties. Le réseau est donc construit pour reproduire, lors de la reconnaissance, ses entrées dans les deux échos de sortie (du moins quand il s'agit de patterns appris). De ce fait, la réinjection des échos visage et contexte est porteuse de sens. On peut alors très bien interpréter les parties "latérales" de l'architecture, consacrées à la reproduction des entrées en sortie, en tant que constitutives d'un système

dynamique. De plus, les résultats présentés au Chapitre I, concernant la tendance à la création d'attracteurs du système par l'algorithme GBP, justifient l'utilisation des réinjections. Il semble donc tout à fait logique d'employer le feedback pour émuler la recherche mnésique.

2) Ces parties "latérales" ont un effet direct sur la fonction "centrale" du réseau qui est l'identification. En effet, le cheminement des informations conduisant à l'identité intègre des informations relatives au visage et à l'association Visage-Contexte au niveau de la couche 2. De plus, les échos visage et contexte sont eux-mêmes partiellement sous la dépendance de l'association V-C. Donc, au cours des réinjections, si les parties latérales du réseau arrivent à rappeler un visage et un contexte adéquat (grâce aux associations V-C), il y a alors de fortes probabilités pour qu'elles imposent, dans la couche 2, des représentations V,C et V-C semblables à celles engendrées au cours de l'apprentissage et conduisent ainsi à l'identification correspondante.

Ce processus de réinjection permettra donc d'étudier la dynamique du réseau multi-couches, mais aussi de simuler (voire même d'émuler ?) l'évolution du mécanisme de recherche mnésique à l'oeuvre lors de l'identification d'une personne à partir de son visage.

III.3. Simulation et résultats

A partir du modèle cognitif de Bruce et Young dont les fonctionnalités ont été retranscrites dans une architecture connexionniste, nous allons étudier explicitement les interactions entre le visage et le contexte dans la construction de l'identité et dans la recherche mnésique. Des indices, quantitatifs et qualitatifs, de la dynamique des effets de contexte intervenant dans le processus d'identification peuvent ainsi être obtenus.

Les premiers tests effectués sur FACENET nous ont conduit à manipuler deux variables expérimentales relatives à la nature de la relation entre un visage et ses contextes d'apprentissage. Nous étudierons donc l'effet de ces différentes modalités d'encodage sur la capacité ultérieure de reconnaissance des visages par le système, ainsi que l'évolution dynamique du réseau au cours des réinjections.

III.3.1 Variables étudiées et procédure expérimentale

Le premier facteur manipulé est la **variabilité** des contextes associés à un visage lors de l'apprentissage. Ce facteur peut revêtir un rôle critique quant à la sémantisation des épisodes en mémoire : en effet, la variabilité contextuelle semble être à l'origine du passage d'une représentation de type épisodique (faits personnellement vécus) à une représentation sémantique (concepts) (Tulving, 1983). On a pu montrer expérimentalement que la reconnaissance d'un visage appris dans des contextes variés est moins sensible au changement de contexte que celle d'un visage encodé plusieurs fois avec un contexte identique (Tiberghien, 1986). Le premier a ainsi pu être abstrait et acquérir une représentation propre indépendante des fluctuations contextuelles, alors que le second est encore lié à son contexte dans un même épisode.

Le second facteur est la **spécificité** du contexte d'encodage : les visages pourront donc être appris dans des contextes qui leurs sont spécifiques ou qu'ils partagent avec d'autres visages. A notre connaissance, cette variable n'a jamais été étudiée dans la reconnaissance des visages, mais comme elle constitue, pour les contextes, le symétrique du premier facteur, elle peut se montrer pertinente. De plus, un phénomène de "fan effect" (propagation diffuse de l'activation à travers les noeuds d'un réseau d'associations sémantiques) a été mis en évidence dans les études sur l'organisation de la mémoire permanente (Anderson, 1983).

Nos hypothèses opérationnelles sont donc :

- L'identification d'un visage encodé dans des contextes variables sera moins perturbée par un changement de contexte lors de la reconnaissance que celle d'un visage appris plusieurs fois dans un contexte identique.
- L'identification d'un visage encodé avec un contexte spécifique sera moins gênée par un changement de contexte que celle d'un visage appris dans un contexte non spécifique.
- hypothèse d'interaction : le facteur "spécificité du contexte d'encodage" n'aura pas les mêmes effets selon les modalités du facteur "variabilité " : si les visages appris dans des contextes variables ont réellement réussi à s'abstraire des influences contextuelles, ils seront les moins gênés par la non spécificité de leurs contextes d'encodage.

La variable dépendante qui nous servira d'indicateur de reconnaissance pour tester ces hypothèses est la valeur d'activation de la cellule qui correspond au visage sur la couche de sortie d'identité (appelée dans ce qui suit : force de l'identité).

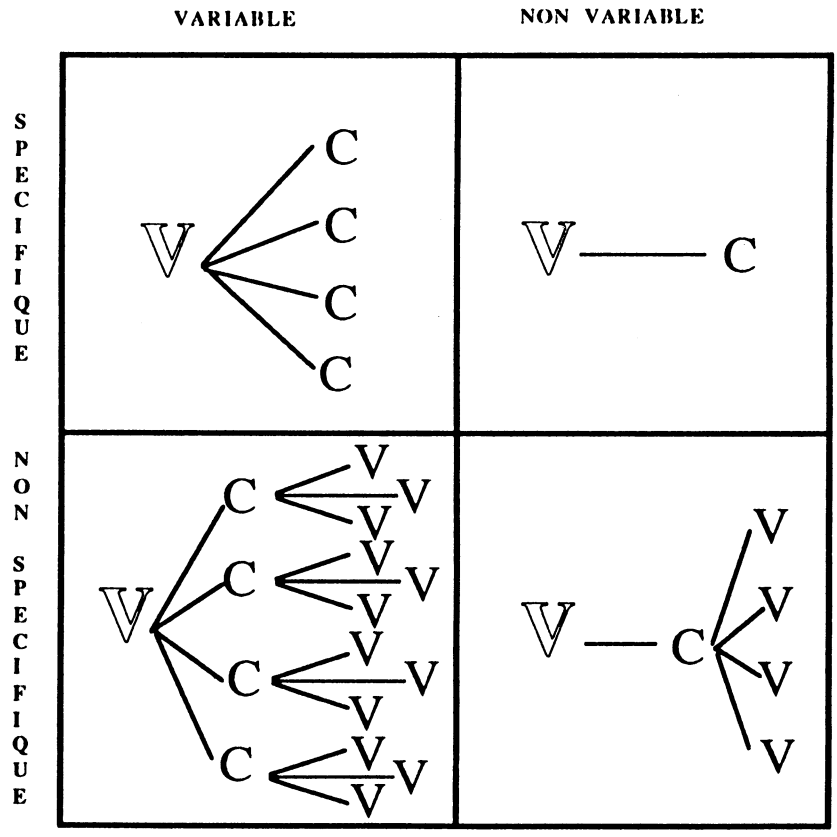


Figure 3.6 : Les quatre conditions d'encodage.

Dans la procédure expérimentale, les deux facteurs indépendants sont croisés, ce qui crée quatre conditions d'apprentissage inter-visages :

1- Visage associé à des contextes variables et spécifiques.
2- Visage associé à des contextes variables et non spécifiques.
3- Visage associé à un contexte non variable et spécifique.
4- Visage associé à un contexte non variable et non spécifique.

La Figure 3.6 illustre les types de rapports entre visages et contextes selon les quatre conditions d'encodage. Ceci devrait permettre de "concrétiser" les facteurs variabilité et spécificité, concepts clés de notre étude.

Chaque condition d'encodage comporte 4 visages, dont 2 sont appris deux fois et 2 sont appris quatre fois par itération d'apprentissage. Chacun des visages a donc un "jumeau" qui est appris exactement dans les mêmes conditions que lui (pour contrôler des effets éventuels de codage). La différence dans le nombre d'apprentissages a été introduite afin d'étudier ponctuellement l'effet des deux facteurs d'encodage à deux moments de l'apprentissage.

La base utilisée pour cette simulation comprend 30 "visages" et 30 "contextes", qui peuvent dans les deux cas se ressembler sur 0, 1 ou 2 blocs. 16 visages sont donc répartis (de façon comparable quant à leur ressemblance) parmi les 4 conditions d'encodage, les 14 autres (buffers) étant appris de manière à conférer à certains contextes leur caractéristique de non spécificité : il faut en effet que les contextes non spécifiques soient vus avec plusieurs visages différents. Tous les visages qui seront à comparer ultérieurement sont bien sûr encodés le même nombre de fois. Au cours de chaque itération d'apprentissage, 139 patterns seront présentés au réseau, étant entendu qu'un pattern correspond à l'association d'un visage et d'un contexte.

III.3.2 L'apprentissage des visages en contexte

L'apprentissage a été effectué sur 139 patterns dont 88 sont différents. Il y a donc 51 patterns dupliqués pour que les visages encodés dans la condition "non variable" soient vus aussi souvent que ceux appris en condition "variable".

Nous avons choisi le mode séquentiel d'apprentissage (au lieu du mode aléatoire). L'ordre de présentation a été fixé au hasard, tout en évitant que les

patterns dupliqués ne soient trop proches sur la liste. Pour cette application, les paramètres d'apprentissage sont les suivants :

- Les poids des connexions ont été initialisés au hasard entre -0,5 et +0,5.
- Le paramètre α_i, qui caractérise chaque fonction sigmoïde (cf Chapitre I), a été initialisé au hasard entre 0,5 et 1 pour chaque cellule. Une fois fixés, ces paramètres a_i sont stockés dans un fichier où ils restent disponibles pour les tests du réseau dans la phase de reconnaissance.
- Le momentum, qui sert à définir la proportion du gradient précédent par rapport au gradient actuel, a été fixé ici à 0,01.
- Le taux d'apprentissage, qui permet de définir le pas de modification dans la direction du gradient a été fixé à 0,3.

Le réseau a été soumis à 1000 itérations d'apprentissage constituées chacune par une présentation de la liste des 139 patterns. Nous avons enregistré le taux d'erreur de chaque pattern toutes les 10 itérations, puis seulement toutes les 100 itérations à partir de la centième. L'apprentissage a été arrêté au bout de 1000 itérations, car au vu des erreurs enregistrées, tous les patterns pouvaient être considérés comme très bien appris : le réseau était alors arrivé à converger vers une configuration stable de poids de connexions qui minimise sa fonction de coût total.

La Figure 3.7 illustre l'évolution des erreurs calculées sur trois patterns différents au cours de l'apprentissage :

- Le pattern A montre un apprentissage très rapide sur les premières itérations et correspond à l'évolution des erreurs recueillies dans la majorité des cas.
- Le pattern B, bien qu'étant présenté le même nombre de fois, ne trouve réellement sa place qu'après 70 itérations. On peut remarquer qu'il perturbe alors légèrement le pattern A, qui commençait à être très bien appris. Ce type d'évolution a été retrouvé sur une dizaine de patterns.
- Le pattern C montre un apprentissage très difficile (un seul cas). Il ne décroche qu'entre les itérations 700 et 800, entraînant alors une très légère perturbation de certains autres patterns (non visible sur la figure).

Quelle que soit la rapidité de l'apprentissage, tous les patterns sont très bien appris par le réseau après 1000 itérations. Etant donné que la rapidité d'apprentissage dépend de nombreux facteurs aléatoires (initialisation du poids

des connexions au départ, ordre de présentation, séparabilité des patterns...), nous n'avons pas étudié l'effet de nos deux variables expérimentales sur la vitesse de l'apprentissage. Nous étudierons ces effets lors de la reconnaissance. Compte tenu du fait que tous les patterns à comparer sont appris au même niveau, les tests au changement de contexte devraient alors être révélateurs de la façon dont le réseau a structuré ses informations pour les apprendre.

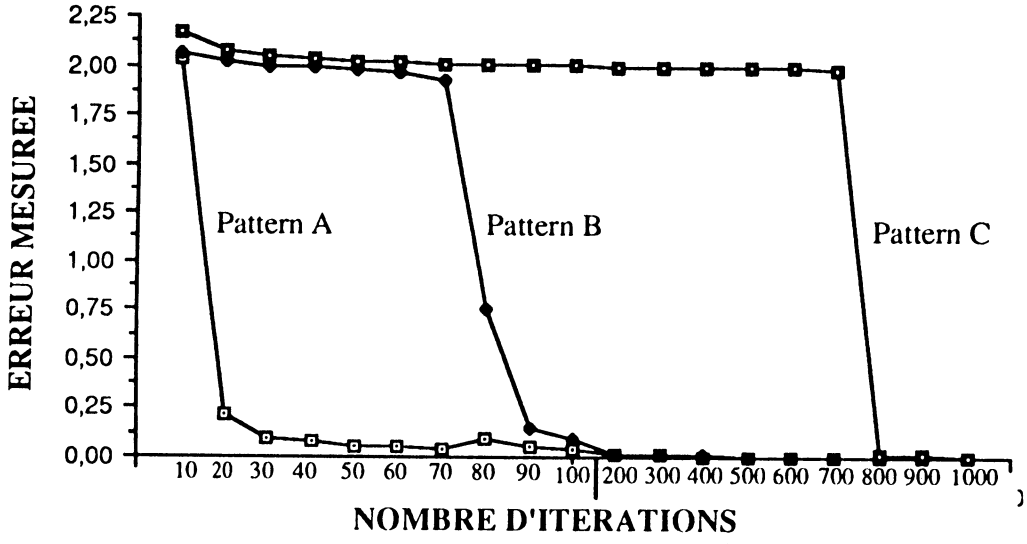


Figure 3.7 : Evolution du taux d'erreur sur 1000 itérations d'apprentissage, pour trois patterns types.

III.3.3 Tests des facteurs manipulés : Variabilité et Spécificité

Nous allons donc étudier l'effet de nos deux variables expérimentales en prenant comme indicateur de la performance mnésique la force de l'identité du visage testé. Ces deux facteurs intervenant au moment de l'apprentissage, les tests de reconnaissance effectués devraient donc être révélateurs de la manière dont le réseau a intégré et structuré ses connaissances au cours de l'encodage.

Reconnaissance dans leur contexte d'encodage :

Cette condition de test doit nous permettre de vérifier que tous les visages que nous comparerons par la suite sont équivalents quant à leur niveau d'apprentissage et que tous les patterns sont égaux quelle que soit la nature de leur relation V-C. Le test de reconnaissance de tous les visages dans leur(s)

contexte(s) d'encodage a donné les résultats suivants :

-Visages vus 4 fois par itération d'apprentissage :

IDENTITE (moy) = 0,982 (+ - 0,005), sans répartition différentielle selon les modalités d'encodage.

- Visages vus 2 fois par itération d'apprentissage :

IDENTITE (moy) = 0,97 (+ - 0,01), sans répartition différentielle selon les modalités d'encodage.

Les résultats semblent donc indiquer que les visages sont équivalents, pour un même nombre d'apprentissages, et il est à noter que même à ce niveau d'apprentissage très fort (1000 itérations), on constate un très léger avantage pour les visages vus 4 fois (à la restriction près que toute analyse statistique est exclue, vu les échantillons par condition). Donc, pour les visages vus le même nombre de fois, le degré d'apprentissage d'un pattern particulier n'interfèrera pas avec les variables manipulées.

Reconnaissance dans un contexte réappareillé :

Ce test permettra d'étudier l'effet des deux variables d'encodage sur la force de l'identité, lorsque le contexte de reconnaissance est déjà connu par le réseau mais n'a jamais été appris en association avec le visage. Pour ce faire, 15 contextes remplissant cette condition ont été choisis au hasard parmi ceux appris (contextes des visages jumeaux et buffers) et serviront pour tous les visages à tester. Ceci nous a permis d'envisager une analyse quantitative des résultats en effectuant une analyse de variance. Cette étude a été effectuée avec les visages appris 4 fois et sur les données d'identité recueillies sur la couche de sortie (lors de la première itération de reconnaissance).

- TABLE DE L'ANALYSE DE VARIANCE :

SOURCE	SC	DL	MC	F	P
VARIABILITE	1.386	1	1.386	10.605	.002
SPECIFICITE	.793	1	.793	6.147	.015
VARIA * SPECI	.616	1	.616	4.775	.031
ERREUR	7.205	56	.129		

Légende :
- SC : Somme des Carrés
- DL : Degré de Liberté
- MC : Moyenne des Carrés = variance inter (SC / DL)
- F : variance inter / variance intra (MC / ERREUR)
- P : Probabilité de rejeter l'hypothèse nulle à tord

- TABLEAU DES MOYENNES :

		ENCODAGE	
		Variable	Non variable
ENCODAGE	spécifique	0,97	0,87
		(0,02)	(0,13)
	Non spécifique	0,94	0,43
		(0,08)	(0,70)

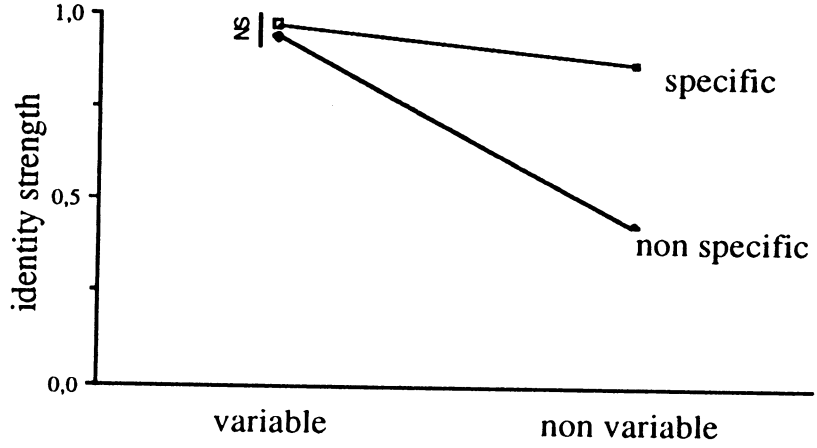


Figure 3.8 : Moyenne des forces d'identité recueillies après la première itération dans la condition ancien contexte réappareillé.

- INTERPRETATION

L'hypothèse de variabilité est vérifiée : en effet, les visages encodés dans des contextes variables sont moins perturbés par un changement de contexte ancien que ceux étudiés dans un contexte toujours identique ($F(1,56) = 10.6$, $p = .002$). Il semblerait donc que la variabilité du contexte durant l'apprentissage ait permis au réseau d'abstraire ces visages des influences contextuelles dans la représentation de l'identité.

L'hypothèse de spécificité est aussi vérifiée ($F(1,56)=6.14$, $p=0.015$) : le fait que le(s) contexte(s) d'apprentissage d'un visage soit associé à d'autres visages semble avoir gêné l'émergence de l 'identité lors de la première itération de reconnaissance.

Enfin, il existe une interaction entre ces deux facteurs : la non spécificité a beaucoup plus gêné les visages encodés de façon non variables ($F(1,56)=4.77$, $p=0.031$). Il est à noter cependant que cette interaction met encore en évidence le rôle prépondérant de la variabilité, car même lorsque les contextes d'encodage sont spécifiques, l'effet de la variabilité est important (0.97 versus 0.87, $T(28)=2.83$, $p=0.008$).

Cette analyse quantitative concernée uniquement par les résultats de la première itération permet de conclure que la variabilité des contextes à l'encodage semble jouer un rôle primordial dans l'intégration des informations par le réseau (abstraction de la représentation de l'identité du visage à partir de ses nombreux contextes). Les résultats relatifs à l'interaction paraissent corroborer cette interprétation. En effet, si la variabilité permet réellement d'abstraire progressivement une identité au visage, on peut comprendre alors qu'à ce degré d'apprentissage fort, on ne retrouve aucune différence entre contexte d'encodage spécifique ou non spécifique. Au contraire, les visages non variables, qui eux sont directement tributaires des informations contextuelles pour leur identité, se montrent d'autant plus sensibles au changement de contexte qu'ils ont été appris avec un contexte non spécifique.

La non variabilité contextuelle induit une représentation d'identité complètement confondue avec le pattern Visage-Contexte appris. Pour avoir une forte sortie d'identité, cette association épisodique V-C unitaire (non séparable) doit être préservée (ou retrouvée). Puisque le réseaux n'a pas pu créer une représentation d'identité indépendante de cette association particulière, tout changement contextuel induira une baisse de performance. Cette interprétation vaut pour tous les visages non variables.

Mais comment expliquer l'effet de spécificité concernant les visages non variables?

- Quand un visage est encodé dans un contexte spécifique, le réseau a appris à donner une identité à cette association particulière, mais les deux composantes n'ont jamais été associées à d'autre identité. Donc la trace épisodique est très distinctive pour le réseau.

- Les visages encodés dans des contextes non spécifiques sont encore plus perturbés par un changement contextuel à la reconnaissance. En effet, les contextes non spécifiques ont été appris à plusieurs reprises avec des visages différents. Ils ont ainsi pu devenir de réels "noeuds attractifs", autour desquels les visages sont organisés. Les contextes non spécifiques ont donc créé des catégories, chaque catégorie contenant, bien sûr, l'identité de tous les visages partageant le même contexte. Par conséquent, puisque les contextes non spécifiques sont à la base d'un processus de catégorisation qui organise l'information dans le réseau, ils jouent un grand rôle dans la création de l'identité. Et ainsi, l'identification de visages non variables et non spécifiques est

très sensible au changement contextuel pendant la reconnaissance.

Reconnaissance dans un contexte nouveau :

Pour trouver certains indices corrobarant ces interprètations, nous avons étudié qualitativement l'effet de nos variables au cours de l'évolution du processus de recherche mnésique, c'est-à-dire sur plusieurs itérations du réseau. Pour analyser la dynamique de reconnaissance, nous avons fixé le seuil d'IDENTITE-TROUVE à 1, afin que les réinjections se poursuivent quel que soit le résultat sur l'identité. Ainsi, tous les visages sont testés suivant la même procédure : ils subissent chacun 15 itérations de reconnaissance et les réinjections se déroulent bien sûr sans focalisation puisque le contexte de reconnaissance est nouveau pour le réseau. Le sytème bouclera donc simplement sur ses sorties .

ETUDE DU FACTEUR "VARIABILITE DE L'ENCODAGE" LORSQUE LES CONTEXTES D'APPRENTISSAGE SONT SPECIFIQUES

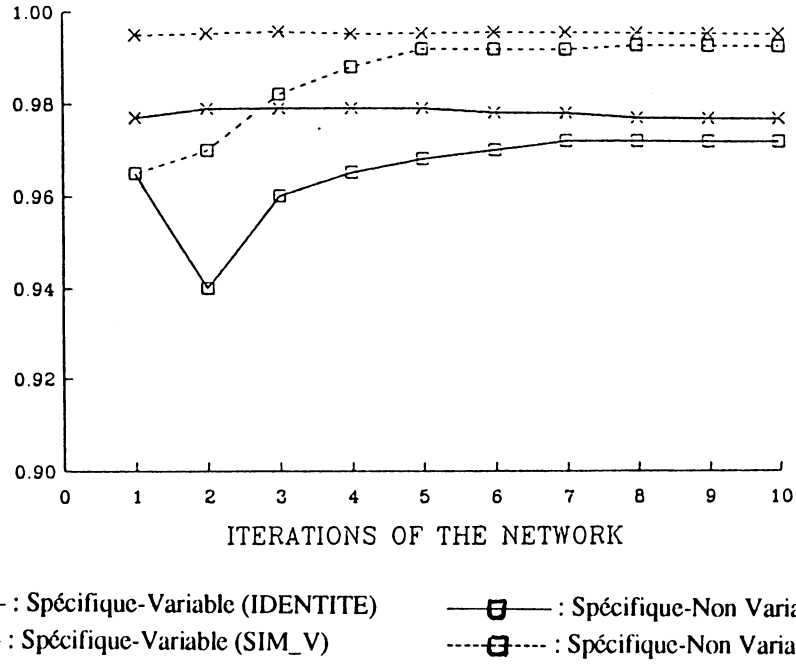


Figure 3.9 : Evolution sur 10 itérations, de l'Identité d'un visage encodé en "Spécifique-Variable" et de l'Identité d'un visage encodé en "Spécifique-Non Variable", lorsque le contexte de reconnaissance est nouveau. Les traits pointillés représentent la Similitude-Visage calculée entre l'écho et le visage initial. Les traits pleins représentent la force de l'Identité recueillie sur la couche 4.

NB : la Similitude et l'Identité sont ici superposées par rapport au même axe de coordonnées uniquement pour comparer la forme de leurs évolutions et non leurs valeurs respectives.

La Figure 3.9 illustre l'effet de la variabilité (à spécificité constante) sur la force de l'identité et sur la similitude entre l'écho-visage et le visage d'entrée. Nous retrouvons ici, sur l'évolution du réseau, l'effet de la variabilité du contexte d'encodage déjà indiqué par l'analyse quantitative effectuée dans la condition précédente :

- Un visage appris dans des contextes multiples et spécifiques se montre insensible à la perturbation introduite par un contexte nouveau lors de la reconnaissance (et ceci tant sur la similitude que sur l'identité).
- Un visage appris dans un contexte toujours identique et spécifique semble gêné par une information contextuelle nouvelle, et ce tant sur la force de l'identité que sur la similitude-visage. Etudions en détail et en parallèle l'évolution de ces 2 indicateurs :

Au cours de la première itération, l'association apprise V-C est perturbée, ce qui retentit sur la force de l'identité et sur l'écho-visage obtenu.

Lors de la seconde itération, les informations réinjectées sont moins pures pour le visage, avec un début de modification du contexte dans le sens de celui réellement encodé avec ce visage. Ceci explique l'inflexion de la courbe d'identité, car l'association V-C est encore trop incomplète et le visage est légèrement bruité.

Au cours des réinjections, cette association V-C se complètera peu à peu en liaison avec le rappel du contexte d'apprentissage, l'identité augmentant alors progressivement. Il faut en outre souligner que la progression de la similitude visage est régulière, ce qui signifie que l'écho-visage n'est jamais perturbé par la récupération du contexte d'encodage permise par l'association V-C. En effet, cette association est encodée de façon distinctive pour le réseau, et ne risque pas d'en activer d'autres. Cette évolution semble donc confirmer que la représentation de l'identité est dans ce cas inextricable de l'épisode Visage-Contexte encodé.

Si un visage appris dans un contexte spécifique non variable semble gêné par un nouveau contexte, il l'est pourtant beaucoup moins que les visages appris dans un contexte non spécifique et non variable, comme montre par le test suivant.

L'effet du facteur "variabilité de l'encodage" (Figure 3.10) sur la force de l'identité et sur la similitude-visage est encore ici manifeste pour des contextes d'apprentissage non spécifiques. Un visage encodé avec des contextes multiples et

non spécifiques se montre insensible au changement de contexte, sur ces deux indicateurs. Il n'en est pas de même pour un visage encodé de façon non variable. Dans le cas d'un visage "non spécifique - non variable", la deuxième itération est très intéressante car nous assistons à une progression de l'identité qui n'est pas attribuable au visage. En effet, au même moment, la similarité visage a diminué. Seul l'echo contexte réinjecté est responsable de cette progression (accès à sa catégorie). Mais pendant les itérations 3 et 4, il y a une chute simultanée de l'identité et de la simularité, qui symbolise un phénomène de "fan-effect". Le visage a activé son contexte d'encodage, mais puisque ce contexte est associé à d'autres visages, l'écho visage est de plus en plus perturbé, parce que le réseau active plusieurs visages en même temps (et d'autre identités émergent). A partir de la cinquième itération, le réseau est finalment capable de choisir, parmi toutes les associations activés, celle qui correspond au visage présenté en entrée.

ETUDE DU FACTEUR "VARIABILITE DE L'ENCODAGE" LORSQUE LES CONTEXTES D'APPRENTISSAGE SONT NON SPECIFIQUES

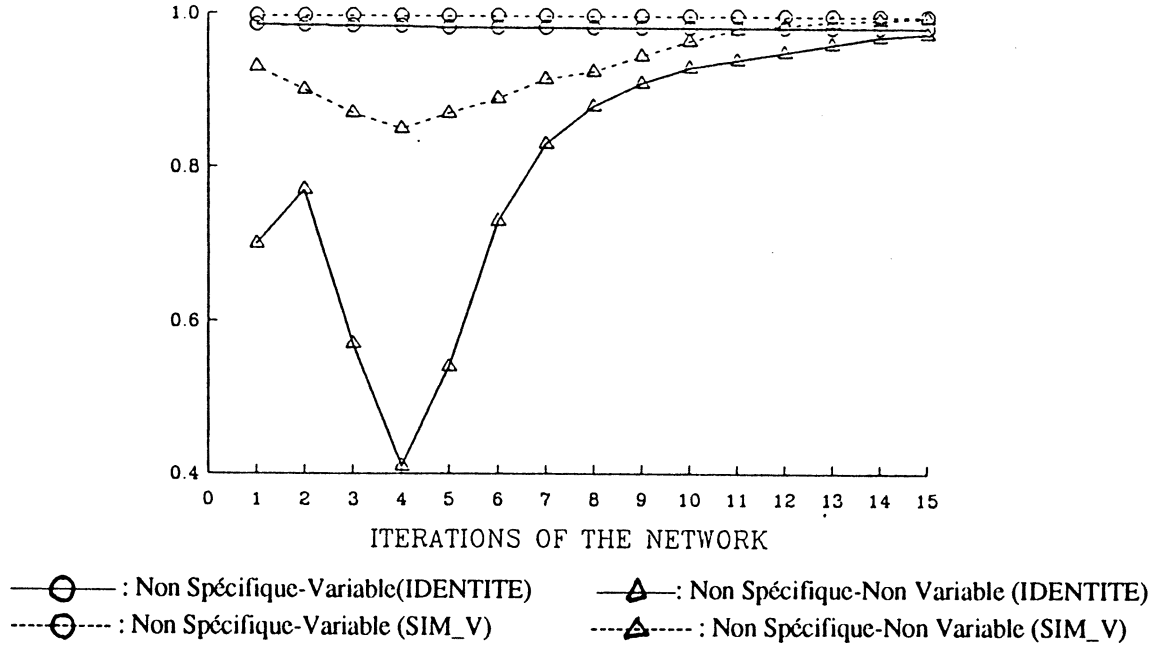


Figure 3.10 : Evolution de l'Identité d'un visage encodé en "Non Spécifique-Variable" et de celle d'un visage encodé en "Non Spécifique-Non Variable", lorsque le contexte de reconnaissance est nouveau.

Il faut souligner que les phénomènes présentés dans les deux figures ci-dessus sont généraux pour tous les visages et les contextes soumis au test, qui satisfont les mêmes contiditions.

En conclusion, le contexte semble bien posséder ici un statut particulier dans la représentation de l'identité. Il est à la base d'un phénomène de catégorisation qui peut jouer un rôle important dans le processus de recherche d'identification : lors d'une perturbation contextuelle, le visage semble pouvoir accéder assez facilement à sa catégorie (voir à plusieurs interdépendantes), ce qui fait augmenter provisoirement son identité. Mais une fois la catégorie activée, un phénomène de "fan-effect" propage l'activation sur les autres membres de cette catégorie. L'identification réussie nécessite alors une différenciation intra-catégorielle, qui ne peut s'avérer possible que si l'information visage est relativement bonne.

Conclusion générale sur les facteurs étudiés :

L'analyse quantitative et les études qualitatives portant sur l'évolution de la recherche au cours des itérations ont mis en évidence le rôle fondamental de la forme de l'encodage sur la représentation mnésique de l'identité. L'efficience du processus de reconnaissance en condition de contexte modifié se montre en effet différente selon les rapports entretenus entre un visage et son contexte à l'apprentissage :

La variabilité contextuelle lors de l'encodage engendre une représentation de l'identité qui s'est abstraite progressivement des épisodes. Cependant, les épisodes particuliers ne semblent pas détruits par ce processus d'abstraction de l'identité, puisqu'en laissant la recherche continuer après l'identification, le réseau peut rappeler une bonne partie des informations contextuelles associées à un visage appris dans des conditions variables. Donc les contextes, dans ces cas, n'ont pas été considérés comme du bruit par le réseau.

Mais qu'est-ce que l'identité pour un visage encodé dans des conditions toujours identiques ? Elle ne peut être envisagée qu'en tant que somme de deux informations liées dans un même épisode :

- Si le contexte d'encodage est spécifique, le visage et le contexte ont le même poids dans l'association V-C qui fonde l'identité. Dans des contextes de reconnaissance erronés, il faut alors que l'intégralité de cette association distinctive soit retrouvée pour qu'émerge une forte valeur d'identité.

- Si le contexte d'encodage est non spécifique, de par ses présentations répétées avec différents visages, il a pu avoir un rôle central d'organisateur de ses associations V-C en catégorie. Mais en aucun cas cette catégorie n'a pu s'abstraire telle quelle dans la représentation d'identité, puisque le réseau a été

forcé de répondre de manière différenciée en fonction des membres de celle-ci. Dans des conditions de reconnaissance avec un contexte erroné, un visage "non variable et non spécifique" doit essayer d'accéder à l'épisode qui représente son pattern d'encodage, en raison même de sa non variabilité. Si le visage réussit à activer sa catégorie, il se produirait alors un phénomène de "fan-effect" qui diffuse l'activation sur les autres visages de la catégorie, ce qui affaiblit sa valeur d'identité et peut momentanément en faire émerger d'autres. Seule la recherche intra-catégorielle de la "bonne association" V-C, guidée par l'information visage, permettra d'aboutir à une seule identité.

III.3.4 Représentations internes de l'identité des visages

On a montré ci-dessus les résultats de la structuration des identités à travers l'étude de la force d'identité. Maintenant, nous alons essayer d'entrer dans le réseau ("la boîte noire") pour regarder comment il a fait pour abstraire l'identité des visages encodés avec des contextes variables (sémantisation des épisodes) et nous comparerons les représentations des visages de différentes modalités d'encodage.

Nous allons analyser les représentations internes dans la couche 3 pour vérifier si le réseau a vraiment abstrait l'identité. La raison pour laquelle on examine cette couche est simple : Les cellules (dans la partie d'identification de la couche de sortie) qui donnent la force d'identité sont connectées seulement aux cellules de la couche 3; Donc la couche 3 contient toutes les informations qui sont à la base de la génération de la force d'identité, de par sa taille réduite, renferme les représentations d'identité les plus généralisées.

Nous remarquons que pour un visage en contextes variables, les représentations internes ont toujours la même forme quel que soit le contexte d'encodage. La Figure 3.11 illustre cette invariance du pattern d'activation dans la couche 3 pour un visage variable présenté dans ses contextes d'encodage (avec seulement quelques micro-variations dues aux différents contextes présentés). A ce niveau-là, le réseau a donc pu construire (ou abstraire) une forme de représentation générale à partir des variations contextuelles.

Il reste à définir le rôle des contextes dans ce cas. Pour clarifier ce point, nous avons examiné les représentations internes en l'absence

d'informations V-C qui viennent de la couche 2 : c'est-à-dire que l'on coupe les connexions reliant la couche 3 et la partie V-C de la couche 2, pendant la phase de reconnaissance. De cette manière la couche 3 ne reçoit aucune information contextuelle pour calculer le pattern d'activation. La Figure 3.12 illustre le pattern d'activation ainsi obtenu, qui correspond au visage ci-dessus. Il est différent de ceux qui sont obtenus en présence des contextes d'encodage sur les cellules 7, 9, 10, 13, 14, 15, 20. Une analyse suplémentaire des poids de connexion entre la couche 3 et la couche 4 montre que les poids des cellules où les plus fortes différences sont observées sont tout à fait comparables avec les autres (Figure 3.13). Nous pouvons conclure donc que le réseau ne prend pas les contextes d'encodage comme du bruit. Il tient compte de la variabilité contextuelle et a réussi à abstaire une représentation sémantique (généralisée) de cette variabilité. Pour les visages non-variables, leur identité est très liée aux informations contextuelles.

L'influence de la modalité d'encodage des visages sur la structuration de l'identité est donc confirmée par les représentations internes au niveau de la couche 3. Il faut ajouter que les résultats ci-dessus montrent aussi la capacité du réseau de résistance aux "pannes". Mais il faut sûrement des tests beaucoup plus raffinés pour voir l'évolution de la performance du système lorsqu'il est progressivement détérioré.

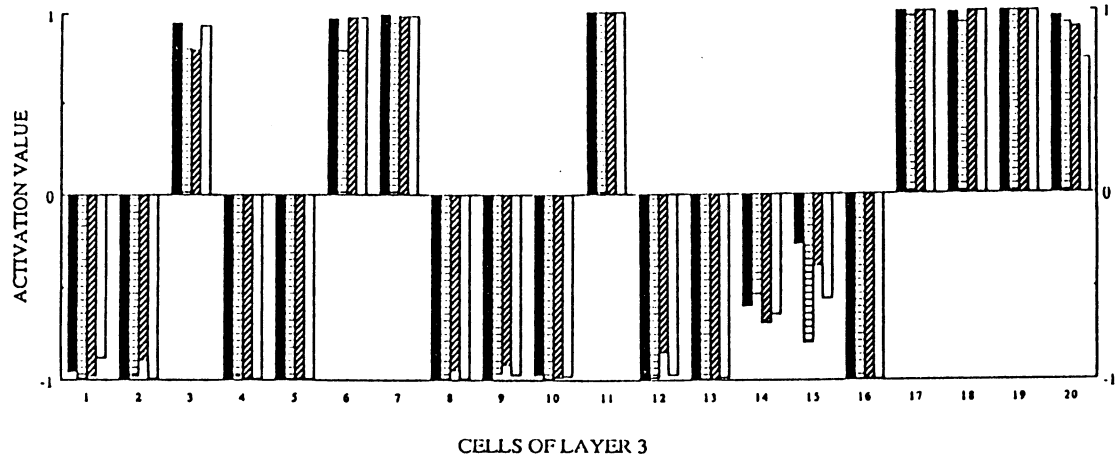


Figure 3.11 : Représentations internes dans la couche 3. Les patterns d'activation sont obtenus par présentation, lors de la reconnaissance, d'un visage variable présenté successivement dans ses quatre contextes d'encodage.

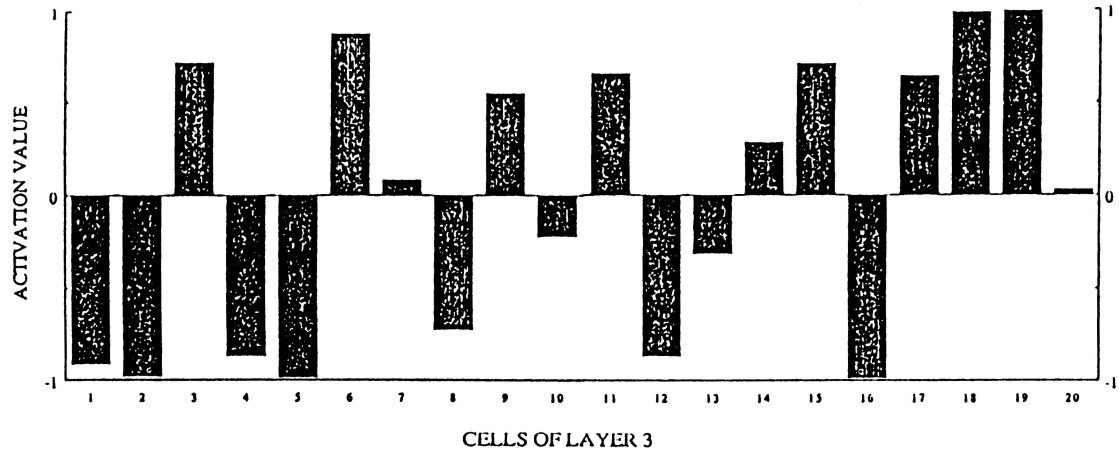


Figure 3.12 : Représentations internes dans la couche 3. Le pattern d'activation est obtenu par la présentation du même visage que celui de la Figure 3.11, quand les connexions entre la couche 3 et la partie VC de la couche 2 soit coupées.

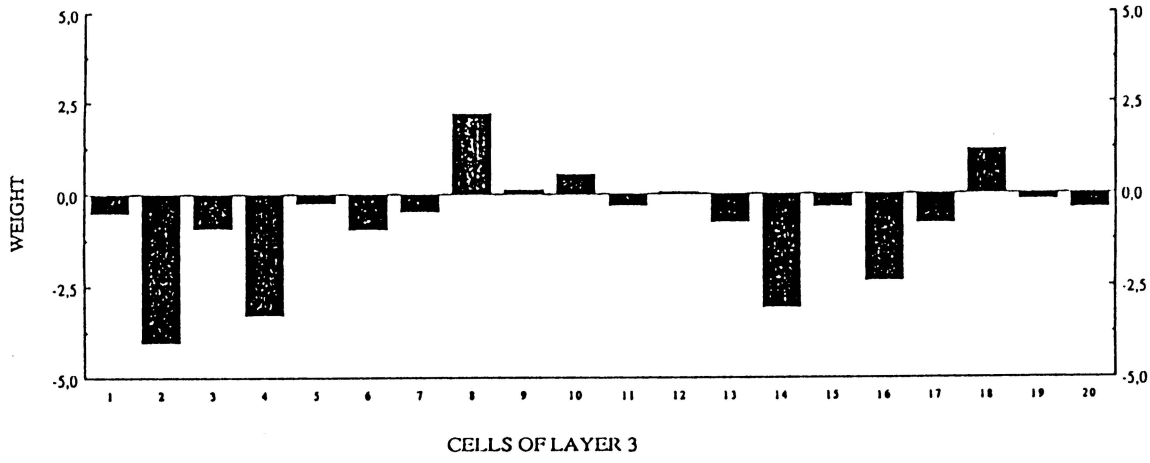


Figure 3.13 : Pattern des poids des connexions reliant les cellules de la couche 3 et la cellule (de la couche 4) d'identité du visage utilisé dans les Figures 3.11 & 3.12.

Il est clair que dans les représentations internes réside une richesse d'information qui n'est pas encore très bien exploitable. Les interprétations de ces représentations sont très difficiles à cause de leur mode distribué. Comme souligné au chapitre I, la résolution de ce problème critique sera le pavé de la voie vers la conciliation des deux principales approches en I.A. .

III.3.5 Tests de la dynamique de la recherche

Les résultats décrits dans le paragraphe III.3.3 proviennent aussi des tests de la dynamique de la recherche, mais ils sont centrés autour de la performance d'identification. Nous nous "focaliserons" désormais sur la forme des échos visage et contexte lors de la recherche de l'identité. Nous étudierons ainsi la pertinence de notre architecture (association V-C, connexions spécifiques, asymétrie) et nous testerons les capacités induites par le processus des réinjections.

Le système sera alors confronté à des visages nouveaux (pour étudier la possibilité de fausses reconnaissances), à des visages hyper-bruités (3 nez, 4 bouches...), et à des informations très affaiblies (valeurs d'activation des entrées divisées par quatre). Nous analyserons plus particulièrement ici sa capacité de rappel associatif, en présentant des visages sans contexte et des contextes sans visage, ainsi que sa capacité de recherche inter-contextuelle (mécanisme supposé être à la base du processus lent de reconnaissance).

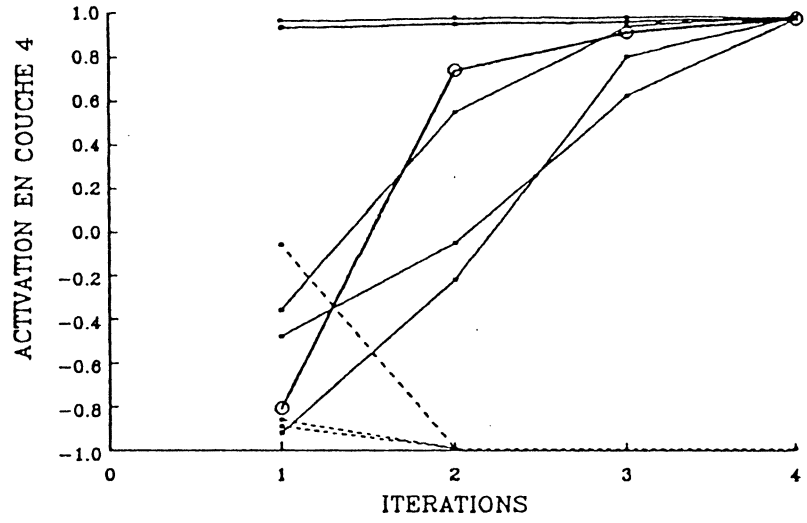
Dynamique des fausses reconnaissances :

Nous étudierons ici comment le réseau peut, au cours des réinjections, converger sur une identité erronée. Ceci ne présage en rien de la réponse finale de fausse identification. En effet, en étudiant le réseau isolément, nous nous situons au niveau des paramètres qui sont accessibles au processus de décision, et non après la phase d'intégration des nombreuses informations qui conduit à la réponse. Par exemple, la recherche mnésique peut très bien aboutir à l'émergence d'une identité erronée et à un écho-visage différent du visage à identifier. Dans ce cas, si une information visage est encore disponible en mémoire à court terme (ou mieux si le visage est perceptivement présent), le processus de décision, par une confrontation de toutes ces informations disponibles, pourra conduire à une réponse du type: "ce visage me rappelle UNTEL, mais ce n'est pas lui".

Quant au déterminisme des fausses reconnaissances, deux facteurs semblent prépondérant au vu des expérimentations rapportées dans la littérature : la ressemblance perceptive entre les visages, et le contexte situationnel et représentationnel de reconnaissance. Nous étudierons donc l'effet conjoint de ces

deux facteurs à travers deux cas : l'un simple qui met directement en évidence la sensibilité du réseau à ces deux variables; l'autre plus complexe, qui rend compte également des difficultés de différenciations intra-catégorielle pour les visages encodés dans la condition "non variable et non spécifique".

PREMIER CAS :



-----•----- : Traits du visage présenté, et qui sont non communs avec le visage encodé.
——•—— : Traits du visage encodé
——⊖—— : Identité du visage encodé.

Figure 3.14 : Dynamique d'une fausse identification par l'effet conjoint de la ressemblance inter-visages et du contexte de reconnaissance. La force de l'identité et les cellules activées dans l'écho-visage sont ici représentées.

Un visage nouveau, ressemblant seulement sur les deux derniers blocs à un visage déjà appris, est présenté dans le contexte d'encodage de ce dernier (contexte spécifique). La Figure 3.14 illustre la dynamique d'une fausse identification, par un effet conjoint de la ressemblance inter-visage et d'un contexte ancien "attracteur". On voit sur la figure l'émergence progressive des traits de l'ancien visage appris, qui s'accompagne d'une augmentation de son identité, ainsi que la diminuition des traits du visage présenté qui ne correspondent pas au visage appris.

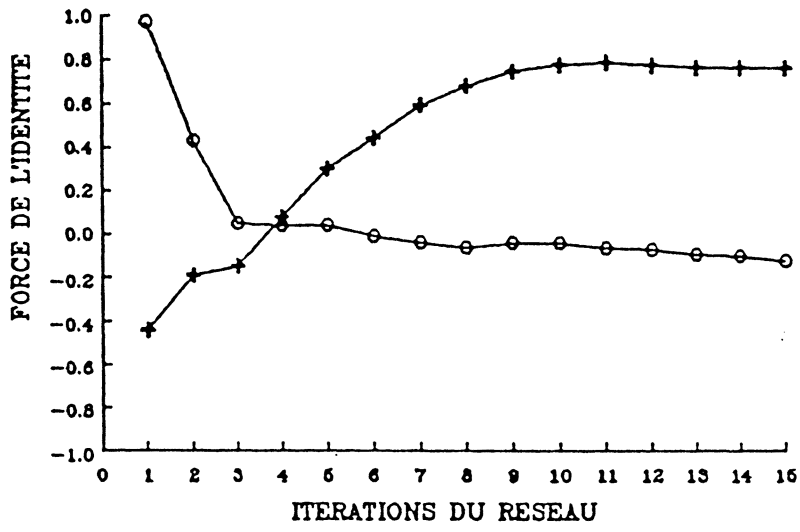
Nous constatons en effet que dès l'issue de la première itération, seuls les deux traits communs sont préservés, les autres étant très affaiblis. En effet,

dès la présentation du pattern, le contexte a commencé à activer dans V-C la représentation de son épisode d'encodage, ceci ayant été facilité par la ressemblance entre les visages. Les traits du visage d'encodage commencent déjà à émerger, ainsi que très légèrement son identité (-0,80). Nous avons par ailleurs pu constater que ce cas de fausse reconnaissance ne se produisait pas avec un contexte blanc, ce qui permet d'avancer qu'ici, le rôle du contexte ancien est important dans l'activation de V-C et qu'il ne s'agit donc pas d'un simple effet de ressemblance inter-visages. Après quatre itération, la recherche aboutit donc à une forte identité (érronée), ainsi qu'à un écho visage "net", en partie différent du visage d'input.

DEUXIEME CAS :

Nous avons présenté ci-dessus un cas de fausse identification induit par le contexte de reconnaissance. Nous nous intéresserons désormais au rôle que peut jouer le contexte d'encodage dans le déterminisme des fausses reconnaissances, de par la structuration des représentations en mémoire qu'il a générée. En effet, nous avons souligné lors de l'étude expérimentale que pour des visages encodés dans la condition "non variable non spécifique" un phénomène de "fan effect" intra-catégoriel pouvait gêner l'identification du visage d'input au profit d'un autre visage de la catégorie.

Pour tester cet éventuel effet, nous disposons de deux visages appris particulièrement intéressants : ils partagent tous deux la même catégorie créée par le contexte d'encodage, se ressemblent sur deux blocs (le premier et le troisième) et l'un est appris deux fois alors que l'autre est appris quatre fois. Nous avons donc présenté le visage le moins appris avec un contexte blanc, ce qui est ici la seule façon de s'assurer que le contexte de reconnaissance ne ressemble en rien à aucun des contextes appris par le réseau.



—○— : Force de l'Identité du visage présenté (appris deux fois).

—✚— : Force de l'Identité d'un visage (appris quatre fois) qui partage la même catégorie créée par le contexte d'encodage.

Figure 3.15 : Dynamique d'une fausse identification par l'effet conjoint de la ressemblance inter-visages et des conditions d'encodage.

La Figure 3.15 représente l'évolution, au cours des réinjections, de l'identité du visage ancien présenté et de celle du visage ancien (mais plus fort) qui lui ressemble.

- Lors de la première itération du réseau, l'identité du visage présenté émerge très nettement puisqu'un contexte blanc ne le gêne en rien pour accéder à sa catégorie, et que le visage est "pur" à l'entrée du premier cycle de traitement. L'identité du visage fort émerge aussi faiblement (-0,5) car il profite de sa ressemblance et du "fan effect" dans la catégorie activée. C'est réellement l'effet conjoint de ces deux facteurs qui permet au visage fort de commencer à se manifester.

- Après la première réinjection, l'identité du visage faible commence à décroître puisque le phénomène de "fan effect" brouille l'information visage. Cet effet est manifeste sur l'écho-visage : un trait du visage initial est en train de s'affaiblir (0,40). Le visage fort, du fait de ses deux traits en commun et de son meilleur degré d'apprentissage, participe et profite du "fan effect" pour faire émerger deux autres traits et augmenter son identité (- 0,2).

- Les itérations suivantes continuerons à favoriser l'installation du visage fort, au dépend du visage présenté. Il possède dès le troisième itération quatre de ces cinq traits. La progression du visage fort s'effectue parallèlement à la baisse

du quatrième trait du visage faible.

- A partir de la douzième itération, les identités des visages fort et faible stagnent respectivement à 0,80 et -0,05. Ceci s'explique au vu des écho-visages, car le visage faible possède encore 3 traits (les deux communs plus son trait fort : cinquième bloc), alors que le visage fort possède tous ses traits à l'exception de celui de son bloc 5. Cette situation n'évoluera plus : le contexte ayant été appris avec les deux visages, il les active donc simultanément par le jeu des associations V-C. Même si le visage présenté est défavorisé par son moindre degré d'apprentissage, il arrive toutefois à conserver au moins son bloc fort. Ceci rejoint la réalité écologique, car il paraît peu probable de perdre un trait aussi fort que la chevelure (dans notre exemple) en cours de recherche mnésique.

Nous avons donc là une illustration du rôle important de la nature de l'encodage sur la dynamique des fausses reconnaissances. En effet, lorque deux visages sont encodés dans un même contexte non spécifique, et uniquement dans celui-ci (deux 'administratifs' appris dans un même bureau), le contexte possède un grand poids dans la représentation V-C. Si de plus ces deux visages se ressemblent un peu, et que l'un est plus appris que l'autre, la différenciation intra-catégorielle lors de la recherche peut s'avérer très difficile, voire même conduire à un sentiment d'identité erroné. Le rôle du processus de décision est ici crucial car il aura à sa disposition des informations difficiles à intégrer et très évolutives avec le temps. En outre sur l'écho-visage, il a un visage différent de celui d'input sur deux traits. Que l'on songe ici aux décisions successives et contradictoires, "la première étant souvent la bonne", nous souffle le sens commun !

Rappel d'un visage à partir d'un contexte :

Nous étudierons ici la capacité du réseau à retrouver un visage à partir d'indices contextuels seuls. Nous soumettrons bien sûr en entrée un contexte appris et spécifique puisqu'un contexte non spécifique n'est absolument plus un indice distinctif pour accéder à un visage (ou une identité) particulier, mais en rappelle plutôt plusieurs simultanément. Ce contexte spécifique sera présenté avec un visage "blanc", car la recherche du souvenir d'un visage (écho-visage) peut être entreprise sans aucun indice facial, en seule présence du contexte approprié. Ce type de situation pourrait s'apparenter à la question courante : "mais quelle est donc la personne que je vois habituellement dans ces

circonstances spécifiques ?".

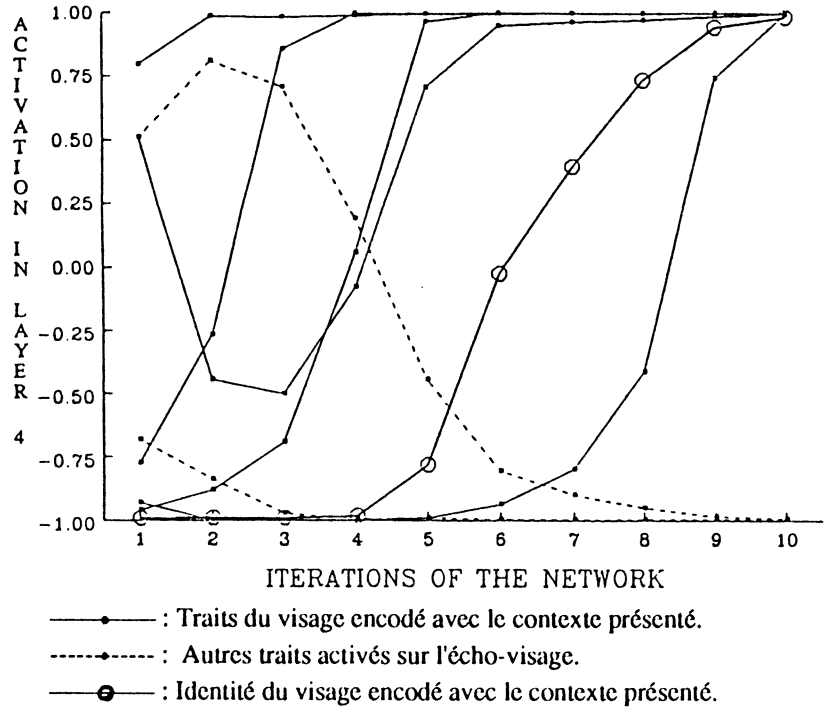


Figure 3.16 : Evolution de l'Identité et de l'écho-visage lorsqu'un contexte ancien et "spécifique" est présenté en l'absence d'indices faciaux.

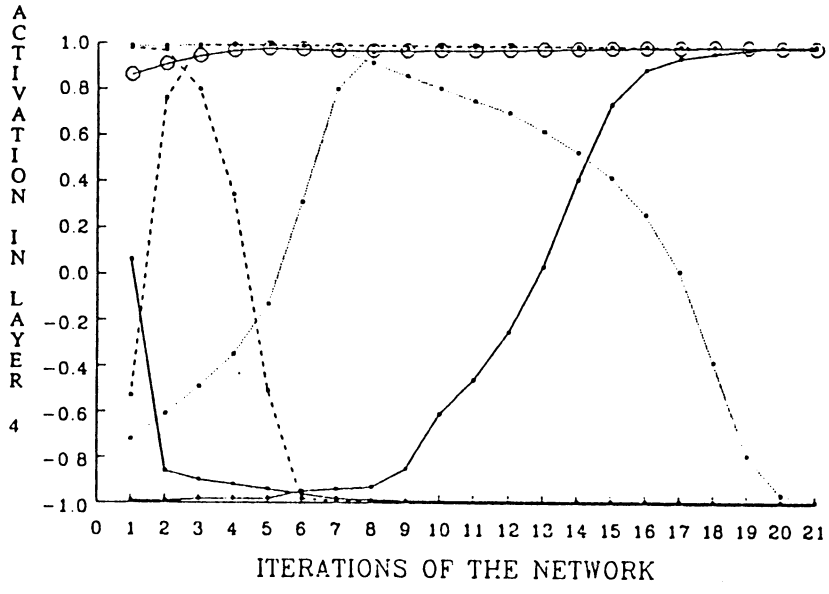
La Figure 3.16 illustre en parallèle l'évolution de la force de l'identité et l'émergence de tous les traits qui apparaissent sur l'écho-visage.

L'écho-visage évolue au cours des réinjections, par baisse progressive des "mauvais" traits et émergence simultanée des traits du visage encodé avec le contexte présenté. L' identité, quant à elle, ne commence à évoluer que lorsque quatre des traits du visage sont retrouvés, et augmente en même temps que le dernier trait. Ce phénomène est dû à l'asymétrie introduite entre les couches 2 et 3 afin de modéliser le statut focal du visage dans le processus d'identification des personnes (à partir de leur visage !). Il n'est donc pas surprenant, dans ce cas, que le contexte ne suffise pas en lui-même à induire une identité; il est par contre tout à fait intéressant qu'il arrive, au cours des réinjections, à rappeler progressivement le visage qui lui est associé spécifiquement.

Recherche inter-contextuelle :

Les théories de la reconnaissance médiatisée par un double processus (Tiberghien, Cauzinille, & Mathieu, 1979; Mandler, 1980) supposent que le

deuxième processus consiste en une recherche, à partir des indices contextuels de reconnaissance, du contexte d'encodage. L'efficience de cette recherche est dépendante de la compatibilité entre les conditions de la mise en mémoire et celles de la reconnaissance. Nous allons donc étudier comment le réseau peut, au cours des réinjections, arriver à effectuer spontanément une recherche inter-contextuelle.



—o— : Identité du visage présenté.

Les traits concurrents sur un bloc sont représentés par le même type de trait :

------•------ : Traits du Bloc 5.

.....•..... : Traits du Bloc 4.

——•—— : Traits du Bloc 1.

Les 2 traits communs aux contextes d'encodage et de reconnaissance ne sont pas représentés.

Figure 3.17 : Dynamique de la recherche inter-contextuelle sur 21 itérations, lorsque la compatibilité contextuelle est assurée par une ressemblance sur 2 blocs (moyens). L'évolution de l'Identité ainsi que le passage d'un trait à un autre (sur un même bloc) sont ici représentés.

Nous avons testé comment un visage ancien, présenté dans un contexte appris mais non adéquat, peut modifier progressivement le contexte de reconnaissance vers son contexte d'encodage. La compatibilité entre les indices d'encodage et de récupération est assurée par une ressemblance sur deux blocs entre les deux contextes (blocs 2 et 3). Pour effectuer ce test dans une condition peu favorable, nous avons présenté un contexte ancien bien appris, en empêchant le système d'effectuer une focalisation (qui normalement aurait pu avoir lieu

puisque nous étions dans le cas EGAL-DEPART-CONTEXTE). En effet, le processus d'atténuation du contexte facilite la recherche inter-contextuelle, en permettant une évolution plus aisée d'un contexte vers un autre, lorsque le contexte de la reconnaissance est très familier et semble ne rien apporter à la recherche de l'identité.

L'évolution dynamique de la recherche inter-contextuelle sur 21 itérations est illustrée par la Figure 3.17, où sont présentées conjointement les valeurs des 2 contextes sur l'écho-contexte ainsi que l'identité. Ici, on observe encore une fois l'émergence des "bons" traits (traits dus contexte d'encodage du visage) et l'affaiblissement des "mauvais" traits (ceux du contexte de reconnaissance qui sont non compatibles). A noter aussi que ce sont toujours les traits des blocs forts(6) qui émergent les premiers. On a testé aussi la vitesse d'émergence en fonction du poids des blocs compatibles au départ (Rousset, Schreiber & Wang, 1988). Il se révèle que la compatibilité assurée par des blocs forts permet toujours une recherche plus rapide que celle assurée par des blocs faibles, ce qui confirme la hiérarchie introduite par les connexions spécifiques.

En conclusion, on constate que le degré de compatibilité contextuelle joue un rôle important dans la recherche mnésique : une plus forte compatibilité permet au réseau de retrouver plus rapidement le contexte d'encodage et l'identité. Ainsi, si l'on considère que le nombre d'itérations du réseau peut être en rapport avec la latence de reconnaissance (le processus de décision ayant alors plus rapidement des informations pour conclure), un visage présenté dans un contexte compatible avec son contexte d'encodage sera plus rapidement identifié que ce même visage présenté dans un contexte moins compatible. Lorsqu'il n'y a aucune compatibilité entre les contextes, la recherche inter-contextuelle s'avère alors très délicate, voire même impossible pour des visages peu appris et présentés dans des contextes forts : dans ce cas, même une forte focalisation ne permet pas toujours au réseau de converger sur la bonne identité (cf § 4.5.1 dans Rousset, Schreiber & Wang, 1988). De plus, une forte compatibilité contextuelle n'est pas le garant d'une bonne identification quand le visage est peu appris et "non variable non spécifique" : en effet, comme nous l'avons exposé au début de cette partie, la récupération de la catégorie d'encodage, si elle n'est pas suivie d'un processus réussi de différenciation intra-catégorielle, peut alors aboutir à une fausse identification.

---

(6) Les connexions spécifiques ont permis en effet de hiérarchiser les blocs de gauche à doite.

III.4. Conclusions et perspectives

On a présenté dans ce chapitre le système FACENET, actuellement en cours de modélisation. La première étape de cette modélisation consiste en une simulation connexionniste de la recherche mnésique d'identité, qui constitue la seconde étape de traitement d'après le modèle général d'identification de la personne proposé par Bruce et Young. Pour ce faire, nous avons construit un réseau multicouches en nous servant des fonctionnalités avancées par ce modèle cognitif.

L'architecture a été soigneusement étudiée pour permettre au réseau de modéliser efficacement plusieurs fonctions : plus spécialement la fonction d'intégration des informations contextuelles dans le processus de structuration de l'itentité des visages, ainsi que l'émulation de la dynamique de la recherche mnésique. Elle permet aussi de modéliser un traitement plus spécifique des entrées visages et des entrées contextes, traitement qui tient compte de la hiérarchie des informations (par les connexions spécifiques).

Une procédure de reconnaissance a été proposée, qui calcule des mesures de similarité entre les échos et les informations d'entrée initiales (supposées être disponibles en mémoire à court terme pendant la recherche). Cette procédure gère les cycles de réinjections, en effectuant dans un cas précis une focalisation sur le visage durant la recherche mnésique. Cette fonctionnalité de réinjection simule la dynamique de la recherche d'identification.

Le système nous a permis d'étudier la dynamique du contexte dans la reconnaissance des visages, sur la structuration de l'identité et la recherche mnésique. Nous avons pour l'instant manipulé la nature de la relation visage-contexte à travers deux facteurs expérimentaux : la variabilité et la spécificité.

Les résultats, tant quantitatifs que qualitatifs (étude de la recherche sur plusieurs itérations) indiquent qu'il existe une interaction entre ces deux facteurs :

- L'identification des visages encodés en contexte variable n'est pas perturbée par un changement de contexte lors de la reconnaissance, et ceci qu'ils aient été appris avec des contextes spécifiques ou non spécifiques.
- Lorsque les visages ont été encodés en condition "non variable", la

spécificité de leur contexte d'encodage joue un rôle dans l'efficience du mécanisme d'identification : l'identification des visages encodés en condition "non spécifique" est beaucoup plus perturbée que celle des visages encodés en condition "spécifique".

- Cependant, les visages "variables" restent toujours supérieurs aux visages "non variables".

Nous avons également effectué des tests de l'évolution propre du réseau en situation de reconnaissance. La procédure de réinjection a doté le système d'un mécanisme de recherche mnésique dynamique, qui lui permet de traiter efficacement des informations bruitées, incomplètes, affaiblies ou partiellement fausses. Par exemple, un visage peut être récupéré à partir de son contexte en quelques itérations de reconnaissance. Le système FACENET a montré également de bonnes capacités de recherche inter-contextuelle, capacités qui sont bien évidemment dépendantes du degré de compatibilité entre les contextes d'encodage et de reconnaissance. Enfin, nous avons pu simuler la dynamique des fausses identifications, propriété non négligeable du point de vue psychologique.

Pour tester la pertinence de l'architecture, nous avons construit un autre réseau sans couche 3. Dans ce réseau, on a relié les cellules d'identité à toutes les cellules de la couche 2. Ici, les contraintes d'identité imposées dans la couche de sortie influencent directement la couche 2. Bien que la base d'apprentissage soit la même que celle de FACENET, ce réseau s'est montré incapable de produire des phénomènes de catégorisation et fan-effect lors des tests de reconnaissance. Ceci met en évidence l'importance de l'architecture dans l'application des réseaux de neurones à la simulation des phénomènes cognitifs.

PERSPECTIVES

Le système actuel offre encore de très nombreuses possibilités de tests expérimentaux d'hypothèses psychologiques. Par exemple, l'effet du degré d'apprentissage sur l'élaboration de l'identité offre à lui seul un vaste champ d'étude. De plus, si l'on considère cet "artéfact connexionniste" comme un sujet virtuel facilement modifiable, il serait intéressant de supprimer certaines de ses composantes pour comparer les résultats ainsi obtenus avec ceux recueillis dans les cas de prosopagnosie.

Certains éléments du système actuel sont aussi à compléter : il faudrait simuler la dégradation progressive des informations disponibles en mémoire à court terme durant la recherche, et bien sûr implémenter le mécanisme de décision d'identification. De plus, la modélisation du réseau perceptif (à partir d'images digitalisées) ainsi que l'intégration de l'accès au nom sont envisagées : l'ensemble du processus d'identification sera alors simulé par le couplage de ces deux éléments avec le système FACENET actuel. Enfin, nous allons implanter un algorithme permettant de construire la notion du temps dans les réseaux multicouches. Nous espérons que cela va offrir bientôt la possibilité de simuler les effets de "priming" à l'oeuvre dans la reconnaissance des visages. Nous pensons également à utiliser les réseaux récurrents pour ce faire.

Chapitre IV
Une application des réseaux multicouches à un problème de reconnaissance des mots prononcés

L'objectif du présent chapitre est d'étudier l'architecture des réseaux multicouches pour des applications au traitement de la parole. Plus particulièrement, nous nous intéresserons au problème du traitement d'informations temporelles. Cette étude est complémentaire à celle présentée au paragraphe I.4 du Chapitre I. Nous allons proposer une architecture à connexions partielles et superposées, qui permet de détecter de manière efficace la régularité dans les variations temporelles. Les expériences ont été faites sur l'hypercube T20 (Wang 1988a).

IV.1. Architecture de connexions partielles pour un traitement plus efficace d'informations temporelles

Le modèle de réseaux multicouches est un outil puissant pour effectuer des traitements d'informations distribuées et des décisions sur des données fortement bruitées. Malgré ces nombreuses qualités, le modèle connaît encore beaucoup de difficultés dans les applications au traitement de la parole : un des problèmes principaux des réseaux est leur manque d'efficacité vis-à-vis des informations temporelles. Dans ce genre d'applications, les données (signaux) qui dépendent du temps doivent être converties et représentées par des données spatiales, qui correspondent souvent à des représentations internes du réseau, de manière à ce que les informations temporelles soient préservées.

En fait, comme souligné au Chapitre I (§I.4), les réseaux multicouches fonctionnant en GBP sont conçus pour traiter des informations plutôt spatiales que temporelles. Dans ce même paragraphe du Chapitre I, nous avons abordé le problème de l'introduction des feedbacks dans les réseaux pour apprendre à produire des séquences, ce qui pourrait avoir des applications telles que la simulation de la *coarticulation*.

Beaucoup de chercheurs utilisent des réseaux à connexions complètes entre les couches consécutives, ce qui résulte en une architecture régulière du réseau et par conséquent rend la programmation plus facile. Cette architecture est extrêmement inefficace vis-à-vis des signaux acoustiques. Dans un problème

de reconnaissance de mots par exemple, une cellule de la couche 1 recevrait des entrées en provenance de toutes les cellules de la couche 0, qui contiennent les informations spectrales d'un mot. Cette cellule devrait donc effectuer une discrimination spatiale sur le spectre tout entier pour extraire une micro-caractéristique qui lui est propre, ce qui est généralement considéré comme la tâche d'une cellule. Cette micro-caractéristique, qui devrait être générée par apprentissage, constitue un problème très délicat si les connexions sont complètes : lorsque le réseau est grand et le nombre des spectres à apprendre (patterns exemplaires) petit, le réseau "apprend" ces spectres comme des vecteurs spatiaux, ignore certaines régularités intrinsèques dans les différents spectres du même mot, et par conséquent ne peut pas généraliser sa connaissance sur des spectres nouveaux; or si le réseau est petit ou/et si le nombre de spectres est trop grand, le réseau aura beaucoup de mal à réussir l'apprentissage car les variations dans les (ou "des") spectres sont trop importantes pour que chaque cellule, qui n'est pas beaucoup plus puissante qu'un classifieur linéaire, puisse déterminer un trait caractéristique d'une manière efficace.

Nous avons proposé une architecture de réseau (Wang, Yé & Robert 1988), qui consiste à utiliser des connexions partielles et superposées. Ces connexions partielles pourraient exister entre deux couches quelconques, mais en principe, il faut au moins que les connexions entre la couche d'entrée et la première couche de traitement soient partielles (Figure 4.1).

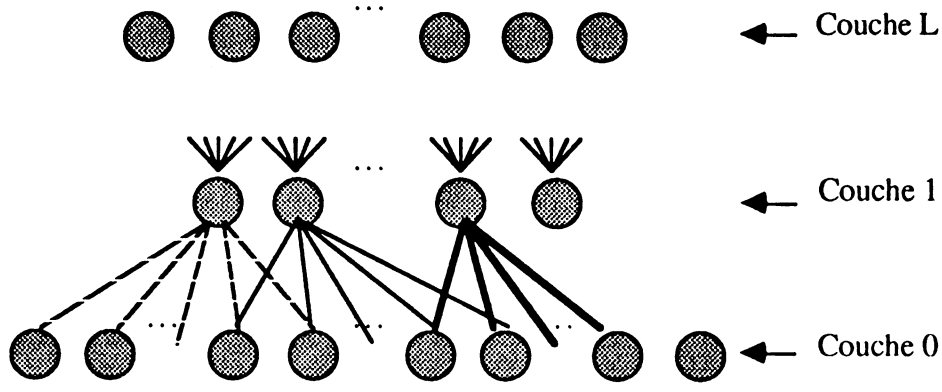


Figure 4.1 : Réseau avec des connexions partielles et superposées au moins entre la couche 0 et la couche 1. Chaque cellule de la couche 1, qui ne reçoit que des signaux d'une courte durée (ou d'un petit intervalle de fréquence), lui permet d'extraire des traits caractéristiques temporelles.

Dans un réseau à connexions partielles, chaque cellule de la couche 1 ne reçoit des entrées qu'en provenance d'un sous-ensemble de cellules (pseudo)consécutives dans la couche 0. On espère que la transformation, effectuée par cette couche de l'espace spectral à un espace spatial, soit efficace au sens qu'une variation temporelle d'un spectre ne peut avoir qu'une influence locale et limitée dans sa représentation spatiale. Par l'utilisation des connexions partielles et superposées, chaque cellule (au moins celle de la couche 1) traite seulement des informations dans un court intervalle de temps. Une petite variation en un ou quelques points d'entrée n'a donc qu'une influence partielle sur la décision finale du réseau.

Comment construire un réseau à connexions partielles? Par quel critère peut-on juger leur efficacité? Nous ne pouvons pas encore donner une méthode systématique, car c'est un problème qui dépend beaucoup de l'application d'autant plus que les connaissances sur le comportement des réseaux sont loin d'être suffisantes. Nous proposons quelques principes que nous avons suivis dans nos expériences.

Nous avons supposé qu'il y ait au moins une couche intermédiaire dans un réseau à connexions partielles, bien que cela ne soit pas strictement obligatoire du point de vue de l'architecture. La première couche joue un rôle très important dans le traitement des informations temporelles. Il est nécessaire que chaque cellule ne soit connectée qu'avec un sous ensemble de cellules dans la couche d'entrée. Il pourrait y avoir, bien sûr, une superposition partielle entre les différents sous-ensembles de cellules (connexions superposées), ou même un seul ensemble de cellules d'entrée pour plusieurs cellules intermédiaires ou de sortie (superposition totale). Dans des applications réelles, les spectres d'entrée qui sont de longueur différente, devraient être "rallongés" pour s'adapter au nombre fixe des cellules d'entrée. Comme ce rallongement consiste souvent en des ajouts de zéros, les informations du début de chaque pattern d'entrée seraient plus pertinentes que celles de la fin. Donc, il faut mieux, dans ce cas, que la densité de connexion ne soit pas la même partout. Les mêmes principes s'appliquent aussi aux connexions entre les couches supérieures.

Voici quelques exemples qui servent à illustrer des modes de connexion pour les différentes buts. La Figure 4.2 montre un cas de connexions partielles. La cellule en haut est connectée seulement avec des cellules (d'entrée) qui sont

"chargées" par des spectres échantillonnés à l'instant t_{n-1}, t_n et t_{n+1}. Le mode de connexion illustré par la Figure 4.3 favorise l'influence des informations dans les spectres de basse fréquence. Dans la reconnaissance, les spectres de haute fréquence sont moins importants. Les deux cellules qui partagent le même ensemble de cellules d'entrée (Figure 4.4) permettent éventuellement d'augmenter la capacité de discrimination du réseau sur des informations dans le même intervalle de temps. Les connexions illustrées dans la Figure 4.5 permettent à la cellule du haut "d'examiner" les variations dans une bande spectrale particulière. La Figure 4.6 illustre un exemple qui est conforme au principe concernant la densité des connexions, c'est-à-dire que les informations au début de chaque pattern d'entrée imposent une influence plus importante sur les représentations intermédiaires au travers desquelles elles apportent des contributions plus significatives à la décision du réseau. Enfin, la Figure 4.7 montre un mode de connexions partielles à deux niveaux qui permettrait de calculer des traits phonétiques dans le spectre d'entrée.

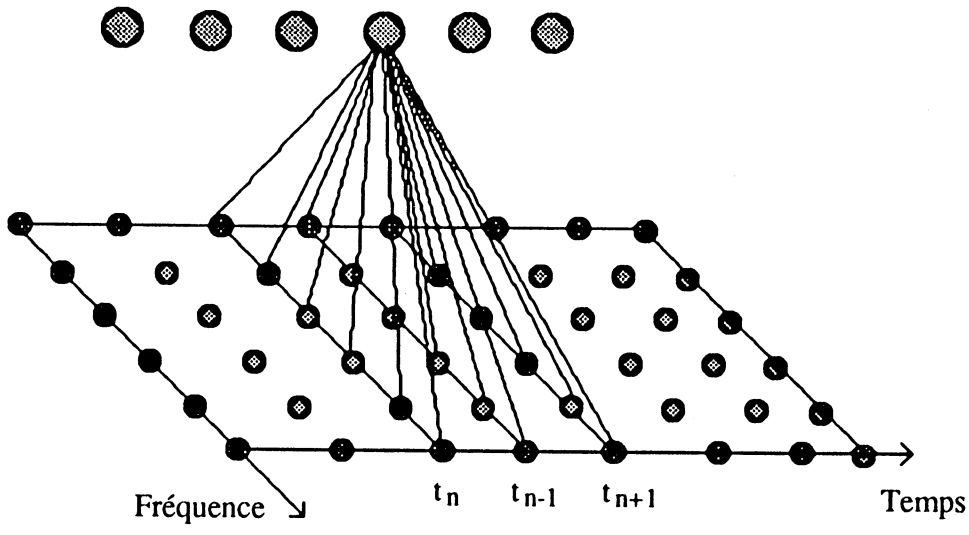


Figure 4.2 : Les connexions partielles qui permettent de traiter des informations temporelles (spectres échantillonnés à l'instant t_{n-1}, t_n, t_{n+1}).

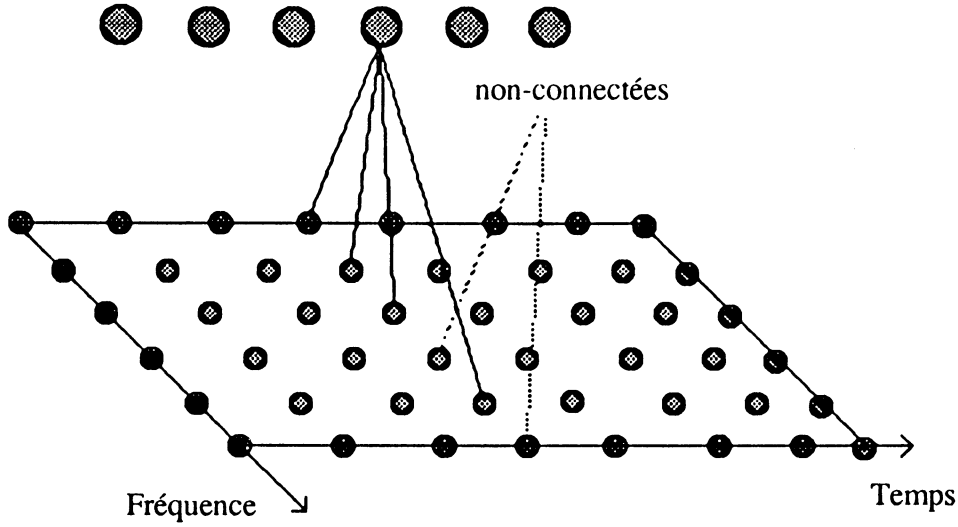


Figure 4.3 : Les connexions qui favorisent des informations dans les spectres de basse fréquence.

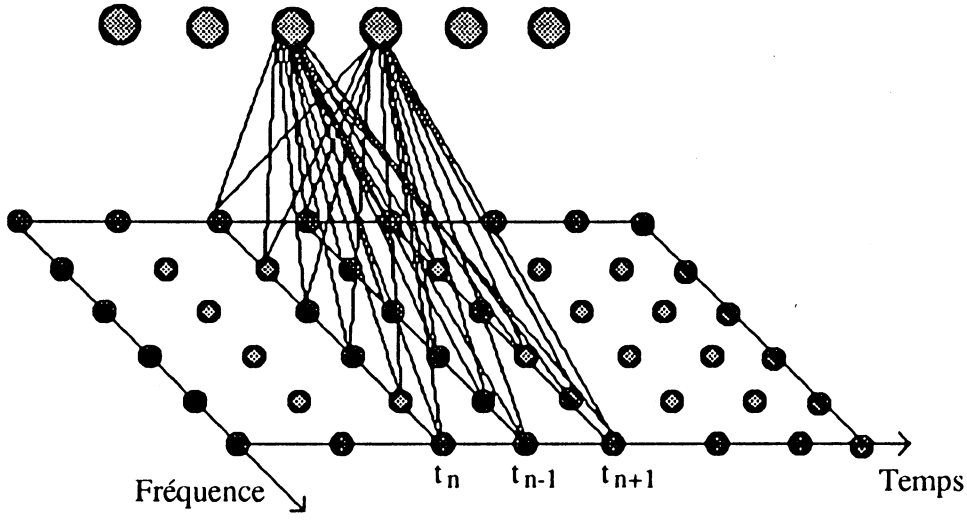


Figure 4.4 : Deux cellules qui sont connectées au même sous-ensemble de cellules permettent d'augmenter la capacité de discrimination du réseau.

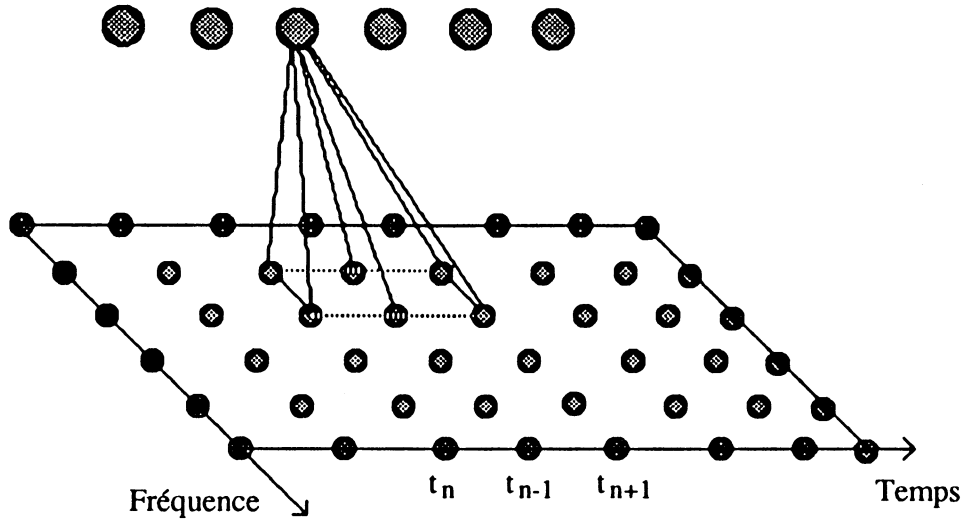


Figure 4.5 : Les connexions partielles qui permettent d'examiner les variations dans une bande spectrale particulière.

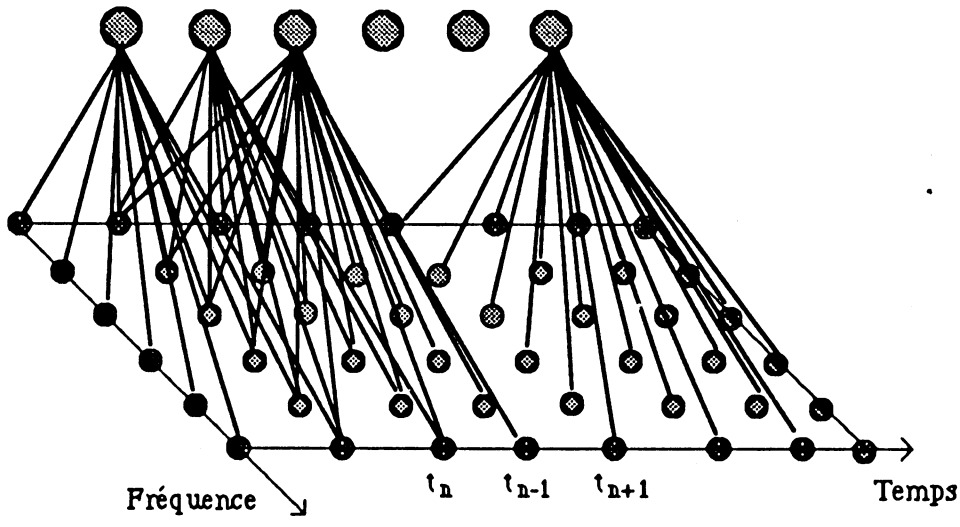


Figure 4.6 : Connexions fortes au début et faibles à la fin, un principe à respecter si les spectres d'entrée n'ont pas la même longueur et sont "rallongés" par des valeurs non pertinentes.

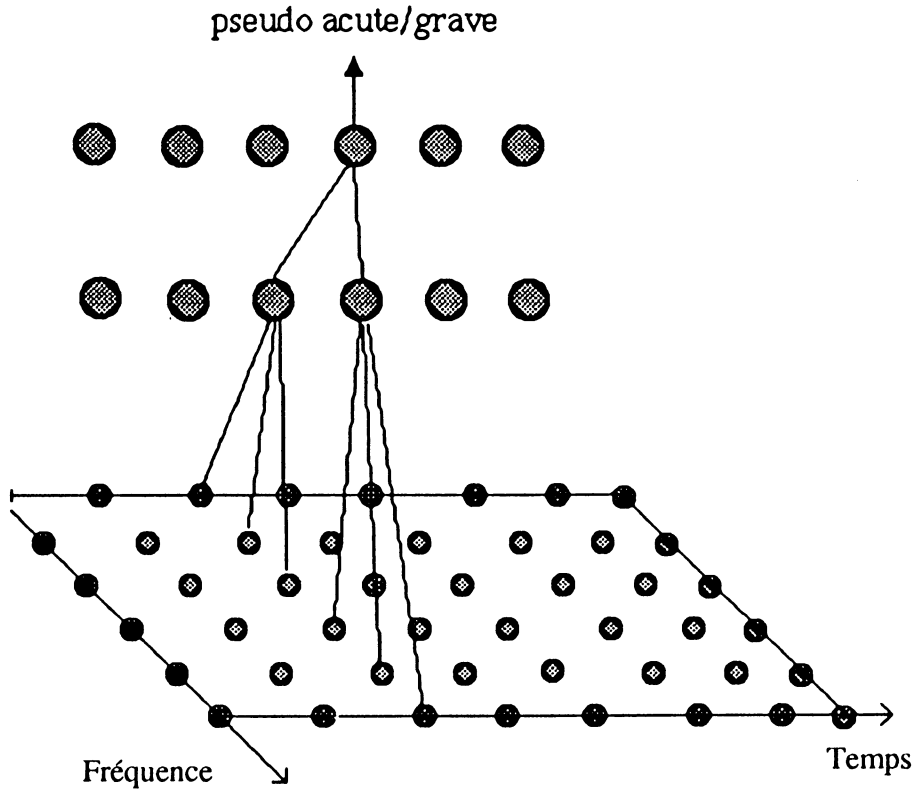


Figure 4.7 : Un mode de connexion qui permet de calculer des traits phonétiques dans le spectre d'entrée.

IV.2. Application d'un réseau à connexions partielles à la reconnaissance de mots

Le problème de la reconnaissance de mots a été choisi comme première application du réseau à connexions partielles au traitement de la parole. Par rapport aux méthodes classiques, nous esperons bien sûr atteindre une performance comparable, mais l'objectif principal est de mieux comprendre la necessité architecturale pour que les réseaux puissent être mieux adaptés à ce type d'applications.

En général, le problème de reconnaissance des mots a deux propriétés importantes : la première est la variabilité acoustique d'un même mot selon les locuteurs, l'environnement, le sexe etc; la seconde est la forte non-linéarité temporelle. Il existe principalement deux méthodes classiques bien connues pour résoudre ce problème. L'une entre celles est la programation dynamique (Myers *et al.*, 1980), l'autre est le modèle de chaine de Markov (Levinson *et al.*, 1983). L'inconvénient de la méthode de programmation dynamique est le temps de calcul qui est trop lent. Le modèle markovien, quant à lui, nécessite une base volumineuse de données pour l'apprentissage. Les réseaux de neurones multicouches constituent une troisième méthode qui surmonte, dans une certaine mesure, ces difficultés.

Il faut préciser le contexte dans lequel les réseaux sont appliqués. La Figure 4.8 montre un modèle simplifié du système auditif humain, qui est constitué par l'oreille pour percevoir les signaux de son, la membrane basilaire pour filtrer des informations en fréquence au cours du temps et par le cerveau qui effectue la reconnaissance et le traitement des informations issues de la membrane. La Figure 4.9 présente un système artificiel auditif qui peut être considéré comme une approche du système auditif humain. Dans ce système, les signaux sont prétraités par un analyse de P.S.A. (Perceptually based Spectral Analysis, voir la thèse de Yé (1989)) qui modélise le système périphérique, tandis que les réseaux à connexions partielles auraient la fonction correspondant à celle du système central. Ce modèle est, bien sûr, très approximatif et manque encore de précision.

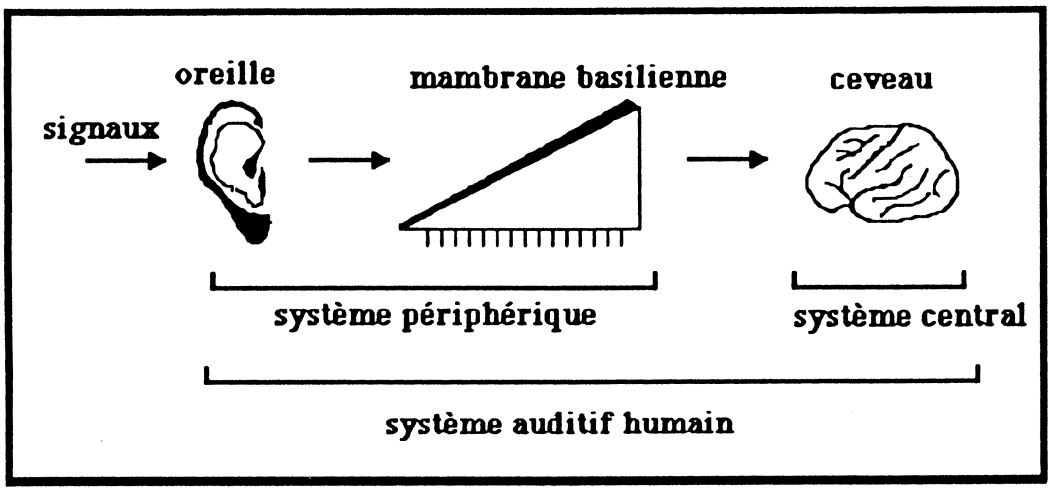


Figure 4.8 : Un modèle simplifié du système auditif humain.

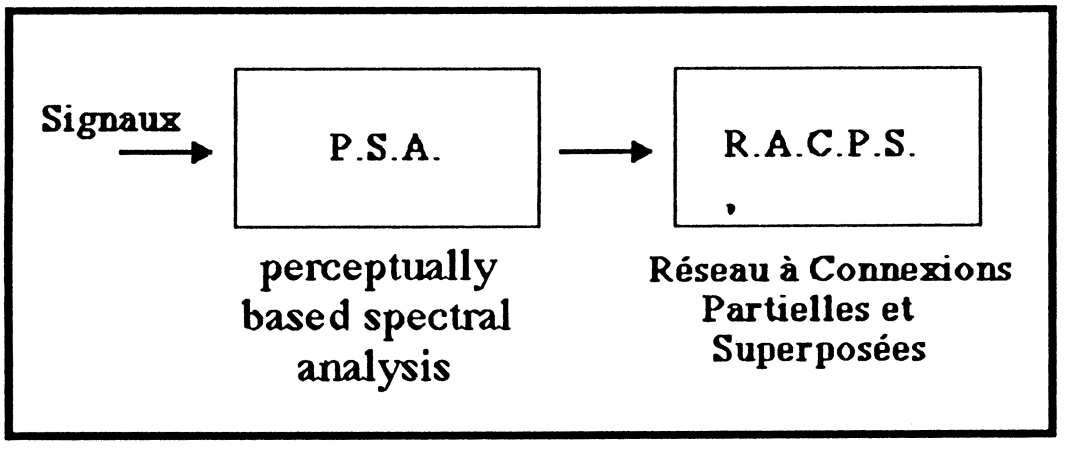


Figure 4.9 : Un système artificiel d'audition qui simule le système humain

Les expériences qui seront présentés ici, concernent un problème de reconnaissance des nombres chinois de 0 à 10. Chaque nombre a été prononcé dix fois par un locuteur et une locutrice. Il y a donc un ensemble de 220 patterns. On distingue le nombre prononcé par l'homme et celui prononcé par la femme.

La langue chinoise est un language mono-syllabique, un caractère est toujours composé d'un CV (Consonne-Voyelle). Le Tableau 4.1 donne l'IPA (pour International Phonetic Alphabet) et le Pinyin (Mandarin) de chaque nombre.

Chiffres	PinYin (Mandarin)	IPA
0	ling	[lin]
1	yi	[i]
2	re	[ɣ]
3	san	[san]
4	si	[si]
5	wou	[u]
6	liu	[lio]
7	qi	[tɕʰi]
8	ba	[ɓa]
9	jiu	[ɖzio]
10	shi	[ʂi]

Tableau 4.1 : La prononciation de chaque nombre transcrite en PinYin et en IPA.

Nous avons plusieurs tâches pour cette expérience : 1° Trouver un réseau capable de mémoriser un sous-ensemble de spectres (à déterminer); 2° Le réseau doit posséder une capacité de généralisation qui consiste, ici, à tout simplement, pouvoir donner des bonnes réponses pour des patterns qui ne sont pas dans l'ensemble d'apprentissage. 3° Examiner la représentation condensée des signaux pour analyser l'extraction des traits caractéristiques par le réseau.

Voici quelques paramètres concernant la prétraitement des données : Le filtre utilisé est de 20Hz-4500Hz; les signaux sont échantillonnés à 10KHz; La fenêtre d'analyse est de 25.6 µS avec un shift de12.6 µS.

Il est généralement considéré que les informations spectrales et temporelles dans la bande spectrale critique sont assez riches pour faire de la reconnaissance de la parole. Les paramètres de MFCC (Mel Frequency Cepstral Coefficients) ont été utilisés pour générer les patterns d'entrée du réseau. L'organigramme suivant (Figure 4.10) illustre le procédé utilisé pour obtenir ces paramètres : les signaux enregistrés ont d'abord subi une analyse spectrale (un FFT de 256 points), puis un traitement d'integration (critical band of Zwicker) et enfin une transformation en cosinus.

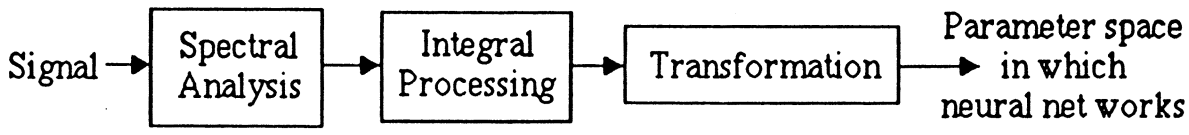


Figure 4.10 : Le procédé de prétraitement des signaux pour "alimenter" le réseau.

Ce traitement simule le traitement effectué dans l'oreille interne et sert de procédure de réduction des données. Le début et la fin de chaque mot sont déterminés de manière automatique. Il en résulte que, pour un même mot, les dix répétitions n'ont pas le même nombre de points dans les patterns d'entrée car la durée de chaque prononciation peut ne pas être la même.

Le réseau utilisé dans notre expérience a trois couches : 550 cellules d'entrée; 15 cellules intermédiaires; et 22 cellules de sortie. Les 550 cellules dans la couche d'entrée correspondent au nombre maximum des points dans les patterns (spectres) d'entrée. Si la longueur d'un pattern est inférieure à 550 points, une constante zéro sera ajoutée afin de ralonger le pattern. Les 22 cellules de sortie correspondent au nombre de mots multiplié par deux. Un mot prononcé par une personne (l'homme ou la femme) est représenté à la sortie du réseau par un pattern d'états d'activation contenant un 1, tous les autres étant à -1. Il est clair qu'une transformation fortement non-linéaire devrait être effectuée par le réseau.

Pour chaque nombre, nous avons choisi au hasard 14 patterns pour effectuer l'apprentissage, 7 masculins et 7 féminins. Il y a donc, au total, 154 (=14*11) patterns sur les 220 qui sont utilisés pour l'apprentissage et 66 pour tester la performance de généralisation. Les poids de connexions sont initialisés dans l'intervalle (-0.5, 0.5), le taux d'apprentissage est choisi à 0.3 et le momentum est égal à zéro (nous avons pas utilisé le momentum dans cette application). Pour la reconnaissance, un pattern d'entrée sera considéré comme un mot prononcé par l'homme ou par la femme si la cellule de sortie correspondante a un état supérieur à 0.5. Quand il y a plusieurs cellules de sortie qui admettent l'état supérieur à 0.5, le pattern est considéré d'avoir été identifié comme plusieurs mots (ou le même mot de différentes personnes). Ce cas correspond à ce que nous appellons, par la suite, la *confusion*.

Expérimentation 1

Dans cette expérimentation, nous avons utilisé des connexions complètes entre couches consécutives. 145 patterns sur 154 sont appris au bout de 5000 itérations d'apprentissage, mais le résultat de généralisation est assez pauvre : 10 patterns seulement sont correctement reconnus parmi 66, soit 15% de réussite.

Nous pensons que cette contre performance provient du fait que les connexions dans le réseau sont complètes. En fait, pour chaque mot, ce réseau aurait beaucoup de mal à extraire des régularités dans les patterns à cause de la variation temporelle, d'autant plus que les valeurs non pertinentes ajoutés à la fin de chaque pattern participent activement à la décision finale. C'est la raison pour laquelle il faut utiliser des connexions partielles afin de rendre le traitement des informations temporelles plus efficaces et pour atténuer l'influence des cellules ayant souvent de valeurs non pertinentes

Expérimentation 2

Dans cette expérimentation, nous avons utilisé des connexions partielles entre la couche d'entrée et la couche 1. La façon dont les cellules des deux couches sont connectées est illustrée par le Tableau 4.2. Au bout de 5000 itérations d'apprentissage, comme précedemment 145 patterns sur 154 sont appris, mais le résultat de généralisation est nettement meilleur : 60 cas parmi 66 sont correctement reconnus, soit 91%.

Le Tableau 4.3, appellé tableau de généralisation et de confusion, donne une statistique du nombre de "bonnes" généralisations et de celui des confusions. L'étiquette du côté gauche correspond aux patterns à tester, par exemple M_5 représente les trois patterns du nombre 5 prononcés par l'homme. Le symbole en haut correspond aux patterns reconnus. Le nombre 3 qui se trouve au croisement (M_5, $\overset{M}{5}$) signifie que les trois patterns sont tous bien reconnus comme le nombre 5 prononcé par l'homme, tandis que le 2 qui se trouve dans le croisement (M_10, $\overset{W}{7}$) signifie que deux patterns, qui devraient être reconnus comme le 10 prononcé par l'homme, sont finalement reconnus comme 7 prononcé par la femme. Certaines confusions entre l'homme et la femme sont dues probablement à la similarité de certaines prononciations entre les personnes. Il existe aussi des cas où un pattern est reconnu comme deux nombres ou un même nombre prononcé par les deux personnes (confusion). Le fait que toutes les valeurs diagonales ne soient pas égales à 3 signifie, bien sûr, l'échec du réseau pour la reconnaissance de certains patterns. L'utilisation des connexions partielles a apporté une amélioration radicale à la performance de généralisation.

cellule de couche 1	cellules de la couche 0 auxquelles la cellule de couche 1 est connectée
1	de 1 à 30
2	de 21 à 50
3	de 41 à 70
4	de 61 à 100
5	de 91 à 130
6	de 121 à 160
7	de 151 à 190
8	de 181 à 220
9	de 211 à 240
10	de 231 à 270
11	de 271 à 320
12	de 321 à 370
13	de 371 à 420
14	de 421 à 470
15	de 471 à 550

Tableau 4.2 : Les connexions partielles entre la couche 1 et la couche 0.

recog. nb. of spect.	M | 0	W | 0	M | 1	W | 1	M | 2	W | 2	M | 3	W | 3	M | 4	W | 4	M | 5	W | 5	M | 6	W | 6	M | 7	W | 7	M | 8	W | 8	M | 9	W | 9	M | 10	W | 10
M_0	2		1																			
W_0		1												1								
M_1			3																			
W_1				3																		
M_2					2																	
W_2						3																
M_3							3		1													
W_3								1						1								
M_4									3													
W_4										3												
M_5											3											
W_5												3										
M_6													3									
W_6														3								
M_7															3							
W_7																3						
M_8					1												3					
W_8																		3				
M_9																			3			
W_9					1														1	3		
M_10																2					2	
W_10																						3

Tableau 4.3 : Tableau de généralisation et de confusion de l'Expérimentation 2.

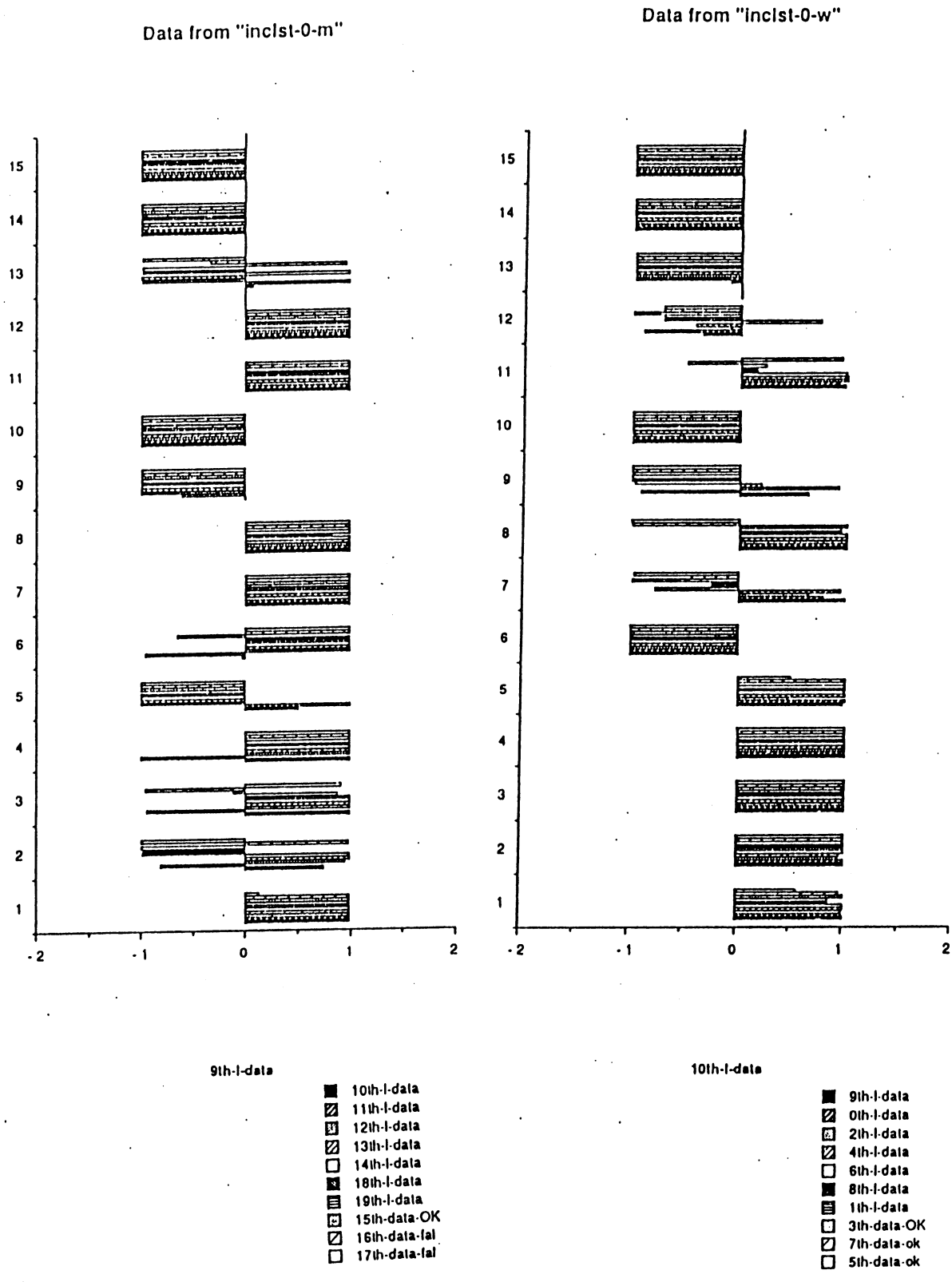


Figure 4.11 : Deux histogrammes pour le nombre 0 prononcé par l'homme et par la femme.

Pour mieux comprendre le fonctionnement du réseau, nous avons établi des histogrammes d'activité des cellules intermédiaires. Deux exemples d'histogramme sont montrés par la Figure 4.11. L'axe x correspond au numéro de la cellule et l'axe y correspond à sa valeur d'activité. Les histogrammes d'un mot sont présentés sur un même dessin pour faciliter la comparaison. On peut noter, sur ces histogrammes, que certaines cellules ont des activités plus homogènes, tandis que certaines autres ont des activités plus variées.

Le Tableau 4.4 donne les valeurs moyennes d'activité de chaque cellule pour chaque mot. Ces valeurs sont tout simplement calculées par la somme d'activité pour les dix répétitions divisée par 10. L'activité d'une cellule pour chaque répétition est souvent proche de 1 ou de -1. Si l'activité d'une cellule est presque toujours la même, la moyenne n'est pas loin de 1 ou de -1, sinon la moyenne est petite (en valeur absolue) ou proche de 0. Donc les valeurs moyennes d'activité fournissent une indication sur la spécialisation des cellules pour un mot.

	0-man	0-woman	1-man	1-woman	2-man	2-woman	3-man	3-woman	4-man	4-woman	5-man
Hidden unit 1	0.91	0.92	0.91	1.00	0.96	0.52	0.76	0.90	0.80	0.88	0.99
Hidden unit 2	-0.01	0.99	1.00	1.00	0.67	-1.00	0.70	-0.19	0.49	0.34	0.74
Hidden unit 3	0.48	1.00	-0.99	0.94	-1.00	-1.00	1.00	-0.42	0.97	1.00	-0.16
Hidden unit 4	0.80	1.00	0.98	-0.94	-1.00	-1.00	-0.95	-0.78	-0.73	0.76	1.00
Hidden unit 5	-0.58	0.89	-1.00	-1.00	1.00	0.11	0.84	0.31	0.54	-1.00	1.00
Hidden unit 6	0.54	-1.00	0.91	-1.00	0.45	-1.00	0.12	1.00	-1.00	0.99	1.00
Hidden unit 7	1.00	-0.03	1.00	1.00	-1.00	-1.00	-0.76	0.41	0.98	1.00	1.00
Hidden unit 8	0.98	0.59	0.99	1.00	-0.83	-1.00	-0.98	-0.87	-1.00	-1.00	-1.00
Hidden unit 9	-0.86	-0.51	-0.93	-0.93	-1.00	-1.00	-0.84	-0.40	-1.00	0.66	-0.88
Hidden unit 10	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-0.99	-1.00	-0.98	-0.99
Hidden unit 11	1.00	0.59	0.84	0.52	-0.68	-0.33	-0.03	0.25	-0.22	-0.34	0.49
Hidden unit 12	0.99	-0.49	0.43	-0.75	0.64	-0.45	1.00	-0.74	0.48	-0.66	-0.62
Hidden unit 13	-0.22	-0.90	-0.95	-0.98	-0.98	-0.98	-0.07	-0.98	-0.87	-0.98	-0.98
Hidden unit 14	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
Hidden unit 15	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00

5-woman	6-man	6-woman	7-man	7-woman	8-man	8-woman	9-man	9-woman	10-man	10-woman
0.87	0.77	0.51	0.85	0.85	0.85	0.67	0.75	0.80	0.66	0.54
0.48	-0.92	0.39	0.97	0.99	0.17	-0.83	0.99	0.99	-0.72	-0.98
-0.97	-0.51	0.62	-0.10	-0.53	-0.98	-0.99	-0.72	-0.89	0.94	0.70
-0.09	0.90	0.88	-0.92	-0.75	-0.99	-1.00	0.03	0.68	0.57	0.67
0.89	-0.30	0.35	-0.71	-0.85	0.85	-0.97	-0.98	-0.04	-0.57	-0.87
1.00	0.86	-0.09	0.49	-0.85	0.54	0.51	0.63	-0.49	-0.61	0.41
-0.89	-0.99	-0.99	0.81	-0.46	-1.00	-1.00	-0.92	-0.93	-0.79	-0.49
-0.79	-0.89	-0.95	1.00	0.98	0.74	0.37	-0.89	-0.97	0.22	0.99
0.55	-0.49	0.67	0.54	0.20	-0.75	-0.07	-0.38	0.79	-0.67	0.54
-0.98	-0.96	-0.95	-0.97	-0.98	-0.98	-0.99	-0.97	-0.96	-1.00	-0.99
0.97	0.49	0.87	0.80	-0.70	-0.01	-0.34	0.23	0.16	-0.48	0.72
0.75	-0.83	-0.01	0.54	0.15	-0.63	0.73	-0.80	0.36	0.63	-0.60
-0.79	-0.98	-0.98	-0.98	-0.98	-0.98	-0.48	-0.98	-0.98	-0.98	-0.98
-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00

Tableau 4.4 : Activité moyenne de chaque cellule pour chaque mot.

La Figure 4.12 montre le pourcentage de cellules dont les moyennes d'activité en valeur absolue sont supérieures à un seul variable, p=**P(v>x)**. Ici **v** représente la valeur absolue de moyenne d'activité, **x** sert de seuil variable et **P** est le pourcentage des cellules dont la valeur **v** est supérieure à **x**. Comme la moyenne est calculée sur dix répétition, la condition v>0.9 signifie que l'activité d'une cellule pour la plupart de répétitions d'un mot a la même signe et v>0.5 signifie que pour plus d'une moitié des répétitions, l'activité d'une cellule a la même signe.

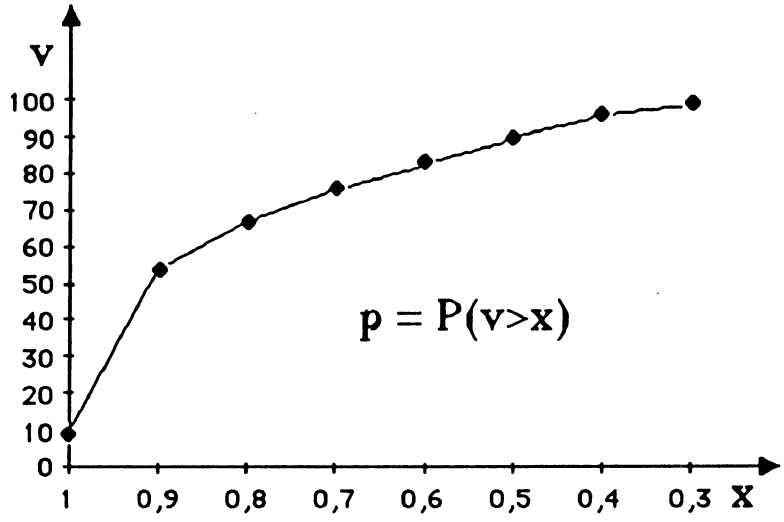


Figure 4.12 : Pourcentage de cellules dont l'activité moyenne v, en valeur absolue, est supérieure à x.

Nous avons, dans cette figure, **P(v>0.9)≈54%** et **P(v>0.5)≈90%**, ce qui signifie que la plupart des cellules ont des valeurs d'activité proches de 1 et -1. Ceci indique que les cellules ne sont pas activées au hasard. Mais pour une interpretation plus complète, il faudrait développer des techniques analytiques et statistiques plus avancées.

IV.3. Conclusion

Le problème le plus critique pour appliquer un réseau multicouches à un problème de traitement de la parole réside dans le traitement des informations temporelles. Du fait de la structure statique du réseau, certains mécanismes devraient lui être ajoutés pour permettre de passer, de manière efficace, des informations temporelles à des représentation spatiales.

La stratégie de connexions partielles donne une solution à ce problème. Les expériences présentées dans ce chapitre montre la supériorité des connexions partielles par rapport aux connexions complètes. En résumé, les connexions partielles ont les avantages suivants : 1° Relativement moins de connexions donc moins de calculs; 2° Efficacité du traitement des informations temporelles; 3° Influence affaiblie des perturbations irrégulières non-perceptives.

Une autre technique permettant de munir les réseaux multicouchés de la capacité de traitement des informations temporelles consiste à introduire du retard dans les cellules. Le time-delay-network developpé par Weible *et al* (1988) constitue une approche importante dans ce domaine.

Il serait donc intéressant de combiner plusieurs techniques pour traiter des problèmes plus compliqués. Dans cette optique, nous avons alors utilisé la technique des connexions partielles et le principe du réseau séquentiel (chapitre I §I.4) pour construire un réseau qui peut apprendre à détecter les formants dans la parole continue (Bailly, Wang & Yé en cours de préparation). Jusqu'à maintenant, la plupart des applications de réseaux de neurones ne concernent encore que des mots ou des phonèmes isolés.

Chapitre V
Amélioration de l'algorithme de rétro-propagation du gradient

V.1. Introduction

Le mérite de l'algorithme GBP réside dans sa simplicité et sa capacité à résoudre les problèmes très complexes. Cependant, le défaut principal de cet algorithme est sa vitesse lente d'apprentissage (qui est souvent très contraignante). L'inefficacité de cet algorithme constitue un des principaux obstacles d'utilisation des réseaux multicouches.

En effet, l'algorithme GBP est dérivé de la méthode du gradient. De nombreuses études théoriques et expérimentales ont été effectuées pour la mise au point de cette méthode du gradient. Les résultats sont présentés dans de multiples livres sur la théorie des filtres adaptatifs ou d'optimisation (e.g. Honig & Messerschmitt, 1984; Haykin, 1986). Pour les filtres linéaires seulement, on connaît déjà deux raisons pour lesquelles la méthode ne fonctionne pas bien : l'une est l'effet d' "overshoot" (c'est-à -dire, qu'à chaque étape d'amélioration dans la direction du gradient, les paramètres avancent trop ou pas assez car il est difficile de trouver les "pas" optimaux); l'autre provient du fait que la direction du gradient ne pointe pas sur le minimum de la surface d'erreur; il n'y a que lorsque la surface est ronde, que l'on peut espérer que le minimum se situe dans la direction du gradient.

En fait, le problème avec la méthode du gradient provient de l'ignorance de la caractéristique de la fonction à minimiser dans la région autour du point de départ pour la recherche de chaque itération, seule la direction du gradient étant prise en compte. Dans ce domaine, les méthodes comme celles de Newton ou du Gradient Conjugué ont des performances bien meilleures. Ces méthodes emploient des informations du second ordre, ce qui leur permet de rendre compte des variances de la fonction dans une région locale.

Pour un problème d'optimisation d'une fonction non-convexe, ce qui est notre cas, il est pratiquement impossible d'obtenir des informations sur la distribution globale de la fonction. Un algorithme itératif et déterministe est donc considéré comme efficace si à chaque étape d'itération il réalise une bonne

descente de la fonction dont le critère varie selon les cas.

Pour la classification des algorithmes d'apprentissage, il existe la notion d'algorithme de calcul local[7] (Robert A.J., 1988). Ces types d'algorithmes modifient chaque paramètre de la fonction d'erreur selon des informations locales en relation directe avec le paramètre. L'algorithme GBP est ainsi considéré comme un tel algorithme. L'avantage des algorithmes de calcul local est qu'ils sont simples et relativement faciles à implanter sur des architectures parallèles, mais leur désavantage est lié à leur manque d'efficacité. Les algorithmes basés sur les méthodes de Newton, ou du Gradient Conjugué sont considérés comme des algorithmes de calcul non-local (pour ne pas utiliser le mot *global*).

L'amélioration de l'algorithme GBP (ou d'apprentissage pour être plus général) constitue un des sujets centraux de la recherche en réseaux de neurones. Les auteurs se divisent en deux écoles dont l'une travaille sur des techniques heuristiques, en se restreignant au cadre des algorithmes de calcul local (e.g. Barto & Sutton, 1981; Sutton 1986; Robert A.J., 1988), et l'autre essaie d'adapter des méthodes plus sophistiquées au problème de l'apprentissage (Pham Dinh, Wang & Yassine 1988; Becker & Le Cun 1988).

Dans le paragraphe §V.2., nous allons présenter certaines heuristiques pour l'amélioration de l'algorithme GBP, et dans le paragraphe §V.3. nous verrons une adaptation de la méthode de Région de Confiance qui se montre beaucoup plus efficace que le GBP, surtout vis-à-vis des cas difficiles pour le GBP.

---

[7] Attention, il ne faut pas confondre avec la notion des calculs locaux introduite au chapitre II.

V.2. Méthodes heuristiques

Les heuristiques présentées ici concernent essentiellement les taux d'apprentissage. Sauf indication spécifique, le terme "une étape d'apprentissage" peut signifier un parcours de tous les patterns à apprendre ou une application du GBP à un seul pattern. Par contre, le terme "une itération d'apprentissage" renvoie toujours à un parcours de tous les patterns à apprendre, comme cela a été défini dans le chapitre I.

Dans la version originale du GBP, le taux d'apprentissage à chaque étape est le même pour tous les poids (correspondant aux composants du gradient), ce qui est conforme à la méthode du gradient. Pourtant, les poids ne jouent pas le même rôle dans la formation de la surface d'erreur. La première idée qui vient à l'esprit est de choisir un taux pour chaque poids :

Si en deux étapes successives d'apprentissage, les deux variations d'un poids $\Delta w_{ij}(t-1)$ et $\Delta w_{ij}(t)$ possèdent un signe opposé, alors le poids w_{ij} oscille et le taux d'apprentissage pour ce poids doit être diminué; Dans le cas contraire, le taux doit être augmenté.

Pour que l'évolution des poids soit compétitive, la somme des taux pour tous les poids doit être une constante. La pertinence de ces idées est discutée dans le papier de Sutton (1986). Voici quatre points qui permettent de résumer ces heuristiques :

1) Chaque poids doit avoir son taux individuel d'apprentissage. Dans l'espace de la recherche, un taux d'avancement approprié pour une dimension particulière ne l'est peut-être pas autant pour une autre dimension.

2) Il faut autoriser la variation du taux d'apprentissage pour chaque dimension de poids au cours du temps, car la propriété de la surface de la fonction d'erreur varie généralement selon la région de la même dimension.

3) Quand deux variations d'un poids $\Delta w_{ij}(t-1)$ et $\Delta w_{ij}(t)$ possèdent le même signe, il arrive souvent que la pente de la surface d'erreur dans la dimension w_{ij} soit petite et par conséquent le poids va continuer à avancer sur un certaine distance dans la même direction. Donc, une augmentation du taux d'apprentissage lui permet de parcourir cette distance en moins d'étapes d'apprentissage.

4) Quand deux variations d'un poids $\Delta w_{ij}(t-1)$ et $\Delta w_{ij}(t)$ possèdent un signe opposé, il arrive souvent que la pente de la surface d'erreur dans la dimension w_{ij} soit grande. La diminution du taux d'apprentissage empêche l'oscillation du

poids et réduit le "risque" que le poids aille trop loin dans la direction non appropriée.

Ces heuristiques sont très intuitives et n'assurent pas toujours la réussite de l'accélération du procédé d'apprentissage. La raison est simple : Le gradient ne détermine la variation de la surface d'erreur que pour une région extrêmement proche du point actuel dans l'espace des poids. Même la variation entre deux gradients consécutifs est loin d'être suffisante pour caractériser la surface. Le même signe de $\Delta w_{ij}(t-1)$ et $\Delta w_{ij}(t)$ pourrait donc très bien donner l'impression fausse que le poids va continuer à évoluer dans ce sens.

La concrétisation de ces heuristiques permet de déduire certaines règles d'apprentissage qui constituent des variations importantes de l'algorithme GBP. Par exemple, l'introduction du momentum dans la formule pour modifier les poids (Rumelhart, Hinton, & Williams 1986) peut être considérée comme une application de ces heuristiques. On va présenter aussi brièvement les règles Delta-Delta et Delta-Barre-Delta.

Momentum :

La formule (1.11') dans le chapitre I peut être réécrite sous la forme :

$$\Delta w_{ij}(t) = -\gamma \frac{\partial C_t(W)}{\partial w_{ij}(t)} + \alpha \Delta w_{ij}(t-1) = -\gamma \sum_{l=0}^{t} \alpha^l \frac{\partial C_{t-l}(W)}{\partial w_{ij}(t-l)}$$

où $0 \leq \alpha < 1$ et $C_t(W)$ représente la fonction d'erreur du réseau pour une paire de patterns présentée à l'instant t. Donc $\Delta w_{ij}(t)$ est représentée comme une somme exponentiellement pondérée (par α^l). Comme α^l décroît rapidement quand l augmente, on peut dire que si les gradients consécutifs :

$$\frac{\partial C_{t-l}(W)}{\partial w_{ij}(t-l)}$$

dans les dernière étapes d'itération sont de même signe, la valeur absolue de la somme a plutôt tendance à être augmentée; et si les gradients oscillent, la valeur absolue de la somme sera diminuée. Ces tendances nous permettent de considérer la technique de momentum comme une réalisation des heuristiques présentées au-dessus.

La règle Delta-Delta :

L'idée de cette règle est de modifier chaque poids selon la règle de l'algorithme GBP mais avec un taux d'apprentissage variable dont la règle de variation est déduite de l'application de la méthode du gradient à une autre fonction d'erreur qui dépend de l'ensemble des taux d'apprentissage. Plus précisément, si la formule de modification des poids s'écrit comme :

$$\Delta\overline{w}_{ij}(t)=-\gamma_{ij}\frac{\partial C_t(W)}{\partial w_{ij}(t)}$$

avec $\Delta\overline{w}_{ij}(t)=\overline{w}_{ij}(t+1)-w_{ij}(t)$ et $\Delta w_{ij}(t)=w_{ij}(t+1)-w_{ij}(t)$ quand $\gamma_{ij}=\gamma_{ij}(t)$, $\gamma_{ij}(t+1)=\gamma_{ij}(t)+\Delta\gamma_{ij}$, on peut introduire une (seconde) fonction d'erreur :

$$E_t(\Gamma(t))=0.5\ \|S(\overline{W}(t+1))-Y(t)\|^2$$

ici $\Gamma(t)$ représente l'ensemble des taux γ_{ij}, $S(\overline{W}(t+1))$ représente le vecteur d'output du réseau qui dépend de l'entrée $X(t)$ à l'instant t, de $\overline{W}(t+1)$, ensemble des poids $\{\overline{w}_{ij}(t+1)\}$ qui dépendent du $\Gamma(t)$, et de $Y(t)$ étant encore la sortie désirée pour l'entrée $X(t)$ à l'instant t.

On considère $E_t(\Gamma(t))$ comme une fonction de $\Gamma(t)$. L'objectif est de trouver la variation $\Delta\gamma_{ij}$ telle que le taux de l'étape suivante $\gamma_{ij}(t+1)=\gamma_{ij}(t)+\Delta\gamma_{ij}$ soit modifié dans la direction qui minimise la fonction $E_t(\Gamma(t))$.

On peut calculer (d'une manière très semblable à la déduction de l'algorithme GBP dans le chapitre I) le gradient de $E_t(\Gamma(t))$ par rapport à $\Gamma(t)$:

$$\frac{\partial E_t(\Gamma(t))}{\partial\gamma_{ij}(t)}=-\frac{\partial C_t(\overline{W})}{\partial\overline{w}_{ij}(t)}\frac{\partial C_t(W)}{\partial w_{ij}(t)}.$$

La partie $\dfrac{\partial C_t(\overline{W})}{\partial\overline{w}_{ij}(t)}$ est formelle et ne peut être réellement calculée. Elle sera approché par $\dfrac{\partial C_{t+1}(W)}{\partial w_{ij}(t+1)}$. Ainsi on aura la formule :

$$\frac{\partial E_t(\Gamma(t))}{\partial\gamma_{ij}(t)}\approx-\frac{\partial C_{t+1}(W)}{\partial w_{ij}(t+1)}\frac{\partial C_t(W)}{\partial w_{ij}(t)}$$

Cette formule permet de proposer une modification de $\gamma_{ij}(t)$ par la formule suivante :

$$\Delta\gamma_{ij}(t) = \varepsilon \frac{\partial C_{t+1}(W)}{\partial w_{ij}(t+1)} \frac{\partial C_t(W)}{\partial w_{ij}(t)}.$$

Cette formule signifie que si deux gradients consécutifs $\frac{\partial C_{t+1}(W)}{\partial w_{ij}(t+1)}$ et $\frac{\partial C_t(W)}{\partial w_{ij}(t)}$ ont le même signe, alors le taux d'apprentissage γ_{ij} sera augmenté; sinon il sera diminué. Donc cette règle réalise les heuristiques. La même règle a été déduite par Sutton et Robert (Barto & Sutton, 1981; Robert, A. 1988) dans le cadre de la règle de LMS (Linear Mean Square) de Widrow & Hoff (1960).

La règle Delta-Barre-Delta :

Représentons :

$$\delta(t) = \frac{\partial C_t(W)}{\partial w_{ij}(t)}$$

et

$$\bar{\delta}(t) = (1-\theta)\delta(t) + \theta\bar{\delta}(t-1)$$

ici $0<\theta<1$ et $\bar{\delta}(t)$ est une moyenne entre la dérivée $\delta(t)$ à l'instant t et la moyenne $\bar{\delta}(t-1)$ de l'étape précédente. La règle s'écrit comme :

$$\Delta\gamma_{ij}(t) = \begin{cases} \kappa & \text{si } \bar{\delta}(t-1)\delta(t)>0 \\ -\phi\gamma_{ij}(t) & \text{si } \bar{\delta}(t-1)\delta(t)<0 \\ 0 & \text{sinon} \end{cases}$$

ici si $\bar{\delta}(t-1)$ et $\delta(t)$ ont le même signe γ_{ij} est incrémenté d'un facteur κ; sinon il est diminué d'un facteur ϕ de son ancienne valeur.

On peut encore proposer beaucoup d'autres réalisations heuristiques pour l'amélioration de l'algorithme GBP. Mais une question se pose naturellement : est-ce que ces techniques apportent vraiment une remarquable accélération au GBP? On sait que pour le problème d'apprentissage d'un classifieur linéaire, ces techniques sont très efficaces (Robert, A. 1988) : en fait, on peut même en prouver mathématiquement l'efficacité. Mais pour des problèmes compliqués (e.g. pour des réseaux dont les neurones sont non linéaires), il sera très difficile de les justifier. Faute de preuves théoriques, il faut appliquer ces techniques à plusieurs réseaux et comparer leurs

performances avec celles du GBP original afin de dégager les conclusions statistiques. D'après les résultats rapportés dans Sutton (1986) et Robert (1988) il semblerait que pour certains cas, le nombre des étapes d'apprentissage soit largement réduit.

En conlusion, les heuristiques basées sur des calculs locaux constituent une approche très importante à l'amélioration de l'algorithme d'apprentissage notamment celui de GBP. Bien qu'ils manquent de rigueur, ils sont efficaces dans certains cas. Les informations utilisées sont bien locales, ce qui rend ces techniques relativement faciles à implanter.

V.3. Une méthode basée sur la technique de Région de Confiance

Dans le paragraphe §V.1., on a déjà discuté les principaux problèmes qui rendent l'algorithme du GBP inefficace. Ce sont les choix du taux d'apprentissage et l'orientation de la direction du gradient. Une des conséquences de l'inefficacité du GBP oblige l'architecture du réseau à être soigneusement conçue pour réussir des applications : une modification mineure peut causer une dégradation très importante de la performance (Wang 1988; Pham Dinh, Wang et Yassine, 1988; Schreiber *et al.* 1988). Donc une étude approfondie de la nature du problème d'application s'impose avant qu'une architecture "appropriée" puisse être trouvée, autrement dit la conception des réseaux est très dépendante des problèmes d'application.

Dans le paragraphe précédent, des heuristiques ont été proposées pour choisir des paramètres d'apprentissage afin de rendre la recherche à chaque étape d'apprentissage plus "incorporée" à la surface d'erreur. Mais le handicap de ces heuristiques est qu'elles sont basées seulement sur des informations du gradient, qui, selon la manière dont elles sont extraites, sont loin d'être suffisantes pour caratériser la surface. Il faut donc faire appel aux informations du second ordre, ou leurs approchées, pour éviter la recherche aveugle et augmenter l'efficacité de manière systématique (voir aussi Becker & Le Cun 1988).

Un algorithme basé sur la technique de Région de Confiance (RC) a été développé. C'est une technique itérative comme celle de Newton, spécialement adaptée pour des problèmes de minimisation des fonctions non-convexes. La fonction d'erreur est successivement approchée par une série des formes quadratiques et est réduite via la direction qui minimise chacune de ces formes. La taille de la région sur laquelle la fonction est approchée est déterminée dynamiquement selon la qualité de l'approximation précédente. Comme l'approximation par des formes quadratiques est du second ordre, dépendant non seulement du gradient mais aussi de la Hessienne (en pratique, on utilise l'approximation de la Hessienne), la recherche à chaque étape d'apprentissage n'est plus dirigée uniquement par la direction du gradient.

Dans les paragraghes §V.3.1. et §V.3.2., nous allons présenter la méthode de Région de Confiance en sa forme originale avec les propriétés

associées. Dans les paragraghes §V.3.3. et §V.3.4., nous allons discuter la mise au point des différentes étapes de la méthode et les résultats théoriques sur la convergence. L'algorithme pratique pour des problèmes d'apprentissage sera donné au paragraphe §V.3.5.. Les résultats expérimentaux seront donnés au paragraphe §V.3.6..

V.3.1. Présentation de la méthode de Région de Confiance

Une méthode pour la minimisation sans contraintes :

On considère le problème d'optimisation sans contrainte :

$$\min \{ f(x) : x \in \mathbb{R}^n \} \qquad (P)$$

où f est une fonction de $\mathbb{R}^n$ à $\mathbb{R}$. Nous représentons le gradient de f au point x_k par g_k, la Hessienne de f en x_k (ou une approximation de la Hessienne) par H_k, qui donc sera appellé "quasi-Hessienne". L'objectif principal du paragraphe est de décrire et d'analyser une technique pour la résolution du problème (P).

L'approche que nous allons présenter est bien connue (Moré, 1983; Sorensen, 1981). A chaque nouvelle itération le point est obtenu par un procédé de minimisation d'un modèle (forme) quadratique local à la fonction objectif sur une région, restreinte et sphérique, centrée autour du point de l'itération actuelle. Le diamètre de cette région est augmenté et contracté d'une manière contrôlée, qui est basée sur la qualité de prédiction du comportement de la fonction objectif par le modèle local. Il est possible de contrôler les itérations de manière à ce que la convergence soit forcée à partir de n'importe quel point de départ, lorsque certaines conditions raisonnables de la fonction sont satisfaites.

Nous allons présenter dans le paragraphe §V.3.2. quelques propriétés de convergence pour cette méthode. Il est montré que l'on pourrait espérer (mais pas assurer) que les itérations vont converger vers un point satisfaisant les conditions nécessaires du second ordre pour un minimum.

Schéma général de la méthode :

La méthode de RC calcule une séquence de points (vecteurs) via la résolution à chaque étape d'un problème quadratique avec une contrainte de norme euclidienne

$$\min \{q_k(d) : \|d\| \leq \delta_k\} \qquad (P_k)$$

où q_k est une approximation quadratique de la variation de f au point x_k définie comme :

$$q_k(d) = \langle g_k, d\rangle + \frac{1}{2}\langle H_k d, d\rangle.$$

δ_k, qui est une valeur strictement positive, est le rayon de "confiance".

Dans les algorithmes de RC (Région de Confiance), le calcul du gradient de la fonction objectif est demandé. L'utilisation des informations du premier ordre conduit en général au point stationnaire du premier ordre. En incorporant les information du second ordre $H_k=\nabla^2 f(x_k)$, ces algorithmes pourraient satisfaire les conditions nécessaires du second ordre pour (P). Dans ce cas la méthode de RC peut être considérée comme la méthode de Newton modifiée et appliquée à la recherche du zéro du gradient de la fonction objectif. Le procédé peut être décrit comme suit : Calculer la direction d_k, qui est la solution de (P_k), puis examiner la qualité de l'approximation locale $q_k(d)$ à partir de laquelle une décision sera prise :

* Si l'approximation est satisfaisante, alors la solution d_k de (P_k) constitue un nouveau point d'itération et le rayon de confiance est augmenté ;

* Sinon, le point actuel reste inchangé (pas d'itération), le rayon de confiance sera diminué, et chercher à nouveau la solution d_k de (P_k). Ce processus de recherche se répètera jusqu'au moment où q_k fournit une approximation satisfaisante (qui nécessairement existe au moins pour des rayons très petits car le premier terme du premier ordre dans q_k deviendra dominant si $\|g_k\| \neq 0$).

Il est clair que ce schéma général peut être réalisé de plusieurs façons différentes. La qualité de l'approximation est généralement examinée via le calcul de la quantité suivante qui s'appelle le "coefficient de qualité " :

$$r_k = \frac{f(x_k) - f(x_k+d_k)}{q_k(0) - q_k(d_k)} \qquad (5.1)$$

Le numérateur représente la réduction actuelle de f si x_k est remplacé par x_k+d_k, le dénominateur représente la réduction prédite selon l'approximation quadratique. Il est clair que, dans un algorithme de RC, l'effort principal est focalisé sur la résolution du problème (P_k) pour déterminer le point d'itération à chaque étape.

Les algorithmes de RC se différencient par leurs stratégies pour résoudre approximativement (P_k).

L'algorithme abstrait :

En résumé, le schéma ci-dessus peut être écrit sous forme d'un algorithme abstrait :

1. Supposons $x_0 \in \mathbb{R}^n$, $\delta_0 > 0$ et H_0 sont donnés
2. Pour k=0,1,2,...

a) Calculer $g_k = \nabla f(x_k)$. Si $g_k = 0$, x_k sera pris comme solution optimale. Arrêter! (voir §2.3) ;

b) Déterminer une solution d_k du problème (P_k) et calculer le coefficient de qualité r_k via la formule (5.1) ;

c) Mettre à jour l'itération ;

d) Mettre à jour le rayon de confiance et la quasi-Hessienne, puis aller à l'étape 2 ;

V.3.2. Caractéristiques du problème de recherche locale

Le problème local (P_k) est réduit à celui de minimisation d'une forme quadratique dans une boule :

$$\min \{ \langle g,d \rangle + \frac{1}{2} \langle d,Hd \rangle : \|d\| \leq \delta \} \qquad \text{(LP)}$$

où le g est un vecteur de dimension n, H est une matrice symmetrique n*n et δ est une valeur positive.

Comme la contrainte $\{d \in \mathbb{R}^n : \|d\| \leq \delta\}$ est un ensemble compact, le problème (LP) admet toujours une solution. Si en plus H est définie positive, alors la solution est unique. Nous allons donner ici une discussion de l'aspect théorique du problème (LP) afin d'exposer la nature des difficultés de calcul qui peuvent être rencontrées.

Lemme 5.1 (Gay 1981, Sorensen 1982) d^* est une solution de (LP) si et seulement s'il existe un $\mu \geq 0$ tel que :

(i) $(H+\mu I)$ semi définie positive ;

(ii) $(H+\mu I)d^* = -g$

(iii) $\|d^*\| \leq \delta$ et $\mu(\|d^*\| - \delta) = 0$. ◊

La solution de (LP) est simple si (LP) n'admet pas de solution sur la frontière de $\{d\in\mathbb{R}^n : \|d\|\leq\delta\}$, car dans ce cas, H est définie positive et $\|H^{-1}g\|<\delta$. Le scalaire non-négatif μ dans **Lemme 5.1** est appellé le multiplicateur de Lagrange associé à la contrainte $\|d\|^2\leq\delta^2$ ($<=>\|d\|\leq\delta$). Le problème (LP) représente un cas intéréssant des problèmes d'optimization non convexes.

On définit une fonction ϕ sur $\{\mu\in\mathbb{R}: H+\mu I$ non singulière$\}$ par

$\phi(\mu)=\|d(\mu)\|$

où $d(\mu)$ est la solution de $(H+\mu I)d= -g$.

Notons par $\lambda_1\leq\lambda_2\leq...\leq\lambda_n$ les valeurs propres de H et par $u_1,u_2,...,u_n$ les vecteurs propres correspondants. Notons également $\mathcal{N}(H-\lambda_1 I)$ l'espace nul de $H-\lambda_1 I$ (ou l'espace des vecteurs propre par rapport à λ_1), et $J_1=\{i : \lambda_i =\lambda_1\}$.

Il est clair que la solution du problème (LP) est en relation étroite avec l'équation non linéaire $\phi(\mu)=\delta$ pour μ dans $]-\lambda_1, +\infty[$. Pour être plus précis, il s'agit du cas où (LP) admet une solution sur la frontière de la région de contrainte et il y a un $\mu>\max(0,-\lambda_1)$ dans le **Lemme 5.1**.. Dans ce cas, l'algorithme de Hebden (présenté plus loin) est fiable et efficace pour résoudre $\phi(\mu)-\delta=0$. Ce qu'on appelle le "cas dur" correspond à la situation où le multiplicateur μ dans le **Lemme 5.1** doit être égal à $-\lambda_1$.

Considérons maintenant la solution de l'équation

$$(H+\mu I)d=-g \qquad \text{pour } \mu\geq-\lambda_1 \qquad (5.2)$$

utilisant la base composée des vecteurs propres de H. On obtient alors que si $\mu>-\lambda_1$ alors la solution à (5.2) est définie par :

$$u_i^t d = \frac{-u_i^t g}{(\lambda_i+\mu)} \qquad \text{pour } i=1,..., n.$$

par conséquent $\phi(\mu)=\|d(\mu)\|=\left[\sum_{i=1}^{n}\frac{(u_i^t g)^2}{(\lambda_i+\mu)^2}\right]^{1/2}$.

La fonction $\phi(\mu)$ est positive et strictement décroisante dans $]-\lambda_1, +\infty[$, avec $\lim_{m\to\infty}\phi(\mu) = 0$, l'équation $\phi(\mu)=\delta$ a donc au maximum une solution dans cet intervalle. Les résultats dans le lemme suivant sont très utiles pour comprendre l'algorithme décrit par la suite pour résoudre (LP).

Lemme 5.2 Si $g \neq 0$ dans (LP)

1° Si $\lambda_1 > 0$ (i.e. H est définie positive) alors :

(i) Si $\|H^{-1}g\| \leq d$ alors $d = -H^{-1}g$ est la solution unique de (LP) (cas $\mu = 0$ dans **Lemme 5.1** et $\phi(\mu) - \delta = \|d(\mu)\| - \delta \leq 0$) ;

(ii) Sinon (cas $\|H^{-1}g\| > \delta$), si $\phi(0) - \delta = \|d(0)\| - \delta > 0$, alors il y a un unique solution $\mu > 0$ à $\phi(\mu) = \delta$.

2° Si $\lambda_1 = 0$ (i.e. H est seulement semi définie positive) alors :

(i) Si $\|(H-\lambda_1 I)^+ g\| \leq \delta$ et $(H-\lambda_1 I)(H-\lambda_1 I)^+ g = g$ (cette égalité aura lieu si et seulement si g est perpendiculaire à $N(H-\lambda_1 I)$) alors chaque $d = -(H-\lambda_1 I)^+ g + u$, avec $u \in \mathcal{N}(H-\lambda_1 I)$ et $\|d\|^2 = \|(H-\lambda_1 I)^+ g\|^2 + \|u\|^2 \leq \delta^2$ est une solution du problème (LP), (cas $\mu = 0$ dans le **Lemme 5.1**).

(ii) Si $\|(H-\lambda_1 I)^+ g\| > \delta$, alors $\phi(\mu) - \delta = 0$ admet une unique solution dans $]-\lambda_1, +\infty[$;

3° Si $\lambda_1 < 0$ alors :

(i) Si $\| (H-\lambda_1 I)^+ g \| \leq \delta$ et $(H-\lambda_1 I)(H-\lambda_1 I)^+ g = g$ alors chaque $d = -(H-\lambda_1 I)^+ g + u$, avec $u \in \mathcal{N}(H-\lambda_1 I)$ et $\|d\|^2 = \|(H-\lambda_1 I)^+ g\|^2 + \|u\|^2 = \delta^2$ est une solution du problème (LP), ($\mu = -\lambda_1 > 0$ dans le **Lemme 5.1**).

(ii) If $\|(H-\lambda_1 I)^+ g\| > \delta$ alors $\phi(\mu) - \delta = 0$ admet une unique solution dans $]-\lambda_1, +\infty[$, ($\mu > -\lambda_1 > 0$ dans le **Lemme 5.1**).

Si $g = 0$ dans le problème (LP), alors l'ensemble des solutions de (LP) est :

(i) $\{0\}$ si $\lambda_1 > 0$;

(ii) $\{d \in \mathcal{N}(H-\lambda_1 I) : \|d\| \leq \delta\}$ si $\lambda_1 = 0$;

(iii) $\{d \in \mathcal{N}(H-\lambda_1 I) : \|d\| = \delta\}$ si $\lambda_1 < 0$ ($\mu = -\lambda$ dans le **Lemme 5.1**). ◊

(pour les interprétations et la démonstration voir Rockafellar 1970; Lancaster 1969; Pham Dinh, Wang & Yassine, 1988). La situation où $g = 0$ dans (LP) est étroitement lié à la théorie des variations spectrales.

V.3.3. Algorithmes pour la mise à jour des différentes étapes

Dans les cinq sous-sections suivantes, nous allons discuter les trois points, a), b) et d), mentionnés dans l'algorithme abstrait ci-dessus et proposer des algorithmes pratiques pour les résoudre. Certains résultats peuvent être trouvés dans (Gay 1981, Sorensen 1982, Moré 1983, Dennis & Schnabel 1983).

Mise à jour du rayon de confiance :

Le rayon de confiance est calculé selon la règle suivante : Supposons que $0<\mu<\eta<1$ et $0<\gamma_1<\gamma_2<1<\gamma_3$ soient des constantes prédéfinies,

1. Si $r_k \leq \mu$ alors $\delta_{k+1}=\Delta \in [\gamma_1 \delta_k, \gamma_2 \delta_k]$
2. Si $r_k \leq \eta$ alors $\delta_{k+1} \in [\gamma_2 \delta_k, \delta_k]$
   sinon $\delta_{k+1} \in]\delta_k, \gamma_3 \delta_k]$

Ici le coefficient de qualité r_k prend μ comme seuil inférieur au-dessous duquel le rayon δ_{k+1} sera forcément diminué, η comme seuil supérieur au-dessus duquel le rayon δ_{k+1} sera forcément augmenté, et entre les deux δ_{k+1} peut rester inchangé ou légèrement varier. Les choix de ces paramètres dépendent du problème, ils n'ont d'influence que sur la vitesse de convergence. Dans le paragraphe §V.3.4. nous donnerons des choix pratiques pour nos applications.

Mise à jour de l'itération :

Dans la méthode de RC, la mise à jour de chaque itération est souvent conditionnée par un paramètre s tel que $0<s<0.25$, qui doit (c'est vivement conseillé) être constant tout au long des itérations.

1. Si $r_k<s$ alors $x_{k+1}=x_k$
2. Si $r_k \geq s$ alors $x_{k+1}=x_k+d_k$

Il faut remarquer qu'une descente significative de la fonction est exigée ($r_k \geq s$) pour permettre de passer de x_k à x_k+d_k.

Mise à jour de la quasi-Hessienne H_k *:*

Il est souvent très difficile et très coûteux de calculer la vraie Hessienne d'une fonction, ce qui est notre cas avec la fonction d'erreur dans les réponses d'un réseau. Heureusement, cette Hessienne dans la méthode de RC peut être remplacée par les informations "accumulées" des variations successives du gradient. Dans cette section nous donnons deux algorithmes pour mettre à jour H_k : la quasi-Hessienne.

Supposons que $\gamma_k=\nabla f(x_{k+1})-\nabla f(x_k)$, $s_k=x_{k+1}-x_k$ et que H_0 ait été initialisée comme une matrice symétrique.

(i) Formule de rang 1 (DFP)

$$H_{k+1}=H_k+\alpha_k u_k u_k^t$$

où $\alpha_k=-1/s_k^t.(H_k s_k-\gamma_k)$ et $u_k=H_k s_k-\gamma_k$

(ii) Formule de rang 2 (BFGS)

$$H_{k+1}=H_k+\alpha_k u_k u_k^t+\beta_k v_k v_k^t$$

où $\alpha_k=1/s_k^t\gamma_k$, $u_k=\gamma_k$ et $\beta_k=-1/s_k^t.H_k s_k$, $v_k=H_k s_k$.

Il est prouvé (Minoux 1983, Gill & Murray 1972) que sous les conditions x_k-x^* et $\nabla^2 f(x^*)$ définie positive, les H_k calculés par les formules ci-dessus convergent vers $\nabla^2 f(x^*)$, la vraie Hessienne.

Les algorithmes pour résoudre le problème local (LP) :

Il faut distinguer globalement trois classes d'algorithmes qui sont utilisées pour traiter les différents cas du problème (LP); les cas où la matrice H est définie positive, ou semi définie positive, ou ni l'un ni l'autre. Ces 3 cas sont censés être traités de manières différentes.

Quand H est définie positive, ce cas correspond au 1° du **Lemme 5.2** et il suffit de résoudre un système souvent bien conditionné. Nous allons en revanche présenter l'algorithme de Hebden qui est généralement appliqué quand H est semi définie positive. Il faut ajouter que pour ce deuxième cas, il existe aussi des méthodes basées sur la technique de projection (voir Pham Dinh, Wang & Yassine 1988). Le cas le plus difficile est le "cas dur" où H n'est ni définie positive ni semi définie positive, ce qui correspond aux 2° (i) et 3° (ii) dans le **Lemme 5.2**.

Dans ce troisième "cas dur", sauf quand le système des vecteurs propres de H s'obtient facilement, reconnaître chacune des situations 2° (i) et 3° (ii) dans le **Lemme 5.2** pourrait aussi être très coûteux à cause du calcul de $(H-\lambda_1 I)^+ g$. Heureusement, il y a une alternative tout à fait acceptable due à Shultz, Schnabel & Byrd (1985) (Voir §2.2.5, et l'algorithme donné par la suite). Leur technique d'approximation pourrait effectuer la recherche en pratique aussi bien que les méthodes exactes qui sont, elles, coûteuses. Elle nécessite tout simplement le calcul d'un vecteur propre de H associé à λ_1.

L'algorithme de Hebden :

Nous allons décrire cet algorithme pour la résolution du problème (LP) dans le cas où l'équation non linéaire : $\phi(\mu)-\delta = 0$, avec

$$\phi(\mu)=\left[\sum_{i=1}^{n} \frac{(u_i^t g)^2}{(\lambda_i+\mu)^2}\right]^{1/2}$$

admet une racine unique dans $]-\lambda_1, +\infty[$.

(Reinsch 1971) et (Hebden 1973) ont observé indépendamment que pour résoudre (5.2), il faut tenir compte du fait que la fonction $\phi^2(\mu)$ est une fonction rationnelle de μ avec les pôles du second ordre sur un sous-ensemble des négatifs des valeurs propres de H (voir (5.2)).

La méthode de Newton qui est basée sur une approximation localement linéaire de $\phi(\mu)$ ne semble donc pas être la meilleure méthode pour résoudre (5.2) car elle ignore la structure rationnelle de $\phi^2(\mu)$. Au lieu de cela, l'itération peut être déduite selon une approximation localement rationnelle de ϕ. On introduit une fonction $\phi^*(\mu)=\gamma/(\alpha+\mu)$ avec les contraintes $\phi^*(\mu)=\phi(\mu)$, $\phi^{*'}(\mu)=\phi'(\mu)$ où μ est l'approximation actuelle de la racine μ^*. Cette approximation est améliorée par un $\hat{\mu}$ qui satisfait $\phi^*(\hat{\mu})=\delta$. Ceci résulte en l'itération suivante :

$$\mu_{k+1}=\mu_k+ \left(\frac{\phi(\mu_k)}{\phi'(\mu_k)}\right) * \left(\frac{\delta-\phi(\mu_k)}{\delta}\right) \qquad (5.3).$$

En effet, cet algorithme peut être vu comme l'algorithme de Newton appliqué à l'équation

$$\psi(\mu)=\frac{1}{\delta} - \frac{1}{\phi(\mu)} =0 \qquad \text{pour } \mu\in\,]-\lambda_1, +\infty[$$

La taux local de convergence de cette itération est quadratique au voisinage de la solution. Le nombre nécessaire d'itérations est souvent très petit pour produire une approximation acceptable de μ^*.De plus, l'itération (5.3) peut être implantée sans connaissance explicite du système des vecteurs propres de H. Cette remarque importante due à Hebden (1973) rend possible l'implantation de (5.3) seulement par la résolution d'un système linéaire, avec $(H+\mu I)$ comme matrice de coefficients.

$\phi(\mu)=\|d(\mu)\|$

$$\phi'(\mu)=-(\frac{1}{\phi(\mu)})\, d(\mu)^t\, (H+\mu I)^{-1}\, d(\mu)$$

où $(H+\mu I)\, d(\mu)=-g$. Par conséquent, l'algorithme de Hebden peut être décrit comme suit : étant donné $\mu_0>0$ avec $H+\mu_0 I$ définie positive et $\phi(\mu_0)>0$. pour k=0,1,2,...faire :

1) Factoriser $(H+\mu_k I)=R^tR$

2) Résoudre $Rp=-g$ et $R^tq=p$

3) Posons $\mu_{k+1}=\mu_k+\left(\frac{\|p\|}{\|q\|}\right)^2\left(\frac{\|p\|-\delta}{\delta}\right)$

k = k+1 et retouner à 1).

Dans cet algorithmes R^tR est une factorisation de Cholesky de la matrice $(H+\mu I)$ avec R qui est triangulaire supérieure. La fonction ψ est convexe et strictement décroissante sur $]-\lambda_1, +\infty[$ (Reinsch 1971), ce qui signifie que la méthode de Newton, débutée en un point quelconque $\mu_0 \in]-\lambda_1, +\infty[$ avec $\psi(\mu_0)>0$ (i.e. $\phi(\mu_0)-\delta>0$), génère une suite monotone et croissante qui converge vers la solution de $\phi(\mu)-\delta=0$.

La méthode de la plus forte courbure négative :

La méthode de Hebden est utilisée quand la matrice H est au moins semi définie positive. Si H n'est ni définie positive, ni semi définie positive, il est préférable d'utiliser la stratégie suivante (due à Shultz, Schnabel & Byrd 1985) pour trouver une solution approximative au problème (LP). C'est le "cas dur" (correspondant à $\mu= -\lambda_1$ dans les **Lemme 5.1** & **5.2**).

1° Prendre un $\alpha \in]-\lambda_1, c*\max(|\lambda_1|, \lambda_n)]$, où c>1 est une constante prédéfinie ;

2° Résoudre $p=-(H+\alpha I)^{-1}g$;

3° Si $\|p\|>\delta$, alors appliquer la méthode de Hebden à $H=H+\alpha I$ pour trouver une direction d ;

4° Si $\|p\|=\delta$, alors d=p.

5° Si $\|p\|<\delta$, alors $d=p+\kappa u_1$, où u_1 est un vecteur propre de H associé à λ_1 et κ est choisi de telle manière que $\|d\|=\delta$ et $\text{signe}(\kappa)=\text{signe}(u_1^t\, p)$.

Comparée avec l'algorithme de Hebden, la seule modification introduite par cette méthode se trouve à l'étape 5) où on procède comme si $\mu=-\lambda_1$ dans les **Lemmes 5.1** & **5.2**.

La direction d obtenue par cette méthode donne une descente de forme quadratique et ce d'autant plus que la direction négative de u_1 est suffisament penchée. Elle satisfait les **conditions** pratiques **1** et **2** qui seront présentées ultérieurement.

Tests de convergence (conditions d'arrêt) :

Nous pouvons utiliser les critères suivants pour arrêter l'itération de l'algorithme de RC :

* Le test de la condition nécessaire du premier ordre :

Si $\nabla f(x_k)=0$ alors arrêter.

* Le test de la condition suffisante du second ordre : Quand la condition nécessaire du premier ordre est satisfaite, les itérations peuvent être arrêtées ou continuées jusqu'au moment où les points trouvés vérifient la condition suffisante du second ordre comme suit :

i) Si H_k est non définie positive, alors continuer les itérations (Voir **Lemme 5.2**, pour la résolution de (LP) dans le cas où g=0) ;

ii) Si $\nabla^2 f(x_k)$ est définie positive, alors arrêter le processus d'itération; Sinon $H_k=\nabla^2 f(x_k)$ et recommencer l'algorithme à partir de l'étape d) (Voir §V.3.1).

En pratique, la condition suffisante du second ordre n'est pas souvent employée car l'évaluation de la Hessienne $\nabla^2 f(x_k)$ n'est pas toujours possible.

V.3.4 Les résultats de convergence

Nous terminerons ce paragraphe par quelques résultats très connus sur la convergence de l'algorithme de RC.

Théorème 5.3.(Moré 1983; Sorensen 1983)

Si la fonction est dérivable, bornée inférieurement sur $\mathbb{R}^n$, et si ∇f est continue uniformément alors :

$\lim_{k\to+\infty} \|\nabla f(x_k)\| = 0$

Théorème 5.4. (Moré 1983; Sorensen 1983)

Si la fonction f est deux fois continûment dérivable et bornée inférieurement sur $\mathbb{R}^n$, et si $\nabla^2 f$ est bornée sur l'ensemble $\{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ alors les algorithmes de RC avec $H_k = \nabla^2 f(x_k)$ possèdent les propriétés suivantes :

1° $\lim_{k \to +\infty} \|\nabla f(x_k)\| = 0$;

2° Si $\{x_k\}$ est borné alors il existe un point limite x^* avec $\nabla^2 f(x^*)$ semi définie positive ;

3° Si x^* est un point limite isolé de $\{x_k\}$ alors $\nabla^2 f(x^*)$ est semi définie positive ;

4° Si $\nabla^2 f(x^*)$ est non singulière pour un certain point de limite x^* de $\{x_k\}$ alors :

a- $\nabla^2 f(x^*)$ est définie positive ;

b- $\lim x_k = x^*$ et il existe un $\varepsilon > 0$ et un entier K tel que $\delta_k > \varepsilon$ pour tous $k > K$;

c- La convergence est superlinéaire.

Dans les deux théorèmes ci-dessus, nous faisons l'hypothèse implicite que chaque étape de l'algorithme de RC peut être effectuée d'une manière exacte. Par exemple, d_k est supposé être exact. Apparemment, ceci peut ne pas être toujours vrai en pratique. Une question se pose : quelles sont les conditions que chaque direction calculée doit satisfaire pour que les propriétés de convergence dans les théorèmes ci-dessus soient préservées ?

Avant de proposer ces conditions pratiques de convergence, on définit d'abord la notation suivante : Représentons la direction calculée par $d(g, H, \mathrm{d})$ (dépendant des g, H et d),

$$\mathrm{pred}(g, H, \delta) = -\langle g, d(g, H, \delta)\rangle - \frac{1}{2}\langle d(g, H, \delta), Hd(g, H, \delta)\rangle$$

Les conditions, que la stratégie de la sélection de la direction à chaque étape doit satisfaire, sont :

Condition 1

Il existe $c_1, s_1 > 0$ tel que $\forall\, g \in \mathbb{R}^n$, $\forall\, H \in \mathbb{R}^{n \times n}$ symétrique et $\forall\, \delta > 0$:

$$\mathrm{pred}(g, H, \delta) \geq c_1 \|g\| \min\left(\delta, s_1 \frac{\|g\|}{\|H\|}\right).$$

Condition 2

Il existe $c_2>0$ tel que $\forall\, g\in\mathbb{R}^n$, $\forall\, H\in\mathbb{R}^{n\times n}$ symétrique et $\forall\, \delta>0$:
$pred(g, H, \delta) \geq c_2.(-\lambda_1(H)).\delta^2$.

Ici λ_1 est la plus petite valeur propre de H.

Condition 3

Si la matrice H est définie positive et $\|H^{-1}g\|\leq\delta$ alors :
$d(g, H, \delta) = -H^{-1}g$.

Théorème 5.5. (Shultz, Schnabel & Byrd 1985)

Supposons que $f : \mathbb{R}^n\mapsto\mathbb{R}$ soit deux fois continûment dérivable et bornée inférieur, et que $H(x)=\nabla^2 f(x)$ satisfasse $\|H(x)\|\leq\beta_1$ pour tous $x\in\mathbb{R}^n$. Supposons aussi qu'un algorithme pratique de RC soit appliqué à f(x), en commencant par un point $x_0\in\mathbb{R}^n$, et génère une suite $\{x_k\}$, $x_k\in\mathbb{R}^n$, k=1,2,..., alors :

1° Si d(g, H, δ) satisfait la **Condition 1** et $\|H_k\|\leq\beta_2$ pour tous k, alors $\nabla f(x_k)$ converge vers 0 (le point stationnaire de la convergence du premier ordre).

2° Si d(g, H, δ) satisfait les **Conditions 1** et **3**, $H_k=H(x_k)$ pour tous k, H(x) est continue satisfaisant la condition de Lipschitz avec le coefficient L, et x^* est un point limite de $\{x_k\}$ avec $H(x^*)$ définie positive, alors x_k converge q-quadratiquement vers x^*.

3° Si d(g, H, δ) satisfait les **Conditions 1** et **2**, $H_k=H(x_k)$ pour tous k, H(x) est uniformément continue, et $\{x_k\}$ converge vers x^*, alors $H(x^*)$ est semi définie positive (le point stationnaire de la convergence du second ordre, avec le 1°)

V.3.5. L'algorithme pratique adapté au problème d'apprentissage

On présente dans ce paragraphe, une version pratique de l'algorithme de RC adapté pour notre problème d'apprentissage.

En appliquant la méthode de RC au problème d'apprentissage dans des réseaux multicouches, il est nécessaire de prendre quelques précautions spéciales dans la programmation car la méthode a besoin du gradient (représenté explicitement) de la fonction d'erreur par rapport à tous les poids de connexions au lieu du gradient par rapport aux entrées totales (des variables intermédiaires,

voir chapitre I) comme cela est le cas pour l'algorithme de GBP. La fonction d'erreur doit aussi très souvent être évaluée pour l'ensemble des poids temporels (pour le calcul du coefficient de la qualité).

Pour accorder les notations, nous utiliserons encore ici f pour représenter la fonction d'erreur (à minimiser), c'est-à-dire f(W)=C(W), où W est le vecteur composé des poids de connexions. L'algorithme pratique est le suivant :

0° Initialisation :
$W^{(0)}$: L'ensemble des poids qui sont initialisés au hasard entre -1. et 1..
$\delta_0>0$: Le rayon initial de confiance ;
$\varepsilon>0$: le seuil pour "zéro" (pour les tests en général) ;
$\varepsilon_g>0$: le seuil pour "zéro" pour tester la norme du gradient $\|g\|$;
$\varepsilon_f>0$: le seuil pour "zéro" pour tester la valeur de la fonction f;
H_0 : Une matrice symétrique dont les composants prennent de petites valeurs, ou, si possible affecter H_0 par $H(W_0)$, Hessienne de la fonction d'erreur à W_0. Il faut évaluer la plus petite valeur propre λ_1 de H_0 et un vecteur propre v_1 associé [8];
k=0; compteur d'itérations.

1° Calculer $f_k=f(W^{(k)})$, $g_k=\nabla f(W^{(k)})$.

2° Si $\|g_k\|\leq\varepsilon_g$ ou $f_k\leq\varepsilon_f$ alors $W^{(k)}$ est considéré comme la solution, stop ;

3° Calculer d_k :
Si $\lambda_1>0$, résoudre $H_k d= -g_k$;
si ($\|d\|\leq\delta_k$) alors $d_k=d$;
sinon utiliser l'algorithme de Hebden pour trouver un $\mu>0$ tel que la solution de $(H_k+\mu I)d=-g_k$ satisfasse $|\,\|d\|-\delta_k|\leq\varepsilon$, alors $d_k=d$;
sinon (cas où $\lambda_1\leq 0$) alors prendre $\alpha_k= -\lambda_1+0.0001$ et résoudre $p_k= -(H_k+\alpha_k I)^{-1}g_k$.
si ($\|p_k\|>\delta_k$) alors appliquer l'algorithme de Hebden à $H=H_k+\alpha_k I$ afin de trouver un $\mu>0$ tel que $\|(H_k+(\mu+\alpha_k)I)^{-1}g_k\|= \delta_k$ et puis $d_k= -(H_k+(\mu+\alpha_k)I)^{-1}g_k$;
sinon si ($\|p_k\|=\delta_k$) alors $d_k=p_k$.
sinon (le cas où $\|p_k\|<\delta_k$) alors $d_k=p_k+\kappa u_1$, où κ est choisi tel que $\|d_k\|=\delta_k$ et $\text{signe}(\kappa)=\text{signe}(u_1^t\, p_k)$.

4° Calculer

[8] C'est la méthode de la puissance itérée que l'on a utilisé dans nos expériences.

$$r_k = \frac{f(x_k) - f(x_k+d_k)}{q_k(0) - q_k(d_k)}$$

Si ($r_k \geq 0.1$) alors $W^{(k+1)}=W^{(k)} + d_k$
sinon $W^{(k+1)}=W^{(k)}$ (rester inchangé)

5° Si $r_k \leq 0.25$ alors $\delta_{k+1}=\delta_k/2$
sinon si ($r_k \geq 0.75$) alors $\delta_{k+1} = 2\delta_k$
sinon $\delta_{k+1} = \delta_k$ (rester inchangé)

6° Mettre à jour H_{k+1} par les algorithmes de formule de rang 1 ou de rang 2, donnés dans §V.3.3 . Utiliser n'importe quelle méthode pratique pour trouver la plus petite valeur propre et un vecteur propre associé de H_{k+1}.

7° k=k+1 et retouner à l'étape 1°).

V.3.6 Les performances de l'algorithme : tests et résultats

Trois expériences seront expliquées ci-dessous. La première montre une comparaison de la performance moyenne entre l'algorithme de GBP et celui de RC pour deux architectures différentes de réseaux. La seconde a pour but de montrer l'avantage de l'algorithme de RC par rapport à celui de GBP pour des cas particuliers, qui sont vraiment des cas très difficiles pour l'algorithme de GBP. La troisième montre la robustesse de l'algorithme de RC : sa performance est beaucoup moins influencée que celle de GBP par une architecture de réseau ne favorisant pas la tâche d'apprentissage.

Le premier problème d'apprentissage est celui de la relation XOR. Pour l'algorithme de GBP, l'ordre de présentation des patterns est fixé au départ et le taux d'apprentissage est initialisé au hasard, ou fixé selon le but du test. La pente α de chaque fonction sigmoïdale a été initialisé entre 0.5 et 1.5. Lorsqu'il y a une initialisation des poids de connexions, les valeurs sont toujours prises au hasard entre -1.0 et 1.0.

Comparaisons de la performance moyenne :

Les deux réseaux illustrés par la Figure 5.1 (pour xor_r_211) et la Figure 5.2 (pour xor_r_241) ont des structures fixées. La cellule "blanche"sur ces figures est celle qui sert de cellule de seuil avec 1.0 comme état permanent.

Toutes les autres cellules (de traitement) sont connectées à cette cellule spéciale.

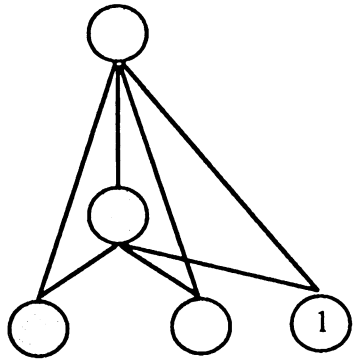


Figure 5.1 : xor_r_211, réseau pour apprendre la relation XOR avec deux cellules d'entrée, une cellule intermédiaire et une cellule de sortie.

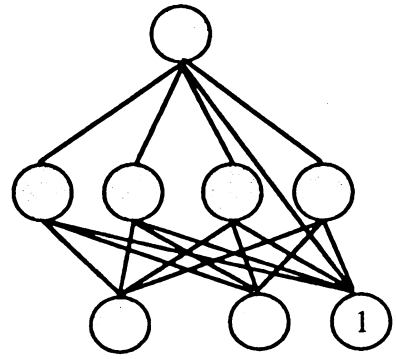


Figure 5.2 : xor_r_241 : réseau pour apprendre la relation XOR avec deux cellules d'entrée, quatre cellules intermédiaires et une cellule de sortie.

On fait varier les conditions de départ en initialisant 100 fois les poids de connexion pour chaque réseau. Pour chaque ensemble de poids initiaux, l'algorithme de RC et celui de GBP sont appliqués pour effectuer l'apprentissage. Le nombre maximum d'itérations d'apprentissage est fixé à 1000. Pour la méthode de GBP, le taux d'apprentissage est initialisé entre 0 et 0.4 pour chaque nouvel ensemble de poids, et reste fixé tout au long des 1000 itérations. Les deux Figures 5.3 et 5.4 montrent l'évolution moyenne[9] de la fonction d'erreur dans chaque réseaux, de la première itération jusqu'à la centième. La moyenne reste presque la même après la centième itération.

Il faut remarquer que la moyenne des erreurs est très vite descendue vers zéro dans les deux cas lorsque l'algorithme de RC est utilisé, mais seulement dans le second cas (le réseau xor_r_241) lorsque l'algorithme de GBP est utilisé. L'algorithme de GBP subit un échec (statistiquement) dans le processus d'apprentissage (au moins jusqu'à 1000 itérations d'apprentissage). En fait, le réseau xor_r_211 a un nombre trop faible de connexions (des cellules) pour apprendre la relation XOR. Il faut que la condition initiale soit très favorable pour que l'algorithme de GBP soit capable d'apprendre la relation à ce réseau.

(9) La moyenne des 100 erreurs à chaque itération d'apprentissage pour les 100 ensembles des poids au départ.

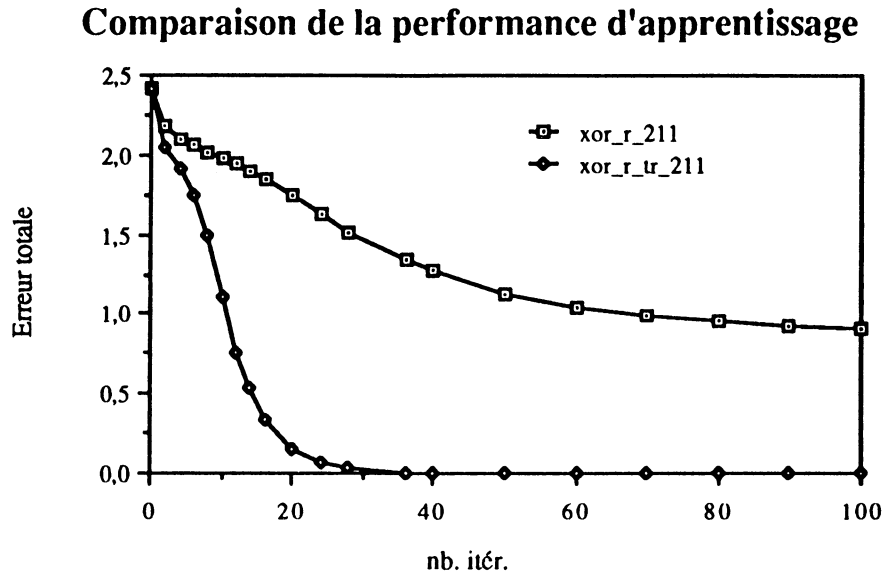


Figure 5.3 : Comparaison de la performance moyenne des deux algorithmes sur 100 ensembles de poids initialisés au hasard, pour le réseau illustré dans la Figure 5.1 (xor_r_211).

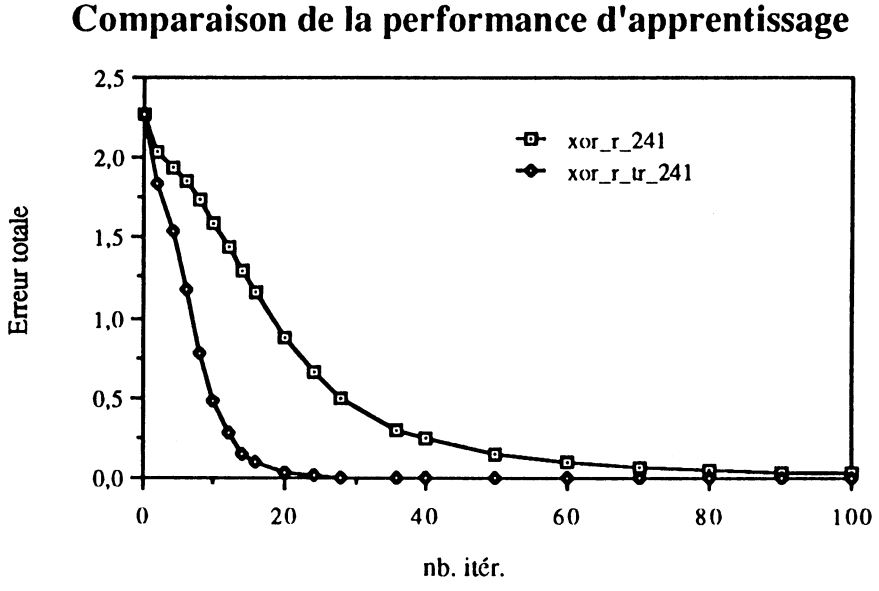
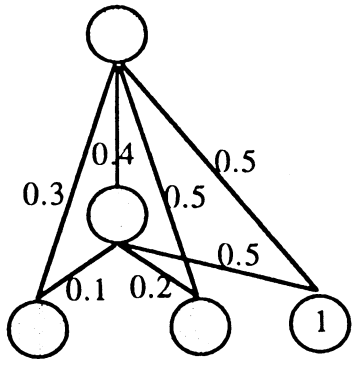


Figure 5.4 : Comparaison de la performance moyenne des deux algorithmes sur 100 ensembles de poids initialisés au hasard pour le réseau illustré dans la Figure 5.2 (xor_r_241).

Comparaison de la performance sur des réseaux particuliers :

Deux réseaux xor_S_1 et xor_S_2, dont les architectures et les poids sont fixés, sont illustrés ci-dessous par les Figures 5.5 et 5.6. Ils sont parmi les cas difficiles pour l'algorithme de GBP. Le taux d'apprentissage pour le GBP a été fixé à 0.2. Aucun de ces deux réseaux ne peut apprendre en moins de 1000

itérations par l'algorithme de GBP.



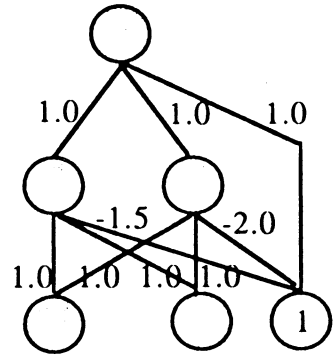


Figure 5.5, Figure 5.6 : xor_S_1 et xor_S_2 sont deux réseaux spéciaux avec, respectivement deux cellules d'entrée, une ou deux cellules intermédiaires, une cellule de sortie et les poids de connexion fixés.

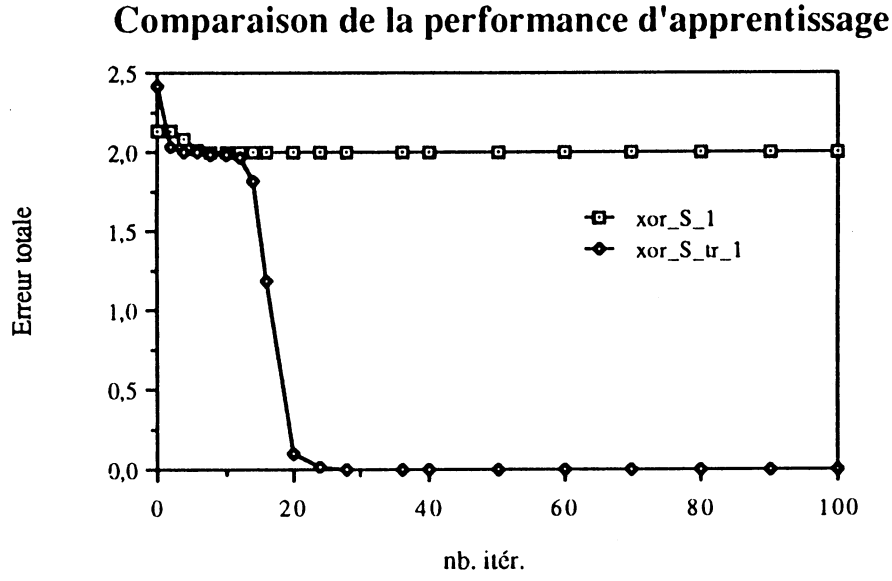


Figure 5.7 : xor_S_1 correspond au réseau de la Figure 5.5 qui a appris par l'algorithme GBP. xor_S_tr_1 correspond au réseau de la Figure 5.5 qui a appris par l'algorithme de RC.

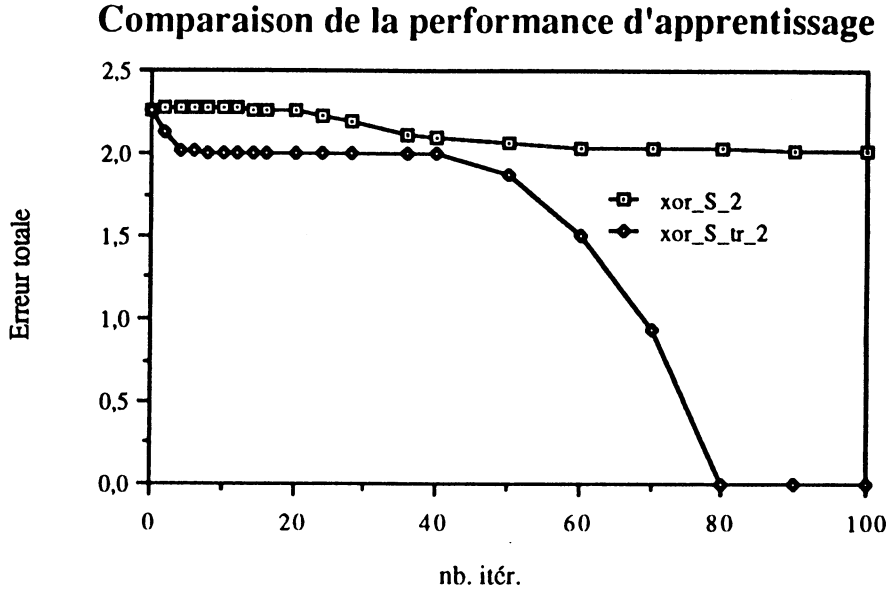


Figure 5.8 : xor_S_2 correspond au réseau de la Figure 5.6 qui a appris par l'algorithme GBP. xor_S_tr_2 correspond au réseau de la Figure 5.6 qui a appris par l'algorithme de RC.

Notons que ces deux exemples ne sont pas extrêmement rares : ils correspondent tout simplement aux cas où le nombre de connexions est petit par rapport au nombre de patterns à apprendre, et la majorité des poids initiaux ont le même signe. Il est aussi à remarquer que l'algorithme de RC peut rencontrer des difficultés et il lui faut un nombre d'itérations nettement supérieur à la moyenne 20 approximativement montrée par les Figures 5.3 et 5.4 pour réussir l'apprentissage (Figure 5.8); mais même dans ce cas il manifeste une supériorité par rapport à l'algorithme de GBP.

L'indépendance de l'architecture :

Actuellement, beaucoup d'études sur les réseaux multicouches concernent les architectures de réseau, pour rendre l'algorithme de GBP plus efficace. Dans des applications réelles, on est obligé de trouver des architectures "intelligentes" (spécialement pour les connexions qui entrent dans des cellules cachées) pour faciliter l'apprentissage et pour favoriser les tâches de généralisation. Ceci est quelque peu paradoxal si l'on en croit ce que l'on dit généralement : le GBP permet à un réseau d'apprendre automatiquement ce qu'on lui demande de faire. Des architectures spéciales ne peuvent être conçues que si l'on possède suffisamment de connaissances sur la nature du problème à traiter (ce qui n'est clairement pas souvent le cas). La nécessité d'avoir un

algorithme efficace quelle que soit l'architecture de réseau est donc évidente.

L'exemple suivant décrit la réalisation d'une fonction vectorielle de type boléen par un réseau 6-3-3. La fonction F est définie comme suit :

$F : \{-1, 1\}^6 \rightarrow \{-1, 1\}^3$

$F(x_1, x_2, x_3, x_4, x_5, x_6)^t = (f(x_1, x_2), f(x_3,x_4), f(x_5, x_6))^t$.

avec

$f(x_1, x_2) = 1$ si $x_1=1$ et $x_2=-1$ ou si $x_1=-1$ et $x_2=1$

$= -1$ si $x_1=1$ et $x_2=1$ ou si $x_1=-1$ et $x_2=-1$

Elle est composée en fait à partir de trois fonctions XOR. Il y a, au total, 64 patterns d'association à apprendre.

Par expérience, nous savons que si l'on utilise l'algorithme de GBP, un des meilleurs choix pour l'architecture du réseau est celui illustré par la Figure 5.9. C'est un choix logique et naturel parce que chaque cellule de sortie ne dépend que deux cellules d'entrée et n'a besoin d'aucune information provenant d'autres cellules d'entrée. Les trois cellules intermédiaires sont également "attribuées à" chacune des trois parties afin de donner à chaque partie la même capacité à traiter le problème de séparabilité de XOR. Cette architecture est aussi la plus simple : toutes les connexions supplémentaires ne peuvent créer que des circulations d'informations inutiles. On peut dire que dans ce réseau, il n'y a pas de connexions aberrantes. Pourtant, dans une application réelle, nous ne pourrons certainement pas obtenir suffisamment d'informations pour déterminer un tel réseau "intelligent". Par conséquent, nous choisirons une architecture du réseau avec beaucoup de connexions inutiles, ce qui rend le problème d'apprentissage très difficile. La Figure 5.10 illustre un cas où chaque cellule intermédiaire est connectée à toutes les cellules d'entrée, et chaque cellule de sortie est connectée à toutes les cellules d'entrée et à toutes les cellules intermédiaires. Nous allons voir que l'algorithme de RC fonctionne bien pour les deux architectures, mais que l'algorithme de GBP fonctionne déjà très mal avec la première (Figure 5.9) et pas du tout avec la seconde (Figure 5.10).

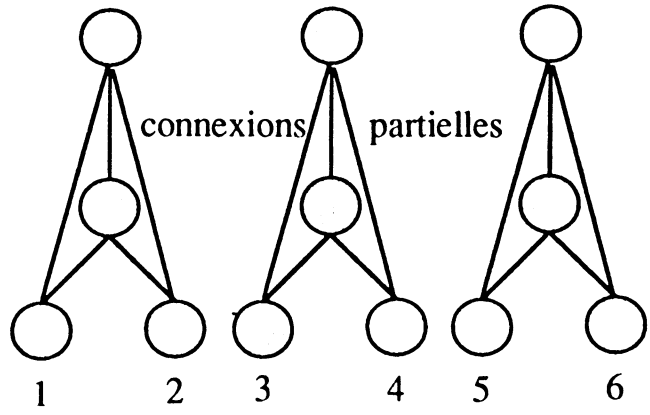


Figure 5.9 : Un réseau 6_3_3 qui contient six cellules d'entrée, trois cellules intermédiaires, et trois cellules de sortie. Il est en fait composé de trois réseaux 2_1_1 illustrés dans la Figure 5.1.

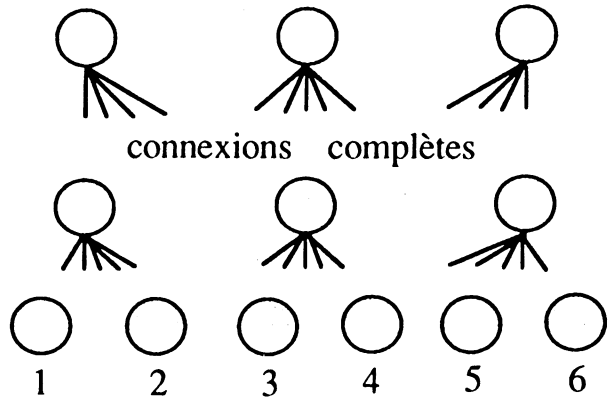


Figure 5.10 : Un réseau 6_3_3 qui contient six cellules d'entrée, trois cellules intermédiaires, et trois cellules de sortie. Chaque cellule reçoit des informations de toutes les cellules du dessous.

Dans les Figures 5.9 et 5.10, nous avons ignoré, pour accentuer la partie propre de l'architecture, la cellule de seuil (cellule blanche dans les figures des réseaux précédents) et toutes les connexions issues de cette cellule. Le fait que chaque cellule autre que celles d'entrée soit automatiquement connectée à une cellule spéciale d'état 1 devient presque une convention : Les poids de ces connexions correspondent au seuil de celle-là.

Expérimentalement, si dans le réseau de la Figure 5.9 les poids ne sont pas "trop mal" initialisés, la fonction F pourrait être apprise bien que le nombre d'itérations soit très élevé. Mais le résultat statistique n'en est pas pour autant satisfaisant. En effet, nous avons effectué, pour ces deux architectures, les

mêmes expériences que celles décrites dans le paragraphe ce-dessus pour les Figure 5.3 et 5.4.

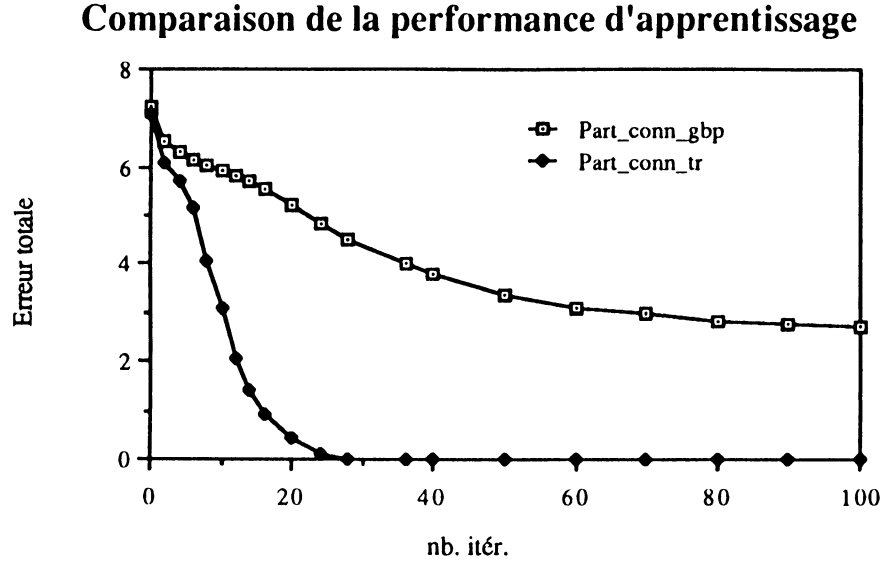


Figure 5.11 : Comparaison de l'évolution des erreurs du réseau illustré dans la Figure 5.9 pour les deux algorithmes d'apprentissage. Part_conn_gbp : correspond à l'algorithme de GBP. Part_conn_tr : correspond à l'algorithme de RC.

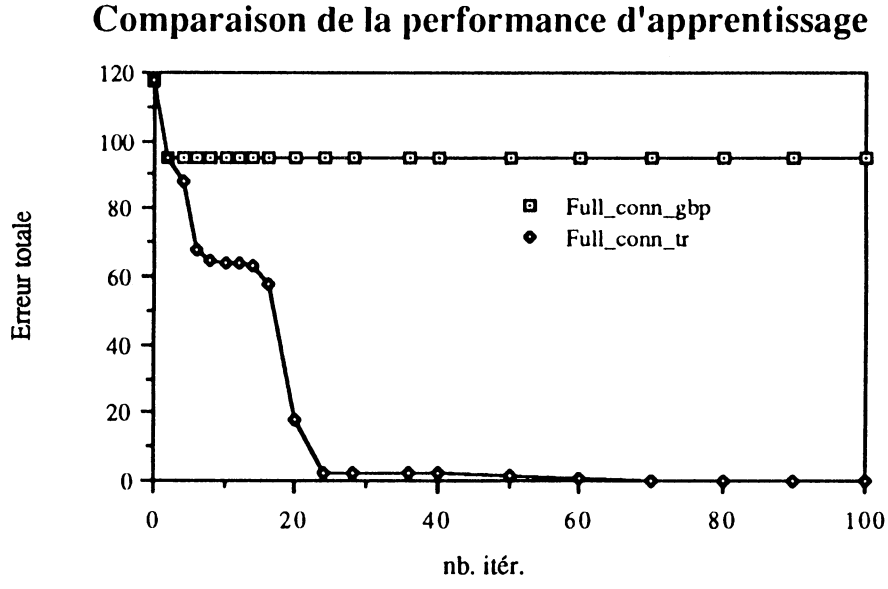


Figure 5.12 : Comparaison de l'évolution des erreurs du réseau illustré dans la Figure 5.10 pour les deux algorithmes d'apprentissage. Full_conn_gbp : correspond à l'algorithme de GBP. Full_conn_tr : correspond à l'algorithme de RC.

Les Figures 5.11 et 5.12 montrent l'évolution des erreurs moyennes des deux architectures. A noter que la Figure 5.11 a presque la même forme que la Figure 5.3. Ceci est dû au fait que le réseau se décompose en trois parties

séparées, dont chacune correspond à un réseau de la Figure 5.1. Donc l'évolution illustrée par la Figure 5.11 correspond presque une accumulation de trois évolutions du genre de celle de la Figure 5.3. Pour la seconde architecture (Figure 5.10), on voit encore plus nettement dans la Figure 5.12 une supériorité de l'algorithme RC. Il semble qu'il n'y ait aucune chance que ce réseau arrive à "s'entrainer" par l'algorithme GBP. Bien que la première architecture constitue aussi un très bon choix pour l'algorithme de RC, les résultats montrent qu'elle n'est pas indispensable.

V.4. Discussion et Conclusions du chapitre

Il faut remarquer que nous avons mesuré la performance des algorithmes d'apprentissage en termes d'erreur au cours des itérations, et non pas en temps réel d'exécution des algorithmes. En réalité, une itération avec l'algorithme de RC est beaucoup plus lente qu'une itération avec l'algorithme de GBP. En moyenne la première est de 10 à 15 fois plus lente que la dernière. Ceci dépend beaucoup de la machine et de la façon dont on implante les algorithmes. Ainsi on pourrait interpréter nos résultats comme suit :

1). On remarque que si un réseau a beaucoup de liens de connexions (beaucoup de cellules) et que s'il est loin de l'état de "saturation", alors la convergence de l'algorithme de GBP est plus rapide car, dans ce cas, la "surface d'erreur" a une dimension relativement grande et les variables ont beaucoup de liberté : le gradient calculé à chaque étape peut donner une descente considérable à la fonction d'erreur. Comme le temps pour faire une itération d'apprentissage est beaucoup plus court pour le GBP que pour le RC, il est plus rapide d'utiliser le GBP, bien qu'il exige un nombre d'itérations plus élevé. Ceci est le cas de la Figure 5.4, où l'algorithme de GBP nécessite en moyenne 80 itérations et celui de RC 20 itérations.

Or dans des applications pratiques, on peut demander à un réseau d'apprendre des associations d'entrée-sortie extrêmement compliquées (en termes de non linéarité et par rapport à sa "capacité") et d'en stocker le plus possible; dans ce genre de situations l'algorithme de GBP connaît souvent des échecs. Ceci est le cas des Figures 5.3 et 5.12, où l'algorithme de RC arrive à apprendre en un peu plus de vingt itérations alors que l'algorithme de GBP ne le peut pas en moins de 1000 itérations. La raison en est qu'il tombe souvent dans des minimums locaux ou dans des états d'oscillation. Dans ces cas, même au delà de 1000 itérations, on ne peut pas non plus espérer que l'algorithme de GBP converge très rapidement.

2). L'algorithme de RC offre une plus grande possibilité pour exploiter les capacités des réseaux multicouches. Comme le montrent ces résultats, on peut remarquer que si un réseau peut apprendre par l'algorithme de GBP, il le peut aussi par celui de RC; de plus, même si le réseau ne peut pas apprendre par le GBP, il reste encore une possibilité qu'il apprenne par l'algorithme de RC. L'algorithme de RC montre une supériorité très nette par rapport à celui de

GBP pour traiter des conditions initialement très difficiles.

3). L'algorithme de RC réalise une bonne utilisation des informations du second ordre. Théoriquement cela permet d'aboutir à une solution du problème avec une forte propriété du second ordre : la solution admet une Hessienne (semi-) définie positive. On a de bonnes raisons pour espérer que cette propriété persiste dans notre algorithme pratique.

4). Une des plus importantes propriétés de l'algorithme de RC est son efficacité vis-à-vis des architectures de réseau. Il se montre beaucoup moins dépendant de l'architecture spéciale pour aider la tâche d'apprentissage. Il peut donc permettre de surmonter beaucoup de difficultés (souvent rencontrées si on utilise l'algorithme de GBP) dans les choix d'architecture, et réduire considérablement "l'intervention humaine" dans l'apprentissage. Il rend le terme "apprentissage automatique" par des réseaux de neurones plus significatif.

La longue durée de chaque itération n'est pas vraiment un défaut de l'algorithme de RC car celle-ci est largement compensée par la réduction spectaculaire du nombre total d'itérations d'apprentissage. Le calcul de la Hessienne est aussi efficacement remplacé par des approximations. *Le vrai défaut de cet algorithme est la nécessité d'une grande quantité de mémoire physique pour stocker la matrice Hessienne* (ou l'approximation). Ceci constitue d'ailleurs un problème pour toutes les méthodes globales qui utilisent des informations du second ordre. La quantité, représentée en nombre réels, égale deux fois le carré du nombre des poids de connexion, ce qui est énorme quand le réseau devient grand. C'est peut-être la raison principale qui fait que la plupart des gens (y compris moi-même) utilisent les méthodes locales alors même qu'ils en connaissent la faible efficacité.

Le principal défaut pourrait partiellement être compensé par une utilisation très efficace des réseaux de taille moyenne, comme cela est promis par l'efficacité de l'algorithme. Mais pour trouver une solution complètement satisfaisante, nous pensons qu'il faut appliquer le principe de Région de Confiance à un système de variables intermédiaires beaucoup moins volumineux que celui des poids, et faire passer des prédictions intermédiaires aux poids à modifier, comme ceci est le cas dans l'algorithme de GBP où on calcule le gradient de la fonction d'erreur par rapport à l'ensemble des variables

représentant les "entrées totales" au lieu de celui des poids. Or, pour appliquer la même idée à la méthode qui utilise des informations du second ordre il faut trouver des solutions à plusieurs problèmes théoriques et pratiques concernant le "passage" entre des variables intermédiaires et des variables de poids.

Dans ce chapitre, on s'intéresse au problème de l'amélioration de l'algorithme de GBP pour l'apprentissage dans des réseaux multicouches, en se limitant au cadre de la même classe de cellules sigmoïdales. Nous avons d'abord présenté des approches locales qui consistent à introduire des heuristiques dans l'algorithme de GBP pour mieux adapter les taux d'apprentissage. Ces heuristiques peuvent, dans certains cas, apporter des améliorations à l'algorithme GBP, dont l'ampleur dépend du problème traité. Nous avons ensuite présenté en détail une approche globale qui consiste à utiliser des informations du second ordre pour prédire d'une manière plus exacte la forme (locale) de la surface d'erreur, afin de mieux diriger la recherche. Les résultats expérimentaux illustrent l'efficacité de cette méthode.

Chaque approche, globale ou locale, a son avantage et son inconvénient. Certes, la conciliation des deux approches, ou au moins des points de rencontre, doivent être trouvés pour réaliser une amélioration complètement satisfaisante. Il est clair qu'il y a encore un très long chemin à parcourir pour arriver à cet objectif. Ce sujet est, et restera encore, un des principaux axes de recherche dans le domaine des réseaux de neurones.

Conclusion Générale

Les travaux de cette thèse regroupent trois aspects de recherche concernant les réseaux de neurones multicouches : implantation, applications et amélioration de l'algorithme d'apprentissage par rétro-propagation de gradient.

Les calculs dans des réseaux de neurones sont massivement parallèles. Ce potentiel de parallèlisme massif n'a pas encore été pleinement exploité dans les applications faute de disposer de machines réellement adaptées. Ceci est considéré comme une des raisons pour lesquelles les réseaux neuronaux se font encore battre par des modèles classiques dans certaines applications. Les technologies actuelles dans le domaine du calcul scientifique ont atteint un niveau très élevé en vitesse de calcul, mais elles sont spécialement adaptées aux problèmes où la structure des données est très régulière. La performance se dégrade lourdement s'il faut effectuer des calculs sur des données à structure irrégulière, comme tel est le cas pour la plupart des applications de type réseaux de neurones. Il faudrait donc développer une étude méthodologique et algorithmique plus approfondie pour analyser l'implantation de ces réseaux sur les architectures de machines existantes. Ceci permettrait de dégager des conclusions sur le besoin architectural en machines dédiées pour les réseaux neuronaux.

Notre démarche a consisté à implanter un modèle de réseau d'architecture (relativement) régulière sur un anneau des processeurs puis à conduire une étude de la parallélisation de l'algorithme d'apprentissage sur un hypercube pour des réseaux d'architecture moins régulière. La conclusion que nous pouvons tirer de ces études est que, pour le modèle régulier, l'anneau permet de réaliser un taux très important d'exploitation du parallélisme des réseaux, tandis que pour le modèle de réseau d'architecture générale, l'hypercube peut être utilisé pour la simulation, de manière que sa capacité de communication soit exploitée au maximum. La future orientation de cette étude est également discutée au Chapitre II.

Nous avons présenté deux applications des réseaux multicouches dans les Chapitres II et III. Ces applications s'inscrivent dans le cadre général de la recherche en Science Cognitive, ce qui est en fait le but principal du développement des réseaux de neurones.

Plus particulièrement, notre simulation de la mémoire des visages

permet de modéliser l'effet contextuel dans la construction des identités de visages, et de fournir plus de précisions sur le modèle de Bruce et Young, (un modèle qui sert de cadre théorique général à la recherche). Cette simulation permet de contrôler de manière systématique certains facteurs, afin d'étudier finement le rôle des variables dont on désire connaître l'effet. Elle permet aussi l'émulation de la dynamique de la recherche mnésique et de modéliser un traitement plus spécifique des entrées visages et des entrées contextes, traitement qui tient compte de la hiérarchie des informations. Il faut souligner que les tentatives effectuées auparavant pour simuler ce type de modèles cognitifs par des systèmes computo-symboliques n'ont pas très bien abouti, et que le modèle connexionniste semble clairement constituer un outil plus naturel et pertinent pour ce genre de simulations ;

L'application des réseaux de neurones à la reconnaissance de la parole nécessite la résolution du problème du traitement des informations temporelles. L'introduction des feedbacks (par Jordan), les connexions partielles et superposées (Chapitre III) et le Time delay network de Weibel *et al* constituent des approches pour différents problèmes spécifiques. Il faut noter que la combinaison de ces techniques permet d'ouvrir une nouvelle perspective pour le traitement de la parole continue ;

Dans les deux applications, nous savons pu montrer l'importance de l'architecture des réseaux pour modéliser les fonctionnalités dont on a besoin.

L'étude du comportement dynamique du réseau (§I.3) et l'amélioration de l'algorithme de GBP (Chapitre V) constituent le troisième aspect de notre recherche. L'étude du comportement des réseaux fonctionant en GBP a été inspirée par notre modélisation de la mémoire des visages. Elle porte sur les propriétés qu'un réseau pourrait exhiber après apprentissage. Nous comprenons mieux maintenant pourquoi les réseaux peuvent "généraliser les connaissances" sur des entrées nouvelles, et surtout pouquoi ils peuvent être utilisés pour simuler la mémoire. L'amélioration de l'algorithme GBP par la méthode de Région de Confiance (RC) permet d'augmenter considérablement la capacité d'apprentissage d'un réseau. L'algorithme de RC est particulièrement efficace si le nombre de patterns à apprendre est élévé par rapport au nombre de cellules et de connexions disponibles. Il réalise une bonne utilisation des informations du second ordre. Sa supériorité sur l'algorithme de GBP apparait aussi dans son efficacité vis-à-vis des architectures de réseau : l'algorithme de RC est beaucoup moins dépendant d'une architecture spéciale destinée à aider la tâche

d'apprentissage. Néanmoins, pour le rendre plus pratique, il faudrait encore résoudre le problème de la taille de mémoire nécessaire pour stocker la matrice Hessienne.

Les préoccupations et les développements de cette thèse participent donc au vaste mouvement actuel visant à l'amélioration et à la meilleure compréhension des réseaux neuronaux. Nous somme convaincus, en concluant ce travail, que ce domaine va encore connaître des développements décisifs dans les années à venir. Nous souhaitons, vivement, continuer à y contribuer.

BIBLIOGRAPHIE

Abdi, H. (1988). A generalized approach for connectionnist auto-associative memories : interpretation, implication and illustration for faces processing. In J. Demongeot , T. Herve, V. Rialle & C. Roche (Eds.), *Artificial intelligence and cognitive science.* Manchester : Manchester University Press.

Amari, S. I. (1972) "Learning patterns and pattern sequences by self-organizing net of threshold elements", *IEEE Trans. Computer*, Vol 21, N° 11.

Amit, D., Gutfreund, H. & Sompolinsky, H. (1985). Storing infinite numbers of patterns in a spin glass model of neural networks. *Physical Review Letters*, **55**, 1530-1533.

Amy, B. & Tiberghien, G. (1988). Contexte, cognition et machines contextuelles. *Rapport de recherche, Laboratoire LIFIA.* IMAG, Grenoble.

Anderson, J.R. (1983). *The architecture of cognition.* Cambridge, MA : Harvard University Press.

Auslender, A. (1976), *Optimisation, méthodes numériques.* Masson, Paris.

Baddeley, A. & Woodhead, M. (1982). Depth of processing, context, and face recognition. *Canadian Journal of Psychology*, **36**, 148-164.

Barto, A. G. & Sutton, R. S. (1981). Goal seeking components for adaptive intelligence : An initial assessment (Air Force Wright Aeronautical Laboratories/Avionics Laboratory Tech. Rep. AFWAL-TR-81-1070). Ohio: Wright-Patterson AFB.

Becker & Le Cun, Y. (1988). dans *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, Publishers.

Benguerel, A.P. & Cowan, H.A. (1974). Coarticulation of upper lip protusion in French. *Phonetica*, 30, 41-55.

Bruce, V. (1979). Searching for politicians : an information-processing approach to face recognition. *Quarterly Journal of Experimental Psychology*, **31**, 373-396.

Bruce, V. (1982). Changing faces : visual and non visual coding processes in face recognition. *British Journal of Psychology*, **73**, 105-116.

Bruce, V. (1983). Recognizing faces. *Philosophical Transactions of the Royal Society of London* : serie B, **302**, 423-436.

Bruce, V. (1986a). Recognising familiar faces. In H.D. Ellis, M.A. Jeeves, F. Newcombe & A.W. Young (Eds.), *Aspects of face processing.* Dordrecht : Martinus Nizhoff.

Bruce, V. & Valentine, T. (1986). Semantic priming of familiar faces. *Quarterly Journal of Experimental Psychology*, **38A**, 125-150.

Bruce, V. & Young, A. (1986). Understanding face recognition, *British Journal of Psychology*, **77**, 305-327.

Bruyer, R. (1987). *Les modèles de la reconnaissance des visages.* Grenoble : PUG.

Bruyer, R. (1987a). Naming faces without recognition : a direct relationship and a new line in the model ? *(CPC) European Bulletin of Cognitive Psychology.* **7**, 309-313.

Cea, J. (1971), *Optimisation : Théories et algorithmes*, Dunod.

Christie, D., Ellis, H.D. (1981). Photophit constructions versus verbal description of face. *Journal of Applied Psychology*, **66**, 358-363.

Conn, A. R., Gould, N. & Toint, Ph. (1986). Testing a class of methods for solving minimization problems with simple bounds on the variables. *Report n°86-3*, University of Waterloo.

Cosnard, M., Tourancheau, B. et Villard. G. (1987). Presentation de l'Hypercube T20 de FPS.

Cottrell, G.W., Munro, P. & Zipser, D. (1987). Image Compression By Back Propagation : An Example of Extensional Programming, Feb. 1987, ICS Report 8702,

Davies, G. M. (1986).Context effect in episodic memory : a review. *(CPC) European Bulletin of Cognitive Psychology*, **6**, 157-174.

Davies, G.M. (1988). Faces and Places : Laboratory research on context and face recognition. In G. Davies & D.M. Thomson (Eds.), *Memory in Context : Context in Memory*. London : Wiley .

Davies, G.M., Ellis, H.D., Fline, R., Milne, A. & Shepherd, J.W. (1982). *Face retrieval techniques*. Report to Home Office SRDB.

Davies, G.M. & Milne, A. (1982). Recognizing faces in and out of context. *Current Psychological Research*, **2**, 235-246.

Dennis, J.E., Schnabel, R.B. (1983), *Numerical methods for unconstrained optimization and nonlinear equations*. Printice-Hall.

Diamond, R. & Carey, S (1986). Why faces are and are not special : an effect of expertise. *Journal of Experimental Psychology : General*, **115**, 107-117.

Duff, I.S., Nocedal, J. & Reid, J.K. (1987). The use linear programming for solution of sparse sets of nonlinear equations. *SIAM J. SCI. STAT.COMPUT. vol 8, N°2*, pp. 99-108.

Duranton, M. & SIRAT J.A. (1989). Learning on VLSI : A general purpuse digital neurochip. Communication à *IJCNN 89*, Washington D.C., juin 1989.

Ekeland, I., & Temam, R. (1974), *Analyse Convexe et problèmes variationnels*. Dunod, Gauthier-Villars.

Ellis, H.D. (1981). Theoretical aspects of face recognition. In G. Davies, H.D. Ellis, & J. Shepherd (Eds.), *Perceiving and Remembering Faces*. London : Academic Press.

Ellis, H.D. (1986). Processes underlying face recognition. In R. Bruyer (Ed.), *The neuropsychology of face perception and facial expression*. Hillsdale, New Jersey : Laurence Erlbaum Associates.

Ellis, A.W., Young, A.W. & Hay D.C. (1986). Modelling the recognition of face and words. In P. E. Morris (Ed.), *Models of Cognition*. London : J. Wiley.

Estes, W.K. (1988). Toward a Framework for Combining Connectionist and Symbol-Processing Models. *Journal of Memory and Language*, **27**, 196-212.

Elman, J. L. & Zisper, D. (1987). Learning the Hidden Structures of the Speech. *ICS Report 8701*, University of Californie.

Fletcher, R. (1980), Pratical Methods of Optimization, vol 1, John Wiley, New York.

Fogelman-Soulie, F. (1985a). Contribution à une théorie du calcul sur réseau, Thèse d'état, Grenoble.

Fogelman-Soulie, F. (1985b). Cerveau et machines : des architectures pour demain? In *Actes du forum Cognitiva*, Paris.

Fogelman-Soulie, F., Gallinari, P., Le Cun, Y., Thiria, S. (1987). Automata networks and artificial intelligence. In F. Fogelman-Soulie, Y. Robert, M. Tchuente (Eds.), *Computing on automata networks*, Manchester University Press.

Foldiak, P. (1989). Adaptive network for optimal linear feature extraction. Physiological Laboratory, cambridge, U.K..

Fowler, C. A. (1980). Coarticulation and theories of extrinsic timing. *Journal of Phonetics*, 8, 113-133.

Fox G.C., Otto S.W. et Hey A.J.G. (1987) "Matrix algorithms on a hypercube I : Matrix multiplication", in Parallel Computing 4 (1987) 17-31, North-Holland.

FPS T20 Manuel Utilisateur, Laboratoire TIM3 (Juin 1987)

Gallinari, P., Le Cun, Y., Thiria, S., Fogelman-Soulie, F. (1987). Mémoires associatives distribuées : une comparaison. *Acte du Forum Cognitiva*, Paris

Gastinel, N. (1966), *Analyse numérique linéaire*, Hermann, Paris.

Gay, D.M. (1981), Computing optimal constrained steps. *SIAM J. Sci. Stat. Comput. 2*, pp 186-197.

Ghosh, J et Hwang, K. (1988) "Critical Issues in Mapping Neural Networks on Message-Passing Multicomputers," *Proceedings of the 15th Annual International Symposium on COMPUTER ARCHITECTURE*, COMPUTER ARCHITECTURE.

Gill, P.E. & Murray, W. (1972)), Quasi-Newton methods for unconstrained optimization. *The Journal of the Institute of Mathematics and its Applications*, vol 9, pp. 91-108.

Gill, P.E., Murray, W. & Wright, M.H. (1981), *Pratical Optimization*, Academic Press.

Gluck, M.A. & Bower, G.H. (1988). Evaluating an Adaptive Network Model of Human Learning. *Journal of Memory and Language*, **27**, 166-195.

Godden, D. & Baddeley, A. (1980). When does context influence recognition ? *British Journal of Psychology*, **71**, 99-104.

Goles-Chacc, E. (1985) "Comportement dynamique de réseaux d'automates", Thèse d'état, Grenoble.

Graf, H. P., Jackel, L. D., Hubbard, W.E. (1988). VLSI Implementation of a Neural Network Model. *IEEE Computer*, Mars 1988, 41-49.

Grossberg, S. (1967). Nonlinear difference-differential equations in prediction and learning theory. *Proceedings of the National Acadamy of Sciences*, 58, 1329-1334

Grossberg, S. (1988). Nonlinear Neural Networks : Principles, Mechanisms, and Architectures. *NEURAL NETWORKS*, Vol 1, N°1, 1988.

Guérin, A. (1987). C.R.A.S.Y. Un Calculateur de Réseaux Adaptatifs Systolique : Application au calcul neuromimétique. Thèse de Docteur-Ingénieur, INP Grenoble.

Guez, A., Protopopsecu, V. & Barhen, J. (1988). On the Stability, Storage Capacity, and Design of Nonlinear Continuous Neural Networks. *IEEE trans. on sys., man, and cyber*, **18**, 1.

Hammerstrom, D., Maier, D. & Thankhar, S. (1986). The Cognitive Architecture Project. *Computer Architecture News* , **14**, 1, 4-8.

Hay, D. C. & Young, A. W.(1982). The human face. In A.W. Ellis (Ed.), *Normality and pathology in cognitive functions*. London : Academic Press.

Haykin, S. (1986). *Adaptive filter theory*. Englewood Cliffs, NJ:Prentice-Hall.

Hebb, D.O. (1949). *The Organization of Behavior*. New York : Wiley.

Hebden, M.D. (1973), An algorithm for minimization using exact second derivatives. *Atomic Energy Research Establishment report T.P. 515*, Harwell, England.

Hinton, G. E., Sejnowsky, T. J. & Ackley, D. H.(1984). Boltzmann Machines: Contraint Satisfaction Networks that learn.*Technical Report CMU-CS-84-119, May 1984.*

Hinton, G.E. (1986). Learning Distributed Representations of Concepts. *8th Annual Conference of the Cognitive Science Society*, Amherst.

Hinton, G.E., McClelland, J.L. & Rumelhart, D.E. (1986). Distributed Representations. In D.E. Rumelhart, J.C. McClelland (Eds.), *Parallel Distributed Processing*, vol.1. Cambridge, MA : MIT Press.

Hinton, G. E. (1987). Connectionist Learning Procedures. *Tecnical Report CMU-CS-87-115* : Computer Science Department, Carnegie-Mellon University.

Hintzman, D.L. & Stern, L.D. (1978). Contextual variability and memory for frequency. *Journal of Experimental Psychology : Human Learning and memory*, **4**, 539-549.

Hofstadter, D. (1987). Cognition, Subcognition : Sortir du rêve de Boole. *Le débat*, **47**, 26-44.

Honig, M. L., & Messerschmitt, D. G. (1984). *Adaptive filters : Structures, algorithms, and applications*. Boston : Kluwer Academic Publishers.

Hopfield, J. J. (1982). Neural Networks and Physical Systems with emergent collective computational abilities. *Proceedings National Academy of Sciences*, **79**, 2554-2558.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Procedings of the National Academy of Science USA*, **81**, 3088-3092.

Hopfield, J. J., Tank, D. W. (1985). "Neural" Computation of Decisions in Optimiszation Problems. *Biological Cybernetics*, Springer-Verlag, 141-152.

Hurat, Ph. (1989). APLYSIE : Un Circuit Neuro-Mimétique, Réalisation et Integration sur Tranche. *Thèse de doctorat*, Institut National Polytechnique de Grenoble.

Jordan, M.I. (1986). Serial Order : A Parallel Distributed Processing Approch. *ICS Report 8604*.

Jordan, M.I. (1988). Supervised learning and systems with excess degrees of freedom. MIT, *COINS Technical Report 88-27*.

Jutten, C. (1987). Calcul neuromimétique et Traitement du Signal. Analyse en Composantes Indépendantes. Thèse d'Etat es Sciences Physiques. INP-USTM Grenoble.

Kaniel, S. & Dax A. (1979), A modified Newton's method for unconstrained minimization. *SIAM J. Num. Anal.*, pp. 324-331.

Klee, M., Leseaux, M., Malai, C., Tiberghien, G. (1982). Nouveaux effets de contexte dans la reconnaissance de visages non familiers. *Revue de Psychologie Appliquée*, 32, 109-119.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetic*, 43, 59-69.

Kohonen, T. (1984). *Self-organisation and associative memory*. Berlin : Spinger-Verlag.

Kung, S. Y., Hwang, J. N. (1988) Digital VLSI Architectures for Artificial Neural Nets. communication personnelle.

Lancaster, P. (1969), *Theory of Matrix*, Academic Press, New York and London.

Lashley, K.S. (1951), The problem of serial order in behavior. In L. A. Jeffress (Ed.), *Cerebral mechanisms in behavior*, pp. 112-136, New York : Wiley.

Laughery, K.R., Duval, C., Wogalter, W.S. (1986). Dynamic of facial recall In H.D. Ellis, M.A. Jeeves, F. Newcombe & A.W. Young (Eds.), *Aspects of face processing*. Dordrecht : Martinus Nizhoff.

Laughery, K.R., Rhodes, B.T., Batien, G.W. (1981). Computer Guided recognition and retrieval of facial images. In G. Davies, H.D. Ellis, & J. Shepherd (Eds.), *Perceiving and Remembering Faces*. London : Academic Press.

Laurent, P.J. (1972), Approximation et Optimisation. Hermann, Paris.

Le Cun, Y. (1987). Modèles connectionnistes de l'apprentissage. *Thèse de doctorat*, Université de Paris VI.

Levy, P. (1987). L'univers du calcul : calculer, percevoir, penser. In J.L. Lemoigne (Ed.), *Intelligence des mécanismes et mécanismes de l'intelligence*. Paris : Fayard.

Llis, H.D. (1981). Studies of cue saliency. In G. Davies, H.D. Ellis & J. Shepherd (Eds.), *Perceiving and Remembering Faces*. London : Academic Press.

Lindsay, P.H. & Norman, D.A. (1980). *Traitement de l'information et comportement humain : Une introduction à la psychologie*. Montréal : Editions Etudes Vivantes.

Lippmann, R. P. (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine, avril* 1987, 4-22.

Marr, D. (1982). *Vision*. San francisco : Freeman.

Mandler, G. (1980) Recognizing : the judgment of previous occurrence. *Psychological Rewiev*, 87, 252-271.

Massaro, D.W. (1988). Some Criticisms of Connectionist Models of Human Performance. *Journal of Memory and Language*, **27**, 213-234.

McBryan,O.A. (1987) "Matrix and vector operations on hypercube parallel processors", Parallel Computing 5 (1987), pp117-125, North-Holland.

McClelland, J.L. (1979). On the time-relation of mental processes : An examination of systems of processing in cascade. *Psychological Review*, **86**, 287-330.

McClelland, J.L. (1988). Connectionist Models and Psychological Evidence. *Journal of Memory and Language*, **27**, 107-123.

McCulloch, W. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematics and Biophysics.*, **5**, 115-133.

Memon, A. & Bruce, V. (1985). Context Effects in Episodic Studies of Verbal and Facial Memory. *Current Psychological Research and Reviews*, Winter, 349-369.

Minoux (1983), *Programmation Mathematique*. Tome1, Dunod.

Minsky, M. & Papert, S. (1969). *Perceptrons*. Cambridge, MA : MIT Press.

Moll, K.L. & Daniloff, R.. G.(1971). Investigation of the timing of velar movements during speech. *Journal of the Acoustical Society of America*, 50, 678-684.

Moody, J. & Darken, C. (1988), Learning with Localized Receptive Fields, *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, Publishers.

Moré, J.J. (1978), The Levenberg-Marquart algorithm : implementation and theory, *Lecture Notes in Mathematics 630*, G. A. Waston, ed., Springer-Verlag, Berlin- Heidelberg-New

York,pp. 105-116.

Moré, J.J. (1983), Recent developments in algorithm and software for Trust Region Methods. *Mathematical Programming, The State of the Art*, Springer, Berlin, pp 258-287.

Moré, J.J. & Sorensen, D.C. (1979). On the use of directions of negative curvature in a modified Newton method. *Math.Prog. 16*, pp. 1-20.

Moré, J.J. & Sorensen, D.C.(1981), Computing a trust region step. *Argonne National Laboratory report*, Argonne, Illinois.

Morton, J.(1969). Interaction of information in word recognition. *Psychological Review*, **76**, 165-178.

Mukai, H. & Polak, E. (1978). A second order method for unconstrained optimization. *J.O.T.A.* vol 26, pp. 501-513.

Nakano, K. (1972) "Associatron : a model of associative memory", *IEEE Trans. Syst, Man and Cybernetics*, Vol SMC-2, N° 3.

Nestor, (1987). Nestor Learning Systems : A Technology Overview.

Oliver A. McBryan et E. F. Van de Velde. (1987) "Matrix and vector operations on hypercube parallel processors", in Parallel Computing 5 (1987) 117-125, North-Holland.

O'Toole,A., Millward, R.B. & Anderson, J.A.(1988). A physical system approach to recognition memory for spatially transformed faces. *Neural Networks*, **1**, 179-199.

Paturle, I., Morens, V., Hullard, V., Tiberghien, G. (1984). Familiarité, contexte et reconnaissance des visages. *Mémoire non publié*, Université de Grenoble, Département de Psychologie.

Penot, J. P. & Roger, A. Updating the spectrum of a real matrix. *Mathematics of Computation.*

Peretto, P. (1984) Collective properties of neural networks, a statistical physics approach. *Biol. Cybernetics*, 50, pp 51-62.

Perez, J.C. (1988). *De nouvelles voix vers l'Intelligence Artificielle : Pluri-disciplinarité, Auto-organisation, Réseaux neuronaux.* Paris : Masson.

Peris, J.L. (1986). Reconnaissance et méconnaissance. *Thèse non publiée*, Université de Grenoble, Département de Psychologie.

Pham Dinh T, Wang S. et Yassine A. (1988) TRAINING A MULTI-LAYERED NEURAL NETWORK WITH A TRUST-REGION BASED ALGORITHM. -apparaître dans *Mathematical Modeling and Numerical Analysis.*

Powell, M.J.D. (1975), Convergence properties of a class of minimization algorithms. O. L. Mangazarian, R.R. Meyer, S. M. Robinson Editors, *Nonlinear programming 2* pp 1-27, Academic press, New York.

Rabinowitz, J.C., Mandler, G., Barsalou, L.W. (1977). Recognition failure: another case of retrieval failure. *Journal of Verbal Learning and Verbal Behavior*, **16**, 639-663.

Reinsch (1967), Smoothing by spline functions, *Numer. Math.* 10, 177-183.

Reinsch (1971), Smoothing by spline functions II, *Numer. Math.* 16, 451-454.

Robert, A.J. (1988). Increased Rates of Convergence Through Learning Rate Adaptation, *Neural Networks*, Vol. 1, N° 4, pp 295-307.

Robert, F. (1986). *Discrete Iterations, a metric study*, Springer Verlag.

Robert, F. & Wang, S. (1988). Implementation of a Neural Network on a Hypercube F.P.S. T20. dans *Parallel Processing,* M. Cosnard, M.H. Barton and M. Vanneschi (eds), North-Holland (1988).

Rockafellar, R.T. (1970) *Convex Analysis*, Princeton University Press, Princeton, New Jersey.

Roger, A. (1987), Mise à jour du spectre d'une matrice symétrique. *Rapport de recherche SNEA(P)*, n° AR/87-970.

Rosenberg,C.R., Sejnowski, T.J. (1986). Practice on NetTalk, a massively parallel network that learns to read aloud. *Connectionnist Models : a summer school* : Carnegie Mellon University.

Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organisation in the brain. *Psychological Review*, **65**, 386-408.

Rousset, S., Schreiber, A.C. & Wang, S. (1988) Modélisation et simulation connexionniste de l'identification des visages en contexte : le système FACENET. *RR 742 -M-. IMAG* Grenoble.

Rumelhart, D.E. & Norman, D. A. (1982). Simulating a skilled typist : A study of skilled cognitive-motor performance. *Cognitive Science*, 6, 1-36.

Rumelhart, D.E. & McClelland, J.L. (1986). PDP Models and General Issues in Cognitive Science. In D.E. Rhumelhart, J.C. McClelland (Eds.), *Parallel Distributed Processing,* vol.1. Cambridge, MA : MIT Press.

Rumelhart, D.E., Hinton, G. & Williams, R. (1986). Learning internal representations by error propagation. In D.E. Rumelhart, J.C. McClelland (Eds.), *Parallel Distributed Processing,* Cambridge, MA : MIT Press.

Saad, Y. et Schultz, M.H. (1985) "Data communication in Hypercubes", Research Report YALEU/DCS/RR-428.

Sanger, T. D. (1988). Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network", MIT AI Lab., NE43-743, Cambridge.

Seamon, J.G. (1982). Dynamic facial recognition : Examination of a natural phenomenon. *American Journal of Psychology*, **95**, 363-381.

Sejnowski, D.E. & Rosenberg, C.R. (1986). NETTALK : A Parallel network that learns to read aloud. *Technical Report JHU/EECS-86/01.* Baltimore : Johns Hopkins University.

Sergent, S. (1984). An investigation into component and configural processes underlying face perception. *British Journal of Psychology*, **75**, 221-242.

Shaffer, L. H. & Hardwick, J. (1970). The basis of transcription skill, *Journal of Experimental Psychology*, 84, 424-440.

Shastri, L. & Feldman, J. A. (1984). Semantic Networks and Neural Nets. *Technical Report. 133.* Dept. of Computer Science, University. of Rochester, Rochester, New York.

Shastri, L. & Feldman, J. A. (1985). Evidential Reasoning in Semantic Networks: a formal Theory. *Proceedings of IJCAAL*, 467-474.

Shepherd, J.W. (1986). An interactive computer system for retrieving faces. In H.D. Ellis, M.A. Jeeves, F. Newcombe & A.W. Young (Eds.), *Aspects of face processing.* Dordrecht : Martinus Nizhoff.

Schreiber, A., Rousset, S., Tiberghien G., et Wang, S., & Robert., F. (1988). Modelling Face Recognition : Simulation of Context Effects in Face Identification, soumis dans *Cognitive Science* (90),

Shultz, G.A., Schnabel, R.B. & Byrd, R.H. (1985), A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. *SIAM Journal on Numerical Analysis 22, pp 47-67.*

Shultz, G.A., Schnabel, R.B. & Byrd, R.H. (1988), Approximate solution of the trust region problem by minimization over two-dimensional subspaces Mathematical Programming, Vol 40, pp 247-263, North-Holland.

Smith, S.M. (1982). Enhancement of recall using multiple environmental contexts during learning. *Memory & Cognition*, **10**, 405-412.

Sorensen, D. C. (1982), Newton's method with a model trust region modification, *SIAM J. Numer. Anal.* vol 19. N° 2, pp 409-426.

Shaffer, L. H. (1976). Intention and performance. *Psychological Review*, **83**, 375-393.

Sternberg, S., Monsell, S., Knoll, R. L., & Wright, C. E. (1978). The latency and duration of rapid movement sequences : Comparisons of speech and typewriting. In G. E. Stelmach (Ed.), *Information processing in motor control and learning*, pp. 117-152. New York, Academic Press.

Stewart, G.W. (1973), *Introduction to matrix computation*, Academic Press, New York.

Stonham, T.J. (1986). Practical Face Recognition and Verification with Wisard. In H.D. Ellis, M.A. Jeeves, F. Newcombe & A.W. Young (Eds.), *Aspects of face processing*. Dordrecht : Martinus Nizhoff.

Sutton, R., S. (1986). Two problemes with backpropagation and other steepest-descent learning procedures for networks. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 823-831.

Thakoor, A. P., Lamb, J. L., Moopenn, A., Lambe J. (1986) Binary Synaptic Connections Based on Memory Switching in a-Si:H, dans J.S. Denker (ed.) AIP Conference Proceedings 151, Neural Network for Computing, Snowbird, Utah, USA, 1986, 426-431.

Thiria, S. (1989). L'apprentissage supervisé dans les modèles connexionnistes. *Thèse de doctorat*, Université Réné Descartes (Paris V).

Thomson, D.M., Robertson, S.L. & Vogt, R. (1982). Person recognition : the effect of context. *Human Learning*, **1**, 137-154.

Tiberghien, G. (1983). La mémoire des visages. *L'année Psychologique*, **83**, 153-198.

Tiberghien, G. (1985). Mais où sont les stimulus d'antan ? *Psychologie Française*, **30**, 177-183.

Tiberghien, G. (1986a). Context and Cognition : Introduction. *(CPC) European Bulletin of Cognitive Psychology*, **6**, 105-119.

Tiberghien, G. (1986b). Contextual effects in face recognition : some theoretical problems. In H.D. Ellis, M.A. Jeeves, F. Newcombe & A.W. Young (Eds.), *Aspects of face processing*. Dordrecht : Martinus Nizhoff

Tiberghien, G. (1988). What is face semantics, What is face processing ? In W. Young & H.D. Ellis (Eds.), *Handbook of Research in Face Processing*. Amsterdam : North Holland

Press.

Tiberghien, G. (1988a). Modèles de l'activité cognitive : introduction. In C. Bastien, J.P. Caverni, P. Mendelsohn, G. Tiberghien (Eds), *Les Modèles de la Cognition.* Grenoble.: Presses Universitaires de Grenoble.

Tiberghien, G., Cauzinille, E. & Mathieu, J. (1979). Pre-decision and conditionnal research in long term recognition memory. *Acta Psychologica*, **43**, 329-343.

Tiberghien, G.& Lecocq, P. (1983). *Rappel et Reconnaissance.* Lille : Presses universitaires de Lille.

Tulving, E. (1983). *Elements of episodic memory.* New York, Oxford University Press.

Tulving, E. & Thomson, D.M. (1973). Encoding specificity and retrieval processes in episodic memory. *Psychological Review*, **80**, 352-373.

Valentine, T. (1988). Computationnal vision and computer applications : implication for psychological research into face processing. In V. bruce, Report on the Fifth Grange over sands meeting in Face Perception.

Villard, G. (1988) Calcul Formel et Parallèlisme : RESOLUTION DE SYSTEMES LINEAIRES. *Thèse de doctorat*, Institut National Polytechnique de Grenoble.

Wang, S. (1988a). Implementation of threshold automata networks with multi-layers on a HYPERCUBE F.P.S. T20. *RR 725 -M-*. IMAG, Grenoble.

Wang, S. (1988b). Training the multi-layered neural nets using T.R. Based Algorithm. *NEURAL NETWORKS*, Vol 3, Supplement 1.

Wang,S., YE, H. & Robert, F.(à paraître). A PNML neural network for isolated words recognition. *Proceedings of nEuro '88.* First european conference on neural network, 6-9 Juin 1988 : Paris.

Wang, S. (1989) REDUCING THE COMMUNICATION COST IN SIMULATING LAYERED NEURAL NETWORKS ON A HYPERCUBE MACHINE. à paraître dans *PARALLEL COMPUTING 1989*, G.R. Joubert, D.J. Evans & F.J. Peters (eds), North-Holland (1989).

Wang S., Schreiber A., Rousset S. (1989) Connectionist Modelling of a Cognitive Model of Face Identification : Simulation of Context Effects. *Proceedngs of IJCNN-89* (International Joint Conference of Neural Networks).

Weinfeld, M. (1988). A Fully Digital CMOS Integrated Hopfield Neural Network Including the Learning Algorithm, International Workchop on VLSI for Artificial Intelligence, University of Oxford, UK, Juillet 1988.

Wickelgren, W. A. (1969). Context-sensitive coding, associative memory, and serial order in (speech) behavior. *Psychological Review*, 76, 1-15.

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *1960 WESCON Convention Record Part IV*, 96-104, IRE.

Yin, R.K. (1969). Looking upside-down faces. *Journal of Experimental psychology*, **81**, 141-145.

Young, A.W., Hay, D.C. & Ellis, A.W.(1985). The face that launched a thousand slips : everyday difficulties and errors in recognizing people. *British Journal of Psychology*, **76**, 495-523.

Young, A.W., McWeeny, K.H., Hay, D.C & Ellis, A.W.(1986). Access to identity-specific semantic codes from familiar faces. *Quaterly Journal of Experimental Psychology*, **38A**, 271-295.

Young, A. W., Hellawel, D. & Hay, D.C. (1987). Configurational information in face perception. *Perception*, **16**, 747-759.

Young, A.W. & Ellis, A.W. (1988). Semantic Processing. In A.W. Young & H.D. Ellis (Eds.), *Hanbook of Research on Face Processing*. Amsterdam : North Holland.

Yuan, Y. (1984), An example of only linear convergence of trust region algorithms for nonsmooth optimization, *IMA Journal of Numerical Analysis 4*, pp. 327-335.

Yuan, Y. (1985), On the superlinear convergence of a trust region algorithm for nonsmooth optimization, *Mathematical Programming, vol 3*, pp. 269-285.North-Holland.

INPG département des études doctorales

AUTORISATION de SOUTENANCE

VU les dispositions de l'Arrêté du 23 novembre 1988 relatif aux Etudes doctorales

VU les rapports de présentation de Messieurs

. F. FOGELMAN, Professeur
. M. COSNARD, Professeur

Monsieur WANG Shengrui

est autorisé(e) à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, spécialité " Informatique "

Fait à Grenoble, le 21 septembre 1989

Pour le Président de l'I.N.P.G.
et par délégation,
le Vice-Président
P. VENNEHEAU

Résumé :

Les travaux de cette thèse s'inscrivent dans un cadre d'expérimentation de réseaux neuronaux multicouches. Ils concernent l'implantation du modèle sur une machine hypercube, les applications en mémoire des visages et en reconnaissance de mots lus (parole), l'étude du comportement de ces réseaux et l'amélioration de l'algorithme d'apprentissage par rétro-propagation du gradient.

En adoptant une méthode générale de distribution du réseau de cellules, nous avons proposé des algorithmes de communication sur un anneau et sur un hypercube qui résolvent les problèmes de multi-accumulation et de diffusion "all-to-all". Nous montrons que ces algorithmes sont, de plus, asymptotiquement optimaux. Les résultats expérimentaux obtenus sur la machine FPS T40 confirment l'efficacité de ces algorithmes.

Notre simulation de l'identification des visages en contexte constitue une tentative d'utilisation du modèle connexionniste comme nouveau paradigme pour modéliser des phénomènes cognitifs. Elle nous a permis de simuler plusieurs effets dans la reconnaissance des visages qui n'ont pas encore été très bien pris en compte dans d'autres modèles. Par ailleurs, notre application des réseaux multicouches à la reconnaissance de mots met en valeur l'intérêt de l'architecture à connexions partielles et superposées pour traiter des informations temporelles.

L'étude du comportement des réseaux fournit des indices plus clairs sur la dynamique des fonctions réalisées par l'apprentissage et permet d'expliquer la conception de l'architecture du réseau pour notre application.

Enfin, l'adaptation de la méthode de Région de Confiance nous a permis d'obtenir un nouvel algorithme d'apprentissage plus efficace et plus robuste que la rétro-propagation de gradient.

Mots-clés : Réseaux de Neurones, Rétro-Propagation de Gradient, Machine Hypercube, Mémoire des Visages, Reconnaissance des Mots, Dynamique d'Attracteurs, Méthode de Région de Confiance

Abstract :

In this thesis we deal with several topics concerning layered neural networks such as implementation on a hypercube machine, applications to the memory modelling and to words recognition, the study of the dynamic behavior of such networks and the improvement of the learning algorithm.

The implementation of layered neural networks on parallel distributed memory machines presents two key problems concerning the communication between processors : multi-accumulation and "all-to-all" broadcasting. We have proposed several algorithms to solve these problems on rings and on hypercubes of processors. The algorithms on the hypercube are asymptotically optimal. The experimental results obtained on the machine FPS T40 confirm the efficiency of these algorithms.

Our simulation of "face identification in context" constitutes an attempt to use the connectionist model as a new paradigm for modelizing cognitive phenomena. It allowed us to simulate some effects in face recognition that other models did not consider completely. Moreover, our application of layered networks to "spoken words recognition" highlights the importance of partial and overlaped connections in temporal information processing.

The study of the networks behavior helps us to clarify several properties of the function resulted from learning and allows us to explain the conception of the network architecture for our application.

Finally, the adaptation of the "Trust-Region" method has resulted in a new learning algorithm which is more efficient and more robust than the gradient back propagation.

Key words : Neural Neworks, Gradient Back Propagation, Hypercube Machine, Face Memory, Words Recognition, Attractor Dynamics, Trust-Region Method