Construction de réseaux de neurones

Didier Wenzek

THÈSE

présentée par

DIDIER WENZEK

pour obtenir le titre de DOCTEUR

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

(arrété ministériel du 30 mars 1992)

(Spécialité : Informatique)

« CONSTRUCTION DE RÉSEAUX DE NEURONES »

soutenue le 30 septembre 1993

Composition du jury :	Président :	FRANÇOIS ROBERT
	Rapporteurs :	MICHEL COSNARD
		MAX DAUCHET
	Examinateurs :	PATRICK GALLINARI
		DENIS TRYSTRAM

Thèse préparée au sein du
LABORATOIRE DE MODÉLISATION ET DE CALCUL

This thesis was typeset by $\mathcal{AMS}$-LaTeX,
the LaTeX macro system of the American Mathematical Society.

Je tiens à remercier :

Monsieur François Robert qui m'a accueilli au LMC et m'a fait l'honneur de présider le jury de cette thèse.

Messieurs Max Dauchet, Michel Cosnard et Patrick Gallinari qui ont pris le temps de lire et juger mon travail.

Monsieur Denis Trystram en tant que directeur de thèse.

Je tiens également à remercier toutes les personnes du laboratoire, permanents et thésards, qui m'ont apporté leur amitié et soutien durant ces quatres années. Je citerai plus particulièrement Jean-Marc Vincent pour l'intérêt particulier qu'il a porté à mon travail et Hervé Frydlender avec qui j'ai eu la joie de partager bureau et idées.

Je remercie enfin Elisabeth pour sa patience et ses encouragements.

Introduction

Avant-propos

La dénomination de réseaux de neurones recouvre tout un ensemble de mécanismes de calcul inspirés initialement par des modèles issus de la neurobiologie et qui intéressent aussi bien les chercheurs essayant de comprendre le fonctionnement des systèmes nerveux que les techniciens qui y voient une source d'inspiration pour la construction de systèmes automatiques. Similaires quant à leur centre d'intérêt, ces deux démarches se distinguent par leurs objectifs et leurs contraintes. Le but des chercheurs est de trouver un modèle microscopique physiologiquement plausible et rendant compte des caractéristiques macroscopiques du système nerveux. Les réseaux de neurones constituent donc un domaine de recherche où interviennent des chercheurs d'origines très diverses : biologistes qui fournissent la matière première avec une description des éléments et des mécanismes internes, psychologues décrivant les caractéristiques émergentes de l'ensemble et physiciens dont les outils ont déjà permis, dans d'autres domaines, de franchir le fossé entre descriptions microscopique et macroscopique.

De leur côté, les techniciens sont intéressés par un modèle techniquement réalisable et permettant d'améliorer les performances des systèmes automatiques. Ils ne retiendront donc que les propriétés d'ensemble qui leur semblent intéressantes et pourront prendre des libertés sur le plan de la mise en œuvre. De plus, alors que pour le biologiste ou le psychologue il est intéressant d'accumuler des faits même sans avoir, à ce jour, élucidé leurs articulations, le technicien ne peut retenir que des éléments dont les liens sont pleinement compris, quitte à laisser de côté des propriétés intéressantes a priori. Du travail du chercheur, le technicien ne retient donc qu'une très faible partie, le problème étant alors de trouver une mise en œuvre des principes retenus permettant leur pleine utilisation et de savoir quelles sont les performances du système obtenu. Le travail développé dans cette thèse se situe dans cette deuxième perspective.

Cette classification des tâches est bien sûr trop catégorique, chaque chercheur étant amené à utiliser les différents niveaux de langage y compris celui du techni-

cien, ne serait-ce que pour la validation des mécanismes proposés. Elle est toutefois nécessaire pour bien situer cette thèse et éviter les écueils dûs à l'appellation de réseaux de neurones qui peut faire penser que ces modèles captent une grande part des propriétés de comportement du règne animal. Le but de ce travail est de cerner en quoi et comment ces modèles peuvent apporter à la conception de systèmes automatiques.

Afin de se faire une idée des principes directeurs, parcourons maintenant, de manière informelle, les diverses démarches qui ont été mises en œuvre dans le cadre de la recherche technologique sur les réseaux de neurones.

Bref historique des réseaux de neurones

Les différents modèles de réseau de neurones prennent tous leur origine dans le modèle proposé en 1943 par W. S. Mc CULLOCH et W. A. PITTS [34]. Contemporain de la naissance de l'informatique et de l'automatique, ce modèle se place d'emblée sur le plan du signal et de son traitement et laisse de côté le support chimique des signaux ainsi que leur codage en fréquence. A l'époque, son intérêt était de proposer un modèle de neurone physiologiquement plausible et permettant en principe de construire un ordinateur.

Un système nerveux est schématisé par un ensemble de neurones reliés entre eux par des synapses qui modulent les interactions entre neurones. Chaque neurone est un automate à deux états : un de repos et un d'activité au cours duquel le neurone influence ses voisins. Chaque neurone calcule une somme pondérée des signaux fournis par les autres cellules auxquelles il est connecté et ce résultat est comparé à une valeur de seuil. Si le seuil est franchi, le neurone s'active et cette activation est communiquée aux autres neurones. En notant x_i l'état du neurone i, ω_i son seuil d'activation et ω_{ij} le coefficient pondérateur du signal provenant de la cellule j et afférent à la cellule i, le calcul effectué par un neurone i de Mc CULLOCH et PITTS s'exprime donc comme suit :

$$x_i = \begin{cases} 1 & \text{si } \sum_j \omega_{ij} x_j > \omega_i \\ 0 & \text{sinon.} \end{cases}$$

Parmi tous les neurones, on distingue des cellules réceptrices, dont l'état d'activité est régi par l'environnement du système et des cellules effectrices dont l'activité influe sur l'évolution de l'environnement. On désigne parfois ces neurones par les termes de cellules d'entrée et de sortie pour utiliser le langage de la théorie des systèmes. Les paramètres pondérateurs des signaux inter-neurones sont aussi qualifiés de coefficients synaptiques par analogie biologique.

Les neurones sont donc simplifiés à l'extrême et toute la complexité du comportement du réseau repose sur l'organisation des neurones. Lorsqu'on envisage

une organisation de ces neurones en un réseau sans boucle, on obtient un système parcouru par un flot d'information allant des cellules réceptrices aux cellules effectrices. On obtient donc un circuit calculant une fonction booléenne dont les données sont matérialisées par l'état des cellules d'entrée et le résultat par celui des cellules de sortie. Par ailleurs, toute fonction booléenne peut être calculée par un tel circuit et un même réseau peut calculer différentes fonctions suivant la valeur de ses coefficients synaptiques.

Un premier intérêt de ce modèle est de permettre d'adapter la fonction d'un réseau à une fonction désirée en modifiant les coefficients synaptiques. Des algorithmes dits d'apprentissage ont été développés dans ce but. Ils ajustent la réponse du réseau sur un ensemble de points qualifiés d'exemples et sont donc des algorithmes d'interpolation. On retrouve ici la démarche initiée par F. ROSENBLATT (1961) [45] avec le PERCEPTRON et qui après les violentes critiques de M. MINSKY et S. PAPERT (1969) [35] s'est poursuivie sous une forme différente avec les réseaux dits multi-couches et l'algorithme de rétro-propagation du gradient de Y. LE CUN (1985) [6] et de D. PARKER (1985) [38] ainsi que D. RUMELHART, G. HINTON et R. WILLIAMS (1986) [48, 47]. Le succès le plus connu de cette démarche est sans aucun doute le réseau NET-TALK présenté par T. SEJNOWSKI et C. ROSENBERG (1986) [49, 50]. Ce réseau d'une centaine de neurones est capable de lire à voix haute un texte en anglais. A partir d'un ensemble d'exemples d'un millier de mots, représentatifs du point de vue de la prononciation, le système arrive à un taux de réussite de quatre-vingt pour cent sur un ensemble comprenant vingt-mille mots.

L'intérêt de cette approche est double. Elle permet d'aborder la réalisation de systèmes calculant une fonction que l'on n'arrive pas à spécifier de manière algébrique. De plus on obtient un système rapide et dont on peut facilement imaginer une réalisation sous forme d'un circuit électronique. Telle quelle, cette démarche est toutefois naïve pour différentes raisons. En effet, si l'on ne fait intervenir aucune notion de structure, il n'y a aucune raison (et cela est même plus qu'improbable) pour que l'algorithme d'apprentissage permette au réseau d'interpoler correctement la fonction souhaitée en des points non spécifiés parmi les exemples. Les critiques de M. MINSKY et S. PAPERT [35] portaient justement sur ce point et insistaient sur le fait que la structure extrêmement simple du PERCEPTRON limitait sérieusement son domaine d'application. L'algorithme de rétro-propagation du gradient permet certes d'appliquer ce principe d'apprentissage à une classe plus importante de réseaux. Mais le lien entre le schéma d'organisation de ces réseaux et la structure des fonctions qu'ils permettent de réaliser est très mal compris. La difficulté est donc seulement reportée. De plus, on peut mettre en question la validité de cette approche d'interpolation par une fonction lorsque la réponse au problème n'est pas univoque (comme dans le cadre de la reconnaissance de caractères typographiés ou manuscrits où l'interprétation

d'un signe dépend de tout un contexte, sans la donnée duquel plusieurs réponses sont a priori envisageables).

Si l'on envisage une organisation des neurones en un réseau comportant des boucles, le modèle de Mc CULLOCH et PITTS a besoin d'être complété. En effet, il devient alors nécessaire de préciser la manière par laquelle se synchronisent les différents changements d'état dans le réseau. Combien de temps met un signal pour se propager d'un neurone à un autre? A quelle vitesse réagit un neurone face à une variation de son entrée? Répondre à ces questions devient crucial car le comportement du réseau dans son ensemble va alors dépendre de cet aspect dynamique. Imaginons, par exemple, deux neurones reliés de telle manière que l'activité de l'un soit nécessaire et suffisante pour maintenir l'activité de l'autre. Partant d'une situation où seulement un des deux neurones est actif, on peut imaginer trois évolutions possibles: le neurone actif entraîne l'activité du second et les deux neurones se trouvent dans un état stable d'activité mutuelle. Son activité n'étant pas entretenue, le neurone actif peut aussi devenir inactif avant d'avoir pu activer le second neurone et le réseau se trouve dans un état stable d'inactivité mutuelle. Et enfin, si les deux événements se synchronisent, les deux neurones voient leur état alterner indéfiniment (ou du moins tant qu'il n'y a pas de rupture dans la synchronisation, auquel cas le système se stabilise dans l'un des deux états décrits précédemment). Il est donc nécessaire de compléter le modèle de Mc CULLOCH et PITTS par une description de la dynamique de l'ensemble du réseau.

Il y a deux manières d'aborder cette difficulté, ces deux approches évoquant par ailleurs la dualité décrite précédemment entre la nécessité de tenir compte d'une réalité biologique que l'on veut comprendre et la liberté d'interprétation de cette même réalité vue cette fois comme source d'inspiration. Ainsi, on peut envisager un retour à la source, le neurone, et modéliser plus finement le signal, l'influx nerveux, ainsi que sa génération et sa propagation. On peut aussi considérer que le modèle de Mc CULLOCH et PITTS décrit une situation d'équilibre et par suite que n'importe quel système dynamique ayant les mêmes équilibres convient a priori, ce qui revient à présupposer que les propriétés que l'on veut capter sont résumées essentiellement par les positions d'équilibres. C'est ce point de vue qui est adopté pour la plupart des utilisations technologiques des réseaux de neurones et de nombreuses dynamiques différentes respectant ce principe ont été décrites et utilisées (les différences essentielles provenant de l'espace d'états et du temps, discret ou continu, du plus ou moins grand parallélisme et synchronisme entre neurones ainsi que du caractère déterministe ou non, voire stochastique, de ces dynamiques).

D'un point de vue technique et outre les considérations de mise en œuvre effective, ce qui va justifier le choix d'une de ces dynamiques va être l'interprétation

que l'on peut donner à une évolution du système obtenu. Comment interpréter, en termes de résultat, l'état d'équilibre atteint par le réseau? Comment interpréter une évolution qui ne se stabilise jamais? Peut-on rendre compte des situations où de multiples réponses sont à priori envisageables? Ce sont les réponses à ces questions qui vont guider le choix d'une dynamique particulière et justifier telle ou telle autre utilisation du réseau.

En 1982, J. HOPFIELD [25, 26] est le premier à avoir proposé une dynamique de réseau de neurone avec son interprétation en soulignant une analogie avec les verres de spin[1], centrée autour de la construction, pour une certaine classe de réseaux, d'une fonction dépendant de l'état des neurones et dont la valeur décroît au cours de toute évolution du réseau. Utilisés dans un cadre déterministe, ces réseaux permettent donc de trouver les minima locaux (localité en un sens qu'il n'est, pour l'instant, pas utile de préciser) de cette fonction, appelée fonction d'énergie pour poursuivre l'analogie. Ce principe a été étendu par G. HINTON, T. SEJNOWSKI et D. ACKLEY (1984) [24] avec la machine dite de BOLTZMANN où une dynamique stochastique permet d'obtenir un état stationnaire au cours duquel la probabilité d'observer une configuration donnée est d'autant plus forte que l'énergie associée à cette configuration est faible. Complétée avec un principe de recuit simulé, encore une fois inspiré par la physique statistique, cette méthode permet de trouver les minima de la fonction d'énergie.

La première utilisation qui peut être faite de ce mécanisme se situe naturellement dans le cadre de l'optimisation combinatoire. Le principe est de coder le problème de façon a pouvoir identifier la quantité à minimiser avec l'énergie d'un réseau dont on obtient ainsi les coefficients synaptiques. Cette approche, initiée par J. HOPFIELD [29, 30], a été poursuivie par de nombreux auteurs dont E. AARTS et J. KORST [2] ainsi que J.J. NIEZ [20] et L. HÉRAULT [19]. Ce dernier a notamment utilisé cette approche dans le cadre de la mise en correspondance de graphes et l'a surtout étendue à la réalisation d'un système perceptif en adoptant le point de vue de la théorie du gestaltisme[2]. Le but de ce système est d'extraire, parmi un nuage de traits, ceux pouvant correspondre à une ligne la plus régulière possible. Le résultat souhaité est défini sous forme d'une contrainte exprimant la nécessité de rejeter au moins un élément de tout couple de traits qui ne sont pas cocirculaires et ce avec d'autant plus de force que ces points sont proches et leurs directions incompatibles. Cette contrainte est ensuite traduite sous une forme

[1]Le terme de verre de spin (spin glass) fut crée vers 1970 pour désigner des matériaux comportant des impuretés magnétiques qui leur confèrent un comportement magnétique très riche.

[2]Le gestaltisme (gestalt-theorie) est une théorie de la perception qui est née en Allemagne au début de ce siècle et qui s'appuie sur la notion de forme optimale. Le terme *Gestalt* désigne en allemand un modèle, une forme ou encore une configuration et se rapproche donc du mot anglais *pattern*. Cette théorie postule que les ambiguïtés rencontrées lors de l'interprétation d'une image sont résolues par un mécanisme sélectionnant les formes les plus régulières.

identifiable à l'énergie d'un réseau. En faisant décroître son énergie le système évolue vers des états de moins en moins contraints, ce qui correspond, dans ce cas, à un nuage de traits de plus en plus cohérent. Le mécanisme de recuit permet alors de raffiner ce principe.

D'une manière duale à cette utilisation des réseaux de HOPFIELD où les coefficients synaptiques sont construits de manière algébrique, on trouve la démarche suivie par G. HINTON et T. SEJNOWSKI qui consiste à utiliser un algorithme d'apprentissage [22, 23]. Pour clarifier l'exposé, plaçons-nous dans le cadre de la reconnaissance de caractères manuscrits et isolés où plusieurs réponses sont a priori envisageables. On veut donc réaliser un système associant à un signe une lettre de l'alphabet, la suite de lettres ainsi fournie étant utilisée par une autre machine, disons un dictionnaire, pour retrouver le mot d'origine et corriger d'éventuelles erreurs. L'idée est d'utiliser une machine de BOLTZMANN qui, au cours de son état stationnaire, va associer une lettre à un signe avec une fréquence d'autant plus grande que cette utilisation du signe est fréquente. Le but est donc de faciliter le travail de la machine dictionnaire en l'orientant dans une direction qui a plus de chance d'aboutir. Un algorithme d'apprentissage pour une machine de BOLTZMANN est donc un algorithme d'identification qui consiste à adapter la distribution stationnaire d'états de cette machine à une distribution de couples représentant un objet et une de ses interprétations possibles. En phase d'utilisation, un mécanisme supplémentaire permet d'obtenir la distribution sur l'espace des interprétations conditionnellement à l'objet à identifier. Là aussi se pose le problème de la structure des distributions accessibles par un réseau donné.

Dans le cadre des dynamiques déterministes, on trouve une démarche similaire depuis la généralisation par L. B. ALMEIDA (1987) [3, 4] et F. J. PINEDA (1987) [40, 41, 42, 43] de l'algorithme de rétro-propagation du gradient à des réseaux comportant des boucles de rétro-action. Les différents attracteurs accessibles par le réseau sont considérés comme autant de réponses a priori possibles et les entrées du système jouent le rôle de paramètres de commande qui, en altérant le profil de l'espace des phases, sélectionnent les réponses adéquates. Ce principe a, entre autres, été utilisé par B. VICTORRI (1988) [51, 14, 44] dans un cadre linguistique pour le problème bien particulier de l'interprétation sémantique à donner à l'adverbe *encore* suivant son emploi. L'espace d'états du système représente les diverses directions suivant lesquelles peut se déployer le sens de l'expression étudiée. L'espace d'entrée permet de prendre en compte les indices qui influent sur la signification de l'expression. La nécessité de comprendre la structure de l'espace des phases du réseau et surtout la manière suivant laquelle elle se modifie en fonction des paramètres de contrôle devient évidente sur cet exemple.

Dans cette lignée on peut aussi citer les travaux de K. FUKUSHIMA et son système, le NEOCOGNITRON [15], proposé en 1980 et amélioré en 1988 avec un

mécanisme d'attention sélective [16]. Le NEOCOGNITRON a été conçu pour la reconnaissance de caractères et une attention particulière a été portée au cas ou plusieurs caractères sont entremêlés dans l'image de départ. Bien que effectivement réalisée uniquement pour la reconnaissance de chiffre, cette étude est intéressante car elle aborde une difficulté essentielle rencontrée dans le cadre de la lecture de l'écriture manuscrite: les symboles porteurs de sens doivent être extraits pour pouvoir être reconnus. Schématiquement, le NEOCOGNITRON est composé de deux réseaux entremêlés. Le premier, allant des cellules réceptrices vers les cellules codant l'interprétation de l'image, permet de dépister plusieurs réponses possibles. Le second, dont le flot d'information parcourt le système en sens inverse, sélectionne, parmi les cellules du premier réseau, celles ayant conduit à l'interprétation la plus sollicitée. D'une manière imagée, le premier réseau propose des solutions que le second sélectionne. Le mécanisme proposé étant entièrement déterministe, le choix effectif d'un caractère parmi tous ceux présents sur l'image, va dépendre de l'état initial du réseau. On peut alors basculer d'un choix vers un autre en perturbant l'état d'équilibre du système.

A la vue de ce panorama, non exhaustif, du domaine de la mise en œuvre des modèles des réseaux de neurones dans un cadre technologique, on peut souligner plusieurs points: diversité des sources d'inspiration et des analogies utilisées, usage de méthodes d'interpolation et d'identification pour adapter un système à l'aide d'exemples, difficulté pour expliciter des structures et, par suite, rupture avec la démarche traditionnelle en informatique de décomposition en étapes du problème à résoudre[3]. Au cours de ce bref historique, on a surtout cité des exemples d'applications tournées vers la reconnaissance de formes. Mais ces techniques commencent, depuis peu, à être utilisées dans des domaines aussi divers que la gestion de plans, la prévision de données économiques ou le contrôle en robotique. Les réseaux de neurones constituent un sujet de travail actuellement en vogue au sein de la communauté de recherche sur la conception de machines pouvant être qualifiées de machines intelligentes. Il en est de même pour de nombreuses autres techniques qui sont présentées comme des verrous ouvrant de nouvelles portes sur l'intelligence artificielle. On peut ainsi citer les logiques floues ou les algorithmes génétiques. Toutes ces techniques introduisent une part d'intuition et de non dit. Les algorithmes génétiques invoquent une sélection par le hasard. La logique floue introduit l'artifice qui consiste à quantifier de manière intuitive le qualificatif. Les réseaux de neurones, quant à eux, s'appuient sur leur souplesse qui permet a priori de les façonner à volonté à l'aide d'algorithmes d'apprentissage.

---

[3]On peut aussi remarquer un usage souvent peu approprié de termes provenant des domaines qui sont à l'origine des analogies. Par exemple, le terme d'apprentissage implique une notion d'échafaudage d'un modèle qui permet d'aller au-delà de ce qui a été pris comme exemple. Or rien n'assure aux algorithmes dits d'apprentissage une quelconque propriété qui pourrait être interprétée en ces termes, à la seule vue de la manière dont ils sont formulés actuellement.

Pourquoi ce besoin d'introduire une part d'intuition et de non dit? En quoi la notion de calcul serait-elle inadaptée à la conception de machines en interaction avec notre environnement? Avant de mettre un terme à cette introduction, faisons un bref détour sur la démarche constructive propre à l'informatique et sur les difficultés liées au champ d'investigation qu'est la construction de machines intelligentes.

Difficultés d'une démarche constructive dans les domaines d'applications visées pour les réseaux de neurones

Reprenons le schéma de base d'une démarche constructive pour le calcul d'une fonction d'un ensemble de données dans un ensemble de réponses. Le but d'une telle démarche est double. Il s'agit simultanément de *définir* cette fonction et de *construire* un mécanisme, de proposer une méthode permettant d'obtenir la réponse associée à une donnée. La solution la plus simple pour arriver à cette fin est d'expliciter directement la réponse associée à chaque entrée. La fonction est alors définie par la donnée exhaustive de toutes les associations, un graphe. Et le mécanisme informatique permettant de calculer cette fonction est tout simplement une mémoire retrouvant la réponse demandée. Lorsque l'espace d'entrée et celui de sortie deviennent trop grands la réponse ne peut plus être simplement retrouvée, elle doit être construite. L'idée est de n'envisager que des ensembles de données obtenues par combinaisons d'éléments si simples que toutes les opérations pouvant porter sur eux puissent matériellement être définies explicitement. Traditionnellement en informatique, la combinaison la plus utilisée est la juxtaposition. On construit ainsi des listes d'objets et les seuls ensembles envisagés sont des ensembles de listes. De même un des ensembles d'objets atomiques le plus simple que l'on pourra envisager est celui ne comportant que deux éléments, 0 et 1. Pour définir une fonction sur un tel ensemble, défini par combinaison d'éléments atomiques, dans un autre ayant lui aussi une telle structure, on procède alors par fermeture transitive : on définit explicitement l'image des éléments atomiques ainsi que, pour chaque constructeur, l'image d'une construction. Le calcul de la valeur d'une fonction consiste alors à décomposer l'information initiale en divers éléments puis d'exprimer la réponse associée comme une combinaison de résultats partiels obtenus avec chacun de ces éléments. L'intérêt de cette approche est de spécifier le travail que l'on attend du système tout en décrivant un chemin pour aboutir à une réponse.

Cette démarche constructive est intrinsèquement liée à la notion de calcul. Née avec l'algèbre et la manipulation des nombres (la représentation d'un nombre dans une base est la première étape vers une spécification des opérations sur les nombres à l'aide de tables et de combinaisons d'opérations élémentaires), cette approche a permis la réalisation de calculateurs automatiques. Ces ordinateurs ont naturellement d'abord été utilisés pour la manipulation de nombres (calcul de lo-

garithmes ou autres fonctions), puis pour diverses transformations d'informations structurées (compilateurs, bases de données, système d'inférences logiques...) et simulations de modèles physiques (météorologie, systèmes mécaniques...). Pour toutes ces utilisations, le calcul reste dans le *domaine clos* d'un modèle. C'est à l'utilisateur de justifier le modèle et surtout d'interpréter les résultats obtenus. On tente maintenant d'utiliser ces méthodes pour des systèmes en *interaction* avec notre univers ou nos comportements (déplacement d'un robot dans une pièce ou lecture d'un texte manuscrit). Malgré des succès indéniables, la tâche s'avère ardue, le pas à franchir entre simulation et interaction étant plus important que prévu.

Prenons l'exemple d'un robot devant se déplacer dans une usine. Si on demande au robot de décrire une trajectoire précise, il est possible de concevoir un processus effectuant cette tâche. Il est même possible de tenir compte des écarts de trajectoire, inévitablement dus aux jeux mécaniques des pièces constituant le robot, à l'aide de capteurs et de signaux guides. Mais un obstacle peut toujours avoir été malencontreusement oublié sur le trajet du véhicule et on sort alors du cadre spécifié par le modèle. On peut raffiner le système et prévoir ce type de situation ainsi qu'un processus de contournement. Le problème devient déjà beaucoup plus complexe car on ne sait, a priori, ni où ni quel obstacle peut être rencontré. De plus, parmi tous les trajets de contournement certains risquent à leur tour de s'avérer impraticables. La difficulté semble devenir insurmontable lorsque le nombre de causes potentielles de déroutement augmente et ce d'autant plus que ces différentes causes vont interférer entre elles. Comment imaginer un robot dans une cuisine? L'ensemble des situations possibles devient si complexe qu'il devient inimaginable de décrire un modèle les décrivant toutes. On est contraint de se contenter de modèles partiels.

Si l'on considère maintenant le problème de la reconnaissance de l'écriture manuscrite, la difficulté provient du fait qu'un même signe peut devoir être compris de façons différentes suivant les autres signes qui lui sont accolés, suivant la personne les ayant écrits ou même suivant le sens de la phrase. Il est ainsi impossible de simplement considérer la lecture d'un mot comme la mise bout à bout des résultats obtenus lettre après lettre. Dans le cas d'une écriture liée, la découpe d'un mot en une suite de lettres semble même ne pouvoir être faite qu'une fois le mot reconnu, qu'une fois connu l'ensemble des lettres à rechercher dans le signe. Ce problème de contexte implique qu'une modélisation de l'écriture des lettres manuscrites isolées restera nécessairement ambiguë. Deux symboles traduits par un même signe ne pourront plus être distingués par un modèle ignorant le contexte. Celui-ci peut au mieux retrouver l'ensemble des symboles pouvant avoir été ainsi retranscrits. On est contraint de se contenter de modèles ambigus.

Quelles sont les réponses qui ont été apportées à ces deux difficultés, incomplétude et ambiguïté du modèle? Commençons par la première: l'impossibilité

pour le modèle de décrire toutes les situations potentielles. Il est nécessaire de confondre plusieurs entrées en une seule. Pour cela on traduit la notion de similitude entre entrées à l'aide d'une distance et face à une entrée impropre à la construction d'une réponse à l'aide du modèle on cherche un ensemble cohérent de données proche de l'entrée effective, voir un des plus proches possible, puis l'on construit la réponse à partir de cette approximation de la situation. La réponse est similaire pour la seconde difficulté : le modèle ne permet pas de distinguer toutes les situations qui devraient l'être. On fait alors un choix parmi l'ensemble des situations confondues en une même entrée et l'on construit la réponse à partir de ce choix. On fait donc une hypothèse sur le contexte et l'on peut quantifier ce choix en fonction de la fréquence de chaque contexte. Pour tenir compte de l'incomplétude et de l'ambiguïté du modèle permettant de construire une réponse, on complète donc cette approximation de l'univers, l'ensemble des situations, d'une notion de proximité entre situations et une notion de fréquence de ces situations. Face à une situation donnée, le système se ramène donc à une entrée permettant la construction d'une réponse en s'appuyant sur la notion de proximité et fait un choix relativement à la notion de fréquence entre les différentes manières qui lui sont offertes par le modèle pour construire sa réponse.

On peut concevoir qu'avec cette approche un système automatique puisse avoir un comportement en moyenne adapté à son environnement. On peut même concevoir que l'on puisse améliorer le comportement de ce système en modifiant les critères d'approximation de la situation et ceux d'extrapolation d'un contexte. Mais cette approche est difficilement compatible avec une démarche constructive. En effet, on ne peut plus alors simplement décomposer une tâche en une succession de tâches élémentaires, réaliser de manière indépendante autant de petits systèmes réalisant ces tâches élémentaires, puis connecter les uns aux autres ces systèmes partiaux pour obtenir le système global. Le système ainsi obtenu va, en effet, être gêné par le fait qu'un sous-système qui se trouve en aval peut être amené à devoir distinguer deux cas qui ont été confondus par un sous-système en amont. Éclairons nos propos par un exemple pris dans le domaine de la reconnaissance de caractères manuscrits isolés.

Considérons que la première étape de cette reconnaissance est l'extraction dans l'image d'un ensemble de traits et des liens entre ces traits. Les choix essentiels à faire au cours de cette description de l'image concernent la connexion des différents éléments : par exemple, ces deux traits doivent-ils être considérés comme connectés malgré le fait qu'un léger espace les sépare ? Au cours de cette étape on va donc confondre différentes situations en utilisant une notion de proximité. C'est-à-dire que des images différentes vont donner lieu à une même interprétation en termes de traits. La deuxième étape consiste alors à utiliser l'information structurelle obtenue pour construire la réponse. Que faire lorsque l'ensemble des traits est incohérent, qu'il ne correspond à aucun caractère ? Il va falloir faire un choix, enlever tel trait ou rajouter tel lien, de façon à se ramener dans un

cas prévu par le modèle. Mais même si l'on dispose d'un contexte permettant d'énoncer que tel choix est plus probablement correct, notre choix restera nécessairement arbitraire, puisqu'à ce niveau on ne dispose pas de l'image. On essaye donc de distinguer arbitrairement des situations qui ont été confondues tout aussi arbitrairement. Trancher entre une présence ou une absence de lien en un point de l'image demande une information de cohérence relative aux autres points de l'image et cette information de contexte n'est obtenue qu'après la seconde étape. Un exemple d'image entraînant une telle situation est donné par la figure .1.

La difficulté, rencontrée lors d'une spécification constructive d'un système en interaction avec un environnement incomplètement modélisé, est donc que des choix ou des hypothèses doivent être faits *a priori* à un moment où l'on ne dispose pas de suffisamment d'information. On aboutit alors à une succession de problèmes mal posés. Face à ce problème la tentation des techniques faisant appel aux réseaux de neurones est de jouer sur leur adaptabilité pour essayer de concevoir le système dans sa *globalité*, puisque toute décomposition semble arbitraire, puis de le façonner *in situ* à partir d'exemples, puisque l'on n'arrive pas à exprimer de manière algébrique le travail à faire.

Organisation de la thèse

Le premier objet de cette thèse est de faire ressortir le fait que les réseaux de neurones ne nous dispensent nullement d'une spécification préalable à tout apprentissage. Plus le problème que l'on veut faire résoudre mécaniquement par un réseau sera ambitieux, plus il sera vain d'espérer obtenir un système correct à partir d'un premier système choisi a priori et que l'on améliore sur un certain nombre d'exemples. Non seulement il importe de choisir le système initial en fonction du problème à résoudre, mais il importe aussi de restreindre l'espace des systèmes explorés au cours de l'apprentissage.

Le deuxième objet de cette thèse est de dégager les principes permettant d'aborder cette spécification de manière constructive : comment décomposer une spécification en spécifications partielles réalisables, puis combiner les réseaux y répondant pour obtenir un réseau répondant à la spécification globale ?

Une fois un cadre de travail cerné par un premier chapitre exposant les différents modèles de réseaux de neurones, leurs modes d'utilisation ainsi que divers points techniques, seront présentés dans un second chapitre quelques exemples de réalisation. Outre de montrer comment ces méthodes peuvent aboutir à une réalisation effective, le but de ces exemples est d'acquérir un savoir faire vis à vis du problème de la construction de réseaux. Le chapitre suivant aborde alors le problème de l'amélioration d'un système sur la base d'une suite d'expériences et

fait ressortir le fait que les algorithmes d'apprentissage n'éludent pas la question de l'adéquation de la structure du réseau à améliorer à la structure du problème à résoudre mécaniquement par ce réseau. Enfin, le dernier chapitre propose un cadre de constructions associé à un cadre restreint de spécifications : celui de la recherche, dans un ensemble fini, d'un élément vérifiant un certain nombre de contraintes.

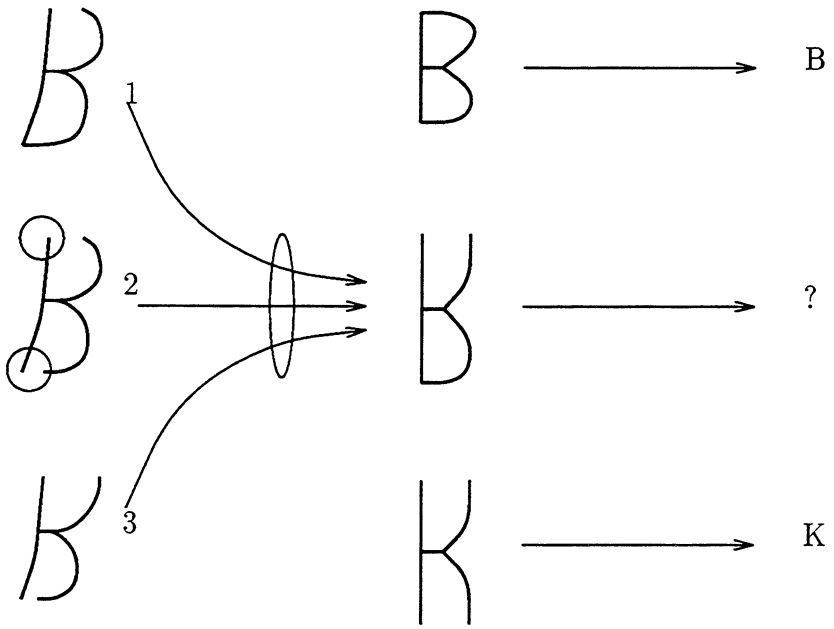

FIG. .1 - . Juxtaposition d'un système transformant une image en une description en termes de traits, avec un système associant à un ensemble de traits une lettre de l'alphabet. Lors de la phase d'extraction des traits et de leurs liens, les images n° 1, 2 et 3 sont confondues et donnent lieu à la même interprétation en termes de traits. Ainsi, lorsque l'entrée du système est l'image n° 2, seul l'espace de la partie inférieure de l'image est inférieur au seuil critique utilisé et donc seul l'espace supérieur est effectivement considéré comme une absence de lien. L'ensemble des traits obtenu est alors incohérent pour le second système. Celui-ci doit arbitrairement soit faire l'hypothèse de la présence de la lettre K en considérant que le lien inférieur est de trop, soit faire le choix d'ajouter le lien supérieur et de signaler la présence de la lettre B. Si ce choix est fait sans retour d'information à la partie amont du système, il ne peut que rester arbitraire. Ainsi, dans le cas de l'image n° 3, qui fournit la même description et qui est donc indistinguable pour le système aval de l'image n° 2, l'hypothèse de la lettre K semble plus appropriée alors que le choix de la lettre B semble plus raisonnable pour l'image n° 1. De même, les choix de la partie amont ne peuvent que rester arbitraires s'il ne sont pas guidés par l'aval qui est le seul à détenir la notion de cohérence d'un ensemble de traits et de liens. Il ne suffit donc pas de se ramener à une entrée cohérente en fonction de critères de distance et de fréquence : il faut coordonner ces choix.

CHAPITRE I

Mécanismes de base et principes de mise en œuvre

Introduction

Une des caractéristiques du domaine de recherche que constituent les réseaux de neurones est la multiplicité des modèles utilisés. Les différences sont parfois minimes. On trouve ainsi divers types de codage pour les états des cellules ainsi que divers choix pour les fonctions d'activation contrôlant l'activité des neurones. D'autres différences sont plus techniques. On peut citer, par exemple, le choix d'une dynamique, à temps continu ou discret, qui va surtout influer sur les domaines d'attraction des points d'équilibre. Certaines différences vont, par contre, être beaucoup plus fondamentales et changer les interprétations que l'on pourra avoir de l'évolution de ces systèmes. L'approche stochastique apporte ainsi un point de vue totalement nouveau.

Cette diversité au niveau des modèles et des mécanismes de base se répercute au niveau de leur mise en œuvre au sein d'un système automatique. Ces différentes exploitations des réseaux de neurones ont pour point commun une conception analogique de la notion de calcul. Elles se fondent sur une correspondance entre les états du réseau et les réponses envisageables pour le problème considéré. Une évolution de l'état du réseau correspond alors à un calcul et l'état d'équilibre éventuellement atteint au résultat recherché. Outre les différents codages possibles des réponses par les états du réseau, la source des diversités réside en la manière d'établir le lien entre les données du calcul et le choix d'une évolution particulière.

Pour que cette conception analogique du calcul aboutisse effectivement à la réalisation de systèmes, il faut pouvoir maîtriser l'allure de l'espace des phases du système dynamique utilisé. Vu l'importance accordée aux points d'équilibre il est ainsi nécessaire de connaître les conditions garantissant qu'une évolution aboutisse à un point d'équilibre. Il est aussi important de pouvoir cerner les conditions permettant à un point d'équilibre particulier d'être atteint. Les réponses à ces questions permettront d'établir le lien, d'une part, entre réponses et états et, d'autre part, entre données et conditions pour atteindre un point d'équilibre particulier. Ce lien étant établi, on pourra alors interpréter, en termes de calcul,

l'évolution de l'état du réseau.

Le but de ce chapitre est de présenter un modèle le plus général possible de réseau de neurones. Aussi la définition proposée distingue les trois niveaux suivants : caractérisation des points d'équilibre, mécanisme permettant d'obtenir ces points d'équilibre et interprétation de ces points en termes de réponses à un calcul. Les trois première sections, axées sur les réseaux de neurones déterministes, décrivent le cadre, les principes d'utilisation puis les points techniques de convergence vers un état d'équilibre. A ce point de l'exposé, on a ainsi une vue des différents niveaux de difficulté rencontrés lors de l'utilisation technologique des réseaux de neurones. L'objet des deux dernières sections est de décrire les approches asynchrone et stochastique en les situant vis-à-vis de l'approche déterministe.

1. Vers une définition générale des réseaux de neurones

Un réseau de neurones est un ensemble de systèmes élémentaires, ou neurones, interagissant entre eux. Il nous faut donc décrire deux points : le comportement d'un neurone et l'interaction entre neurones. La description des systèmes élémentaires, comme celle du système global, se fait en termes de liens entre une entrée et une sortie ou réponse. Le comportement d'un neurone i est donc décrit par le lien entre son entrée u_i et sa réponse x_i. L'interaction entre neurones est décrite par le lien entre le vecteur x, décrivant la réponse de tous les neurones, et le vecteur u résumant l'entrée de tous les neurones. Un vecteur de réponses élémentaires x est alors considéré comme une réponse du système global, le réseau, si ces deux liens sont vérifiés simultanément (*i.e.* la réponse de tout neurone correspond à son entrée et celle-ci est bien induite par la réponse des autres neurones.).

Précisons maintenant les choix communs à presque tous les modèles de réseaux de neurones qui se trouvent dans la lignée des travaux de W. S. Mc Culloch et W. A. Pitts [34]. Premièrement l'entrée u_i et la réponse x_i d'un neurone i sont considérées comme des quantités scalaires (appartenant à $\mathbb{R}$) et liées par un lien fonctionnel : une fonction dite de seuillage f. Le premier lien que doit vérifier le vecteur des réponses élémentaires pour constituer une réponse globale du réseau est donc le suivant :

$$x_i = f(u_i) \quad \text{pour tout neurone } i. \tag{1}$$

Deuxièmement, le lien entre l'entrée u_i d'un neurone i et la réponse, ou état, des autres neurones est choisi linéaire (ou affine si l'on veut tenir compte de l'influence sur ce neurone i d'une entrée du système global). Étant donné un vecteur x de réponses élémentaires correspondant à une réponse globale du réseau, le vecteur u des entrées de chaque neurone est alors l'image du vecteur x par un opérateur

affine Ω, c'est à dire :

$$u_i = \sum_j \omega_{ij} x_j + \omega_i \quad \text{pour tout neurone } i. \tag{2}$$

On construit donc un réseau de neurones en se donnant un ensemble I de n neurones et un certain nombre de connexions dirigées et pondérées par des coefficients inter-neurones (ω_{ij}), dits coefficients synaptiques. Les coefficients ω_i sont soit donnés par construction et jouent alors le rôle de seuil, soit modulés par l'extérieur du réseau et jouent alors le rôle d'entrée. Un vecteur de réponses élémentaires x est alors considéré comme une réponse du système global si il vérifie simultanément les liens 1 et 2 :

$$x_i = f(\sum_j \omega_{ij} x_j + \omega_i) \quad \text{pour tout neurone } i.$$

Nous retiendrons la définition structurelle suivante :

DÉFINITION (MCCULLOCH & PITTS). *Un* réseau de neurones *est défini à partir*

- *d'un ensemble I de n éléments appelés cellules ou* neurones,

- *d'un sous-ensemble Q de $\mathbb{R}$ codant les différents états accessibles par un neurone,*

- *d'une fonction f de $\mathbb{R}$ dans Q,*

- *d'un graphe (I, S) décrivant les connexions entre ces neurones,*

- *pour chaque couple (i,j) de S, d'un réel ω_{ij},*

- *d'un ensemble d'entrées E,*

- *et pour chaque élément i de I, une fonction ω_i de E dans $\mathbb{R}$.*

Un réseau de neurones construit à partir de ces données, est alors un système dont la réponse à une entrée e de E est un vecteur x de $\mathbb{R}^n$ vérifiant :

$$\text{pour toute cellule } i, \quad x_i = f\Big(\sum_{j \ tq \ (i,j) \in S} \omega_{ij} \, . \, x_j + \omega_i(e) \Big) \quad .$$

On appelle, respectivement, synapses *et* coefficients synaptiques *les éléments (i,j) de S et les réels ω_{ij}. Une fonction ω_i est qualifiée de* seuil[1] *lorsqu'elle est constante sur E. La fonction f est appelée* fonction de seuillage.

[1]Nous donnons donc au terme *seuil* un sens différent de celui usuel : une quantité qui est comparée au signal reçu par le neurone. Nous considérons ici le seuil comme un signal particulier résumant tous les signaux reçus par un neurone et qui ne sont pas émis par le réseau lui même.

Pour achever la définition d'un réseau de neurones il faudra spécifier le mécanisme lui permettant de trouver effectivement un vecteur de réponses correct. On utilisera souvent la définition plus manipulable qui suit et qui revient à s'abstraire de la structure en réseau des neurones et de la manière suivant laquelle une entrée est transformée en un vecteur de seuils.

DÉFINITION. *Étant données une application affine* Ω *de* $\mathbb{R}^n$ *dans* $\mathbb{R}^n$ *et une application* f *de* $\mathbb{R}$ *dans un sous-ensemble* Q *de* $\mathbb{R}$, *on appelle réseau de neurones un système dont la réponse à une entrée* e *de* $\mathbb{R}^n$ *est un vecteur* x *de* $\mathbb{R}^n$ *vérifiant l'équation implicite :*

$$x = F(\Omega(x) + e) \quad ,$$

où F *est la fonction de* $\mathbb{R}^n$ *dans* Q^n *définie par :*

$$F_i(u) = f(u_i) \quad .$$

On notera ω_{ij} *les éléments de la matrice de la partie linéaire* $\mathcal{L}(\Omega)$ *de l'application affine* Ω, *exprimée dans la base canonique de* $\mathbb{R}^n$. *On appellera* matrice de connexion *du réseau cette matrice.*

On notera ω_i *les composantes du vecteur* $\Omega(0)$ *exprimé dans la base canonique de* $\mathbb{R}^n$. *On appellera* vecteur des seuils *du réseau ce vecteur.*

Plusieurs points de cette définition restent à préciser. Quels choix peut-on faire pour l'espace d'états d'un neurone et la fonction de seuillage? Quelles sont les mécanismes permettant de trouver un vecteur de réponses?

1.1. Espace d'états des neurones et fonction de seuillage

Contemporain de la naissance de l'informatique, le modèle de MC CULLOCH et PITTS considère un espace d'états discret de deux éléments. L'état d'un neurone est interprété comme la réponse à une question alternative et les signaux reçus par un neurone comme différents votes, ayant plus ou moins de poids, pour l'une des deux alternatives. Afin de rester dans la lignée de ce principe, la fonction de seuillage f est souvent choisie croissante et symétrique par rapport au point $(0, f(0))$. Dans le but d'utiliser les outils du calcul différentiel ou d'être plus proche du modèle électrique qui permettra la réalisation finale du système, on envisage souvent un espace d'états continu et on prend une fonction de seuillage dérivable. L'espace d'états Q reste le plus souvent borné et les choix les plus usuels sont les suivants :

$$f(u) = \begin{cases} 1 & \text{si } u > 0 \\ 0 & \text{sinon} \end{cases} \qquad \text{et} \quad Q = \{0,1\} \quad ,$$

$$f(u) = \begin{cases} 1 & \text{si } u > 0 \\ -1 & \text{sinon} \end{cases} \qquad \text{et} \quad Q = \{-1, 1\} \quad ,$$

$$f(u) = 1/(1 + \exp(-2\lambda u)) \quad \text{où } \lambda \text{ est un réel positif} \quad \text{et} \quad Q =]0, 1[$$

et

$$f(u) = \text{th}(\lambda u) \qquad \text{où } \lambda \text{ est un réel positif} \quad \text{et} \quad Q =]-1, 1[\quad .$$

Ces choix sont fortement liés entre eux. En effet le changement de repère affine B, qui à un vecteur x fait correspondre le vecteur x' tel que $x_i' = 2x_i - 1$, permet de passer de manière biunivoque d'un réseau de neurones ayant $]0, 1[^n$ comme espace d'états et $1/(1 + \exp(-2\lambda u))$ comme fonction de seuillage à un réseau ayant $]-1, 1[^n$ comme espace d'états et $\text{th}(\lambda u)$ comme fonction de seuillage. Au cours de ce changement de repère, l'opérateur affine Ω du premier réseau est transformé en l'opérateur affine $\Omega \circ B^{-1}$. Ce même changement de repère établit aussi une correspondance biunivoque dans le cas d'un espace d'états discret.

Par ailleurs, lorsque le paramètre λ tend vers l'infini la valeur des deux fonctions de seuillage $1/(1 + \exp(-2\lambda u))$ et $\text{th}(\lambda u)$ tendent respectivement vers les fonctions :

$$f(u) = \begin{cases} 1 & \text{si } u > 0 \\ 0 & \text{sinon} \end{cases} \qquad \text{et} \qquad f(u) = \begin{cases} 1 & \text{si } u > 0 \\ -1 & \text{sinon} \end{cases}$$

en tout point u non nul.

L'existence de ces liens montre le peu d'importance qu'a le choix de l'intervalle codant les états des neurones et permet surtout de justifier l'utilisation d'un domaine continu lorsqu'on envisage de coder la réponse à une question alternative par l'état d'un neurone. En effet, dans le cadre d'une application, on pourra choisir un coefficient λ suffisamment grand pour pouvoir départager les réponses du réseau.

Comme exemple de fonction de seuillage faisant intervenir un espace d'états non borné et rompant par ailleurs la symétrie, on peut citer la fonction suivante utilisée par K. FUKUSHIMA pour le NEOCOGNITRON [15] :

$$f(u) = \begin{cases} u \text{ si } u > 0 \\ 0 \text{ sinon} \end{cases} \quad .$$

En réalité, les neurones du NEOCOGNITRON ont un comportement plus complexe : les signaux inter-neurones sont tous positifs et il est donc nécessaire de

départager les signaux qui ont un rôle d'inhibition de ceux ayant un rôle d'activation. Le signal u reçu par un neurone est construit comme suit, à partir d'une combinaison linéaire (à coefficients positifs) e des signaux d'activation et d'une combinaison linéaire (toujours à coefficients positifs) h des signaux d'inhibition :

$$u = \frac{1+e}{1+h} - 1 \quad .$$

On pourra vérifier qu'un tel neurone a un comportement similaire à celui d'un neurone de Mc CULLOCH et PITTS : il n'est actif que si le signal d'activation e est plus fort que le signal inhibant h. L'auteur K. FUKUSHIMA ne donne pas la raison de ses choix, mais on peut penser qu'ils ont pour but de se rapprocher au mieux du système électronique choisi pour réaliser le NEOCOGNITRON (La restriction à des signaux positifs semble ainsi être dû à un codage en fréquence).

On retiendra de ce dernier exemple la liberté que l'on a pour le choix du comportement d'un neurone, le choix final pouvant même être dicté par des raisons techniques de réalisation. Il semble uniquement nécessaire de considérer qu'un neurone est un système ayant un état résumable par une quantité scalaire et que cet état puisse être modulé par des signaux activateurs et inhibiteurs, ayant pour action respective d'augmenter et de baisser l'état d'activation du neurone. En retour cette activation du neurone agit sur les neurones auxquels il est connecté.

1.2. Mécanismes

Nous avons défini un réseau de neurones en spécifiant uniquement le lien que doivent vérifier les réponses élémentaires de chaque neurone pour pouvoir être considéré comme une réponse du réseau. Cherchons maintenant à compléter la définition d'un réseau par la donnée d'un mécanisme permettant de trouver effectivement un vecteur de réponses élémentaires vérifiant le lien implicite que nous avons pris comme caractérisation des réponses du réseau.

Plus précisément, on cherche un mécanisme contrôlant au cours du temps l'évolution de la réponse x_i de chaque neurone i de façon à aboutir, pour tout vecteur d'entrée e, à un vecteur de réponses x tel que :

$$(3) \qquad x_i = f(u_i) \quad \text{pour tout neurone } i \text{ et avec } u_i = \sum_j \omega_{ij} x_j + \omega_i(e),$$

en reprenant les notations de la définition structurelle.

D'un point de vue technique, les critères de choix vont être la possibilité de simuler ce mécanisme avec un système électrique ou de calculer les trajectoires avec un système informatique. Dans le deuxième cas, il est souhaitable d'obtenir un algorithme rapide et parallèle, c'est à dire permettant la mise à jour des états de chaque cellule d'une manière la plus indépendante possible. D'un point de vue plus théorique, les critères de choix vont être la possibilité de cerner l'ensemble

des vecteurs de réponses qui pourront effectivement être atteints et surtout la possibilité de pouvoir garantir que toute évolution aboutit effectivement à un vecteur de réponse.

Passons maintenant en revue les différents choix usuels en utilisant les notations de la seconde définition. Lorsque la fonction de seuillage f est continue, on peut choisir une dynamique à temps continu comme suit[2] :

$$\dot{x} = F(\Omega(x) + e) - x \quad ,$$

ou à temps discret avec un pas de temps τ :

$$x^{t+1} = (1 - \tau) \, . \, x^t + \tau \, . \, F(\Omega(x^t) + e) \quad ,$$

ou encore, en prenant τ égal à 1 :

$$x^{t+1} = F(\Omega(x^t) + e) \quad ,$$

cette dernière dynamique étant celle utilisée lorsque la fonction de seuillage est discrète. On vérifiera que pour tous ces mécanismes les seuls vecteurs de réponses (ou d'états) x qui sont stables sont ceux vérifiant l'équation (3).

Ces deux dernières dynamiques à temps discret, ou itératives, sont qualifiées de parallèles car à chaque itération toutes les composantes du vecteur d'état du réseau évoluent simultanément. Dans le cadre d'une implémentation sur une machine séquentielle, ce type d'itération impose la recopie du vecteur d'état avant sa mise à jour. Afin de réduire la place mémoire utilisée, on peut envisager une mise à jour séquentielle des états. Au cours d'une telle itération, une seule composante x_i de l'état $x = (x_1, \ldots, x_n)$ évolue à la fois et ce, à tour de rôle et dans un ordre établi. Le comportement dynamique de cette itération séquentielle pouvant être très différent de celui d'une itération parallèle, on peut essayer le compromis qui consiste à faire évoluer l'état du réseau par blocs de cellules.

En pratique, il faudra s'assurer que l'état du système reste borné. Dans le cas d'une itération avec un pas de temps τ inférieur à 1, si l'image de $\mathbb{R}$ par la fonction de seuillage f est un intervalle borné Q, alors toute trajectoire, issue de l'hypercube Q^n, y reste. Ceci découle du fait que l'hypercube Q^n est convexe et que l'état du système à l'instant $t + 1$ se trouve sur le segment entre l'état précédent x^t et le point $F(\Omega(x^t))$ de l'hypercube. Dans le cas d'une dynamique continue, si l'image de $\mathbb{R}$ par la fonction de seuillage f est un intervalle borné et ouvert Q, alors toute trajectoire, issue de l'hypercube Q^n, y reste. Il suffit de vérifier qu'en tout point de la surface de l'hypercube Q^n la variation d'état $\dot{x}$ est dirigée vers l'intérieur du cube.

---

[2]Les notations sont les suivantes. Pour une dynamique à temps continu on désigne par $\dot{x}$ la dérivée par rapport au temps de l'état x. Dans le cas discret, on désigne par x^t l'état du système à l'instant t.

Le fait que l'espace d'états d'un neurone soit compris dans un intervalle borné de $\mathbb{R}$ entraîne donc que toute trajectoire du système reste confinée dans un compact de $\mathbb{R}^n$ et, par suite, que toute trajectoire admet un attracteur. Ceci rend possible une interprétation du système fondée sur ses attracteurs. On remarquera aussi que l'horizon de ces systèmes n'est pas borné: toute trajectoire peut être prolongée sur un intervalle de temps aussi grand que l'on veut.

On retiendra là aussi la diversité des mécanismes utilisables *a priori*. Le choix de l'un ou l'autre de ces mécanismes sera dicté par les moyens dont on dispose pour les mettre en œuvre.

2. Principes d'utilisation

Comment utiliser le formalisme des réseaux de neurones dans un cadre informatique? Nous disposons d'un modèle de systèmes dynamiques que nous pouvons simuler sur un ordinateur ou approcher avec une réalisation électronique. L'idée est de considérer l'aboutissement d'une trajectoire comme une réponse à un calcul et il nous faut donc contrôler le choix d'une trajectoire en fonction d'une instance du problème que l'on veut résoudre. Par souci de simplicité, on envisage uniquement des trajectoires qui aboutissent à un point d'équilibre. L'espace des attracteurs auxquels il faut donner une interprétation est donc un sous-ensemble de l'espace d'états Q du réseau. Pour simplifier encore plus la démarche d'interprétation, supposons que le choix d'un état initial soit le seul moyen dont on dispose pour choisir une trajectoire. Il suffit alors de disposer de deux fonctions de codage, $\mathcal{D}$ et $\mathcal{R}$, sur l'espace d'états Q du système et respectivement à valeurs dans l'espace D des entrées et celui R des réponses. L'association réalisée par le réseau est alors constituée de tous les couples (d, r) de l'ensemble $D \times R$ tels qu'il existe deux états x^o et x^∞ avec:

$$\begin{aligned} \mathcal{D}(x^o) &= d \quad , \\ \mathcal{R}(x^\infty) &= r \end{aligned}$$

et tels que x^∞ soit le point d'équilibre atteint en partant du point x^o.

Pour construire ces deux fonctions de codage, $\mathcal{D}$ et $\mathcal{R}$, l'approche usuelle est de choisir, parmi l'ensemble I des neurones du réseau, deux sous ensembles, E et S, servant respectivement de support à la fonction codant les entrées et celle codant les réponses. On entend par le terme de *support* le fait suivant. Étant donnés deux états $x = (x_i)_{i \in I}$ et $y = (y_i)_{i \in I}$ du réseau, alors:

$$\mathcal{D}(x) = \mathcal{D}(y) \quad \text{si } x_i = y_i \text{ pour tout neurone } i \text{ de } E$$

et, de même,

$$\mathcal{R}(x) = \mathcal{R}(y) \quad \text{si } x_i = y_i \text{ pour tout neurone } i \text{ de } S.$$

C'est-à-dire que deux états ne différant pas sur les cellules de E (respectivement sur celles de S) sont interprétés de manière identique en termes de données de calcul (respectivement en termes de réponses). Les cellules de E sont alors naturellement appelées cellules d'entrée et celles de S cellules de sortie. On qualifie usuellement de cellules cachées les neurones qui n'appartiennent ni à E ni à S et dont l'état n'intervient pas dans le schéma d'interprétation. Présentons maintenant les variantes les plus fréquentes.

2.1. Réseaux multi-couches

La première utilisation qui a été faite des réseaux de neurones est le calcul d'une fonction. Les cellules d'entrée sont alors destinées à recueillir l'information qui est transformée par les cellules cachées et véhiculée jusqu'aux cellules de sortie. Ce mouvement d'information est conçu sans boucle de rétro-action et le graphe permettant à une cellule de recevoir ses données de calcul se résume donc à un arbre dont les feuilles sont des cellules d'entrée. Si l'on classe les différents neurones en fonction du nombre maximum de connexions que doit franchir l'information de départ pour leur parvenir, on obtient une représentation en couches du réseau. On parle alors de réseaux multi-couches et la figure I.2 en donne un exemple.

Une autre caractérisation de ces réseaux est que la matrice de connexion peut être mise sous une forme triangulaire inférieure et avec une diagonale nulle. Il suffit

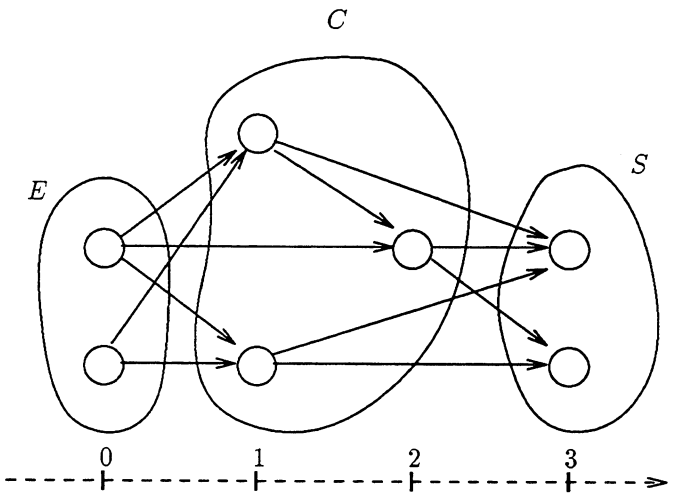

FIG. I.2 - . Exemple de réseau multi-couches, comportant deux couches intermédiaires entre la couche codant l'entrée et celle codant la réponse.

pour cela de numéroter les cellules dans un ordre qui respecte l'ordre des couches. La propriété intéressante de ces réseaux est que le problème de la recherche d'un point d'équilibre ne se pose pas. Quel que soit l'état initial du réseau sur les cellules autres que celles d'entrée, il existe un et un seul point d'équilibre qui peut se calculer de couche en couche et en partant de l'état des cellules d'entrée.

Quelle est la classe des fonctions que l'on peut réaliser à l'aide d'une telle organisation en couches des neurones? Si l'on prend des neurones ayant pour espace d'états l'ensemble discret $\{0,1\}$, toute fonction booléenne peut être calculée par un réseau comportant au plus une couche de cellules cachées. Toute fonction booléenne peut, en effet, être mise sous la forme d'une conjonction de disjonctions, deux calculs qui peuvent être faits à l'aide d'un seul neurone. Le rôle des cellules cachées est donc de calculer les différentes disjonctions et les cellules de sortie n'ont plus qu'un calcul de conjonction à faire. On a un résultat similaire dans le cas de neurones ayant un espace d'états continu. Toute fonction continue, d'un compact de $\mathbb{R}^n$ dans $\mathbb{R}$, peut être approchée à la précision souhaitée par un réseau comportant une couche intermédiaire entre celle codant l'entrée et celle codant la réponse [27].

Si ces réseaux permettent de réaliser une grande classe de fonctions, leur utilisation pose néanmoins des questions en pratique. Ainsi, si l'on peut réaliser n'importe quelle fonction booléenne exprimée sous forme d'une conjonction de disjonctions, on peut se demander quel est l'intérêt d'utiliser un réseau de neurones pour le calcul de cette fonction alors que l'on dispose déjà de nombreuses solutions électroniques. Le seul intérêt probant des réseaux de neurones est qu'ils tolèrent à priori plus de défauts. On pourrait ainsi envisager de les utiliser pour réaliser de gros circuits sur une seule tranche de silicium. En effet, on ne maîtrise actuellement pas suffisamment les techniques de photo-gravure de ces tranches pour garantir une absence de défauts, ce qui limite grandement la taille des circuits réalisables. Un autre intérêt de ce type de réseaux est de pouvoir moduler la fonction calculée en jouant sur la valeur des poids de connexion. On peut alors se demander s'il est possible de faire coïncider la fonction qu'ils calculent avec une fonction dont on ne connaît pas d'expression algébrique et que l'on ne sait donc pas calculer. L'idée est de procéder par retouches successives et on qualifiera d'algorithme d'apprentissage toute méthode permettant d'effectuer ces retouches à l'aide d'exemples. Le chapitre III a pour objet de détailler cette approche.

Les réseaux multi-couches sont ceux les plus utilisés actuellement dans un contexte technologique. Le but est de pallier à un manque de connaissance du problème traité, à l'aide d'algorithmes d'apprentissage. L'exemple le plus fréquent est celui de la reconnaissance de l'écriture où le problème est de faire correspondre un symbole à une image censée représenter une lettre de l'alphabet. On peut alors se demander, avec raison, si vouloir représenter ce travail de reconnaissance par le calcul d'une fonction (à telle image correspond tel symbole) est une approche

correcte. En effet, à un même signe peuvent correspondre différentes interprétations suivant le contexte où ce signe est rencontré. De plus, deux personnes différentes peuvent même ne pas être d'accord sur l'interprétation à donner à un signe. Il y a donc plus qu'une simple méconnaissance de l'association à réaliser et l'approche fonctionnelle semble être trop catégorique et définitive.

2.2. Réseaux récurrents

Concevoir la réponse à un problème comme fonction des données de ce problème étant trop restrictif pour les applications envisagées pour les réseaux de neurones, on peut se placer dans un cadre plus large permettant de rendre compte de réponses multiples pour une seule instance du problème. Ces différentes réponses possibles pour le système sont alors interprétées comme les différentes réponses envisageables. Un tri ultérieur devra donc être fait pour tenir compte du contexte.

Pour retrouver cette possibilité de réponses multiples on rajoute au schéma des réseaux multi-couches des connexions de rétro-action. On choisit usuellement de n'utiliser les cellules d'entrée que pour coder l'instance du problème qui nous intéresse et donc sans modifier leur état durant la phase de calcul. On ne fait donc pas intervenir de connexions afférentes à ces cellules. Un exemple d'un tel réseau récurrent est donné par la figure I.3. Il n'y alors plus unicité du point d'équilibre atteint à partir d'une configuration donnée de l'état des cellules d'entrée du réseau : les états initiaux des cellules cachées et de celles de sortie interviennent aussi sur le point d'équilibre atteint. Ce point d'équilibre ne peut plus être calculé directement, comme cela est le cas pour les réseaux multi-couches, et il devient nécessaire d'utiliser le caractère dynamique du réseau pour obtenir une réponse. Pour obtenir différentes réponses il est nécessaire d'essayer divers choix de l'état initial du réseau.

Signalons une autre utilisation possible pour les réseaux récurrents. Au lieu de partir d'un état quelconque pour les cellules cachées et de sortie, choisissons un état particulier $x^o_{C,S}$, qui sera repris pour chaque calcul en étant complété par l'état x^o_E des cellules d'entrée. On retrouve alors un schéma fonctionnel : à une entrée correspond une et une seule réponse. L'intérêt, par rapport aux réseaux multi-couches, est qu'ainsi on peut obtenir une fonction comportant des points de discontinuité correspondant à des changements d'attracteur. La figure I.4 montre comment ces discontinuités apparaissent.

Plus formellement, si n est le nombre de cellules cachées ou de sortie, le lien entre l'état d'équilibre $x^\infty_{C,S}$ obtenu pour ces cellules et l'état x^o_E des cellules d'entrée s'exprime sous la forme

$$G(x^\infty_{C,S}, x^o_E) = 0 \quad ,$$

où G est une fonction différentiable de $\mathbb{R}^n$ dans $\mathbb{R}^n$. D'après le théorème des

fonctions implicites, l'état d'équilibre $x^{\infty}_{C,S}$ est presque partout lié à l'état initial x°_{E} par une fonction différentiable. Des points de discontinuité peuvent exister si la différentielle $G'_{C,S}(x^{\infty}_{C,S}, x^{\circ}_{E})$ est non inversible ($G'_{C,S}(x^{\infty}_{C,S}, x^{\circ}_{E})$ désignant la différentielle au point $x^{\infty}_{C,S}$ de l'application qui à un état $x_{C,S}$ associe $G(x_{C,S}, x^{\circ}_{E})$).

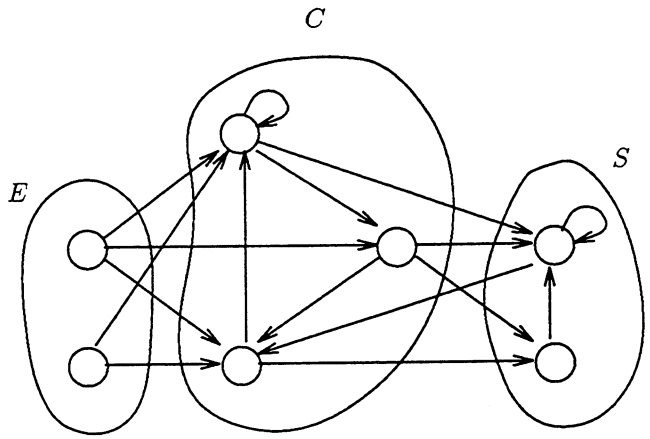

FIG. I.3 - . Un exemple de réseau récurrent.

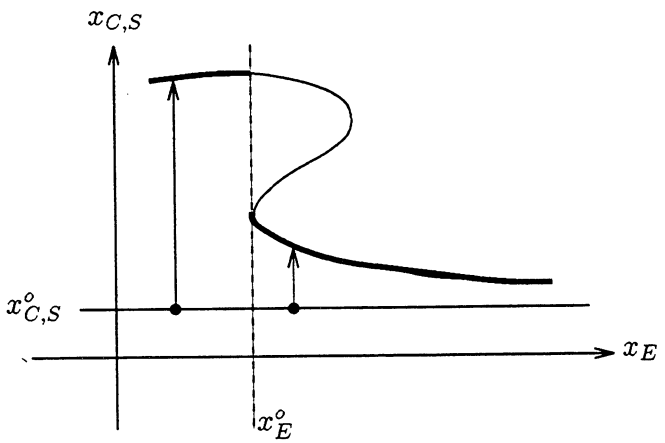

FIG. I.4 - . Apparition de discontinuité lors de l'utilisation de réseau récurrents. En gras est représenté le lieu des points d'équilibre atteignables en partant d'un état prenant la valeur fixée de $x^{\circ}_{C,S}$ sur les cellules cachées et les cellules de sortie. La réponse du réseau est discontinue au point associé à l'entrée x°_{E}.

Savoir si il y a effectivement une discontinuité et pour quelle entrée elle a lieu, est une question plus difficile. Prenons un exemple avec un seul neurone.

EXEMPLE. Considérons un neurone connecté sur lui même par un poids ω et recevant une entrée résumée par un réel e. Prenons comme fonction de seuillage f la fonction tangente hyperbolique th vérifiant l'équation différentielle

$$f' = (1 - f)(1 + f) \quad .$$

Comment varie l'état d'équilibre x de ce neurone en fonction de l'entrée si l'on part d'un état initial x^0 ? Le lieu des points d'équilibre dans l'espace produit (état $\times$ entrée) est donnée par l'équation implicite :

$$G(x, e) = f(\omega x + e) - x = 0 \quad .$$

L'entrée e permettant d'obtenir un point d'équilibre x^∞ est donc unique et est donnée par

$$e = f^{-1}(x^\infty) - \omega x^\infty \quad ,$$

mais en partant de l'état initial x^o seule une partie de ces points d'équilibre peut effectivement être atteinte. La figure I.5 schématise la situation.

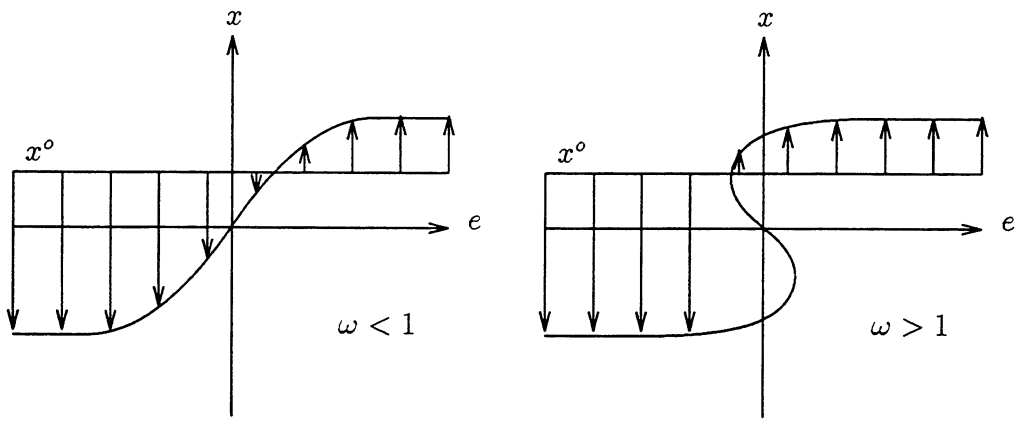

FIG. I.5 - . Lorsque ω est strictement inférieur à un 1, il n'y a qu'un seul point d'équilibre possible par entrée et la réponse du réseau varie de manière continue en fonction de l'entrée. Lorsque ω est strictement supérieur à 1 il y a plusieurs équilibres possibles et la réponse du réseau dépend du bassin d'attraction dans lequel se trouve l'état initial x^o. Dans le cas de l'exemple, autour d'une certaine valeur de l'entrée, celle rendant stable l'état initial, ce point initial change de bassin d'attraction.

La dérivée partielle $G'_x(x^\infty, e)$, en point d'équilibre x^∞ obtenu avec l'entrée e, de l'application qui à un état x associe $G(x, e)$ vérifie

$$G'_x(x^\infty, e) = \omega(1 - x^\infty)(1 + x^\infty) - 1 \quad ,$$
$$\text{avec} \qquad x^\infty = f(\omega x^\infty + e) \quad .$$

Si ω est strictement inférieur à 1, la dérivée $G'_x(x^\infty, e)$ ne peut être nulle et le point d'équilibre x^∞ s'exprime en fonction de l'entrée e à l'aide d'une fonction définie et dérivable sur $\mathbb{R}$. Si ω est supérieur à 1, on a un point de discontinuité entre les deux points :

$$e_1 = f^{-1}(\sqrt{\frac{1}{\omega} - 1}) - \omega\sqrt{\frac{1}{\omega} - 1} \quad ,$$
$$e_2 = f^{-1}(-\sqrt{\frac{1}{\omega} - 1}) + \omega\sqrt{\frac{1}{\omega} - 1} \quad .$$

Plus précisément, le lieu e_d de cette discontinuité est égal à

$$e_d = e_1 \qquad \text{si} \qquad \sqrt{\frac{1}{\omega} - 1} \leq x^o$$
$$e_d = f^{-1}(x^o) - \omega x^o \qquad \text{si} \qquad -\sqrt{\frac{1}{\omega} - 1} < x^o < \sqrt{\frac{1}{\omega} - 1}$$
$$e_d = e_2 \qquad \text{si} \qquad x^o \leq -\sqrt{\frac{1}{\omega} - 1}$$

Dans le premier et le dernier cas, un point d'équilibre apparaît ou disparaît le long de la trajectoire issue de l'état initial lorsqu'on fait varier l'entrée. Dans le cas intermédiaire, la discontinuité est due à un changement de bassin d'attraction.

2.3. Réseaux associatifs

Une autre application des réseaux de neurones est celle des mémoires associatives. La notion de mémoire est usuellement conçue en informatique comme un ensemble de cases dans lesquelles on peut déposer différentes informations que l'on peut retrouver ultérieurement en indiquant la case où elles ont été déposées. Ce sont des mémoires accessibles par une adresse. Les mémoires associatives, elles, sont accessibles par leur contenu, et leur fonction est de compléter une information partielle. Prenons comme exemple le cas d'un ensemble de mémoires associatives mémorisant chacune les différentes caractéristiques d'un objet. Pour obtenir toutes les informations relatives à un objet dont on ne connaît qu'une partie des caractéristiques, il suffit alors de donner à chacune de ces mémoires cet élément d'information et celles dont le contenu vérifie ces spécifications proposent alors la totalité des caractéristiques de leur objet. Il ne nous reste alors plus qu'à faire un choix.

Le principe suivi pour réaliser une mémoire associative à l'aide d'un réseau de neurones est de considérer un point d'équilibre attractif du réseau comme un résumé des informations relatives à un objet. Un réseau pourra donc mémoriser autant d'objets qu'il a de points d'équilibre attractifs. L'information relative à un objet est donc codée par l'état du réseau et l'on prend comme support de cette fonction code un ensemble S de neurones. Ce même ensemble S sert alors aussi comme support E pour coder l'entrée de la mémoire, c'est à dire l'information incomplète. Le bassin d'attraction d'un point d'équilibre, ensemble de tous les états initiaux amenant le réseau à ce point d'équilibre, code donc l'ensemble des spécifications incomplètes ou erronées, qui seront complétées ou corrigées en l'information codée par le point d'équilibre en question. Afin d'avoir un comportement le plus riche possible on peut aussi prendre un ensemble C de neurones intermédiaires. L'état initial de ces neurones cachés établit alors le choix entre un objet parmi tous ceux pouvant répondre à la question. La figure I.6 donne un exemple de réseau destiné à un tel usage.

3. Problèmes techniques de convergence

Lors de notre définition des réseaux de neurones, nous avons imposé uniquement le lieu des points d'équilibre. Mais nous venons de voir qu'au cours de l'utilisation d'un tel système nous sommes intéressés par les attracteurs des différentes trajectoires. Plus précisément nous sommes intéressés par le lien qui existe entre un point d'équilibre, codant une réponse, et les conditions permettant d'obtenir ce point d'équilibre, codant une instance de calcul. Se posent alors les questions

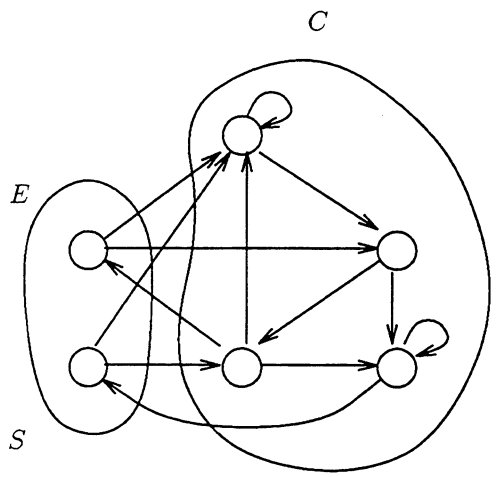

FIG. I.6 - . Un exemple de réseau associatif.

suivantes: quels sont les points d'équilibre attractifs? Sont-ils les seuls attracteurs rencontrés? Vis à vis de la réponse à ces questions, comment intervient le choix d'une dynamique? Quel rôle jouent les coefficients synaptiques du réseau? L'objet des deux sections suivantes est de répondre à ces questions. La première aborde le problème local de l'attractivité d'un point d'équilibre. La seconde décrit les conditions d'ensemble garantissant que toute évolution aboutit effectivement à un point d'équilibre.

3.1. Points d'équilibre attractifs

L'étude des systèmes dynamiques déterministes autour d'un point d'équilibre s'appuie sur la notion de système linéarisé. L'idée directrice consiste à approcher le système $\dot{x} = V(x)$ autour d'un point x^o, solution de l'équation $V(x) = 0$, par le système linéaire $\dot{y} = V'_x(x^o)y$ autour de l'origine (où $V'_x(x^o)$ désigne la différentielle de la fonction V au point x^o). La difficulté est de cerner les propriétés qui seront conservées par cette transformation. Dans le cadre des réseaux de neurones, nous sommes intéressés par le caractère attractif des points d'équilibre. Un tel point étant donné, existe-t-il un voisinage de ce point à partir duquel toute évolution converge vers ce point d'équilibre? Autrement dit, pour atteindre ce point, suffit-il d'en être suffisamment proche initialement? Ou faut-il partir d'un lieu bien précis et donc, au cours d'une simulation du système, veiller à ne pas s'écarter de la trajectoire si on veut effectivement l'atteindre? Ou finalement, ce point est-t-il répulsif et donc hors d'atteinte?

L'étude du système linéarisé nous donne des conditions suffisantes pour pouvoir répondre à ces questions. Le critère de décision va s'appuyer sur les valeurs propres de la différentielle $V'_x(x^o)$. Si aucune de ces valeurs propres n'est purement imaginaire ou nulle, les deux systèmes sont similaires d'un point de vue topologique autour du point d'équilibre considéré et celui-ci sera attractif si l'origine est un point attractif du système linéarisé. De plus le caractère attractif de l'origine pour le système linéaire est lui aussi caractérisé par ces mêmes valeurs propres. L'origine sera attractive si toutes les valeurs propres du linéarisé ont une partie réelle strictement négative. Dans le cadre d'une itération à temps discret la démarche est identique et les valeurs propres associées à l'approximation linéaire devront être de module différent de 1 pour garantir la similitude topologique des deux systèmes autour du point étudié. L'origine sera attractive si toutes les valeurs propres du linéarisé ont un module strictement inférieur à 1.

Revenons au cas des réseaux de neurones et commençons par le cas d'une dynamique à temps continu:

$$\dot{x} = F \circ \Omega(x) - x \quad .$$

Un point d'équilibre x^o sera alors attractif sur un voisinage si toutes les valeurs

propres de la différentielle en x^o de la fonction G, qui à x associe $F \circ \Omega(x) - x$, ont une partie réelle strictement négative.

En notant $F'(\Omega(x^o))$ la différentielle de la fonction F au point $\Omega(x^o)$ et $\mathcal{L}(\Omega)$ la partie linéaire de l'application affine Ω, on peut donc énoncer :

PROPRIÉTÉ. *Un point d'équilibre x^o du système dynamique*

$$\dot{x} = F \circ \Omega(x) - x$$

sera attractif sur un voisinage si toutes les valeurs propres de l'application linéaire

$$G'_x(x^o) = F'(\Omega(x^o)) \circ \mathcal{L}(\Omega) - I$$

ont une partie réelle négative.

Cette condition va être plus restrictive dans le cadre d'une itération parallèle,

$$x^{t+1} = (1 - \tau) \, . \, x^t + \tau \, . \, F \circ \Omega(x^t) \quad ,$$

et ce d'autant plus que le pas de temps τ sera grand. En effet dans ce cas, un point d'équilibre x^o sera attractif sur un voisinage si toutes les valeurs propres de la différentielle en x^o de la fonction H, qui à x associe $(1 - \tau) \, . \, x + \tau \, . \, F \circ \Omega(x)$, ont un module strictement inférieur à 1. Avec les mêmes notations que dans le cas continu, cette différentielle est égale à :

$$H'_x(x^o) = I + \tau \, . \, G'_x(x^o) \quad .$$

En remarquant, alors, que pour tout nombre complexe λ, $1 + \tau\lambda$ est une valeur propre de la différentielle $H'_x(x^o)$ si et seulement si λ est une valeur propre de la différentielle $G'_x(x^o)$ on peut donc conclure :

PROPRIÉTÉ. *Un point d'équilibre x^o du système dynamique itératif*

$$x^{t+1} = (1 - \tau) \, . \, x^t + \tau \, . \, F \circ \Omega(x^t)$$

sera attractif sur un voisinage si toutes les valeurs propres de l'application linéaire

$$G'_x(x^o) = F'(\Omega(x^o)) \circ \mathcal{L}(\Omega) - I$$

sont strictement comprises dans le disque de rayon $1/\tau$ et de centre $-1/\tau$. On remarquera que ce disque est compris dans le demi-plan qui intervient dans le cadre d'une dynamique à temps continu.

L'utilisation d'une dynamique à temps discret a donc pour conséquence principale de faire perdre des points d'équilibre qui auraient été attractifs avec une

dynamique continue. En pratique il faudra donc utiliser un pas de temps τ suffisamment petit pour que toutes les valeurs propres, associées aux points d'équilibre que l'on désire capter, se trouvent dans le disque de convergence de rayon $1/\tau$ et de centre $-1/\tau$.

3.2. Condition pour n'avoir que des points fixes

Quelles sont les conditions garantissant que toute évolution du réseau se stabilise effectivement ? Pour répondre à cette question il est nécessaire d'avoir une vue d'ensemble du portrait de phase du système. L'approche classique s'appuie sur la notion physique d'énergie, quantité associée à l'état d'un système et qui décroît au cours de l'évolution de celui-ci. Exhiber une telle fonction d'état permet de garantir que toute trajectoire du système atteint un point d'équilibre. De plus les points d'équilibre attractifs correspondent alors aux minima locaux de cette fonction que l'on appellera fonction d'énergie.

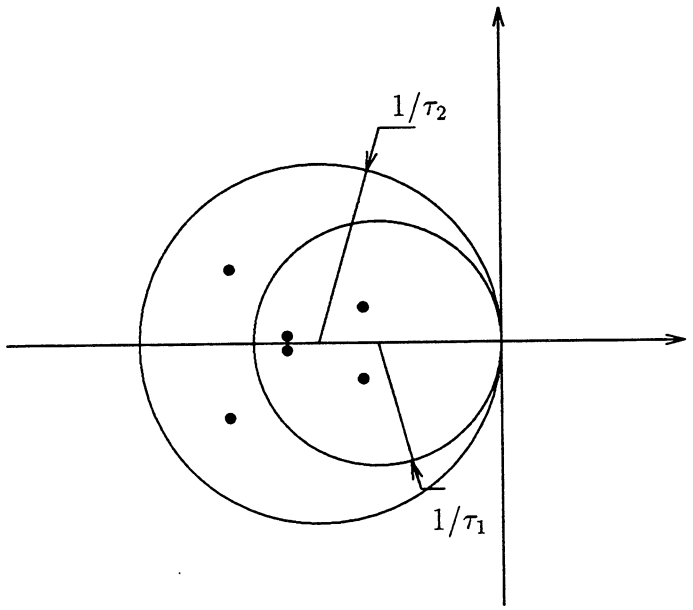

FIG. I.7 - . Disques de convergence pour deux pas de discrétisation τ_1 et τ_2 et valeurs propres associées à un point d'équilibre. Le premier disque laissant échapper des valeurs propres, il est nécessaire d'utiliser le second pas de discrétisation si l'on veut que ce point d'équilibre soit attractif comme c'est le cas pour une dynamique continue puisque toutes les valeurs propres associées à ce point ont une partie réelle strictement négative.

J. HOPFIELD est le premier a avoir introduit ce principe dans le cadre des réseaux de neurones discrets [25] puis continus [26]. Commençons par le cas discret.

Considérons un réseau de neurones pouvant prendre comme état -1 ou 1 et dont la fonction de seuillage est la fonction « sign ». Plaçons-nous dans le cadre d'une itération séquentielle: les états des neurones évoluent un à un, dans un ordre établi et suivant l'équation:

$$x_i^{t+1} = \text{sign}(\sum_j \omega_{ij} \, . \, x_j^t + \omega_i) \quad .$$

On a alors le résultat suivant:

PROPRIÉTÉ (HOPFIELD). *Si la matrice de connexion (ω_{ij}) du réseau est symétrique et à diagonale nulle, alors:*

i) *La quantité E, appelée* énergie *et définie en tout point x de l'espace d'états $\{-1,1\}^n$ du réseau par:*

$$E(x) = -\frac{1}{4} \sum_{ij} x_i \omega_{ij} x_j - \frac{1}{2} \sum_i \omega_i x_i \quad ,$$

est décroissante le long de toute trajectoire du système.

ii) *Toute trajectoire du système atteint un point d'équilibre en un temps fini et les points d'équilibre sont des minima locaux de l'énergie E (la localité étant relative à la relation de voisinage sur l'hypercube $\{-1,1\}^n$: deux points de l'espace d'états sont voisins s'ils ne diffèrent qu'en une seule coordonnée).*

iii) *La différence d'énergie entre deux points x et y qui ne diffèrent que selon leur $i^{ème}$ coordonnée vérifie:*

$$E(x) - E(y) = -x_i \, . (\sum_j \omega_{ij} y_j + \omega_i) \quad .$$

Une fois établie la relation entre l'énergie de deux points ne différant qu'en une et une seule coordonnée, la décroissance de l'énergie le long de toute trajectoire s'établit en remarquant que si le réseau passe d'un point x^t à un point x^{t+1} par un changement d'état de la $i^{\text{ème}}$ cellule on a:

$$E(x^{t+1}) - E(x^t) = -|\sum_j \omega_{ij} \, . \, x_j{}^t + \omega_i| \quad .$$

Les trajectoires étant confinées dans un espace d'états fini, la décroissance de l'énergie entraîne que toute trajectoire atteint un minimum local de l'énergie E en un temps fini. □

Dans le cas d'un espace d'états continu, le résultat de J. HOPFIELD est le suivant :

PROPRIÉTÉ (HOPFIELD). *Si la matrice de connexion $\omega = (\omega_{ij})$ du réseau est symétrique et si la fonction de seuillage f est monotone, alors :*

i) *La quantité E, appelée* énergie *et définie en tout point x de l'espace d'états $\mathbb{R}^n$ du réseau par :*

$$E(x) = -\frac{1}{2} \sum_{ij} x_i \omega_{ij} x_j - \sum_i (\omega_i x_i - g(x_i)) \quad ,$$

(où g est une primitive de la réciproque f^{-1} de f) est décroissante le long de toute trajectoire du système dynamique :

$$\dot{x}_i = f(\sum_j \omega_{ij} x_j + \omega_i) - x_i \qquad \textit{pour tout } i.$$

Si, de plus, la fonction g est bornée alors l'énergie E est bornée et :

ii) *Il n'existe pas d'attracteur autre que des points fixes.*

iii) *Les minima locaux de l'énergie E correspondent aux points d'équilibre attractifs.*

En effet, le gradient en un point x du premier terme de E,

$$-\frac{1}{2} \sum_{ij} x_i \omega_{ij} x_j \quad ,$$

est égal au vecteur $-\omega \,.\, x$, en tenant compte de l'hypothèse de symétrie de ω. Par définition de g, le gradient en un point x du second terme de E,

$$\sum_i (-\omega_i x_i + g(x_i)) \quad ,$$

est le vecteur de coordonnées $-\omega_i + f^{-1}(x_i)$. Si l'on compare maintenant la $i^{\text{ème}}$ coordonnée du gradient de E,

$$-\sum_j w_{ij} x_j - \omega_i + f^{-1}(x_i) \quad ,$$

avec la $i^{\text{ème}}$ coordonnée du vecteur vitesse,

$$f(\sum_j w_{ij} x_j) - x_i \quad ,$$

on remarque en tenant compte de la monotonie de f que ces deux quantités sont de signe opposé ou toutes deux nulles. Pour tout point x où la variation d'état $\dot{x}$ est non nulle, on a alors :

$$\mathrm{grad}_x(E) \,.\, \dot{x} < 0$$

et l'énergie $E(x)$ est donc strictement décroissante le long de toute trajectoire non stationnaire du réseau. □

4. Réseaux asynchrones

Introduisons les réseaux asynchrones par un exemple. Considérons un réseau de n neurones $\{0, \ldots, n-1\}$, prenant leur état dans l'ensemble $\{-1, 1\}$ et dont la fonction de seuillage est la fonction « sign ». On souhaite qu'à l'équilibre du réseau tous les neurones soient dans le même état. Pour cela, il suffit de relier chaque neurone i au neurone $i+1$ modulo n par une connexion pondérée par la quantité 1. Une dynamique séquentielle (évolution de l'état des neurones à tour de rôle et dans l'ordre $0, \ldots, n-1$) permet effectivement d'atteindre l'un des deux points recherchés. Mais cela n'est plus le cas si la dynamique utilisée est une dynamique à temps discret et parallèle : les états des deux cellules basculent simultanément vers celui calculé à partir des signaux reçus. En effet, si l'on part d'un état initial où les cellules n'ont pas toutes le même état, celles-ci vont faire basculer leur état simultanément et aucun équilibre ne peut être atteint. Il semble donc que le fait de vouloir un parallélisme maximum gêne le réseau dans la recherche d'un point d'équilibre. Or dans le cas de l'exemple, il suffit que de temps en temps certains neurones ne changent pas d'état pour obtenir l'un des deux points d'équilibre souhaités. Les choix de ces moments et lieux d'inactivités menant à une réponse correcte étant multiples, l'idée est de ne pas les faire *a priori* mais de manière non déterministe en cours de calcul. Nous allons voir que ces dynamiques non déterministes, ou asynchrones, vont nous permettre d'obtenir un parallélisme important sans gêner la bonne terminaison du calcul.

En restant dans le cadre des neurones à espace d'états discret, il y a deux façons de se représenter un comportement asynchrone d'un réseau. On peut considérer que le temps est discret et qu'à chaque instant un neurone a le choix entre l'attente qui consiste à garder son état et le calcul qui revient à faire basculer son état dans le sens indiqué par la fonction de seuillage. Le point qui deviendra essentiel lorsqu'on s'intéressera à la convergence d'une telle dynamique, est qu'à chaque instant et pour chaque neurone les deux alternatives, attente ou calcul, soient possibles. Par ailleurs, on peut aussi considérer que le temps est continu mais que l'instant où un neurone change d'état n'est pas synchronisé avec les éventuels autres changements d'état du réseau. Dans ce cas un neurone ne doit pas rester indéfiniment dans un état d'instabilité (*i.e.* tel que l'état du neurone soit différent de celui induit par la fonction de seuillage). Ces deux visions sont

en fait similaires : une évolution exprimée avec les premiers termes peut être rendue avec les seconds et *vice versa*. La première forme est plus manipulable alors que la seconde est plus proche d'une réalisation matérielle : en pratique avec un circuit électronique il suffira de ne pas synchroniser les divers changements d'état du système pour obtenir un tel comportement.

Une telle dynamique asynchrone peut être quantifiée en donnant à chaque neurone une probabilité de calculer son état ou non à un top d'horloge quelconque. Avec une telle quantification le réseau est assimilé à un processus de MARKOV ce qui va nous permettre d'étudier son comportement dynamique. Le point important est que les propriétés de convergence vers un état d'équilibre ne vont en fait dépendre que des connexions du réseau et nullement de la quantification probabiliste de l'asynchronisme, du moins tant que l'on ne s'intéresse pas à la durée d'un calcul.

4.1. Réseaux asynchrones discrets

Le cadre de cette section est celui de neurones discrets pouvant choisir leur état parmi deux possibles, 0 et 1, et évoluant dans un temps discret, une succession de tops d'horloge. Un neurone i reçoit un signal ${u_i}^t$ à l'instant t, ce signal étant une combinaison linéaire de l'état des neurones auxquels il est connecté, et fait évoluer son état ${x_i}^t$ suivant le schéma :

$$ {x_i}^{t+1} = \begin{cases} \text{soit} & f({u_i}^t) \\ \text{soit} & {x_i}^t \end{cases} , $$

où f désigne la fonction de seuillage définie par

$$ f(u) = \begin{cases} 1 & \text{si } u > 0 \\ 0 & \text{sinon} \end{cases} . $$

Un point $x = (x_1, \ldots, x_n)$ d'un réseau de n neurones est un point d'équilibre si c'est un état stable quels que soient les choix faits par les différentes cellules. Si la matrice de connexion d'un tel réseau est la matrice $(\omega)_{ij}$ complétée avec un ensemble de seuils ω_i, les seuls points d'équilibre du réseau sont donc les points vérifiant pour tout neurone i

$$ x_i = f(\sum_j \omega_{ij} x_j + \omega_i) $$

On retrouve donc la propriété qui nous avait servi à définir les réseaux de neurones déterministes.

A partir d'un point $x = (x_1, \ldots, x_n)$ le système peut passer à tout point $y =$

$(y_1, \ldots, y_n)$ tel que :

$$y_i = x_i \text{ si } x_i = f(u_i) \text{ avec } u_i = \sum_j \omega_{ij} x_j + \omega_i \quad .$$

Est-ce que toutes les trajectoires aboutissent à un point d'équilibre ? Manifestement non, car il suffit de reprendre l'exemple de réseau donné en début de section et de remarquer que l'évolution parallèle est une évolution possible et ne menant à aucun équilibre. Il faut nous poser la question d'une manière différente. Étant donné un réseau, est-ce que presque toutes ses trajectoires aboutissent à un point d'équilibre ? Plus précisément, il faut munir l'ensemble de toutes les suites de choix possibles d'une mesure et la question devient la suivante : l'ensemble des trajectoires issues d'un état initial donné et n'aboutissant pas à un point d'équilibre est-il de mesure nulle ? La mesure de probabilité que nous prendrons est la suivante : à chaque instant chaque neurone calcule effectivement son état avec une probabilité p et reste dans son état sans faire de calcul avec une probabilité de $1-p$. Étant donné un état $x = (x_1, \ldots, x_n)$ du réseau la probabilité qu'à l'instant suivant l'état du réseau soit $y = (y_1, \ldots, y_n)$ est alors :

$$P(x, y) = p^{|C|} . (1-p)^{|R|} . 0^{|I|} \quad ,$$

où $|C|$ (respectivement $|R|$) désigne le nombre de neurones devant effectuer leur calcul pour que le système passe de l'état x à l'état y (respectivement, ne devant pas effectuer de calcul) :

C est l'ensemble des neurones i tq $y_i = f(u_i)$ et $y_i \neq x_i$,

R est l'ensemble des neurones i tq $y_i \neq f(u_i)$ et $y_i = x_i$.

Enfin, I est l'ensemble des neurones i dont les états x_i et y_i sont tels que la transition de x à y ne soit pas possible :

I est l'ensemble des neurones i tq $y_i \neq f_i(x)$ et $y_i \neq x_i$.

Ainsi quantifié le système est un processus de MARKOV sur l'ensemble d'états $\{0,1\}^n$ du réseau. En particulier, la probabilité qu'un chemin $x^1, \ldots, x^n, x^{n+1}$ d'états soit suivi par le système est alors :

$$P(x^1, \ldots, x^n, x^{n+1}) = \Pi_1^n P(x^i, x^{i+1}) \quad .$$

Nous sommes donc maintenant en mesure de répondre à la question précédemment posée : l'ensemble des trajectoires n'aboutissant pas à un point d'équilibre est-il de mesure nulle ? Pour cela nous allons utiliser la propriété suivante :

PROPRIÉTÉ. *Étant donné un processus de* MARKOV *sur un espace d'états X fini, si pour tout état il existe un chemin de probabilité non nulle et conduisant à un état d'équilibre,*

alors la probabilité d'observer une trajectoire n'aboutissant jamais à un état stable est nulle.

Pour tout état x du réseau, notons respectivement k_x et p_x la longueur du chemin menant de x vers un état d'équilibre et la probabilité pour que ce chemin soit effectivement suivi. L'ensemble d'états X étant fini, le nombre $p = \min_{x \in X}\{p_x\}$ est strictement positif et le nombre $k = \max_{x \in X}\{k_x\}$ est fini. Notons maintenant $\mathcal{P}_x^n$ la probabilité d'observer une évolution partant de l'état x et n'ayant pas abouti à un état d'équilibre au bout de n étapes. De même notons $\mathcal{P}_x^\infty$ la probabilité d'observer une trajectoire issue de l'état x et n'aboutissant jamais en un point d'équilibre. On a alors

$$\mathcal{P}_x^k \leq 1 - p_x \leq 1 - p < 1 \quad \text{pour tout } x \text{ de } X,$$

et donc

$$\mathcal{P}_x^{nk} \leq (1-p)^n \quad \text{pour tout } x \text{ de } X \text{ et tout entier } n.$$

Comme

$$\mathcal{P}_x^\infty \leq \mathcal{P}_x^n \quad \text{pour tout } x \text{ de } X \text{ et tout entier } n,$$

on a donc finalement

$$\mathcal{P}_x^\infty = 0 \quad \text{pour tout } x \text{ de } X. \qquad \square$$

Le point essentiel de ce résultat est qu'il n'est pas nécessaire de connaître la mesure de probabilité donnant au réseau un caractère de processus de MARKOV. Il nous est seulement nécessaire de connaître le graphe de transition d'état qui est uniquement induit par les poids de connexion du réseau. La seule hypothèse à faire est de supposer que l'asynchronisme des neurones du réseau est modélisable par un processus de MARKOV. Pour montrer alors que toutes les trajectoires d'un réseau asynchrone convergent presque sûrement vers un point d'équilibre, il nous suffit donc d'exhiber pour tout état du réseau un chemin partant de ce point, accessible à la dynamique du réseau et menant à un point d'équilibre. Ainsi dans le cas des réseaux de HOPFIELD les points d'équilibre sont les minima locaux de l'énergie et de tout point on atteint un de ces minima en suivant la trajectoire donnée par une itération séquentielle : un réseau de HOPFIELD donne donc presque sûrement une réponse dans le cas d'une dynamique asynchrone.

Ce principe a permis à B. JAKUBCZYK [28] d'exhiber une classe de réseaux de neurones dont la convergence est garantie. Les restrictions imposées aux poids de connexion sont moins restrictives que celles dues à J. HOPFIELD :

PROPRIÉTÉ (JAKUBCZYK). *Si un réseau de neurones asynchrone a une matrice*

de connexion (ω_{ij}) équilibrée, c'est-à-dire telle que pour tout cycle de neurones $(i_0, \ldots, i_n = i_0)$ le produit des poids associés est positif:

$$\prod_{k=1}^{n} \omega_{(i_{k-1}, i_k)} \geq 0 \quad ;$$

Alors l'ensemble des trajectoires n'aboutissant pas à un point d'équilibre en un temps fini est de mesure nulle et ce pour toute valeur des seuils ω_i.

La démonstration de ce résultat comporte plusieurs étapes.

PROPOSITION. *Si tous les poids de connexion du réseau sont positifs ou nuls,*

alors, partant de tout état du réseau il existe un chemin de probabilité non nulle et conduisant à un état d'équilibre.

Le chemin est le suivant et comporte deux phases. Dans une première phase, toutes les cellules se trouvant dans un état instable égal à 0 passent dans l'état 1. Dans une deuxième phase, toutes les cellules se trouvant dans un état instable égal à 1 prennent l'état 0. Un équilibre est alors atteint.

Notons tout d'abord que les deux phases ont une longueur finie : le nombre de cellules pouvant changer d'état au cours d'une étape de la première phase (respectivement , la seconde) est inférieur au nombre de cellules inactives (respectivement, actives) qui décroît d'au moins une unité à chaque étape. Les longueurs de ces deux phases sont donc au plus égales au nombre de cellules que nous avons supposé fini.

Par ailleurs, au début de la seconde phase, toute cellule inactive a un potentiel négatif, et cette propriété reste alors vérifiée durant toute cette seconde phase. En effet, en raison du fait que tous les coefficients synaptiques sont positifs et que les seules variations d'état y sont des inactivations, les potentiels d'action ne font que décroître. Les cellules inactives restent donc dans un état stable. De plus, par construction de la seconde phase, une cellule active ne devient inactive que si cet état est stable. A la fin de ce processus, toute les cellules inactives se trouvent donc dans un état stable. Par construction de la deuxième étape, il en est de même pour les cellules actives. Un équilibre est donc bien atteint. □

PROPOSITION. *Si le réseau est fortement connecté, c'est-à-dire si pour tout couple (i, j) de cellules il existe un chemin, dans le graphe de connexion, menant de la cellule i vers la cellule j,*

et si, de plus, pour tout cycle du graphe de connexion, le produit des poids associés est positif,

alors, il existe un réseau dont tous les poids de connexion sont positifs et une correspondance bijective entre les états de ces deux réseaux qui est respectée en cours d'évolution, c'est-à-dire telle, que si il existe un chemin menant d'un état à un autre pour le premier réseau, alors l'image de ce chemin correspond à un chemin du second réseau.

Le second réseau est construit en choisissant pour chaque neurone i un changement de signe $\epsilon_i \in \{-1, 1\}$. Le graphe de connexion de ce second réseau est le même que celui du premier mais le coefficient synaptique ω_{ij} de la cellule i vers la cellule j est remplacé par le coefficient ${\omega_{ij}}^*$ égal à:

$$ {\omega_{ij}}^* = \epsilon_i \epsilon_j \omega_{ij} \quad . $$

La correspondance qui à un état $x = (x_1, \ldots, x_n)$ du premier réseau associe l'état $x^* = (x_1^*, \ldots, x_n^*)$ du second réseau défini par:

$$ {x_i}^* = \epsilon_i x_i $$

et qui à un vecteur d'entrée $e = (e_1, \ldots, e_n)$ du premier réseau associe le vecteur d'entrée $e^* = (e_1^*, \ldots, e_n^*)$ du second réseau défini par:

$$ {e_i}^* = \epsilon_i e_i \quad , $$

est respectée au cours de toute évolution du réseau. C'est à dire que si pour le premier réseau l'état x peut basculer en l'état y alors pour le second réseau l'état x^* peut basculer en l'état y^*.

Pour obtenir un second réseau dont tous les coefficients sont positifs choisissons pour un neurone particulier i le changement de signe $\epsilon_i = 1$. Le changement de signe associé à un neurone j est alors pris égal au produit des poids le long d'un chemin de i à j. Par hypothèse il existe au moins un tel chemin et le fait que le produit des poids soit positif sur tous cycles nous garantit que tous les chemins de i à j ont un produit de poids de même signe. □

La preuve de la propriété de B. JAKUBCZYK est alors achevée en remarquant que tout réseau équilibré peut se mettre sous une forme hiérarchique de sous réseaux fortement connectés. □

5. Réseaux stochastiques

Les modèles stochastiques ont été introduits par G. HINTON, T. SEJNOWSKI et D. ACKLEY (1984) [24] avec la machine dite de BOLTZMANN. Ces modèles s'inspirent de la physique statistique pour transformer les réseaux de HOPFIELD, dont les trajectoires, rappelons-le, peuvent s'interpréter en termes de recherche d'un minimum local d'une fonction caractéristique du réseau et appelée énergie. Le principe est d'utiliser une dynamique stochastique qui permet d'obtenir un état

stationnaire au cours duquel la probabilité d'observer une configuration donnée est d'autant plus forte que l'énergie associée à cette configuration est faible. Complétée avec un principe de recuit simulé, encore une fois inspiré par la physique statistique, cette méthode permet de trouver les minima de la fonction d'énergie.

Le non-déterminisme qui intervient dans ces modèles stochastiques est plus fort que celui utilisé précédemment avec les machines asynchrones. Pour ces dernières un neurone, pris individuellement, a une loi d'évolution déterminée par les signaux qu'il reçoit et seul est indéterminée la synchronisation des divers changements d'états. Dans le cas des modèles stochastiques, un neurone isolé et recevant un signal donné a à lui seul un comportement non-déterministe : quel que soit son état un neurone peut changer d'état, le signal reçu ne faisant que moduler la probabilité de cette transition. Il n'y a alors plus de point d'équilibre au sens où nous l'avons rencontré jusqu'à maintenant : tout état est accessible de tout autre. Par contre, si l'on suit au cours du temps la distribution de probabilité de chaque état du réseau, celle-ci va converger vers une distribution stationnaire. Les modèles stochastiques se démarquent donc des réseaux que nous avons considérés jusqu'à présent. Avant de présenter la machine de BOLTZMANN, regardons d'abord un résultat sur les processus de MARKOV montrant l'intérêt d'une analogie avec la physique statistique pour un problème technique de recherche d'un optimum.

5.1. Distribution stationnaire de Gibbs

On se place dans le cadre d'un processus de MARKOV sur un espace d'états fini. On note $T(x|y)$ la probabilité d'une transition d'un état y vers un état x. On a donc la condition de normalisation suivante :

$$\forall y \quad \sum_x T(x|y) = 1 \quad .$$

On note maintenant $P(x,t)$ la probabilité d'observer l'état x à l'instant t. L'hypothèse de MARKOV se traduit alors par :

$$P(x,t+1) = \sum_y T(x|y) P(y,t) \quad .$$

On a alors la propriété suivante dont on pourra trouver une preuve dans le livre de Y. KAMP et M. HASLER [31]:

PROPRIÉTÉ. *Si les probabilités de transition sont telles que l'on puisse écrire :*

$$\frac{T(x|y)}{T(y|x)} = \frac{G(x)}{G(y)} \quad ,$$

et si il n'existe pas deux ensembles d'états S_1 et S_2 tels que la probabilité $T(x|y)$ soit nulle pour tout x de S_1 et y de S_2,

alors lorsque $t \to \infty$ *la distribution de probabilité* $P(x,t)$ *tend vers la distribution :*

$$P(x) = \frac{G(x)}{\sum_y G(y)} \quad .$$

Cette distribution de probabilité stationnaire est strictement positive et indépendante de l'état initial du système.

La quantité $G(x)$ étant strictement positive on peut la mettre sous la forme :

$$H(x) = -\frac{1}{\beta} \log G(x) \quad ,$$

où β est une constante positive. La distribution stationnaire s'écrit alors :

$$P(x) = \frac{\exp(-\beta H(x))}{\sum_y \exp(-\beta H(y))} \quad .$$

En mécanique statistique, cette distribution est appelée distribution de GIBBS et la fonction $H(x)$ est appelée hamiltonien du système et représente la notion d'énergie. Toujours en mécanique statistique, la constante β est liée à la température T par la relation $\beta = \frac{1}{kT}$, où k est la constante de BOLTZMANN. En écrivant :

$$\frac{P(x)}{P(y)} = \exp(\beta(H(y) - H(x))) \quad ,$$

on remarque qu'un état est d'autant plus probable à l'état stationnaire que son énergie est faible. De plus cet effet est d'autant plus accentué que la température est faible (que la constante β est grande). A la limite, lorsque la température tend vers 0, seuls les états d'énergie minimale sont observables. Ce principe pour obtenir un minimum d'une fonction est qualifié de *recuit simulé*. On trouvera une étude de la convergence de cette méthode dans l'article [36].

Ce sont ces dernières remarques qui guident l'approche stochastique des réseaux de neurones. L'idée est de construire une dynamique aboutissant à une distribution stationnaire à la GIBBS construite à partir d'une quantité que l'on souhaite minimiser et contrôlée par un paramètre, que l'on appelle par analogie température. En laissant le système atteindre un état stationnaire puis en abaissant la température suffisamment lentement pour rester en un tel état stationnaire, on pourra alors obtenir un des minima recherchés.

5.2. La machine de Boltzmann

La machine de BOLTZMANN a été introduite par G. HINTON, T. SEJNOWSKI et D. ACKLEY (1984) [24] et s'appuie sur les réseaux discrets de J. HOPFIELD. On se place donc dans le cadre suivant :

On considère un réseau de neurones pouvant prendre comme état -1 ou 1 et dont la fonction de seuillage est la fonction sign. La matrice de connexion (ω_{ij}) du réseau est supposée symétrique et à diagonale nulle de sorte que la quantité E, définie en tout point x de l'espace d'états $\{-1,1\}^n$ du réseau par :

$$E(x) = -\frac{1}{4} \sum_{ij} x_i \omega_{ij} x_j - \frac{1}{2} \sum_i \omega_i x_i \quad ,$$

soit décroissante pour toute transition faisant intervenir un seul neurone i suivant l'équation :

$$x_i^{t+1} = \text{sign}(\sum_j \omega_{ij} \, . \, x_j^t + \omega_i) \quad .$$

En effet, la différence d'énergie entre deux points x et y qui ne diffèrent que selon leur $i^{\text{ème}}$ coordonnée vérifie alors :

$$E(y) - E(x) = x_i \, . (\sum_j \omega_{ij} x_j + \omega_i) \quad .$$

On notera :

$$u_i = \sum_j \omega_{ij} x_j + \omega_i \quad .$$

On souhaite décrire un fonctionnement stochastique des neurones permettant d'obtenir une distribution stationnaire de GIBBS dont l'hamiltonien est la quantité E. Dans un premier temps, on considère qu'un seul neurone change d'état à la fois et l'on ne peut alors plus utiliser la propriété citée plus haut (car tout état n'est pas directement accessible de tout autre). Par contre, si le processus de MARKOV est choisi irréductible (tout état est accessible de tout autre par un nombre fini de transitions), comme l'ensemble des états considéré est fini, le processus est alors aussi récurrent (la probabilité que le système revienne à tout état déjà visité est de 1) et par suite, le système admet une et une seule distribution stationnaire qui est atteinte quelque soit la distribution d'états initiale. Il nous suffit donc dans ce cas de vérifier que la distribution de GIBBS $P(x)$ associée à $E(x)$ est stationnaire. Il suffit même de vérifier la condition plus simple suivante (avec les notations de la section précédente) :

$$\forall x, y \quad T(x|y)P(y) = T(y|x)P(x)$$

qui expriment que toute transition est observable avec la même probabilité que la transition inverse. La distribution d'états $P(x)$ est alors stationnaire :

$$\forall x \quad \sum_y T(x|y)P(y) = \sum_y T(y|x)P(x) = P(x)$$

Plusieurs choix pour les probabilités de transitions conviennent. Un choix usuel est le suivant :

Étant donné un nombre strictement positif T et un état x^t des neurones du réseau, un neurone i est choisi suivant une distribution de probabilité strictement positive $p(i)$ sur l'ensemble des neurones puis son état est transformé comme suit :

$$x_i{}^{t+1} = -x_i{}^t \qquad \text{si } x_i{}^t \neq \text{sign}(u_i{}^t)$$

$$x_i{}^{t+1} = \begin{cases} -x_i{}^t & \text{avec une probabilité de } \exp(\frac{-x_i{}^t . u_i{}^t}{T}) \\ u_i{}^t & \text{sinon} \end{cases} \qquad \text{sinon}$$

$$x_j{}^{t+1} = x_j{}^t \qquad \text{pour tout neurone } j \neq i.$$

La probabilité d'une transition est contrôlée par la différence d'énergie qu'elle entraîne ($E(x^{t+1}) - E(x^t) = x_i{}^t . u_i{}^t$)). Ainsi si x et y sont deux états du réseau qui diffèrent en plus d'un neurone, la probabilité d'une transition de y vers x est nulle. Si ces deux états x et y ne diffèrent que suivant l'état du neurone i (*i.e.* $x_i = -y_i$ et $\forall j \neq i x_j = y_j$), alors la probabilité d'une transition de y vers x est égale à :

$$T(x|y) = \begin{cases} p(i) . \exp(\frac{E(y)-E(x)}{T}) & \text{si } E(x) > E(y) \\ p(i) & \text{sinon} \end{cases}$$

Le neurone choisi, suivant la distribution de probabilité $p(i)$, change donc sûrement d'état si cette transformation entraîne une diminution de l'énergie. Dans le cas contraire la transition est d'autant moins probable que l'augmentation d'énergie est élevée. Pour tout état du réseau et tout neurone, la transition correspondant à un changement d'état de ce neurone est possible (la distribution de probabilité $p(i)$, guidant le choix d'un neurone étant strictement positive) et tout état est donc accessible de tout autre (par tirage successif de tout neurone dont l'état est différent d'un état global à l'autre). Il nous reste donc à vérifier que la distribution de GIBBS :

$$P(x) = \frac{\exp(\frac{-E(x)}{T})}{\sum_y \exp(\frac{-E(y)}{T})} \quad ,$$

vérifie l'équation :

$$\forall x, y \quad T(x|y)P(y) = T(y|x)P(x) \quad .$$

Si les deux états x et y diffèrent en plus d'un neurone les deux probabilités $T(x|y)$ et $T(y|x)$ sont nulles. Si x et y ne diffèrent que selon l'état du neurone i, supposons que l'on ait $E(x) \geq E(y)$. On a alors :

$$\begin{aligned} T(x|y)P(y) &= p(i) . \exp(\frac{E(y) - E(x)}{T}) . \frac{\exp(\frac{-E(y)}{T})}{\sum_z \exp(\frac{-E(z)}{T})} \\ &= p(i) . \frac{\exp(\frac{-E(x)}{T})}{\sum_z \exp(\frac{-E(z)}{T})} \\ &= T(y|x)P(x) \quad . \end{aligned}$$

Le processus aboutit donc à la distribution de GIBBS souhaitée et ceci à partir de toute distribution d'états initiale.

Un autre choix usuel est le suivant : un neurone i étant toujours choisi suivant une distribution de probabilité strictement positive $p(i)$ sur l'ensemble des neurones, son état est alors transformé comme suit :

$$x_i^{t+1} = \begin{cases} 1 & \text{avec une probabilité de } \frac{1}{1+\exp(-\frac{u_i^t}{T})} \\ -1 & \text{sinon.} \end{cases}$$

Si deux états x et y ne diffèrent que suivant l'état du neurone i, alors la probabilité d'une transition de y vers x est dans ce cas égale à :

$$T(x|y) = \begin{cases} p(i) \cdot \frac{1}{1+\exp(\frac{-u_i}{T})} & \text{si } x_i = 1 \\ p(i) \cdot (1 - \frac{1}{1+\exp(\frac{-u_i}{T})}) = p(i) \cdot \frac{1}{1+\exp(\frac{u_i}{T})} & \text{si } x_i = -1 \end{cases}$$

$$= p(i) \cdot \frac{1}{1 + \exp(\frac{-x_i u_i}{T})}$$

$$= p(i) \cdot \frac{1}{1 + \exp(\frac{E(x)-E(y)}{T})}$$

On a alors :

$$T(x|y)P(y) = p(i) \cdot \frac{1}{1 + \exp(\frac{E(x)-E(y)}{T})} \cdot \frac{\exp(\frac{-E(y)}{T})}{\sum_z \exp(\frac{-E(z)}{T})}$$

$$= p(i) \cdot \frac{1}{1 + \exp(\frac{E(y)-E(x)}{T})} \cdot \frac{\exp(\frac{-E(x)}{T})}{\sum_z \exp(\frac{-E(z)}{T})}$$

$$= T(y|x)P(x)$$

et la distribution de GIBBS associée à E est donc bien stationnaire.

Le problème rencontré avec ces machines de BOLTZMANN est que le temps pour atteindre une distribution pouvant être considérée comme stationnaire va être important : au fait que ce temps est déjà important pour un processus de MARKOV s'ajoute le fait qu'un seul neurone à la fois change d'état. Peut-on faire évoluer tout les neurones simultanément ?

5.3. Machine de Boltzmann parallèle

Faire évoluer simultanément plusieurs neurones d'une machine de BOLTZMANN est possible. Mais un choix de fond doit être fait : soit on désire obtenir une distribution stationnaire de GIBBS, identique à celle obtenue avec les machines que nous venons de voir et que nous qualifierons de séquentielles, et dans ce cas il faut se contraindre à un parallélisme partiel : seuls deux neurones non connectés peuvent évoluer simultanément. Soit l'on désire une simultanéité d'évolution

portant sur tous les neurones, et dans ce cas la distribution stationnaire obtenue sera toujours une distribution de GIBBS mais construite à partir d'un hamiltonien différent de celui utilisé jusqu'à présent.

Commençons par le cas d'un parallélisme partiel. On considère deux neurones i et j non-connectés ($\omega_{ij} = 0$) ainsi que quatre états différents du réseau a,b,c et d tel que l'on ait $a_k = b_k = c_k = d_k$, pour tout neurone k différent de i et j. Supposons de plus que l'on ait $a_j = b_j$ et $a_i = c_i$ (ce qui entraîne aussi $c_j = d_j$ et $b_i = d_i$). On a alors :

$$E(a) - E(b) = E(c) - E(d) \qquad \text{et} \qquad E(a) - E(c) = E(b) - E(d) \quad ,$$

Ce qui exprime qu'une transition du neurone i a la même probabilité avant ou après une transition du neurone j et inversement. L'ordre de ces deux transitions n'ayant pas d'effet on peut les considérer simultanées. Au lieu de choisir un seul neurone comme candidat à une transformation, on peut donc choisir un ensemble de neurones sans connexions et les faire évoluer de manière indépendante comme dans le cas d'une machine de BOLTZMANN séquentielle. On obtient alors une distribution stationnaire de GIBBS construite à partir de E.

Si l'on veut maintenant une évolution faisant intervenir en parallèle tous les neurones du réseau, on ne connaît plus *a priori* un hamiltonien satisfaisant (on ne dispose plus d'une quantité décroissante le long de toute trajectoire déterministe du réseau). On part donc d'une équation d'évolution, vérifiant les hypothèses de la propriété énoncée au cours de la sous-section 5.1. On obtient alors l'hamiltonien décrivant la distribution stationnaire du processus. Cette démarche a été suivie par W.A. LITTLE [33], P. PERETTO [39] ainsi que O. FRANÇOIS [11]. On en trouvera un exposé dans le livre [31]. Dans le modèle de LITTLE, un neurone i prend l'état x_i, sachant que le réseau se trouve dans un état y, avec une probabilité égale à :

$$\frac{\exp(\frac{x_i u_i(y)}{T})}{\exp(\frac{x_i u_i(y)}{T}) + \exp(\frac{-x_i u_i(y)}{T})} \quad ;$$

de sorte que l'on ait :

$$T(x|y) = \frac{\exp(\frac{\sum_i x_i u_i(y)}{T})}{\prod_i 2\cosh(\frac{u_i(y)}{T})}$$

En tenant compte de la symétrie de la matrice de connexions a alors :

$$\frac{T(x|y)}{T(y|x)} = \frac{G(x)}{G(y)}$$

avec

$$G(x) = \prod_i 2\cosh(\frac{u_i(y)}{T}) \quad ,$$

Que l'on peut aussi mettre sous la forme suivante :

$$G(x) = \sum_y \exp(\frac{\sum_{ij} x_i \omega_{ij} y_j}{T}) \quad .$$

La distribution stationnaire est donc la distribution de GIBBS associée à l'hamiltonien :

$$\begin{aligned} H(x) &= -T \log(G(x)) \\ &= -T \sum_i \log(2 \cosh(\frac{u_i(y)}{T})) \end{aligned}$$

L'inconvénient d'une machine de BOLTZMANN parallèle est donc qu'elle est guidée par un hamiltonien fort différent de celui, quadratique, qui guide une évolution séquentielle et dont l'usage est pratique comme nous le verrons au cours du chapitre II.

Conclusion

Le principe qui est à l'origine des réseaux de neurones peut être résumé comme suit : il consiste à concevoir un système comme l'agencement de systèmes élémentaires, les neurones. Ces derniers ont un état résumé par une quantité scalaire et qui peut être modulée par des signaux activateurs et inhibiteurs. Ces neurones sont alors connectés en un réseau de sorte que chacun puisse moduler l'activité des autres. Cette modulation est contrôlée par l'état du neurone et pondérée par des coefficients caractérisant le réseau. En termes plus précis, cette interaction entre neurones est décrite par l'ensemble des états d'équilibre.

Ce principe offre de très nombreuses latitudes : sur l'espace d'état de chaque neurone, sur la forme que prend l'interaction entre deux neurones et sur l'évolution au cours du temps de l'état du système. Ces diverses combinaisons donnent autant de systèmes dont les trajectoires peuvent être interprétées en termes de calcul : le choix d'une trajectoire est associé à une donnée de calcul et le point d'équilibre éventuellement atteint est considéré comme une réponse à ce calcul. C'est à ce point qu'apparaît la nécessité de faire un choix parmi la diversité des réseaux *a priori* possibles : chaque type de réseau sera plus ou moins approprié à une application ou une autre. Néanmoins, un problème étant donné, des réseaux de formes très différentes peuvent *a priori* y répondre. Comment faire un choix (fonction de seuillage, dynamique), puis comment exhiber un réseau construit à partir de ce choix et répondant au problème initial ? L'approche classiquement suivie consiste à s'appuyer sur le fait qu'un même réseau peut avoir une gamme de comportements très riche suivant les valeurs choisies pour les divers coefficients le caractérisant. On prend donc un réseau dont la gamme de comportements semble suffisamment vaste pour contenir le comportement souhaité. Les coefficients de

contrôle de ce réseau sont alors jugés par comparaison, entre le comportement qu'ils induisent et celui souhaité, puis modifiés par retouches successives. Lorsque ces améliorations sont effectuées avec méthode, on parle d'algorithmes d'apprentissage et ceux-ci feront l'objet du chapitre III.

A l'opposé de cette approche on trouve aussi une démarche constructive essayant de tirer parti au mieux de la connaissance que l'on peut avoir du comportement d'un réseau. Cette approche est suivie essentiellement avec les réseaux de HOPFIELD ou leur extension stochastique, où l'existence d'une quantité qui est diminuée au cours de toute trajectoire donne la possibilité d'aborder des problèmes d'optimisation. De plus, remarquons dès maintenant que les algorithmes d'apprentissage ne peuvent répondre définitivement au problème de l'obtention d'un réseau reflétant un comportement donné. Ils imposent en effet le choix d'un premier réseau ou plus précisément d'une structure de réseau (fonction de seuillage, dynamique de calcul, nombre de neurones, graphes de connexions). Comment s'assurer qu'au moins un choix des coefficients de contrôle pour ce réseau répond au problème à résoudre? Abordons maintenant, avec le chapitre II et sur la base d'exemples, ces problèmes de construction.

CHAPITRE II

Exemples et premières constructions

Introduction

Ce chapitre donne plusieurs exemples de réseaux de neurones. Nous avons essayé de montrer les diverses approches pour obtenir un réseau ayant le comportement souhaité. Outre l'intérêt de montrer comment aborder ce problème de synthèse, le but de ce chapitre est de cerner les recettes et les ingrédients qui permettent de tirer pleinement partie de ces techniques.

La première approche de synthèse est entièrement constructive: on calcule tous les paramètres définissant un réseau à partir des données du problème à résoudre. Cette approche est essentiellement utilisée pour des problèmes tenant de l'optimisation combinatoire et fait appel à des réseaux de HOPFIELD ou à la machine de BOLTZMANN.

A l'opposé, on peut compter sur l'existence d'algorithmes d'apprentissage, comme celui de la rétro-propagation du gradient, qui adaptent les connexions d'un réseau à la résolution d'un problème. C'est l'approche qui a suscité le plus d'engouement lors de l'apparition de la technique des réseaux de neurones. Ainsi, l'espoir mis en l'algorithme de rétro-propagation du gradient lors de son apparition a été tel que les causes de succès ou d'échecs ont été interprétées en termes de nombre de neurones, de connexions et de couches de neurones entre l'entrée et la sortie du système. Lors de la conception d'une application on procédait par retouches successives (ajout d'une couche de neurones ou augmentation du nombres de connexions) suivies de phases d'apprentissage sur une base d'exemples et ce, jusqu'à ce que le réseau ait le comportement souhaité sur une base de tests. La nécessité d'une adéquation entre la structure des liens entre neurones et la structure du problème à résoudre est restée presque complètement occultée jusqu'à ce que la complexité des applications visées soit telle que cette approche empirique ne puisse plus porter de fruits.

Aussi le second exemple présenté dans ce chapitre est le PERCEPTRON qui, malgré sa simplicité, montre bien la nécessité d'une bonne complémentarité entre

construction et apprentissage lors de la synthèse d'un réseau de neurones destiné à une application particulière. On donnera, au passage, une construction montrant comment on peut améliorer le principe du PERCEPTRON à l'aide de boucles de rétro-action.

Enfin nous présenterons plusieurs réseaux partiellement construits ou structurés : on déduit du problème à résoudre une structure de réseau rendant compte des caractéristiques du problème. On établit de même des liens que doivent respecter les différents poids de connexion du réseau pour que celui-ci puisse être un candidat à notre synthèse. Ce n'est qu'après avoir établi une telle structure pour le réseau que l'on fait appel à un algorithme d'apprentissage pour mettre au point les différents choix qu'il reste à faire et que l'on ne sait faire.

1. Recherche d'optimums

Les réseaux de HOPFIELD et les machines de BOLTZMANN ont une dynamique qui s'appuie sur une quantité $E(x)$ déduite des poids de connexion du réseau et définie en tout état x. Dans le cas des réseaux de HOPFIELD cette quantité est décroissante le long de toute trajectoire. Avec une machine de BOLTZMANN on obtient une distribution stationnaire d'états favorisant les points où cette quantité est moindre et cet effet peut être amplifié en modulant un paramètre du réseau qualifié de température, par analogie physique. Suivant la même analogie cette quantité $E(x)$ est appelée énergie de l'état x. Les états minimisant cette énergie étant soit l'aboutissement d'une évolution soit visité plus fréquemment, il est donc naturel d'utiliser ces réseaux dans le cadre de l'optimisation combinatoire. Le principe est de coder le problème de façon à pouvoir identifier la quantité à minimiser avec l'énergie d'un réseau dont on obtient ainsi les coefficients synaptiques.

Cette approche, initiée par J. HOPFIELD [29, 30], a été poursuivie par de nombreux auteurs dont G. HINTON et T. SEJNOWSKI [24, 21]. On peut aussi citer L. HÉRAULT [19] ou encore B. HELLSTROM et L. KANAL [18]. On trouvera un panorama de ces méthodes dans l'article de E. AARTS et J. KORST [2]. Ces divers exemples regroupent en fait deux approches de l'optimisation. La première tient réellement de l'optimisation combinatoire et on cherche à résoudre des problèmes comme celui dit du « voyageur de commerce » [1](recherche dans un graphe pondéré d'un circuit passant par tous les sommets et de poids le plus faible possible) ou celui du K-partitionnement de graphes [20] (recherche d'un partitionnement d'un graphe pondéré en K sous-graphes et induisant un coût d'interconnexion minimal). La deuxième approche est plus empirique et consiste à exprimer un problème non conceptualisé en termes de minimisation d'une fonction. Au problème de la recherche d'un optimum s'ajoute donc le problème de la justification du choix de la fonction. Nous avons choisi pour exposer respectivement ces deux

approches l'exemple de l'extraction du plus grand sous-graphe indépendant [2] ainsi que le système de regroupement perceptif de L. HÉRAULT [19]. Le premier exemple a été choisi car il est très simple et montre parfaitement la démarche à suivre. Le second a été retenu pour son caractère prospectif.

1.1. Extraction d'un sous-graphe indépendant maximal

Soit $G = (X, U)$ un graphe. On suppose que G ne comporte pas de boucle (*i.e.* $\forall i \in X \quad (i,i) \notin U$). Un sous-graphe indépendant de G est une partie Y de X telle qu'aucun couple de Y^2 n'appartienne à l'ensemble des arcs U. On se propose de construire un réseau de neurones permettant de trouver un tel sous-graphe indépendant et maximal au sens de la cardinalité (E. AARTS et J. KORST [2]). Associons à chaque sommet i de X une variable x_i à valeur dans $\{0,1\}$ et exprimant si, oui on non, le sommet i appartient au sous-graphe réponse. Le problème s'écrit donc :

$$\begin{array}{ll} \text{maximiser} & \sum_{i \in X} x_i \\ \text{avec les contraintes :} & x_i x_j = 0 \quad si (i,j) \in U \quad . \end{array}$$

On utilise alors le principe dit de pénalisation et qui revient ici à exprimer le problème sous la forme :

$$\text{maximiser} \qquad F(x) = \alpha \sum_{i \in X} x_i - \beta \sum_{(i,j) \in U} x_i x_j \quad ,$$

où α et β sont deux paramètres positifs. En d'autre termes les configurations x ne codant pas un sous-graphe indépendant sont pénalisées.

Plus rigoureusement, si l'on choisit α et β de sorte que l'on ait $\beta > \alpha > 0$, on a alors les deux propriétés suivantes :

i) Si une configuration x a une valeur $F(x)$ localement maximale (qui ne peut être augmentée en changeant la valeur x_i de la variable associée à un sommet i) alors cette configuration code un sous-graphe indépendant. Il suffit de remarquer que si la configuration ne satisfait pas les contraintes, on augmente la valeur de F de la quantité positive $\beta - \alpha$ lorsqu'on retire un des sommets incriminés.

ii) Le paramètre α étant positif, la valeur associée par F à une configuration satisfaisant les contraintes sera d'autant plus grande que cette configuration comporte un grand nombre de sommets.

Il nous reste maintenant à identifier l'opposé de la quantité F, que nous cherchons à maximiser, à l'énergie E d'un réseau de neurones, qui elle, décroît au cours d'une dynamique séquentielle du réseau. Pour un réseau dont les neurones

prennent leur état dans $\{0,1\}$ et dont les coefficients synaptiques et de seuils sont respectivement notés ω_{ij} et ω_i, cette énergie E est égale à :

$$E(x) = -\frac{1}{2}\sum_{ij} x_i \omega_{ij} x_j - \sum_i \omega_i x_i \quad .$$

Pour que cette énergie soit effectivement décroissante le long de toute trajectoire où les neurones changent d'état un à un, il est suffisant que l'on ait :

$$\forall i,j \qquad \omega_{ij} = \omega_{ji} \quad \text{et} \quad \omega_{ii} = 0 \quad .$$

En prenant $\alpha = 1$ et $\beta = 2$, on obtient donc par identification de E et $-F$ les coefficients suivants :

$$\omega_i = 1$$

$$\omega_{ij} = \begin{cases} 0 & \text{si } (i,j) \notin U \text{ et } (j,i) \notin U \\ -4 & \text{si } (i,j) \in U \text{ et } (j,i) \in U \\ -2 & \text{sinon} \end{cases}$$

En utilisant une dynamique séquentielle comme HOPFIELD, ce réseau nous donne une réponse qui satisfait les contraintes. En cherchant d'abord une distribution stationnaire d'états à l'aide d'une dynamique à la BOLTZMANN, puis en évoluant de manière séquentielle vers un état d'énergie localement minimale à partir d'un état désigné par le processus stochastique, on obtient une réponse satisfaisant les contraintes. De plus, les solutions maximales seront plus probablement atteintes par ce processus. Cet effet peut même être amplifié en diminuant progressivement le paramètre jouant le rôle de température dans le processus stochastique.

D'une manière plus générale, pour obtenir un réseau de neurones permettant de résoudre un problème d'optimisation combinatoire, on procède comme suit. On exprime le critère à optimiser en fonction de variables binaires. Puis l'on utilise le principe de pénalisation pour tenir compte des contraintes du problème et celles éventuellement induites par le code utilisé (toutes les configurations ne codent pas nécessairement une réponse au problème). Le critère F ainsi obtenu doit être quadratique de façon à pouvoir être identifié à l'énergie d'un réseau de neurones. Il doit enfin vérifier les deux points suivants :

i) Si une configuration x a une valeur $F(x)$ localement optimale (qui ne peut être améliorée en changeant la valeur d'une seule des variables) alors cette configuration code une réponse satisfaisant les contraintes.

ii) La valeur associée par F à une configuration satisfaisant les contraintes est d'autant plus importante que la réponse codée est proche de l'optimale.

Le second point permet d'obtenir avec une machine de BOLTZMANN une distribution de probabilité favorisant les réponses qui nous intéressent. Le premier nous permet d'obtenir une réponse satisfaisant les contraintes à l'aide d'une itération séquentielle à la HOPFIELD.

1.2. Un système de regroupements perceptifs

Pour sa thèse de doctorat [19], L. HÉRAULT a présenté un système de perception de courbes s'appuyant sur un mécanisme à la HOPFIELD. Le but de ce travail prospectif est de rendre compte de la facilité que l'on a à extraire d'une image l'information pertinente et la démarche poursuivie s'appuie sur le Gestaltisme. Cette théorie psychologique de la perception est fondée sur la notion de groupements perceptifs et postule que les divers éléments composant une sensation, visuelle, auditive ou autre, sont regroupés suivant certains critères, faisant intervenir des propriétés comme la proximité, la couleur ou la continuité, de façon à être perçues comme un tout unitaire. Dans un tel groupement perceptif ne sont retenus que les composants permettant au tout ainsi construit de satisfaire au mieux ces critères globaux. Pour cette théorie, un ensemble d'éléments d'informations est donc considéré comme un tout si il minimise un critère de cohérence. D'où l'idée de L. HÉRAULT d'utiliser une machine de BOLTZMANN pour reproduire le phénomène de perception. L'exposé qui suit insiste plus sur les articulations de la démarche suivie que sur les points techniques : le lecteur est donc invité à se reporter à l'article d'origine [19].

Le système proposé par L. HÉRAULT a pour but d'extraire, parmi un nuage de traits, ceux pouvant correspondre à une ligne la plus régulière possible. C'est donc un système destiné à épurer une image de contours de tous les points parasites qui n'ont pu être éliminés par des critères de variation du gradient de l'intensité lumineuse dans l'image initiale. Un seul regroupement est recherché et l'on souhaite que les traits conservés retiennent les courbes présentes dans l'image et que soient rejetés tous les traits ne pouvant être raccordés à ces courbes. Le principe consiste à associer à chaque couple (i, j) de traits un coefficient c_{ij} d'autant plus grand que ces deux traits peuvent faire partie d'une même ligne régulière. De même, à chaque trait i est associée une variable x_i dont la valeur, 0 ou 1, signifie que le trait doit être respectivement rejeté ou conservé dans le groupement perceptif. On cherche alors les valeurs de ces différentes variables x_i maximisant la quantité

$$\sum_{ij} x_i c_{ij} x_j - \lambda . (\sum_i x_i)^2 \quad .$$

Le deuxième terme a pour rôle d'empêcher que tous les points soient pris et le nombre de points conservés peut être modulé à l'aide du paramètre positif λ. Un choix devra être fait par le système et un trait aura une possibilité d'être pris d'autant plus grande qu'il peut se raccorder à de nombreux points. La forme du second terme a été choisie de manière à ce que la quantité à maximiser puisse se

réécrire

$$\sum_{ij} x_i(\lambda - c_{ij})x_j \quad ;$$

ce qui permet d'utiliser un réseau de neurones pour la recherche de ce maximum. Il reste maintenant à trouver un critère rendant compte de la possibilité de considérer deux traits comme faisant partie d'une même courbe et de construire ainsi les coefficients c_{ij}.

Le critère de regroupement retenu s'appuie sur la cocircularité entre deux traits et a été proposé par P. PARENT et S. W. ZUCKER [37]. Deux segments de droites seront donc retenus dans un même regroupement perceptif s'il existe un cercle (ou une droite) tangent à ces deux segments en leur milieu. Tel quel ce critère est trop restrictif et les regroupements obtenus serait sans grand intérêt. Plusieurs raffinements sont donc apportés.

On veut ainsi pouvoir regrouper deux traits qui sont presque cocirculaires ; ce qui permet de s'affranchir des erreurs de quantification mais aussi de percevoir des courbes autres que des courbes circulaires par morceaux. Parler de traits presque circulaires revient à choisir une topologie pour l'espace des segments de droite et à associer à chaque trait un voisinage contenant tous les traits pouvant lui être substitués. Deux segments de droite sont alors presque cocirculaires s'il existe dans leur voisinage respectif deux segments cocirculaires. Le choix retenu par L.HÉRAULT est de considérer comme voisin deux traits dont les centres se trouvent distant de moins d'un demi-pixel et dont les directions (modulo π) diffèrent de moins que le pas de discrétisation utilisé pour les angles. Deux traits i et j sont ainsi cocirculaires, aux erreurs de quantification près, si :

$$|\lambda_i + \lambda_j - \pi| \leq \epsilon + 2\arcsin(\frac{d}{d_{ij}}) \quad ,$$

où ϵ et d sont respectivement le pas de discrétisation utilisé pour les angles et le diamètre d'un pixel. d_{ij} est la distance séparant les centres des deux traits i et j. λ_i et λ_j sont respectivement les angles que font les deux traits i et j avec le segment rejoignant les deux centres. Pour palier au caractère brutal de ce critère, selon lequel deux traits sont presque cocirculaires ou ne le sont pas, L.HÉRAULT prend comme coefficient caractérisant la cocircularité des traits le coefficient suivant :

$$c_{ij} = (1 - \frac{\Delta_{ij}^2}{\pi^2}) \cdot \exp(\frac{-\Delta_{ij}^2}{k})$$

avec :

$$\Delta_{ij} = |\lambda_i + \lambda_j - \pi|$$

et le paramètre k étant choisi de telle sorte que le point $\Delta_{ij} = \epsilon + 2\arcsin(\frac{d}{d_{ij}})$ soit un point d'inflexion. L.HÉRAULT corrige aussi ce coefficient c_{ij} par un facteur de

la forme $\exp(\frac{-d_{ij}^2}{2\sigma^2})$ exprimant ainsi que la notion de cocircularité n'a un sens que dans un contexte local.

Pour extraire un ensemble de traits cohérents d'un nuage de traits, il faut donc maintenant calculer pour tout couple (i,j) de traits le coefficient c_{ij}. On obtient alors par identification les coefficients synaptiques d'un réseau de neurones. Puis à l'aide d'une machine de BOLTZMANN ou un réseau de HOPFIELD continu, tous deux contrôlés par un paramètre identifié à une température et que l'on fait décroître, on obtient un ensemble de traits maximisant le critère de cohérence choisi. Les exemples donnés par L.HÉRAULT [19] montrent que ce critère de cohérence reflète effectivement une notion de régularité de courbes.

Mais les choix faits par L. HÉRAULT pour mettre en œuvre son système semblent arbitraires. La part d'intuition est telle que l'on voit mal comment une démarche similaire pourrait être menée dans un autre cadre. De plus et surtout, on calcule les coefficients c_{ij}, et donc les coefficients définissant le réseau, à partir du nuage de traits initiaux, et de même, on associe un neurone uniquement aux traits initiaux. Par suite aucune cohérence n'est recherchée par ajout de traits, ce qui aurait pu aider à l'extraction de courbes régulières. Enfin, on voit mal comment ce système pourrait être connecté à un système amont d'extraction de traits dans une image. Il faudrait pour cela, d'une part, pré-calculer les coefficients de cocircularité c_{ij} pour tous les couples de traits potentiellement détectables par le système amont; puis, d'autre part, exprimer l'interaction des deux systèmes. Le système amont d'extraction de traits depuis une image, doit pouvoir imposer un trait au système aval avec d'autant plus de force que les indices dont il dispose (gradient de l'intensité lumineuse) sont probants. De même, le système aval doit pouvoir ajouter un trait, et par là, aider le système amont à déceler un indice trop faible. On retrouve donc le problème, cité au cours de l'introduction, de la coopération entre les parties amont et aval d'un système.

2. A propos du Perceptron

L'exemple présenté dans cette section, le PERCEPTRON, a été choisi pour sa simplicité et le fait qu'il montre bien ce que l'on peut et doit attendre d'un algorithme d'apprentissage mais aussi parce qu'il a fait couler beaucoup d'encre à la suite des travaux de M. MINSKY et S. PAPERT [35]. Il me semble que ces travaux ont été très mal compris par la communauté des chercheurs sur les réseaux de neurones qui ont souvent pris l'algorithme de rétro-propagation et les réseaux multi-couches comme une échappatoire aux limitations décrites par M. MINSKY et S. PAPERT. Le but de cette section est double. Il s'agit, tout d'abord, de clarifier le débat sur le PERCEPTRON qui se situe au niveau de la construction d'une machine ad hoc et non de la puissance de calcul du modèle. Nous verrons ensuite comment modifier le PERCEPTRON à l'aide de boucles de rétro-action

de façon à pouvoir construire un réseau d'automates reconnaissant le caractère connexe d'une image.

2.1. Le Perceptron

Le Perceptron, introduit par F. ROSENBLATT [45] en 1957, est un modèle de machine tentant de rendre compte des facilités de reconnaissance du système visuel humain ou animal. Un PERCEPTRON particulier est constitué d'une rétine, d'un ensemble de cellules d'association et d'une cellule de décision. La rétine est destinée à recevoir l'image à interpréter et peut être éventuellement infinie ou continue. Elle est constituée de points pouvant prendre l'un des deux états, 0 ou 1, dénotant une absence ou une présence de lumière. La cellule de décision est un neurone de MC CULLOCH et PITTS [34] dont l'état, toujours 0 ou 1, est calculé à partir des résultats des cellules d'association, résultats qui sont pondérés par des coefficients synaptiques. Les cellules intermédiaires sont en nombre fini et calculent chacune la valeur d'une fonction booléenne prédéterminée et dépendant uniquement de l'image reçue par la rétine. Un PERCEPTRON est donc défini par un nombre fini de fonctions A_i associées à des coefficients pondérateurs ω_i. Face à une image x, la réponse de la cellule de décision est alors

$$s(x) = \begin{cases} 1 & \text{si} \quad \sum_i \omega_i A_i(x) \geq 0 \\ 0 & \text{sinon.} \end{cases} ,$$

Un PERCEPTRON opère ainsi une classification de l'ensemble des images en deux parties et l'intérêt de ce modèle est donc de cerner la puissance de calcul d'un neurone de MC CULLOCH et PITTS. Dans ce but, M. MINSKY et S. PAPERT [35] se sont attachés à la construction de Perceptron réalisant diverses classifications, telles la reconnaissance du caractère convexe d'une image. Il nous faut maintenant préciser ce que l'on entend par construction. En effet, les contraintes imposées aux fonctions d'association sont très faibles et on peut toujours concevoir un PERCEPTRON ne comportant que deux cellules intermédiaires et réalisant une classification donnée à priori. Une première cellule, à valeur constamment égal à 1, est associée à un coefficient pondérateur de -1 et une deuxième cellule calcule le résultat souhaité qui est alors pondéré par la valeur 2. La cellule de décision ne fait donc que copier le résultat, correct par définition, de la deuxième cellule d'association et la difficulté est seulement reportée.

Aussi on entend, par construction, une décomposition du problème initial en éléments de complexité moindre. La complexité d'un PERCEPTRON peut être évaluée par la complexité des connexions entre la rétine et les cellules d'association. On peut ainsi limiter le nombre de points de la rétine dont vont dépendre les calculs intermédiaires. On peut aussi limiter en taille la portion de l'image affectant une cellule d'association ou rechercher une régularité dans le schéma de connexions. Par exemple, pour reconnaître l'ensemble des images convexes on

peut se limiter à trois points d'entrée par cellule intermédiaire [35]. Il suffit, en effet, d'associer à chaque triplet de points alignés de la rétine une cellule indiquant si les extrémités sont toutes deux noires alors que le point intermédiaire est blanc. La cellule de décision ne doit alors avoir une réponse positive que lorsqu'aucune anomalie de convexité n'est signalée. Ce principe ne peut, bien sur, être appliqué que dans le cas d'images discrètes mais peut être raffiné à une précision arbitraire.

Pour compléter cette approche de synthèse par décomposition en tâches élémentaires il faut ajuster les valeurs des coefficients de pondération. L'algorithme d'apprentissage présenté à cette fin par F. ROSENBLATT [45] a la propriété remarquable suivante. Si les fonctions d'association ont été suffisamment bien choisies pour qu'il existe un choix des poids de connexion satisfaisant, alors l'algorithme converge vers une solution après un nombre fini d'essais.

PROPRIÉTÉ (ROSENBLATT). *Étant donnés un* PERCEPTRON *de fonction d'association A, à valeur dans $\{0,1\}^n$, et un ensemble X d'images,*

si X est reconnaissable, c'est à dire s'il existe un vecteur de poids ω^* de $\mathbb{R}^n$ et un réel strictement positif l tels que l'on ait pour toute image x de X

$$\omega^* . A(x) > l \quad ,$$

alors la suite $(\omega_i)_{i\in\mathbb{N}}$ de vecteurs de poids construite comme suit à partir d'une suite $(x_i)_{i\in\mathbb{N}}$ d'éléments de X :

$$\begin{aligned} \omega_o &= 0 && , \\ \omega_{i+1} &= \omega_i && si \quad \omega_i . A(x_i) > 0 \\ et \quad \omega_{i+1} &= \omega_i + A(x_i) && sinon, \end{aligned}$$

converge vers un vecteur de poids ω permettant de reconnaître toutes les images de la suite d'exemples à partir d'un certain rang N, c'est-à-dire tel que l'on ait pour tout élément x de la suite d'images $(x_i)_{i\geq N}$:

$$\omega . A(x) > 0 \quad .$$

De plus cette limite est effectivement atteinte après un nombre fini de modifications du vecteur de poids, ce qui se produit lorsque l'exemple présenté x_i est mal classé par le PERCEPTRON *à l'aide du vecteur ω_i.*

L'algorithme d'apprentissage de F. ROSENBLATT pour le PERCEPTRON permet donc de mener correctement une démarche de construction. Il met, en effet, en évidence un critère d'adéquation entre les fonctions d'association et le problème à résoudre. On peut alors choisir les cellules intermédiaires et adapter les poids de connexion avec un nombre fini d'exemples. Cet algorithme rend parfaitement

compte du rôle d'un algorithme d'apprentissage qui est de permettre d'ajuster à une situation précise un mécanisme guidé par des principes généraux. On peut tout de suite noter une faiblesse de cet algorithme d'apprentissage qui lui donne une coloration d'exemple d'école. Prouver qu'un choix des cellules d'association convient, revient en effet, le plus souvent, à exhiber un vecteur de poids adéquat et, par là, à rendre inutile tout apprentissage !

Les limitations du PERCEPTRON et de cette démarche de synthèse ont été pleinement soulignées par M. MINSKY et S. PAPERT qui ont exploré et exposé dans leur livre [35] de nombreux exemples de mise en œuvre. L'exemple négatif le plus connu est sans aucun doute celui de la reconnaissance du caractère connexe d'une image. Ils ont montré que cette reconnaissance ne peut être effectuée par un PERCEPTRON dont aucune cellule d'association ne calcule son état à la vue de l'ensemble de la rétine. Autrement dit, on n'arrive pas à décomposer le problème initial en sous-problèmes de complexité moindre. Pour être précis, M. MINSKY et S. PAPERT ont uniquement montré que l'on ne peut pas réduire la complexité en se bornant uniquement à des portions de l'image et que la propriété de connexité a un caractère de globalité dont on ne peut pas s'affranchir. Ne voyant pas d'autre moyen pour décomposer le problème initial, on aboutit à une impasse.

En modifiant uniquement les poids de connexion de la cellule de décision, l'algorithme d'apprentissage du PERCEPTRON ne permet de tirer profit que des choix de construction faits a priori et ne permet nullement de les affiner. On aimerait pouvoir modifier aussi les fonctions d'association. Un premier pas en ce sens consiste à utiliser des neurones de MC CULLOCH et PITTS comme pour la cellule de décision. Le rôle de ces cellules intermédiaires, ou cachées car leur état n'est pas intéressant en tant que tel, est de calculer les fonctions d'association. Elles sont donc organisées en arbres dont les feuilles recueillent l'information sur la rétine et dont la racine résume l'association faite en descendant l'arbre. On obtient un réseau en couches et sans boucle. En modifiant les coefficients synaptiques associés aux cellules cachées on peut ainsi moduler les calculs intermédiaires. La difficulté rencontrée lors d'un apprentissage pour un tel réseau multi-couches vient du fait que l'on n'a pas de critère pour guider les modifications à faire sur les cellules cachées. L'algorithme de rétro-propagation du gradient, qui a été exprimé initialement dans ce contexte [6, 38, 48, 47], apporte une solution à ce problème.

Avant d'aborder la présentation de l'algorithme de rétro-propagation du gradient (objet du chapitre III), on peut dès maintenant se poser la question : cet algorithme permet-il de s'affranchir de toutes les limitations du PERCEPTRON soulignées par M. MINSKY et S. PAPERT ? Le sentiment général a souvent été que oui. Ma réponse est plus mitigée. Certes, l'algorithme de rétro-propagation permet d'explorer un espace de machines beaucoup plus grand et permet donc a priori un effort de construction préalable moindre. Mais les limitations imposées par une application sur la structure des fonctions d'association restent valable

pour les réseaux multi-couches qui ne sont que des PERCEPTRONS particuliers. La difficulté que l'on avait à construire d'une manière méthodique des fonctions d'association convenables est seulement reportée à la phase d'apprentissage où l'on ne dispose comme outil de vérification que de l'essai sur un exemple. S'il est alors possible d'obtenir un réseau satisfaisant, on n'a aucun moyen de s'en assurer ce qui ne peut être satisfaisant d'un point de vue technologique. De plus, si l'on désire augmenter la taille du problème (par exemple, dans le cas de la reconnaissance de formes, si l'on désire augmenter le nombre de pixels de l'image), on est contraint de recommencer tout l'apprentissage avec un réseau de taille supérieure.

2.2. Un Perceptron récursif reconnaissant la connexité

Avant de quitter le PERCEPTRON et sa simplicité pour des réseaux à la structure de plus en plus complexe, revenons au problème de la reconnaissance de la connexité. Quelle est la raison profonde de notre incapacité à décomposer ce problème en termes de fonctions d'association? M. MINSKY et S. PAPERT ont beaucoup insisté sur le caractère global de la notion de connexité et ont souligné que même pour un lecteur humain ce problème n'est pas évident et demande un certain temps de réflexion[1]. En voulant obtenir un maximum de parallélisme, aurait-on perdu une certaine forme de séquentialité qui nous fait maintenant défaut?

Le calcul effectué par le PERCEPTRON est en effet entièrement parallèle, chaque cellule d'association calculant son état d'une manière totalement indépendante. En imaginant une réalisation du PERCEPTRON comportant autant d'unités de calcul que nécessaire, ceci permet d'obtenir un gain en temps maximum. Or on s'aperçoit rapidement, lorsque l'on veut faire exécuter un algorithme par un ensemble d'unités de calcul, que ce type de situation est plutôt l'exception que la règle, et que la plupart du temps une certaine coopération entre unités de calcul est nécessaire. Des choix doivent être faits de manière cohérente et cette cohérence renvoie à une notion de globalité. Le calcul est décomposé et effectué en divers lieux, par souci d'efficacité, mais ne garde un sens que dans sa globalité. Si un choix de circonstance doit être fait par une unité de calcul particulière, la décision doit être prise en fonction des choix antérieurs qui ont pu être faits en un autre lieu. Cette contrainte, liée au sens de l'algorithme, fait qu'une exécution en parallèle n'échappe pas, en général, à cette forme de séquentialité qu'est la synchronisation entre processus.

Retrouve-t-on ces problèmes de synchronisation lors de la reconnaissance du

[1]Pour l'anecdote, on peut rappeler l'exemple figurant en couverture du livre de M. MINSKY et S. PAPERT [35]. On y trouve, d'une part, deux spirales imbriquées l'une dans l'autre et, d'autre part une image si similaire qu'il faut un certain temps pour réaliser que cette deuxième image est connexe alors que la première est composée de deux parties. On est amené à suivre mentalement le contour de l'image pour vérifier qu'il n'y a qu'une seule ligne entrelacée.

caractère connexe d'une image? La preuve de M. MINSKY et S. PAPERT repose sur l'existence de cas pathologiques, des images construites de manière *ad hoc* de façon à empêcher une prise de décision locale. Mais on peut imaginer des classes restreintes d'images ainsi que les PERCEPTRONS associés permettant de reconnaître si une image est connexe ou non sachant qu'elle appartient à la classe correspondante. On peut ainsi se restreindre à la classe des images ne comportant pas deux pixels coloriés contigus et une image sera alors reconnue comme connexe si elle comporte au plus un seul pixel colorié. Pour reconnaître le caractère connexe d'une image quelconque, un deuxième processus est nécessaire. Il faut déformer l'image d'origine (par exemple, par érosion), en conservant sa connexité, jusqu'à obtenir une image de la classe pour laquelle on sait résoudre le problème avec un Perceptron classique. C'est au cours de cette phase de déformation qu'interviennent les problèmes de synchronisation. En effet, si l'on procède par érosion, deux pixels voisins et formant le dernier lien entre deux parties de l'image ne doivent pas être érodés simultanément.

Le principe que nous allons suivre pour reconnaître le caractère connexe de l'image est donc le suivant: nous allons choisir un sous-ensemble des images, M, pouvant être reconnu par un PERCEPTRON à diamètre limité et tel que le caractère connexe d'une image de M puisse, lui aussi, être reconnu par un PERCEPTRON à diamètre limité. Puis nous allons construire un processus transformant une image quelconque en une image de M ayant le même nombre de composantes connexes que l'image initiale[2]. Le second PERCEPTRON nous permet alors d'obtenir le caractère connexe de l'image initiale et le premier de valider cette réponse en reconnaissant l'aboutissement du processus de transformation. On s'impose de plus que le processus de transformation puisse être assuré par des automates ayant chacun accès uniquement à une partie de l'image. La construction qui suit montre l'aboutissement de cette démarche.

On considère un écran de $n \times n$ pixels. On appelle voisins d'un pixel les huit pixels qui l'entourent. Une image x est définie par la donnée pour chaque pixel i d'une valeur x_i appartenant à $\{0,1\}$. Une image x est dite connexe ssi pour tout couple (i,j) de pixels avec $x_i = x_j = 1$ on peut passer du pixel i au pixel j en ne passant que par des pixels k deux à deux voisins et vérifiant $x_k = 1$. On notera C l'ensemble des images connexes.

[2]C'est à ce niveau qu'apparaît le fait que l'ensemble des images ne comportant pas deux pixels contigus, associé à un mécanisme d'érosion, ne nous convient pas. En effet un anneau suffisamment grand ne pourra pas être réduit à un point par des automates n'en visualisant chacun qu'une partie, sans que l'un d'eux prenne la liberté de faire varier localement le nombre de composantes connexes. Lors de la présentation d'une image connexe et donnant à cet automate une vue locale identique à celle donnée par l'image de l'anneau, il y aura variation du nombre de composantes connexes de l'image. Au lieu de procéder par érosion, nous allons donc colorier l'image de départ.

On se propose de transformer l'image par ajout de pixel ($x_i \leftarrow 1$) sans entraîner de variation locale, à un pixel et son voisinage, du nombre de composantes connexes (et donc sans entraîner de variation globale du nombre de composantes connexes). Soit, pour chaque pixel i de l'image, la fonction V_i associant à une image x une valeur dans $\{0,1\}$ comme suit : $V_i(x) = 1$ ssi le pixel i est vierge et peut être colorié en noir sans entraîner une variation locale du nombre de composantes connexes. Tous les pixels i pour lesquels $V_i(x) = 1$ (et seulement ceux-ci) peuvent être coloriés par ce processus de transformation. L'ensemble M des images fixes par ce processus (ou maximales) est l'ensemble de toutes les images x telles que pour tout pixel i l'on ait $V_i(x) = 0$.

Cet ensemble M est reconnaissable par un PERCEPTRON dont les fonctions d'association sont les fonctions V_i. De plus, en notant que les seules images connexes de M sont l'image totalement vierge et l'écran noir, on peut construire un PERCEPTRON reconnaissant le caractère connexe des images de M[3]. On notera que ces deux Perceptrons (celui reconnaissant M et celui reconnaissant C parmi M) sont à diamètre limité. Il nous reste maintenant à construire le processus de transformation d'images.

Étant donnée une image x, les pixels i qui vont pouvoir être coloriés ($x_i \leftarrow 1$) sans entraîner une variation locale du nombre de composantes connexes sont ceux tels que $V_i(x) = 1$ (par définition des fonctions V_i). Tous ces pixels ne peuvent néanmoins pas être coloriés simultanément. En effet, si deux pixels voisins, i et j, sont coloriés simultanément, il peut très bien y avoir création d'une connexion même si l'on a $V_i(x) = 1$ et $V_j(x) = 1$. Autrement dit l'algorithme parallèle :

$x \leftarrow$ image initiale
tant que x n'appartient pas à M
    simultanément pour tous les pixels
        si $V_i(x) = 1$ alors $x_i \leftarrow 1$
image finale $\leftarrow x$

ne permet pas de conserver le nombre de composantes connexes. Il est donc nécessaire de faire un choix parmi l'ensemble des pixels pouvant être coloriés et

[3]Il suffit de prendre une cellule d'association $A_{(i,j)}$ par couple (i,j) de pixels voisins. Face à une image x la réponse d'une de ces cellules est définie par $A_{(i,j)}(x) = 1$ ssi $x_i = x_j$, c'est-à-dire ssi les deux pixels voisins i et j sont de la même couleur dans l'image x.

utiliser un algorithme de la forme suivante :

$x \leftarrow$ image initiale
tant que x n'appartient pas à M
    Parmi les pixels vérifiant $V_i(x) = 1$
        choisir un ensemble T de pixels qui ne sont pas voisins
    Simultanément pour tous les pixels de T
        $x_i \leftarrow 1$
image finale $\leftarrow x$.

On souhaite maintenant construire un réseau de neurones effectuant le choix de l'ensemble T de manière non déterministe. On associe pour cela à chaque pixel i un neurone t_i dont l'état d'équilibre indiquera si i appartient ou non à T. On connecte chaque neurone t_i à la cellule d'association V_i du PERCEPTRON reconnaissant M et à tous les neurones t_j où j est un pixel voisin de i. Le fonctionnement de ces neurones est le suivant :

$$t_i \leftarrow \begin{cases} 1 & \text{si } u_i > 0 \\ 0 & \text{sinon} \end{cases} \qquad \text{avec une probabilité } 0 < p < 1,$$

où u_i est égal à

$$u_i = -\frac{1}{2} + V_i(x) - \sum_{j \text{ voisin de } i} t_j \quad .$$

Un neurone i est donc à l'équilibre ssi ($t_i = 1$ et $u_i > 0$) ou ($t_i = 0$ et $u_i \leq 0$). On pourra vérifier que tout état d'équilibre de ce réseau code effectivement un choix de l'ensemble T. De plus, un état d'équilibre est atteint au bout d'un temps fini avec une probabilité de 1. En effet de tout point de départ on peut exhiber une suite d'états accessibles à la dynamique du réseau et menant à un état d'équilibre : si initialement deux cellules voisines sont simultanément dans l'état 1 elles passent à l'état nul ; puis , l'une après l'autre, les cellules passent à l'état 1 si elles y sont incitées par les cellules V_i et si aucune de leurs voisines n'est déjà à 1.

On souhaite maintenant contrôler la mise à jour de l'image, en considérant que les pixels sont matérialisés par des neurones. On souhaite de plus faire cette transformation sans attendre la convergence du réseau effectuant un choix de l'ensemble T, ce qui nécessiterait un contrôle global du système. Un pixel i sera colorié dès que $t_i = 1$ et $t_j = 0$ pour tout pixel j voisin de i, c'est-à-dire dès que le choix de l'ensemble T sera fixé localement autour du pixel i. Pour chaque pixel i, le neurone x_i associé est donc relié au neurone t_i correspondant, ainsi qu'à tous les neurones t_j, où j est un pixel voisin de i, et l'état d'un tel neurone x_i est

calculé comme suit :

$$x_i = \begin{cases} 1 \quad \text{si} \quad N \,.\, x_i + t_i - \sum_{j \text{ voisin de } i} t_j - \frac{1}{2} > 0 \\ 0 \quad \text{sinon,} \end{cases}$$

où N est égal au nombre maximum de voisins d'un pixel augmenté de 1 ($N = 9$ dans le cas des pixels carrés où nous nous sommes placés). On pourra vérifier que si $x_i = 0$ alors x_i ne passe à l'état 1 que si $t_i = 1$ et si $t_j = 0$ pour tout pixel j voisin de i. De même on a le fait que si $x_i = 1$ alors quelles que soient les valeurs prises par les neurones t_j, x_i reste à l'état 1, c'est-à-dire qu'un pixel déjà colorié le restera quelle que soit l'évolution du réseau de sélection T.

Un pixel i étant colorié dès que $t_i = 1$ et $t_j = 0$ pour tout pixel j voisin de i, il faut vérifier que t_i ne soit égal à 1 que si $V_i(x) = 1$. On ne peut donc plus prendre n'importe quel état initial pour le réseau de sélection T : il est nécessaire qu'initialement tous les t_i soient nuls. On peut maintenant vérifier que si $t_i = 0$ alors t_i ne peut passer à l'état 1 que si $V_i(x) = 1$. Partant d'une situation initiale où tous les t_i sont nuls, l'image x initiale est donc déformée uniquement par ajout de pixel et cette transformation conserve le nombre de composantes connexes. Le rôle du réseau d'inter-connexion entre les neurones non-déterministes t_i est d'assurer que cette transformation aboutisse en un temps fini à une image de l'ensemble M. On dispose alors d'un premier PERCEPTRON reconnaissant la connexité de l'image obtenue et donc de l'image initiale et d'un deuxième PERCEPTRON validant cette réponse en reconnaissant l'aboutissement du processus de transformation.

L'objet de la construction qui vient d'être présentée est de montrer la lacune principale du PERCEPTRON. En envisageant le calcul des cellules d'association comme le calcul d'une fonction, on gagne certes en parallélisme et donc en vitesse de calcul, mais on perd en pouvoir d'expression et en souplesse de mise en œuvre. Introduire des boucles de rétro-action et donc une certaine forme de séquentialité permet d'utiliser le principe (fondamental en informatique et pourtant souvent oublié avec les réseaux de neurones) qui consiste à transformer le problème initial, en veillant à ne pas perdre l'information que l'on veut recueillir, de façon à se ramener à une situation plus simple.

Les avantages de la construction qui vient d'être présentée, sur un système obtenu par apprentissage, est qu'elle permet de construire le système de reconnaissance quelle que soit la taille de l'écran ou la forme des pixels (on peut très bien prendre des pixels hexagonaux à six voisins) et de garantir son bon fonctionnement. Dans le cas d'un système obtenu par apprentissage on ne pourra que constater que le système ne s'est, pour l'instant, jamais trompé et de plus toute modification de l'écran impliquera une refonte complète du système.

3. Le Neocognitron et les réseaux structurés pour la reconnaissance de caractères

Les exemples support de cette section ont été choisis car ils montrent que les réseaux de neurones constituent une technique maintenant suffisamment élaborée pour pouvoir être introduite dans un contexte industriel. Notre objectif n'est pas de les présenter sous leur forme achevée, mais de décrire les principes qui ont guidé leur réalisation. Pour une description plus complète, le lecteur est donc invité a se reporter aux articles d'origine.

Ces systèmes, le NEOCOGNITRON de K. FUKUSHIMA [15], le NEOCOGNITRON avec attention sélective du même auteur [16] ainsi qu'un prototype industriel reprenant les idées de Y. LE CUN [7, 8], sont tous trois des systèmes de reconnaissance de l'écriture manuscrite. Le dernier permet de traiter des caractères isolés et écrits en majuscules ou en minuscules. Il a été développé au Laboratoire d'Électronique de Philips (LEP) [12] dans le cadre d'un projet de *NoteBook*, ordinateur portable dont le clavier est remplacé par un écran et un stylo. Il est actuellement implanté sur la tablette de saisie PAID (Philips Advanced Interactive Display) et peut reconnaître l'ensemble des 94 caractères ASCII. Le réseau de neurones utilisé est fort semblable, par son organisation, au NEOCOGNITRON mais utilise un schéma plus simple présenté par Y. LE CUN. On pourra se reporter à la thèse de H. FRYDLENDER pour en avoir une présentation [13]. Les deux premiers systèmes sont dédiés à la reconnaissance de chiffres manuscrits. L'intérêt du second est de rendre compte d'une attention sélective grâce à un mécanisme de rétro-action : il est capable d'extraire un symbole particulier d'une image composée de plusieurs symboles et de lui associer le signe correspondant. De plus en perturbant le réseau (*i.e.* en modifiant l'état d'équilibre codant la réponse) on peut obtenir après un second cycle de calcul une nouvelle réponse : un autre symbole présent dans l'image. C'est cette propriété d'attention sélective qui fait l'intérêt du système de K. FUKUSHIMA : il est un bon candidat à un système de reconnaissance de l'écriture manuscrite liée.

Outre leur domaine d'application, ces exemples ont pour point commun les principes qui ont mené à leur réalisation : dans un premier temps on choisit une structure du réseau (exprimée par des contraintes d'égalité entre poids de connexion) de façon à rendre compte du problème de la reconnaissance de formes dans sa généralité. Puis, les valeurs des poids de connexion sont fixées par apprentissage, en tenant compte des contraintes structurelles pré-établies, de façon à se rapprocher au mieux de la classification souhaitée.

En effet, lorsqu'on envisage de fabriquer un système de reconnaissance de l'écriture et même lorsqu'on se restreint aux caractères isolés, il devient impossible d'envisager obtenir ce système uniquement par apprentissage tant l'espace des images devient grand dès que l'on prend une discrétisation suffisamment fine

pour que la reconnaissance puisse être fiable. On peut, à la rigueur, espérer reconnaître les quelques exemples qui ont été choisis pour l'apprentissage ainsi que quelques images proches de ces exemples. Cette proximité, au sens de la métrique euclidienne de l'ensemble des images discrétisées considéré comme un espace vectoriel comportant autant de dimensions qu'il y a de pixels sur l'écran récepteur, ne permet de rendre compte que d'une forme très simple de bruit, le bruit blanc. Deux images, différant ne serait-ce que d'une translation, peuvent être si éloignées l'une de l'autre pour cette métrique (sur laquelle repose les algorithmes d'apprentissage) qu'il est vain d'espérer reconnaître l'une en ne présentant que l'autre lors de l'apprentissage des poids de connexion. Et il est tout à fait impensable de présenter toutes les images (qui sont, par exemple, au nombre de 10^{67} pour un écran de 15×15 pixels) sachant que le temps d'apprentissage d'une dizaine de milliers d'images se compte en heures sur une station de travail, à la puissance de calcul déjà considérable. Invoquer l'utilisation de super-calculateurs ou d'ordinateurs parallèles ne change rien au problème.

Il est donc nécessaire d'imposer un lien entre l'ensemble des machines, que l'on explore par apprentissage, et le problème dont on cherche une réalisation. Quelle forme doit donc prendre ce lien? La réponse proposée par K. FUKUSHIMA avec le NEOCOGNITRON [15] ou Y. LE CUN [7] est la suivante. Indépendamment de l'ensemble des formes à reconnaître, la réponse du système doit rester invariante lorsque l'on fait subir certaines transformations à l'image d'entrée. Il est ainsi souhaitable qu'une translation de l'image sur l'écran n'entraîne pas de variation de sortie de la part du système. De même lors de l'étirement d'une boucle, la réponse du réseau doit être pratiquement constante, du moins tant qu'un certain seuil n'est pas franchi. Ces transformations de l'entrée devant laisser invariante la réponse du système caractérise le problème dans sa généralité, dans notre cas la reconnaissance de symbole manuscrit. Si l'on impose ces contraintes à toutes les machines explorées au cours d'un apprentissage, le rôle de l'algorithme d'apprentissage est alors de choisir la machine reflétant le mieux le code particulier que l'on souhaite obtenir. On se trouve ainsi devant un problème d'interpolation mieux posé que précédemment. Il nous reste maintenant à caractériser le problème de la reconnaissance d'un symbole à l'aide d'un ensemble de transformations de l'entrée devant laisser invariante la réponse du système. Il reste aussi à choisir une structure de réseau qui décrive un ensemble de systèmes vérifiant ces contraintes d'invariance.

C'est à ce point que se différencient les trois systèmes. Celui du LEP et le NEOCOGNITRON rendent compte tout deux d'une invariance par rapport à une translation et d'une invariance vis-à-vis des distorsions locales de l'image. Mais le contrôle de cette invariance est plus explicite pour le NEOCOGNITRON. Enfin la version améliorée du NEOCOGNITRON, à l'aide de son mécanisme d'attention sélective, rend compte d'une invariance à une superposition d'images : la réponse à un symbole reconnu comme formant un tout est inchangée lorsqu'on superpose

à cette image un autre symbole. Regardons maintenant les solutions qui ont été retenues pour imposer ces invariances aux réponses du réseau.

3.1. Invariance vis à vis d'une translation

Les trois réseaux présentés sont organisés en couches et la structure plane de l'entrée, ou rétine, est conservée de couches en couches. Oublions pour l'instant le caractère discret de la rétine et de diverses couches et plaçons nous dans un cadre continu. Tous les neurones d'une même couche calculent une même fonction de l'image mais sur une portion restreinte. De plus, à un facteur d'échelle λ près, le support de calcul sur la rétine $S_{T(n)}$, d'un neurone $T(n)$ d'une de ces couches et obtenu par translation du neurone n de cette couche, est le translaté $T(S_n)$ du support S_n du neurone de référence n :

$$S_{T(n)} = \lambda T(S_n) \quad .$$

En d'autres termes, les connections ω_{ij}, d'une couche amont vers une couche aval, sont obtenues par translation :

$$\omega_{T(i),\lambda T(j)} = \omega_{ij} \quad ;$$

ainsi une couche intermédiaire donne la distribution d'une information locale de l'image et constitue ainsi un filtre. De plus, si l'on translate l'image d'entrée sur la rétine, la réponse de chaque couche est elle-même translatée.

Lorsqu'on se replace dans le cadre discret, une translation de l'image d'entrée n'entraîne pas nécessairement une translation de la réponse d'une couche: la réponse de cette couche reste constante si l'amplitude de la translation de l'image est faible. Cet effet est amplifié par les facteurs d'echelle entre couches et par la présence de couches intermédiaires. En jouant sur ces derniers paramètres, on peut ainsi rendre la réponse des cellules de sortie insensible à une translation de l'entrée. Avec le NEOCOGNITRON, K. FUKUSHIMA [15] contrôle plus précisément l'invariance d'une couche vis-à-vis d'une translation de faible amplitude de la couche aval. Chaque couche de traitement est dupliquée: à une couche effectuant l'extraction des caractéristiques d'une image est associée une couche de focalisation qui gomme les petites délocalisations d'une caractéristique. Ce sont les informations ainsi corrigées qui sont transmises aux couches de traitement amont.

Ainsi lorsqu'on considère le système, de couche en couche et de l'amont vers l'aval, les caractéristiques extraites de l'image sont de plus en plus globales et de moins en moins localisées.

3.2. Invariance vis-à-vis d'une distorsion locale

La structure donnée au réseau de façon à rendre ses réponses insensibles à une translation de l'image sur la rétine, lui confère aussi une certaine insensibilité aux distorsions locales de l'image. En effet, l'insensibilité vis-à-vis d'une translation de la réponse d'un neurone chargé de l'extraction d'une caractéristique est locale au support de calcul de ce neurone. Une distorsion, qui correspond à la translation de portions isolées de l'image et dans des directions différentes, laissent donc invariante la réponse de toute cellule telle que la variation de l'image induite sur leur support de calcul est une translation. Par suite, les neurones qui utilisent uniquement la réponse de ces dernières cellules pour leur calcul, ont une réponse insensible à cette distorsion de l'image.

Ainsi, le fait que les caractéristiques extraites de l'image sont de plus en plus globales et de moins en moins localisées lorsqu'on considère le système de couche en couche et de l'amont vers l'aval, confère au système une certaine insensibilité aux distorsions locales de l'image. Le principe reste néanmoins très empirique.

3.3. Invariance vis-à-vis d'une superposition

Schématiquement, le NEOCOGNITRON avec attention sélective est composé de deux réseaux entremêlés. D'une manière imagée, le premier réseau propose des solutions que le second sélectionne. Plus precisément, le premier réseau, dont le flot d'information circule des cellules réceptrices vers les cellules codant l'interprétation de l'image, permet de dépister plusieurs réponses possibles, suivant les principes précédemment exposés. Le second réseau, dont le flot d'information parcourt le système en sens inverse, sélectionne, parmi les cellules du premier réseau, celles ayant conduit à l'interprétation la plus sollicitée.

Le principe de cette sélection est le suivant. Dans un premier temps, le rôle du second réseau est de reconstruire une image à partir d'une interprétation. Alors que le premier réseau est un système de reconnaissance, le second est donc un système de synthèse de caractères. Aussi, ce réseau de sélection par reconstruction est construit avec la même structure que le réseau de détection : même nombre de neurones, même graphe de connections (sauf que ces dernières sont dirigées dans le sens contraire) et même interprétation en termes de caractéristiques de l'image (d'entrée pour l'un, de sortie pour l'autre). Mais les solutions à une telle synthèse de caractères sont nombreuses. Dans un second temps, les deux réseaux sont donc connectés de sorte que la reconstruction s'appuie sur les indices détectés par le réseau de reconnaissance. Plus précisément, le réseau de reconstruction fait le choix d'un symbole à reconstruire parmi tous les symboles dont le réseau de détection signale la présence possible. Puis seuls les indices détectés au cours de la première phase et nécessaires à la reconnaissances de ce caractère, sont retenus par le réseau de reconstruction. De plus, les indices manquant sont ajoutés. L'image de sortie de ce réseau donne alors l'image du symbole reconnu. Enfin, on ajoute

des connexions du réseau de reconstruction vers le réseau de détection, de telle sorte que soient inhibés les neurones détectant des caractéristiques non utiles à la reconstruction et que soient activés les neurones n'ayant pas détecté un indice nécessaire à la reconstruction. La coopération entre les fonctions de détection et de sélection est ainsi achevée.

En réalité, il n'y a donc pas à proprement parler une phase de détection suivie d'une phase de reconstruction : tous les neurones du système présenté par K. FUKUSHIMA évoluent simultanément. Aussi, l'interprétation de cette machine repose sur ses points d'équilibre : en un tel point une partie de l'image est sélectionnée et une interprétation en est donnée ; le réseau de reconnaissance assure que l'interprétation découle du symbole extrait ; quant à lui, le réseau de sélection assure que le symbole peut être reconstruit à partir de son interprétation.

En pratique, la mise en œuvre effective de ce principe est délicate. Pour cela, K. FUKUSHIMA introduit des mécanismes d'interaction entre cellules peu utilisés habituellement dans le cadre des réseaux de neurones (contrôle du gain de sortie modulant la réponse d'un neurone, diminution progressive du seuil d'activation d'un neurone). On retiendra de ce système la possibilité de reconnaître un symbole malgré la présence de signes étrangers à ce symbole. De plus, le choix effectif d'un caractère parmi tous ceux présents sur l'image dépend de l'état initial du réseau. On peut donc basculer d'un choix vers un autre en perturbant l'état d'équilibre du système. Il est ainsi possible d'envisager d'utiliser une telle machine comme point de départ pour un système de reconnaissance de l'écriture manuscrite liée. En effet, la difficulté de ce problème vient essentiellement du fait que l'on doit isoler les différents symboles porteurs de sens et que l'on doit pouvoir aussi envisager des interprétations multiples pour un même signe et basculer de l'une à l'autre de ces interprétations, en fonction du contexte où ce symbole a été extrait.

conclusion

Le premier point que nous retiendrons de ce chapitre est celui de l'apport des boucles de rétro-action. En introduisant ce mécanisme dans le PERCEPTRON on arrive ainsi à exhiber un système reconnaissant le caractère connexe d'une image, problème qui a été montré comme insoluble dans le cadre d'origine. De même l'usage de la rétro-action par K. FUKUSHIMA lors de la réalisation du NEOCOGNITRON avec attention sélective, montre l'intérêt de boucles dans le graphe de connexions d'un réseau dès lors que l'on souhaite pouvoir extraire une partie cohérente d'un ensemble d'informations ou que l'on souhaite pouvoir obtenir *a priori* plusieurs réponses de manière à pouvoir faire un choix ultérieur en fonction du contexte.

Par ailleurs, les exemples présentés mettent bien en évidence le fait que les al-

gorithmes d'apprentissage n'éludent pas la nécessité d'une phase de construction. Entre autres, on retiendra du PERCEPTRON la nécessité d'une bonne complémentarité entre cette phase de construction et celle d'apprentissage lors de la synthèse d'un réseau de neurones. Avant d'aborder toute amélioration par apprentissage, il nous faut cerner une structure commune à tout les réseaux que l'on se propose d'explorer. Les techniques de synthèse d'un réseau de HOPFIELD nous donnent un premier moyen pour construire une telle ébauche de réseau. Une seconde approche est d'imposer une invariance de la réponse du réseau vis-à-vis de diverses transformations de son entrée. On obtient alors un graphe de connexions entre un ensemble de neurones ainsi qu'un certain nombre de liens que doivent vérifier les poids de ses connexions. Il reste alors à déterminer un nombre restreint de paramètres de contrôle.

Nous pouvons donc maintenant aborder le problème de l'apprentissage proprement dit: comment explorer un ensemble de réseaux ayant une structure commune, chacun étant particularisé par un choix des paramètres de contrôle, à la recherche du choix nous convenant le mieux.

CHAPITRE III

Principes des algorithmes d'apprentissage

Introduction

Un des attraits principaux du formalisme des réseaux de neurones dans le cadre d'une utilisation technologique provient de la possibilité de modifier la fonction d'un réseau en contrôlant ses coefficients synaptiques. Peut-on alors adapter le comportement d'un réseau de neurones à une fonction particulière en le façonnant par retouches successives? On appelle algorithme d'apprentissage une méthode permettant de progresser dans un ensemble de systèmes de calcul à la recherche d'une machine satisfaisant au mieux un certain nombre de critères d'évaluation. La trame d'un tel algorithme consiste à évaluer la réponse du système à adapter sur un exemple puis de modifier les constituants de ce système de façon à améliorer sa réponse. Dans le cadre des réseaux de neurones, l'espace des machines explorées est spécifié (outre le type des neurones et leur dynamique d'évolution) par un graphe de connexion et est constitué de tous les réseaux correspondant aux différentes valeurs des poids de connexion. Partant d'une instance particulière de ces poids, le but de l'algorithme est alors d'améliorer la réponse du réseau en modulant ses coefficients de contrôle.

La première difficulté rencontrée est d'exprimer le lien entre la réponse du système à améliorer et les paramètres qui le définissent, de façon à évaluer la direction suivant laquelle il est préférable d'évoluer. Ce lien étant établi, il apparaît que l'information donnée par un seul exemple n'est pas nécessairement pertinente. Il se peut, en effet, que de nombreuses directions d'amélioration n'arrivent pas à être départagées à la seule vue de cet exemple, toutes ces directions apportant sur ce point une amélioration similaire. Si l'on essaye, en réponse à ce problème, de tenir compte simultanément de plusieurs exemples, il se peut alors que l'on obtienne des informations contradictoires, que chaque exemple indique des directions d'amélioration opposées et qu'aucune direction ne permette au système d'être simultanément amélioré sur tous les points. Le deuxième niveau de difficulté est donc de tenir compte de tous les exemples dont on dispose pour naviguer dans l'espace des machines. Un algorithme d'apprentissage devra rendre compte de ces deux niveaux de difficulté: calcul de la direction d'évolution permettant

de tenir compte d'un exemple puis choix d'un compromis pour tenir compte de tous les exemples.

Le troisième niveau de difficulté ne se situe pas, à proprement parler, sur le plan de l'algorithme, mais plutôt sur le plan de sa mise en œuvre. En effet, en fixant un graphe de connexion, et par là, l'espace des machines visité par l'algorithme d'apprentissage, on peut très bien, par inadvertance, ne pas y inclure la machine que l'on souhaite obtenir ou, au contraire, se donner un espace de recherche si grand que les quelques exemples dont on dispose ne sont plus pertinents. Dans les deux cas notre recherche ne peut aboutir. Dans un espace trop confiné, on ne peut qu'obtenir une machine réalisant un vague compromis et avec trop de liberté on risque d'obtenir une machine au comportement fort éloigné de celui souhaité aux points non spécifiés comme exemple. Le choix de l'espace de recherche doit être fait en tenant compte des limites des algorithmes d'apprentissage et du problème spécifique que l'on souhaite résoudre.

Si actuellement ces trois niveaux de difficulté sont pris en compte par la communauté de recherche sur les réseaux de neurones, cette prise de conscience s'est faite progressivement. En 1949, D.O. HEBB [17] proposa un principe pouvant guider les modifications des systèmes nerveux. Ce principe reste essentiellement au premier niveau : celui du lien entre une situation et la modification des coefficients synaptiques qu'elle entraîne. Selon l'hypothèse de HEBB, une forte corrélation temporelle entre les activités pré et post-synaptique entraînerait une augmentation de l'efficacité de la synapse concernée. Cette idée peut être utilisée dans un cadre technologique pour la conception de mémoires associatives pouvant compléter une information partielle. Le fil directeur de cette construction est de coder l'ensemble des mots à retenir avec un certain nombre de chiffres binaires et d'associer une cellule à chacun de ces chiffres. Le coefficient de contrôle entre deux cellules est alors pris d'autant plus fort que les valeurs des chiffres associés sont corrélées parmi l'ensemble des mots. Pour avoir une étude approfondie de cette démarche et des difficultés qu'elle engendre, on pourra se reporter au livre de Y. KAMP et M. HASLER [31].

Le principe de HEBB est aussi à l'origine de l'algorithme d'apprentissage développé par F. ROSENBLATT pour le PERCEPTRON [45]. L'objectif de cet algorithme est très simple : il ne doit améliorer que les connexions afférentes aux neurones réponses et dont on connaît donc le comportement souhaité. F. ROSENBLATT arrive ainsi à rendre compte des trois niveaux de difficulté d'un algorithme d'apprentissage. Un lien est établi entre l'erreur commise sur un exemple et les corrections à effectuer, puis ce lien est réutilisé d'exemple en exemple. Le critère de convergence permet, lui, de choisir un cadre correct de mise en œuvre. M. MINSKY et S. PAPERT [35] ont néanmoins objecté qu'il est difficile, voire impossible, de construire a priori et de façon méthodique un ensemble de cellules intermédiaires autorisant la mise en œuvre de cet algorithme d'apprentissage.

En réponse à cette objection, on trouve l'algorithme de rétro-propagation du gradient de l'erreur présenté par Y. LE CUN [6, 32] et D. PARKER [38] et dont D. RUMELHART, G. HINTON et R. WILLIAMS [48, 47] ont fait une large diffusion. Sous sa forme d'origine, cet algorithme s'applique aux problèmes de classification et à des réseaux dits multi-couches qui se trouvent dans la lignée du PERCEPTRON. Ces réseaux ne comportent pas de boucle et une classification est conçue comme une cascade de classifications élémentaires. Plus récemment, L. B. ALMEIDA [3, 4] ainsi que F. J. PINEDA [40, 41, 42, 43] ont proposé une généralisation permettant la mise en œuvre de cet algorithme avec des réseaux comportant des boucles de rétro-action. Le principe est de mesurer l'erreur commise par le réseau pour une certaine instance de ses paramètres de contrôle et de faire varier ces derniers de manière à faire diminuer au maximum l'erreur commise. De façon à pouvoir exprimer de manière simple la variation des réponses du réseau induite par une modification des poids synaptiques, ces auteurs font intervenir un espace continu d'états qui permet l'utilisation des outils du calcul différentiel.

Le but du présent chapitre est d'évaluer la puissance d'adaptation que l'on peut attendre de ces méthodes. L'exposé est centré sur l'algorithme de rétro-propagation du gradient de L. B. ALMEIDA et F. J. PINEDA, dont la généralité permet de capter tous les aspects du problème. Toutefois nous allons le présenter avec plus de recul que dans les articles de références [3, 4, 40, 41, 42, 43]. L'intérêt de cette présentation est de s'éloigner de la structure choisie pour le réseau (fonction de seuillage, rôle des coefficients synaptiques, dynamique de calcul). Ceci permet de bien différencier les différents niveaux de l'algorithme et de faire ressortir ce qui est généralisable à l'adaptation de tout système paramétré.

1. Principes et cadre d'application

La démarche suivie pour la construction d'un réseau de neurones à l'aide d'un algorithme d'apprentissage est la suivante. Une première étape a pour but de se fixer un espace de machines parmi lesquelles on pourra rechercher celle nous convenant le mieux. Pour cela on choisit un réseau de neurones, défini par son graphe de connexion et la fonction de seuillage de chaque neurone, puis on fixe un mécanisme faisant évoluer l'état du réseau à la recherche d'un point d'équilibre et enfin on se choisit un mode d'utilisation du système dynamique ainsi obtenu en répondant aux questions suivantes : Comment choisir une entrée du système à partir d'une donnée du problème à résoudre ? Comment un état du réseau est-il décodé en termes de réponse ? A chaque choix des coefficients synaptiques correspond alors une machine et c'est dans l'ensemble de toutes ces machines que l'on se propose de trouver celle résolvant le problème à résoudre.

Il nous faut maintenant pouvoir évaluer une machine particulière. On effectue pour cela une suite d'expériences permettant d'observer le comportement du réseau. Une expérience consiste à présenter un exemple d'entrée au système dont la réponse est recueillie en fin d'évolution. Cette réponse effective est alors comparée à celle que l'on souhaitait obtenir. On résume ainsi l'évaluation du réseau par la valeur prise par une fonction dépendant des réponses obtenues sur un certain nombre d'entrées. Cette fonction, d'évaluation ou d'erreur, est choisie à valeurs dans $\mathbb{R}^+$, d'autant plus faibles que les réponses correspondantes sont satisfaisantes. Le problème de l'apprentissage consiste alors à trouver, parmi l'ensemble des machines choisies, une machine particulière minimisant le critère d'évaluation fixé. La difficulté provient du fait que la valeur prise par la fonction d'évaluation, pour un certain choix des coefficients synaptiques, ne peut être connue qu'expérimentalement. Regardons donc de plus près le lien entre les paramètres définissant un réseau et la réponse donnée par ce réseau sur une entrée.

Suivant la définition que nous avons prise pour un réseau de n neurones, la réponse à une entrée e de E est un vecteur x de $\mathbb{R}^n$ vérifiant une équation de la forme :

$$\text{pour toute cellule } i, \quad x_i = f\Big(\sum_{j \text{ tq } (i,j)\in S} \omega_{ij} \,.\, x_j + \omega_i(e)\Big) \quad .$$

Plaçons-nous dans un cadre permettant de simplifier l'expression de ce lien : l'ensemble des paramètres du réseau est résumé par un vecteur ω de coefficients de contrôle appartenant à $\mathbb{R}^m$. La réponse x du réseau obtenu avec ces coefficients pour une certaine entrée vérifie alors une équation implicite de la forme :

$$G(x,\omega) = 0 \quad ,$$

où G est une application différentiable de $\mathbb{R}^n \times \mathbb{R}^m$ dans $\mathbb{R}^n$. (Pour que G soit différentiable il suffit de considérer une fonction de seuillage qui le soit.)

Étant donnés x^o et ω^o solutions de l'équation $G(x,\omega) = 0$ (c'est-à-dire que x^0 est la réponse que l'on a obtenue avec les coefficients ω^o), d'après le théorème des fonctions implicites on a le résultat suivant. Si l'application partielle qui à x associe $G(x,\omega^o)$ admet une différentielle en x^o qui est inversible (notée $G'_x(x^o,\omega^o)$), alors l'équation $G(x,\omega) = 0$ est localement équivalente à une équation $x = g(\omega)$, où g est différentiable en ω^o, et on a :

$$\begin{aligned} x'_\omega(\omega^o) &= g'(\omega^o) \\ &= -G'_x(x^o,\omega^o)^{-1} \circ G'_\omega(x^o,\omega^o) \quad . \end{aligned}$$

Ce qui est à retenir de ce résultat est qu'il n'est pas nécessaire de connaître par quel moyen le réseau a obtenu une réponse x^o avec les coefficients ω^o pour pouvoir contrôler les modifications apportées à cette réponse en modulant les coefficients de contrôle. Le lien entre la réponse et les coefficients que nous avons pris comme définition des réseaux de neurones nous suffit malgré le fait qu'il soit implicite.

De façon à pouvoir effectivement utiliser ce lien pour extrapoler le comportement du réseau autour d'un point correspondant à une expérience, il est nécessaire que ce lien soit valable sur un voisinage du point d'équilibre atteint. C'est-à-dire que si la dynamique choisie permet d'atteindre un certain point d'équilibre x^o avec les coefficients ω^o lors d'une expérience, alors, sur un voisinage du couple (x^o, ω^o), si un couple (x, ω) vérifie l'équation $G(x, \omega) = 0$ alors le point x est le point d'équilibre atteint par le réseau, contrôlé par les coefficients ω et mis dans les mêmes conditions d'utilisation. En pratique, cette propriété ne pourra pas être assurée en tous points (entre autres, lorsque $G'_x(x^o, \omega^o)$ n'est pas inversible) mais sera vraie presque partout et pourra donc servir de support au calcul de l'amélioration apportée au fonctionnement du réseau par une modification des coefficients de contrôle.

Lors de la spécification d'un algorithme d'apprentissage, il est donc suffisant de résumer toute la construction préalable par deux points : d'une part, le lien entre l'état codant la réponse du système et les paramètres de contrôle, et d'autre part, une quantification de la qualité d'une réponse. Il est inutile, à ce moment, de connaître comment ce lien entre réponse et paramètres est assuré par le réseau. Il n'est pas plus nécessaire de savoir comment une réponse est codée : pour la spécification de l'algorithme seule importe la donnée d'une fonction d'évaluation. Ce point important n'a pas toujours été bien compris. Ainsi l'algorithme de rétro-propagation classique qui a été spécifié pour des réseaux sans boucle de rétro-action est formellement identique à celui présenté par L. B. ALMEIDA et F. J. PINEDA pour des réseaux comportant de telles boucles. En effet, si dans le premier cas le calcul effectué par le réseau est une simple phase de propagation de signaux alors que dans le deuxième cas une phase de recherche d'équilibre est nécessaire, le lien entre la réponse finale et les coefficients synaptiques s'exprime dans les deux cas par une même équation. Le calcul du choix d'une direction de modification est donc formellement le même pour les deux types de réseaux.

Disposant, presque partout, d'un lien $x = g(\omega)$ entre la réponse x du réseau et les coefficients de contrôle ω, ainsi que d'un critère d'évaluation $E(x)$ (la situation est résumée par la figure III.8), on peut alors calculer la direction d'évolution des poids permettant de diminuer au maximum cette erreur et donc de se rapprocher au maximum d'un système nous satisfaisant. Cette direction est l'opposé du gradient au point ω de la fonction composée $E \circ g$. En suivant cette direction $\Delta\omega$, on obtient alors un système contrôlé par les coefficients $(\omega + \lambda . \Delta\omega)$ qui donnent une réponse plus satisfaisante si le pas λ est choisi suffisamment petit. En itérant ce processus on se déplace donc dans l'espace des machines jusqu'à l'obtention d'un système localement optimal : aucune direction d'évolution ne permet de l'améliorer.

2. L'algorithme du gradient stochastique

On désire améliorer le comportement du réseau sur l'ensemble des entrées (ou sur tous les exemples dont on dispose). Une première solution est de mesurer une qualité globale du réseau sur toutes les entrées possibles en faisant une suite d'expériences, puis de calculer le gradient de cette quantité globale. Le plus simple est de prendre comme mesure d'erreur globale la somme des erreurs mesurées sur chaque exemple. Le gradient total est alors égal à la somme des gradients partiels. Le gros inconvénient de cette méthode est qu'il faut calculer la réponse du réseau sur tous les exemples pour pouvoir calculer une direction de modification des poids. Or l'ensemble des entrées peut avoir une taille gigantesque dans les applications envisagées. De plus, comment choisir le pas λ permettant de contrôler la distance sur laquelle nous allons suivre la direction donnée par le gradient de l'erreur?

Une seconde solution pour améliorer le comportement du réseau sur l'ensemble des entrées est celle du gradient stochastique. Le principe en est le suivant. Les exemples sont pris au hasard (indépendamment des exemples déjà tirés) et présentés successivement au système. Celui-ci est modifié à *chaque* exemple dans la direction diminuant au maximum l'erreur commise sur cet exemple:

$$\omega^{t+1} = \omega^t - \lambda^t \operatorname{grad} E_\omega(e^t, \omega^t)$$

où chaque λ^t est un coefficient positif et où $E(e,\omega)$ désigne l'erreur commise sur l'entrée e avec les coefficients ω. On ne cherche donc plus à évaluer directement

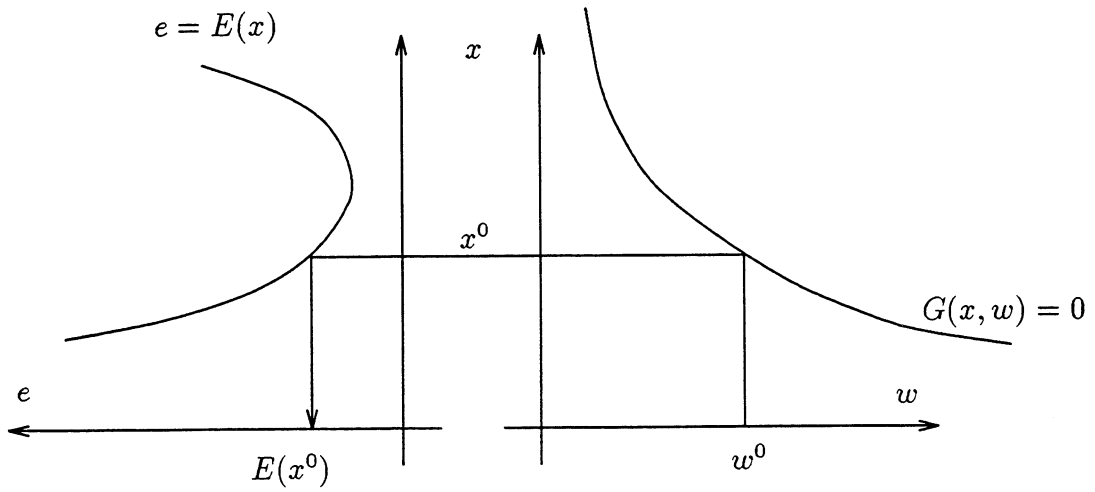

FIG. III.8 - . Lien entre les paramètres de contrôle ω, la réponse x du réseau et l'erreur induite e.

une qualité globale du réseau mais à tenir compte au fur et à mesure de chaque expérience. Un tel processus converge-t-il? Étant donné une distribution de probabilité p sur l'ensemble des entrées, reflétant le choix d'un exemple à chaque pas de l'algorithme, remarquons les trois points suivants:

i) Si l'on veut que le processus converge indépendamment de la fonction E (quitte à obtenir un système médiocre), alors la suite des λ^t doit converger vers 0. En effet, il n'existe pas forcément un choix des paramètres ω pour lequel les différents gradients relatifs aux différents exemples soient simultanément nuls. Si la suite des λ^t ne converge pas vers 0, alors aucun point ω de l'espace des paramètres ne saurait être un candidat de limite.

ii) Un ensemble de poids ω^* vérifiant:

$$\sum_{e \in E} p(e) \,.\, \mathrm{grad}_\omega E(e, \omega^*) = 0$$

c'est-à-dire

$$\mathrm{grad}_\omega \Big(\sum_{e \in E} p(e) \,.\, E(e, \omega^*) \Big) = 0$$

est un bon candidat de limite: la moyenne des déplacements y est nulle. Ceci est intéressant, car si l'on utilise maintenant le système contrôlé par les coefficients ω^* dans une situation où une entrée e est présentée avec une fréquence $p(e)$ alors la quantité localement minimisée est l'erreur moyenne commise: ω^* est un minimum local de l'espérance de l'erreur.

iii) Enfin, la convergence vers 0 de la suite des λ^t doit être suffisamment lente. En effet, dans le cas extrême où la fonction d'erreur ne dépend pas de l'exemple présenté et est égale à:

$$E(e, w) = \frac{1}{2} \quad \|\omega - \omega^*\|^2 \quad .$$

La seule solution à notre problème est ainsi représentée par le choix ω^* des coefficients du réseau. On a alors:

$$\mathrm{grad}_\omega E = \omega - \omega^* \quad ;$$

d'où:

$$\omega^{t+1} - \omega^* = (1 - \lambda^t) \,.\, (\omega^t - \omega^*) \quad .$$

Comme on impose déjà à la suite des λ^t de converger vers 0, on peut prendre $\lambda^t < 1$ et l'on a alors:

$$\|\omega^{t+1} - \omega^*\| = \prod_{i=0}^{t} (1 - \lambda^i) \,.\, \|\omega^t - \omega^*\| \quad .$$

Or, la quantité

$$\prod_{i=0}^{t}(1-\lambda^i)$$

ne tend vers zéro que si et seulement si la série $\sum \lambda^t$ diverge. Une condition nécessaire pour que le processus d'apprentissage converge est donc que la série $\sum \lambda^t$ diverge. En d'autres termes, l'algorithme doit pouvoir parcourir une distance arbitrairement grande dans l'espace des coefficients de contrôle.

3. Expression du gradient de l'erreur

Pour achever la spécification de l'algorithme d'apprentissage du gradient stochastique, il nous faut maintenant calculer le gradient grad $E(e,\omega)$ de l'erreur commise sur une entrée e par rapport à une variation des coefficients ω. Avant de préciser ce calcul dans un cadre plus proche de la structure en réseau et retrouver l'algorithme de rétro-propagation de Y. LE CUN ainsi que deux algorithmes proposés par F. J. PINEDA, exprimons ce gradient dans ce cadre général. Ceci permet, d'une part, d'éviter la longue suite de calcul développée par F. J. PINEDA et Y. LE CUN qui expriment le problème à un niveau trop proche de la forme finale des algorithmes obtenus. Par ailleurs, ceci permet de faire ressortir le côté très général de l'algorithme de rétro-propagation du gradient.

On se place dans le cadre suivant. Un réseau constitué de n cellules est contrôlé par un vecteur ω de m coefficients. L'évolution du réseau de neurones spécifié par une instanciation ω^o du vecteur de contrôle ayant abouti à un état d'équilibre x^o, le lien entre un point d'équilibre x du réseau et les coefficients de contrôle ω est caractérisé sur un voisinage du couple (x^o,ω^o) par l'équation :

$$G(x,\omega)=0 \quad .$$

La qualité de la réponse du réseau est mesurée par une fonction E définie sur un voisinage du point x^o.

On suppose que les fonctions G et E, respectivement de $\mathbb{R}^n \times \mathbb{R}^m$ dans $\mathbb{R}^n$ et de $\mathbb{R}^n$ dans $\mathbb{R}$, sont différentiables au point x^o obtenu avec les paramètres ω^o. On cherche l'expression du gradient de E, considéré comme fonction des paramètres de contrôle ω, au point ω^o caractérisant le système de départ que l'on désire améliorer. Étant donnés x^o et ω^o vérifiant $G(x^o,\omega^o)=0$, si l'application partielle qui à x associe $G(x,\omega^o)$ admet une différentielle en x^o qui est inversible, alors, d'après le théorème des fonctions implicites, l'équation $G(x,\omega)=0$ est localement équivalente à une équation $x=g(\omega)$, où g est différentiable en ω^o, et on a :

$$x'_\omega(\omega^o) = -G'_x(x^o,\omega^o)^{-1} \circ G'_\omega(x^o,\omega^o) \quad .$$

Par composition, on obtient la différentielle de E par rapport à ω :

$$\begin{aligned} {E'}_{\omega}(\omega^o) &= {E'}_{x}(x^o) \circ {x'}_{\omega}(\omega^o) \\ &= -{E'}_{x}(x^o) \circ {G'}_{x}(x^o, \omega^o)^{-1} \circ {G'}_{\omega}(x^o, \omega^o) \end{aligned}$$

et en transposant, le gradient recherché :

$$\begin{aligned} \mathrm{grad}_{\omega} E(\omega^o) &= {E'}_{\omega}(\omega^o)^T \\ &= -{G'}_{\omega}(x^o, \omega^o)^T \,.\, z \quad , \end{aligned}$$

avec :

$$ {G'}_{x}(x^o, \omega^o)^T \,.\, z - \mathrm{grad}_{x}(E) = 0 \quad . $$

L'intérêt de faire intervenir un vecteur intermédiaire est d'éviter le calcul de l'inverse de la différentielle partielle ${G'}_{x}(x^o, \omega^o)$. On calculera directement le vecteur y, image du gradient de E par rapport à x par l'inverse de la transposée de la différentielle ${G'}_{x}(x^o, \omega^o)$, à l'aide d'une méthode itérative faisant intervenir un système dynamique adjoint. On notera que ce vecteur intermédiaire a la même dimension que les vecteurs d'état du système.

Dans le cas où la fonction d'erreur E dépend non seulement des variables d'état x mais aussi explicitement des coefficients de contrôle ω, la démarche est identique. Après avoir exprimé localement le lieu des points d'équilibre x en fonction des paramètres ω :

$$ x = g(\omega) \quad , $$

on obtient une expression F de l'erreur E ne dépendant explicitement que des coefficients ω :

$$ E(x, \omega) = E(g(\omega), \omega) = F(\omega) \quad , $$

puis l'expression du gradient de l'erreur par rapport à ω :

$$ \mathrm{grad}_{\omega}(F) = {x'}_{\omega}^{T} \,.\, \mathrm{grad}_{x}(E) + \mathrm{grad}_{\omega}(E) \quad , $$

ou encore :

$$ \mathrm{grad}_{\omega}(F) = -{G'}_{\omega}^{T} \,.\, z + \mathrm{grad}_{\omega}(E) \quad , $$

avec :

$$ {G'}_{x}^{T} \,.\, z - \mathrm{grad}_{x}(E) = 0 \quad . $$

On obtient donc un gradient similaire au premier cas et, à la variation d'erreur induite indirectement par la variation du point d'équilibre, il suffit d'ajouter un terme rendant compte de la variation d'erreur directement induite par la variation des coefficients du réseau.

Une étape de l'algorithme d'apprentissage par la méthode du gradient stochastique est donc la suivante :

Partant d'un ensemble de coefficients de contrôle ω^t et d'un exemple d'entrée e^t,

- on recherche d'abord un vecteur de réponses x^t associé à l'entrée e^t et qui vérifie donc l'équation $G(x^t, \omega^t)$ caractérisant les réponses du réseau ;

- on calcule alors le gradient $\text{grad}_x(E)$ de l'erreur commise sur cette entrée e^t par rapport au vecteur de réponses au point obtenu x^t ;

- on cherche alors un vecteur z solution de l'équation :

$$G'_x(x^t, \omega^t)^T \,.\, z - \text{grad}_x(E) = 0 \quad ;$$

- on calcule alors le gradient de l'erreur mais cette fois-ci dans l'espace des coefficients de contrôle :

$$\text{grad}_\omega E(\omega^t) = -G'_\omega(x^t, \omega^t)^T \,.\, z \quad ;$$

- on prend finalement comme nouveau vecteur de contrôle le vecteur :

$$\omega^{t+1} = \omega^t - \lambda^t \,\text{grad}_\omega E(\omega^t) \quad .$$

Lorsqu'on exprime sous cette forme le calcul de la direction à suivre, dans l'espace des coefficients de contrôle, pour améliorer le résultat d'une expérience, on fait ressortir le fait que l'algorithme de rétro-propagation du gradient n'est qu'une reformalisation, dans un cadre spécifique, de techniques plus anciennes et utilisées pour l'identification de paramètres régissant un processus. On retrouve, par exemple, cet usage d'un système dynamique adjoint en météorologie, où le problème est de trouver les conditions initiales du système dynamique destiné à simuler l'évolution des conditions atmosphériques. Le système dynamique adjoint permet de tenir compte des mesures faites sur un laps de temps d'observation et d'extrapoler ainsi l'évolution future.

3.1. Cas des réseaux en couches

Dans le cas des réseaux multi-couches cet algorithme d'apprentissage est connu sous le nom de l'algorithme de rétro-propagation du gradient de l'erreur. Il a été présenté par Y. LE CUN [6, 32] et D. PARKER [38]. Une large diffusion en a été faite par D. RUMELHART, G. HINTON et R. WILLIAMS [48, 47].

Dans le cas des réseaux en couches, le lien G entre réponses, poids et entrées

est :

$$G_i(x,\omega) = f(\sum_{j<i} \omega_{ij} x_j + e_i) - x_i = 0 \quad ,$$

si les neurones sont numérotés de façon à ce que la matrice de connexion soit triangulaire inférieure et à diagonale nulle. En notant u_i la quantité $\sum_{j<i} \omega_{ij} x_j + e_i$, on a donc :

$$\frac{\partial G_k}{\partial \omega_{ij}} = \begin{cases} f'(u_k) x_j & \text{si } i = k \\ 0 & \text{sinon} \end{cases}$$

et

$$\frac{\partial G_i}{\partial x_j} = \begin{cases} f'(u_i)\omega_{ij} & \text{si } j < i \\ -1 & \text{si } i = j \\ 0 & \text{sinon} \end{cases}$$

Dans un premier temps on doit trouver un vecteur z vérifiant :

$$G'_x{}^T . z - \text{grad}_x(E) = 0 \quad ;$$

c'est-à-dire :

$$\sum_j \frac{\partial G_j}{\partial x_i} z_j = \frac{\partial E}{\partial x_i} \quad ;$$

ou encore, en tenant compte de la forme particulière de G'_x :

$$z_i = -\frac{\partial E}{\partial x_i} + \sum_{j>i} f(u_j) z_j \omega_{ji} \quad .$$

Ce vecteur peut donc, lui aussi, être calculé directement de couche en couche. Il faut seulement commencer par les cellules de la couche de sortie.

Le gradient recherché est alors le vecteur de coordonnée $\Delta\omega_{ij}$ vérifiant :

$$\Delta\omega = -G'_\omega{}^T . z \quad ;$$

c'est-à-dire :

$$\Delta\omega_{ij} = -\sum_k \frac{\partial G_k}{\partial \omega_{ij}} z_k \quad ;$$

ou encore, en tenant compte de la forme particulière de G'_ω :

$$\Delta\omega_{ij} = -f'(u_i) x_j z_i \quad .$$

En pratique, au vecteur z on préférera le vecteur y défini par :

$$y_i = -f'(u_i) z_i \quad .$$

Dans un premier temps on calculera donc d'abord le vecteur y avec :

$$y_i = f'(u_i).(\frac{\partial E}{\partial x_i} + \sum_{j>i} y_j \omega_{ji}) \quad ,$$

puis les composantes du gradient recherché :

$$\Delta\omega_{ij} = x_j y_i \quad .$$

L'algorithme de rétro-propagation du gradient pour un réseau multi-couches est donc le suivant, les neurones étant numérotés dans un ordre respectant les couches :

i) du premier neurone jusqu'au dernier, calculer :

$$u_i = \sum_{j<i} \omega_{ij} x_j + e_i \quad \text{et} \quad x_i = f(u_i) \quad .$$

ii) du dernier neurone jusqu'au premier, calculer :

$$c_i = \frac{\partial E}{\partial x_i} \quad \text{et} \quad y_i = f'(u_i)(c_i + \sum_{j>i} y_j \omega_{ji}) \quad .$$

iii) pour toute connexion d'une cellule j vers un neurone i augmenter le coefficient synaptique ω_{ij} de la quantité :

$$-\lambda x_j y_i \quad .$$

C'est la deuxième phase de cet algorithme qui lui donne son nom de rétro-propagation du gradient : un gradient est calculé sur les neurones de sortie, puis cette information est rétro-propagée et transformée dans le réseau de la couche de sortie vers celle d'entrée.

3.2. Cas des réseaux récursifs

La particularité des réseaux en couches nous a permis de résoudre au préalable le système :

$$\sum_j \frac{\partial G_j}{\partial x_i} z_j = \frac{\partial E}{\partial x_i}$$

et de calculer ainsi directement le vecteur intermédiaire z (ou y). F. PINEDA et L. ALMEIDA ont, depuis, généralisé cet algorithme aux réseaux quelconques [40, 41, 3, 4] en faisant remarquer que ce vecteur intermédiaire z peut être calculé par une méthode itérative. Dans le cas d'un réseau de neurones comportant des boucles on procède donc comme suit pour le calcul du gradient de l'erreur dans l'espace des poids.

A partir d'un exemple de référence on obtient avec un choix ω des coefficients de contrôle une réponse x vérifiant :

$$G_i(x,\omega) = f(\sum_j \omega_{ij} x_j + e_i) - x_i = 0 \quad .$$

En notant u_i la quantité $\sum_j \omega_{ij} x_j + e_i$, on a toujours :

$$\frac{\partial G_k}{\partial \omega_{ij}} = \begin{cases} f'(u_k) x_j & \text{si } i = k \\ 0 & \text{sinon} \end{cases}$$

et

$$\frac{\partial G_i}{\partial x_j} = \begin{cases} f'(u_i)\omega_{ij} & \text{si } i \neq j \\ f'(u_i)\omega_{ij} - 1 & \text{si } i = j \end{cases}$$

Il nous faut alors chercher un vecteur z vérifiant :

$$G'_x{}^T . z - \text{grad}_x(E) = 0 \quad ;$$

c'est-à-dire :

$$\sum_j \frac{\partial G_j}{\partial x_i} z_j = \frac{\partial E}{\partial x_i} \quad ;$$

ou plus précisément :

$$\sum_j f'(u_j) z_j \omega_{ji} - z_i = \frac{\partial E}{\partial x_i} \quad .$$

Le gradient recherché est alors le vecteur de coordonnée $\Delta\omega_{ij}$ vérifiant :

$$\Delta\omega = -G'_\omega{}^T . z \quad ;$$

c'est-à-dire :

$$\Delta\omega_{ij} = -f'(u_i) x_j z_i \quad .$$

En pratique, au vecteur z on préférera le vecteur y défini par :

$$y_i = -f'(u_i) z_i \quad ,$$

et vérifiant donc :

$$y_i = f'(u_i) . (\frac{\partial E}{\partial x_i} + \sum_j y_j \omega_{ji}) \quad .$$

Les composantes du gradient recherché sont alors données par :

$$\Delta\omega_{ij} = x_j y_i \quad .$$

La recherche du vecteur y se fera suivant une méthode itérative et en pratique on choisira la même méthode que celle permettant de trouver le vecteur de réponses x. En effet, si cette méthode nous permet de trouver effectivement un vecteur x correct alors elle permet aussi de trouver un vecteur y correct (qui, par ailleurs, est alors unique). Cela vient du fait que les deux systèmes dynamiques ont pour systèmes linéarisés deux systèmes qui sont transposés l'un de l'autre et qui admettent donc les mêmes valeurs propres (se reporter à la section 3 traitant des problèmes de convergence).

L'algorithme de rétro-propagation du gradient pour un réseau quelconque est donc le suivant :

i) recherche, suivant la méthode propre au réseau, d'un vecteur x vérifiant :

$$x_i = f(u_i) \quad \text{avec} \quad u_i = \sum_j \omega_{ij} x_j + e_i \quad .$$

ii) recherche, suivant la même méthode, d'un vecteur y vérifiant :

$$y_i = f'(u_i).(c_i + \sum_j y_j \omega_{ji}) \quad \text{avec} \quad c_i = \frac{\partial E}{\partial x_i} \quad .$$

iii) pour toute connexion d'une cellule j vers un neurone i, augmenter le coefficient synaptique ω_{ij} de la quantité :

$$-\lambda x_j y_i \quad .$$

Ici aussi l'appellation de rétro-propagation du gradient est justifiée par le fait que l'information relative aux corrections circule en sens inverse de celle relative aux calculs d'une réponse.

3.3. Cas des réseaux à poids contraints

On a vu au cours du chapitre II qu'il est intéressant de pouvoir imposer à un réseau, outre un graphe de connexions, des contraintes d'égalité entre les poids de connexions, voir des contraintes plus complexes. On peut ainsi exprimer une adéquation entre le réseau et la structure du problème à résoudre ou celle des informations à traiter. La difficulté qui se pose au cours de l'apprentissage est alors de maintenir ces liens entre poids de connexion au fur et à mesure des transformations apportées au réseau. Aussi Y. LE CUN [7] montre comment tenir compte de contraintes d'égalité entre poids avec l'algorithme de rétro-propagation du gradient. Plaçons-nous plutôt dans un cadre général, pour retrouver ensuite le résultat de Y. LE CUN.

On suppose que les contraintes entre les différents poids de connexion ω_{ij} d'un

réseau, sont données sous forme paramétrique en fonction de m paramètres α_i :

$$\omega_{ij} = \omega_{ij}(\alpha_1, \ldots, \alpha_n) \quad .$$

Les coefficients qui nous servent de contrôle pour moduler le fonctionnement du réseau, ne sont donc plus les coefficients synaptiques mais les paramètres α_i. A partir d'un exemple de référence on obtient, avec un choix α des coefficients de contrôle, une réponse x vérifiant :

$$G_i(x, \alpha) = f(\sum_j \omega_{ij}(\alpha) x_j + e_i) - x_i = 0 \quad .$$

En notant u_i la quantité $\sum_j \omega_{ij}(\alpha) x_j + e_i$, on a toujours :

$$\frac{\partial G_i}{\partial x_j} = \begin{cases} f'(u_i)\omega_{ij} & \text{si } i \neq j \\ f'(u_i)\omega_{ij} - 1 & \text{si } i = j \end{cases} \quad .$$

Par revanche, la dérivée par rapport aux coefficients de contrôle prend la forme :

$$\frac{\partial G_i}{\partial \alpha_k} = f'(u_i) \cdot \sum_j x_j \frac{\partial \omega_{ij}}{\partial \alpha_k} \quad .$$

Il nous faut alors chercher un vecteur z vérifiant :

$$G'_x{}^T \cdot z - \mathrm{grad}_x(E) = 0 \quad .$$

Cette contrainte s'exprime comme précedemment :

$$\sum_j f'(u_j) z_j \omega_{ji} - z_i = \frac{\partial E}{\partial x_i} \quad .$$

Seul le gradient recherché prend une forme différente :

$$\Delta\alpha = -G'_\alpha{}^T \cdot z \quad ;$$

c'est-à-dire :

$$\Delta\alpha_k = -\sum_{ij} f'(u_i) z_i \frac{\partial \omega_{ij}}{\partial \alpha_k} x_j \quad .$$

En pratique, au vecteur z on préférera le vecteur y défini par :

$$y_i = -f'(u_i) z_i$$

et vérifiant donc comme précédemment :

$$y_i = f'(u_i) \cdot (\frac{\partial E}{\partial x_i} + \sum_j y_j \omega_{ji}) \quad ;$$

de sorte que le gradient recherché s'exprime :

$$\Delta\alpha_k = \sum_{ij} y_i \frac{\partial \omega_{ij}}{\partial \alpha_k} x_j \quad .$$

Et les variations de connexions induites vérifient :

$$\begin{aligned} \Delta\omega_{ij} &= \sum_k \frac{\partial \omega_{ij}}{\partial \alpha_k} \Delta\alpha_k \\ &= \sum_{lm} y_l \cdot \Big(\sum_k \frac{\partial \omega_{ij}}{\partial \alpha_k} \frac{\partial \omega_{lm}}{\partial \alpha_k}\Big) \cdot x_m \quad . \end{aligned}$$

On retrouve ainsi le résultat de Y. LE CUN [7] où les seules contraintes envisagées sont des contraintes d'égalité entre poids. Soit V_k l'ensemble de poids de connexion devant prendre une même valeur α_k. On alors :

$$\Delta\alpha_k = \sum_{(i,j)\in V_k} y_i x_j \quad ,$$

$$\forall (i,j) \in V_k \qquad \Delta\omega_{ij} = \Delta\alpha_k \quad .$$

4. Mise en œuvre de l'algorithme

Ayant décrit un algorithme d'apprentissage il nous reste maintenant à le mettre en œuvre pour obtenir un réseau ayant un comportement souhaité. La description que nous avons donnée supposait que le comportement souhaité était résumé par une fonction d'évaluation associant à toute réponse une qualité plus ou moins bonne, cette réponse étant obtenue à partir d'une expérience faite avec le réseau. Pour pouvoir rendre compte d'un comportement particulier, il nous faut donc maintenant lui associer une classe d'expériences ainsi qu'une fonction qualifiant le résultat obtenu à la suite d'une telle expérience. Nous allons nous attacher principalement à deux formes de réalisation : comment réaliser un réseau calculant une fonction donnée ? Comment obtenir un réseau ayant les points d'équilibre souhaités ? Dans le premier cas, on souhaite obtenir un point d'équilibre particulier à partir d'une configuration de départ donnée. Dans le second cas, on souhaite obtenir un des points d'équilibre désignés et ce quelque soit le point d'initialisation du calcul.

4.1. Rapprochement vers la réponse désirée

On souhaite réaliser un réseau calculant une fonction donnée. On se donne *a priori* un réseau pour lequel on suppose qu'au moins une instance des coefficients le caractérisant nous convient. Outre les différents choix techniques d'une fonction de seuillage et d'une dynamique de calcul, on a donc aussi fait le choix d'un graphe de connexions dont il nous reste à déterminer la valeur des coefficients

pondérateurs. On a de plus, et surtout, choisit une méthode d'utilisation du réseau en répondant au deux questions : comment une entrée de la fonction détermine le choix d'une trajectoire de calcul ? Comment un point de l'espace des valeurs de la fonction est-il associé à un point d'équilibre ?

Pour juger de la qualité du réseau défini par un choix des coefficients, il nous suffit donc de regarder la valeur obtenue pour différentes entrées, voir toutes les entrées. Une expérience consiste alors à recueillir la réponse du réseau pour une valeur d'entrée donnée. La fonction d'évaluation doit uniquement exprimer que la réponse du réseau doit être la plus proche possible de la réponse souhaitée. Pour cela, on peut utiliser une distance sur l'espace des valeurs de la fonction. En pratique, on utilise la norme euclidienne sur l'espace des réponses du réseau (Ces deux choix sont équivalents si la fonction nous permettant d'interpréter une réponse du réseau est continue pour la norme et la distance respectives des deux espaces.).

Si la réponse souhaitée est codée par l'état $y = (y_1, \ldots, y_n)$ d'un réseau de m neurones dont n seulement servent de support à la réponse, on prend comme mesure d'erreur la fonction qui à tout état $x = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_m)$ associe :

$$E(x) = \frac{1}{2} \sum_{i=1}^{n} (x_i - y_i)^2 \quad .$$

Le gradient de E en un point x est alors donné par :

$$\frac{\partial E}{\partial x_i} = \begin{cases} x_i - y_i & \text{si } i \leq n \\ 0 & \text{sinon} \end{cases} \quad .$$

L'algorithme d'apprentissage est alors complètement spécifié, et le réseau pour lequel l'erreur globale, somme des erreurs sur toutes les entrées possibles, est minimale, est le réseau convenant le mieux parmi tout les réseaux obtenus avec les diverses instances des coefficients de contrôle. De plus si cette erreur globale est nulle, alors le comportement du réseau coïncide avec celui souhaité. L'objection couramment faite à cet algorithme est qu'il ne peut aboutir qu'à un réseau localement optimal. Avec la généralisation de l'algorithme d'apprentissage à des réseaux récursifs se pose en fait un problème de fond : l'existence de points de discontinuité, du lien entre la réponse d'un réseau et ses poids de connexion, pose la question de la validité de l'usage d'un gradient pour améliorer le réseau.

Afin de faire ressortir cette difficulté, plaçons-nous dans le cas simplifié d'un réseau dont la dynamique est caractérisée par une fonction de LYAPUNOV $H(x)$, définie en tout état x du réseau et décroissante le long de toute trajectoire du réseau. Partant d'un point x^o, une évolution aboutit alors à un point x^∞ où H est localement minimale. On suppose que l'on peut moduler cette fonction H à l'aide de paramètres de contrôle et l'on se propose de modifier ces coefficients de

sorte que le point atteint à partir du point x^o soit l'état d. La situation initiale est donnée par la figure III.9 : la réponse désirée d est proche de celle obtenue x^∞. En utilisant un algorithme d'apprentissage on peut alors rapprocher le minimum local x^∞ de H vers d. On peut même espérer que la méthode converge et obtenir un choix des coefficients de contrôle induisant une fonction de LYAPUNOV telle que le point d en soit un minimum local. Mais ce minimum sera-t-il toujours accessible à partir de notre état d'initialisation x^o ? On peut, en effet, aboutir à la situation schématisée par la figure III.10. On a, en fait, détérioré le comportement de notre système.

On peut aussi voir cette difficulté en se plaçant dans l'espace produit « états $\times$ coefficients de contrôle ». La situation est résumée par la figure III.11. Pour améliorer le réseau on s'appuie sur le lien implicite $G(x, \omega) = 0$, entre une réponse x du réseau et ses coefficients de contrôle ω. Désirant obtenir la réponse d et obtenant avec un premier choix ω^o des coefficients de contrôle une réponse x^∞, on utilise l'information donnée par G. Ceci revient, dans notre cas, à suivre la *surface* d'équation $G(x, \omega) = 0$ dans le sens qui nous rapproche du *plan* d'équation $x = d$. On obtient ainsi un second choix $\omega^o + \Delta\omega$ pour lequel d est effectivement un point d'équilibre, mais ne correspond plus à la réponse donnée par le réseau à partir de l'état initial x^o.

Pour éviter cette difficulté, il est nécessaire et suffisant que soit continue la

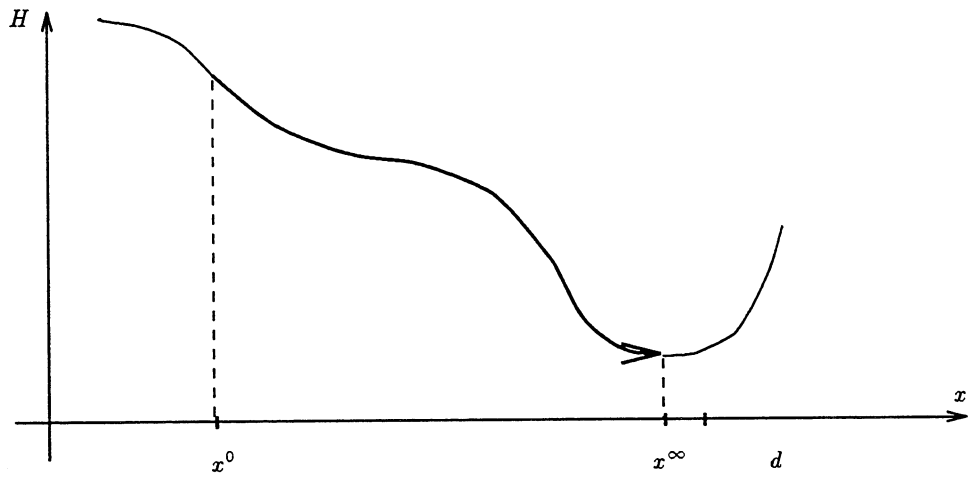

FIG. III.9 - . En partant de l'état initial x^o, la dynamique du réseau aboutit à un minimum local x^∞ de la fonction de LYAPUNOV H caractérisant le réseau. On souhaite obtenir comme aboutissement le point d. Peut-on modifier H de sorte que ce point d en soit un minimum local ?

fonction, qui à un état x et un choix ω des coefficients de contrôle associe le point d'équilibre atteint par le réseau guidé par ω à partir de l'état x. On a vu que cela n'est pas nécessairement le cas pour les réseaux récursifs. L'algorithme de rétropropagation du gradient ne peut donc pas être rigoureusement mis en œuvre pour l'interpolation d'une fonction dans le cadre des réseaux récursifs, malgré le fait qu'il y soit formalisable : le gradient de l'erreur ne donne alors pas suffisamment d'information.

Insistons sur ce manque d'information en nous plaçant dans un cadre concret : celui du réseau que nous avons proposé dans la section précédente pour la reconnaissance de la connexité. Pour obtenir une version définitive de ce réseau, il reste à exhiber les réseaux de neurones calculant les fonctions $V_i(x)$, permettant de décider si le pixel i peut être colorié ou non sans changer la connexité de l'image x. On se propose de synthétiser ces réseaux par apprentissage. On choisit pour cela un réseau organisé en couches, ayant pour entrée les pixels voisins du pixel de référence et dont le neurone de sortie doit décider si oui ou non le coloriage peut avoir lieu. Ce même réseau peut être utilisé pour tous les pixels (à une translation près des cellules d'entrée) et on obtient ainsi un premier réseau récursif candidat à la reconnaissance de la connexité. C'est ce réseau que l'on se propose d'améliorer en lui fournissant diverses images et en comparant ses réponses avec celles attendues. Nous sommes donc *a priori* dans une situation idéale : le problème est

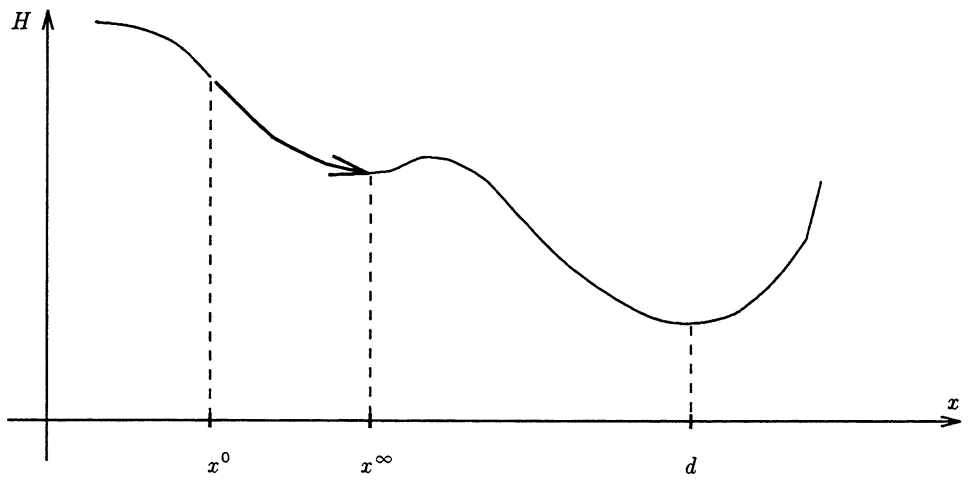

FIG. III.10 - . En modifiant la fonction de LYAPUNOV H caractérisant le réseau, on peut effectivement s'arranger pour que le point d en soit un minimum local. Mais peut-on garantir que ce minimum local soit toujours accessible depuis le point x^o? On risque comme ici de modifier les bassins d'attraction.

déjà structuré. Il reste à trouver un nombre relativement faible de connexions et celles-ci sont utilisées uniformément sur tout le réseau.

Mais en réalité l'information que l'on peut obtenir à un point d'équilibre du réseau est sans intérêt : s'il y a, au cours du processus une coloration ayant entraîné une connexion accidentelle, on ne sait, lorsque l'équilibre est atteint, ni où ni quand cette connexion a eu lieu. Supposons ainsi que la fonction V réalisée soit incorrecte et qu'elle autorise la coloration d'un pixel pour une configuration du voisinage de ce pixel qui soit telle que la connexité de l'image soit affectée. Toute image est alors déclarée connexe par ce processus qui transforme toute image comportant au moins un pixel en l'image entièrement noire. Pour chaque instance V_i de la fonction V, l'entrée est alors uniformément noire. Si l'image à reconnaître n'était pas connexe, modifier les poids de connexion définissant la fonction V reviendrait à ne plus reconnaître l'image toute noire comme maximale et plus aucune image ne serait alors reconnue connexe. Pour synthétiser par apprentissage le système récursif de reconnaissance de la connexité, il est donc nécessaire de concevoir la fonction de contrôle V dans un contexte isolé.

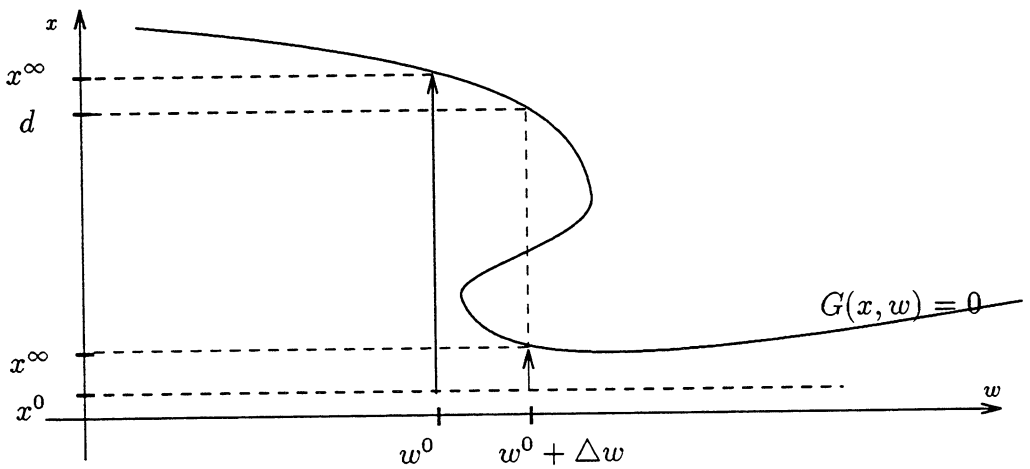

FIG. III.11 - . La difficulté rencontrée par un algorithme d'apprentissage qui s'appuie sur le lien $(G(x,\omega) = 0)$, entre une réponse x du réseau et ses coefficients de contrôle ω, est que l'on a seulement l'implication ($\ll x^\infty$ est le point d'équilibre atteint depuis l'état initial $x^o \gg \Rightarrow G(x^\infty,\omega) = 0$) et non nécessairement la réciproque.

4.2. Transformation en un point fixe

On souhaite maintenant réaliser une mémoire associative, c'est à dire imposer un certain nombre de points d'équilibre à la dynamique du réseau. Comme précédemment il nous faut choisir une classe d'expériences ainsi qu'une fonction qualifiant le résultat obtenu à la suite d'une telle expérience. L'idée est de placer le réseau sur l'un des points d'équilibre souhaités et de regarder si le réseau y est stable ou non. L'erreur à considérer est donc une mesure du vecteur « vitesse » au point test. Nous allons en fait chercher uniquement à imposer un état d'équilibre à une partie de l'ensemble des neurones. Soit V cet ensemble de neurones que l'on qualifie de visibles et C l'ensemble de tous les neurones dont l'état d'équilibre ne nous importe pas et qui sont dits cachés (On notera respectivement ω_V et ω_C les matrices définissant les connexions afférentes aux neurones de V et C.). Étant donné un état d des cellules visibles, on cherche en ensemble de poids de connexion tel qu'il existe un état x_C des cellules cachées pour lequel (d, x_C) soit un point d'équilibre.

L'expérience consiste à mettre les cellules de V dans l'état désiré d et à laisser évoluer *uniquement* les cellules de C jusqu'à l'obtention d'un point d'équilibre x_C de ces cellules. Cette expérience ne reproduit donc pas le fonctionnement du réseau en cours d'utilisation où *toutes* les cellules évoluent. L'erreur prise pour juger de la qualité du réseau est alors une mesure de la qualité de l'équilibre qu'auraient les cellules visibles au point $x = (d, x_C)$ dans les conditions d'utilisation :

$$\begin{aligned} E(x_c, \omega_V) &= \frac{1}{2}\|F_V(\omega_V \,.\, x) - x_V\|^2 \\ &= \frac{1}{2}\sum_{i\in V}[f(u_i) - d_i]^2 \quad , \end{aligned}$$

avec :

$$\begin{aligned} \forall i \in V \qquad u_i &= \sum_j \omega_{ij} x_j \\ &= \sum_{j\in V} \omega_{ij} d_j + \sum_{j\in C} \omega_{ij} x_j \quad . \end{aligned}$$

Le point x_C vérifie le lien implicite :

$$G(x_c, \omega_C) = F_C(\omega_C \,.\, x) - x_C = 0 \quad ,$$

c'est-à-dire :

$$\forall i \in C \qquad G_i = f(u_i) - x_i = 0 \quad ,$$

toujours avec :

$$\forall i \in C \qquad u_i = \sum_j \omega_{ij} x_j$$
$$= \sum_{j \in V} \omega_{ij} d_j + \sum_{j \in C} \omega_{ij} x_j \quad .$$

L'erreur E est bien nulle si et seulement si le point $x = (d, x_C)$ est un point d'équilibre dans le cadre d'une utilisation normale du réseau. La fonction d'erreur dépend explicitement des coefficients synaptiques du réseau (de ceux afférents aux cellules visibles) et seuls les états des cellules cachées interviennent dans le lien implicite $G(x) = 0$. Un pas de l'algorithme d'apprentissage consiste donc à chercher d'abord un vecteur intermédiaire y_C de même dimension que x_C et tel que :

$$\forall i \in C \qquad y_i = f'(u_i).(\frac{\partial E}{\partial x_i} + \sum_j y_j \omega_{ji})$$
$$= f'(u_i) \sum_{j \in V} f'(u_j)\omega_{ji}(f(u_i) - d_i) + \sum_{j \in C} y_j \omega_{ij} \quad .$$

Les composantes du gradient recherché sont alors données par :

$$\forall i \in V \quad j \in V \qquad \Delta\omega_{ij} = (d_i - f(u_i))f'(u_i)d_j$$
$$\forall i \in V \quad j \in C \qquad \Delta\omega_{ij} = (d_i - f(u_i))f'(u_i)x_j$$
$$\forall i \in C \quad j \in V \qquad \Delta\omega_{ij} = y_i d_i$$
$$\forall i \in C \quad j \in C \qquad \Delta\omega_{ij} = y_i x_j \quad .$$

L'intérêt de cet algorithme d'apprentissage est d'une part de montrer qu'une expérience faite sur le réseaux pour juger de sa qualité, ne doit pas nécessairement reproduire les conditions d'utilisation prévues pour ce réseau. De plus il fournit un cadre correct pour l'amélioration d'un réseau récursif. En effet, le lien $G(x, \omega)$ sur lequel s'appuie la spécification de l'algorithme donne cette fois-ci toute l'information nécessaire : on cherche ω tel que l'état d soit un point d'équilibre. Et par définition de G, on a l'implication $G(d, \omega) = 0 \Rightarrow \ll d$ est un point d'équilibre du réseau contrôlé par $\omega \gg$. La différence avec la section précédente est que l'on ne cherche plus à atteindre ce point d'équilibre depuis un état initial particulier, fait dont ne peut pas rendre compte le lien $G(x, \omega) = 0$.

On peut aussi voir les choses d'un autre point de vue : en donnant au réseau des exemples de points d'équilibre, on lui donne bien plus d'informations qu'en lui donnant seulement une entrée et la valeur qui lui est associée. Pour expliciter ce point reprenons le réseau récursif destiné à reconnaître la connexité. Rappelons que pour obtenir une version définitive de ce réseau, il reste à exhiber les réseaux de neurones calculant les fonctions $V_i(x)$, permettant de décider si le pixel i peut

être colorié ou non sans changer la connexité de l'image x. On prend comme cellule visible tous les neurones codant un pixel. Les exemples fournis au réseau sont cette fois-ci des images qui ont été au préalable entièrement coloriées sans en changer la connexité et on demande que ces images soient stables. Cela revient donc à fournir aux cellules chargées de calculer les fonctions V_i, des exemples de situation où un coloriage entraînerait un changement de connexité. L'information fournie au réseau est donc bien plus importante que précédemment : on lui transforme au préalable l'image. Ainsi le réseau dispose de suffisamment d'informations pour pouvoir savoir qu'une image est maximale (*i.e.* qu'elle ne peut plus être coloriée sans en changer la connexité). Il ne dispose, par contre, d'aucune information relative aux images non maximales. Pour remédier à ce problème il faudrait pouvoir aussi présenter des exemples négatifs de non-équilibre : des images devant être encore transformées avant de pouvoir conclure sur leur connexité. Il faut alors, non seulement fournir l'image, mais aussi le lieu des pixels à colorier. Un tel processus d'apprentissage reviendrait en fait à concevoir la fonction de contrôle V dans un contexte isolé.

Conclusion

En utilisant le lien implicite entre les états d'équilibre d'un réseau et les coefficients décrivant ce réseau, on arrive donc à exprimer des algorithmes permettant d'améliorer le comportement d'un système en le façonnant *in situ* à l'aide d'expériences permettant de juger de sa qualité. Ces algorithmes peuvent être exprimés dans des cadres très divers indépendamment de la dynamique du réseau permettant d'obtenir un état d'équilibre et de l'interprétation de cette dynamique en termes de calcul. Le calcul d'une modification des coefficients ne s'appuie que sur le lien décrivant les états d'équilibre et une fonction d'erreur résumant l'expérience. En généralisant l'algorithme du PERCEPTRON, ces algorithmes de rétro-propagation permettent de gagner en souplesse d'adaptation. Mais en l'absence de critère de convergence, le problème du choix d'un réseau de départ ne peut plus être abordé avec autant de rigueur qu'avec le PERCEPTRON. Et l'on ne sait ainsi pas dans quelles conditions l'apprentissage peut aboutir au système souhaité.

Toutefois on peut cerner des cadres de mise en œuvre inadéquats. Il apparaît ainsi que l'on ne peut pas ainsi reproduire tout comportement choisi *a priori* : il est nécessaire que ce comportement ne repose que sur le lieu des points d'équilibres et nullement sur les conditions pour obtenir un point d'équilibre particulier. Aussi on ne peut obtenir par apprentissage un réseau récursif calculant une fonction en ne donnant comme exemples au réseau que des couples (donnée, résultat à obtenir). Il apparaît nécessaire de donner au système des informations sur le chemin menant de cette entrée à son image.

CHAPITRE IV

Un cadre de construction de réseaux de neurones

Introduction

Ce chapitre aborde le problème de la construction de réseaux de neurones répondant à une certaine spécification. La difficulté est double. Il faut pouvoir décomposer une spécification de façon à se ramener à un ensemble de problèmes plus simples. Il faut aussi pouvoir composer des réseaux partiels en garantissant le comportement du réseau global. Ces deux niveaux sont imbriqués et la question devient la suivante. Comment décomposer une spécification en spécifications partielles réalisables, puis combiner les réseaux y répondant pour obtenir un réseau répondant à la spécification globale?

D'une manière plus formelle mais tout en gardant une définition intuitive de la notion de spécification, le cadre du problème est le suivant. On se fixe un ensemble $\mathcal{C}_\mathcal{R}$ de constructeurs de réseaux, permettant d'obtenir un nouveau réseau à partir d'un ou plusieurs réseaux et on considère un ensemble de réseaux clos pour ces constructeurs. Ce dernier ensemble $\mathcal{R}$ est l'ensemble de tous les réseaux parmi lesquels on se propose de chercher une réponse à une spécification. De même, on se fixe un ensemble $\mathcal{C}_\mathcal{S}$ de constructeurs de spécifications, permettant d'obtenir une nouvelle spécification à partir d'une ou plusieurs spécifications, et on considère un ensemble de spécifications clos pour ces constructeurs. Ce dernier ensemble $\mathcal{S}$ est l'ensemble de toutes les spécifications que l'on souhaite pouvoir réaliser. L'adéquation entre un réseau r et une spécification s est définie par une relation $\rightarrow$ entre les éléments des deux ensembles $\mathcal{R}$ et $\mathcal{S}$ et dont l'interprétation est la suivante :

$r \rightarrow s$ ssi le réseau r est une réalisation de la spécification s.

En pratique la définition de cette relation va de paire avec la définition de la notion de spécification : une spécification est un ensemble de propriétés que devra vérifier le réseau (ou plus généralement le système de calcul automatique) proposé comme réalisation.

Supposons maintenant que cette relation vérifie un ensemble de propriétés de la forme :

$$\begin{aligned}
&\text{Pour tous réseaux} \quad r_1, \ldots, r_n \quad \text{de } \mathcal{R} \quad , \\
&\text{pour toutes spécifications} \quad s_1, \ldots, s_n \quad \text{de } \mathcal{S} \quad , \\
&\text{si} \quad (r_1 \rightarrow s_1) \quad \text{et} \ldots \text{et} \quad (r_n \rightarrow s_n) \quad , \\
&\text{alors} \quad c_r(r_1, \ldots, r_n) \rightarrow c_s(s_1, \ldots, s_n) \quad ,
\end{aligned}$$

où c_r est un constructeur de réseaux et c_s un constructeur de spécifications.

Pour construire un réseau répondant à une spécification s que l'on arrive à mettre sous la forme $c_s(s_1, \ldots, s_n)$, il suffit alors de construire des réseaux $r_1, \ldots, r_n$, répondant respectivement aux spécifications $s_1, \ldots, s_n$, puis de les combiner à l'aide du constructeur c_r. Pour que cette démarche nous permette de réaliser toute spécification de $\mathcal{S}$, il suffit de connaître une réalisation de chaque spécification primitive de $\mathcal{S}$ (i.e qui ne peut être décomposée) et de disposer d'au moins une règle de construction pour chaque constructeur de spécifications c_s.

Prenons un exemple pour fixer les idées. On prend comme ensemble de réseaux l'ensemble des réseaux organisés en couches et dont les neurones prennent leur état dans l'ensemble $\{0,1\}$. On prend comme ensemble de spécifications l'ensemble des applications d'un ensemble de la forme $\{0,1\}^E$ dans un ensemble de la forme $\{0,1\}^S$. Un réseau r, ayant l'ensemble E comme neurones d'entrées et l'ensemble S comme neurones de sortie, réalise la spécification f de $\{0,1\}^E$ dans $\{0,1\}^S$ si et seulement si il associe à un état $e \in \{0,1\}^E$ de sa couche d'entrée l'état $s = f(e) \in \{0,1\}^S$ de sa couche de sortie.

On peut combiner deux réseaux (réalisant respectivement les applications f_1 de $\{0,1\}^{E_1}$ dans $\{0,1\}^{S_1}$ et f_2 de $\{0,1\}^{E_2}$ dans $\{0,1\}^{S_2}$) de plusieurs façons. On peut juxtaposer les deux réseaux et obtenir une réalisation de l'application f de $\{0,1\}^{E_1 \cup E_2}$ dans $\{0,1\}^{S_1 \cup S_2}$ qui à (e_1, e_2) associe $(f_1(e_1), f_2(e_2))$. On peut aussi combiner ces deux réseaux par superposition en identifiant toutes les cellules d'entrée du second réseau à des cellules de sortie du premier (cette identification étant définie par une application p de E_2 dans S_1). On obtient alors une réalisation de l'application $f_2 \circ P \circ f_1$, où P est l'application qui à $(x_1, \ldots, x_n)$ de $\{0,1\}^{S_1}$ associe $(y_1, \ldots, y_i, \ldots, y_m) = (x_{p(1)}, \ldots, x_{p(i)}, \ldots, x_{p(m)})$ de $\{0,1\}^{E_2}$.

Ce cadre de spécification est celui qu'utilise implicitement F. ROSENBLATT [45] avec le PERCEPTRON : faire intervenir des cellules d'association revient à se donner des spécifications partielles pour lesquelles on cherchera ultérieurement une réalisation. C'est aussi le cadre qui est implicitement utilisé avec les réseaux de neurones organisés en couches et la rétro-propagation du gradient : en réalisant, par exemple, un système associant un symbole à un signe on souhaite pouvoir ultérieurement connecter plusieurs copies de ce système pour associer un mot à une

suite de signes. Mais comme l'ont souligné M. MINSKY et S. PAPERT [35] ce cadre ne permet pas de spécifier facilement des tâches pourtant réalisables (comme le calcul du caractère connexe d'une image). De plus toutes les tâches que l'on souhaiterait confier à des réseaux de neurones ne se spécifient pas nécessairement en termes de fonctions, du moins en l'absence d'informations supplémentaires de contexte (comme dans le cas de la lecture). Il est donc souhaitable d'élargir le domaine de spécification. Les réseaux organisés en couches ne pouvant réaliser que des applications, on est donc aussi amené à utiliser des réseaux pouvant comporter des boucles de rétro-action.

La difficulté va être alors de combiner plusieurs réseaux en un seul en garantissant que cette construction ne va pas altérer le comportement des réseaux initiaux, c'est-à-dire en garantissant qu'ils réalisent la même spécification avant et après la combinaison. En effet, si l'on combine par identification de neurones deux réseaux, une altération peut avoir lieu. Considérons, par exemple, le réseau de la figure IV.12. Ses connexions ont été choisies de façon à ce qu'à l'équilibre les deux neurones soient tous deux dans le même état ; mais des connexions supplémentaires issues d'un deuxième réseau peuvent très bien imposer à ces deux neurones un équilibre brisant cette relation d'égalité. La spécification du premier réseau (imposer un même état) n'est alors plus respectée.

L'organisation du chapitre est la suivante. Dans un premier temps nous allons dégrossir le problème en regardant sur un exemple la forme des spécifications que l'on peut donner aux réseaux de neurones ainsi que l'angle sous lequel on peut envisager la construction de réseaux. Un cadre de spécification étant choisi, nous

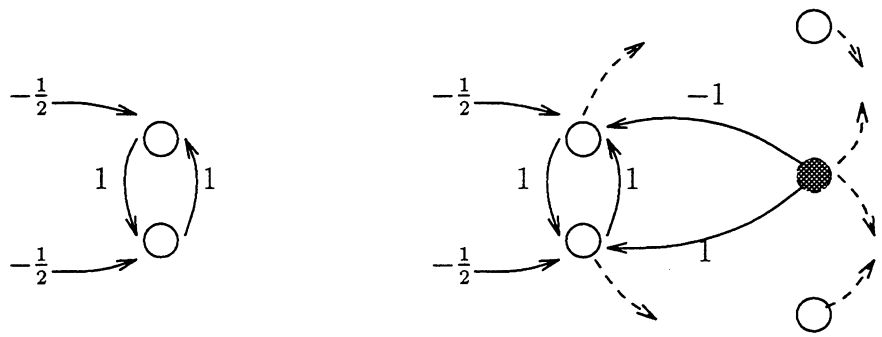

FIG. IV.12 - . Les connexions du neurone grisé altère le fonctionnement du sous-réseau de référence (à gauche). Lorsque ce neurone est actif, il impose un état, $(0,1)$, qui n'est pas un état d'équilibre du sous-réseau isolé, $(0,0)$ ou $(1,1)$.

allons ensuite cerner une classe particulière de réseaux de neurones, notre guide étant la relation de réalisation entre une spécification et un réseau. On aboutira enfin à un cadre de construction tel que toute spécification admet au moins une réalisation.

1. Choix d'un cadre de travail

Si l'on regarde le problème de la lecture d'un texte manuscrit, on n'arrive pas à dégrossir simplement des règles permettant de construire la réponse, un texte, à partir de l'image initiale. A un même indice peuvent correspondre de nombreuses interprétations différentes suivant le contexte où il se situe et même suivant l'origine du document. De plus, ces différentes interprétations ne peuvent être souvent départagées que par des informations que l'on ne peut obtenir qu'une fois le texte lu (Tel mot figure-t-il dans le dictionnaire? Cette syntaxe est-elle correcte?). Cette difficulté se rencontre aussi à des niveaux de traitement considérés comme plus simples. On peut ainsi citer l'extraction des contours d'une image. Il s'agit de délimiter, dans une image, le contour des différents objets présents dans l'image. On dispose bien d'un indice, le gradient de l'intensité lumineuse, mais en s'appuyant uniquement sur celui-ci on peut dissocier un même objet en plusieurs parties ou associer plusieurs objets en une seule zone. Ces erreurs vont alors gêner le système utilisant les contours extraits pour reconnaître les objets présents dans la scène.

Par contre, il est plus facile de vérifier que telle transcription peut correspondre à tel texte manuscrit ou que telle scène peut donner lieu à telle image. Il suffit de pouvoir faire correspondre point à point les diverses composantes du texte transcrit d'une part et du texte manuscrit d'autre part. Au lieu de s'attacher à la description de l'obtention d'une réponse à partir d'une situation, on cherchera donc plutôt à spécifier l'adéquation d'une réponse à une situation. Par exemple, une suite de lettres peut être considérée comme le résultat de la lecture de tel signe, si et seulement si ce signe peut être décomposé en sous-signes pouvant être mis en correspondance avec chacune des lettres du mot. Il reste alors à spécifier la décomposition d'un signe en sous-signes et l'adéquation entre un tel sous-signe et une lettre de l'alphabet. Cette spécification de l'adéquation entre un signe et une suite de lettres peut être aussi raffinée en précisant que la suite de lettres doit faire partie du dictionnaire. Une telle approche est donc intrinsèquement modulaire : les propriétés que devront vérifier les réponses du système sont définies et raffinées par conjonction de propriétés plus élémentaires. Il est alors souhaitable que le système global soit lui-même obtenu par superposition de sous-systèmes réalisant ces spécifications partielles.

La programmation logique et les langages qui lui sont associés sont une mise en œuvre de cette approche qui consiste à spécifier la nature de la réponse que

l'on souhaite obtenir plutôt que le chemin pour y parvenir. Une spécification y est exprimée par une relation que doivent vérifier les différents éléments composant les réponses du système. Une spécification est simplement raffinée par l'ajout de telles relations (ou *règles* pour reprendre la terminologie de PROLOG). Pour éclairer nos propos, prenons l'exemple de la lecture d'un mot manuscrit.

On considère, pour simplifier, qu'un mot manuscrit est décrit par une courbe et que celle-ci peut être résumée par une liste. On peut alors spécifier en PROLOG la lecture d'une courbe en une suite de lettres par les règles suivantes[1] :

$$\begin{aligned}&\text{Lecture-Courbe-Suite-de-lettres}(c,< l >) \quad \rightarrow \\ &\qquad \text{Lecture-Courbe-Lettre}(c,l) \\ &\text{Lecture-Courbe-Suite-de-lettres}(c_1 \bullet c_2, l_1 \bullet l_2) \quad \rightarrow \\ &\qquad \text{Lecture-Courbe-Suite-de-lettres}(c_1,l_1) \\ &\qquad \text{Lecture-Courbe-Suite-de-lettres}(c_2,l_2) \quad .\end{aligned}$$

On peut raffiner le système en ajoutant une règle de la forme :

$$\begin{aligned}&\text{Lecture-Courbe-Mot}(c,l) \quad \rightarrow \\ &\qquad \text{Lecture-Courbe-Suite-de-lettres}(c,l) \\ &\qquad \text{Dictionnaire}(l) \quad .\end{aligned}$$

Il reste alors à spécifier les deux règles :

$$\begin{aligned}&\text{Lecture-Courbe-Lettre}(c,l) \quad \rightarrow \quad \ldots \\ &\text{Dictionnaire}(l) \quad \rightarrow \quad \ldots\end{aligned}$$

La recherche de la transcription *mot* d'un texte manuscrit *texte* donné correspond alors à la requête :

$$\text{Lecture-Courbe-Suite-de-lettres}(\textit{texte},\textit{mot})$$

qui fournit l'ensemble des mots pouvant être mis en correspondance avec le texte.

Notons tout de suite que cette spécification est trop simple pour mener à la réalisation effective d'un système de lecture automatique : l'espace de recherche (grosso modo l'ensemble de toutes les décompositions possibles de la courbe de départ en sous-courbes) étant bien trop grand pour que cette recherche puisse se faire en un temps raisonnable, et ceci même en invoquant un parallélisme de calcul comme nous allons le faire avec les réseaux de neurones. Mais en se délimitant un cadre de spécification et de construction modulaire on pourra toujours essayer d'améliorer les performances d'un système en lui ajoutant des contraintes permettant de restreindre son espace de recherche, l'essentiel étant que tout gain de

[1] Les notations sont les suivantes : $< a_1, \ldots, a_n >$ désigne la liste des n éléments a_i ; $l_1 \bullet l_2$ désigne la concaténation des deux listes l_1 et l_2.

performance, par une heuristique ou une autre, restera un acquis si l'on dispose d'un mécanisme de construction modulaire.

L'intérêt d'une telle spécification portant sur la nature des réponses que l'on souhaite obtenir et non sur les chemins pour y parvenir, est quelle permet de passer outre les problèmes de contexte. Les deux parties du système, « Lecture-Courbe-Lettre(c,l) » et « Dictionnaire(l) », peuvent être conçues indépendamment l'une de l'autre. Chacune des parties est conçue indépendamment de tout contexte en cherchant uniquement à rendre compte de toute les réponses possibles : les mécanismes internes à PROLOG assureront la coopération souhaitée. Les difficultés liées au contexte sont ainsi reportées. On ne les trouve plus au moment de la spécification d'un système mais en cours de sa réalisation : il faut proposer un mécanisme de coopération. C'est cette voie que nous nous proposons d'explorer. Mais le fait d'utiliser des réseaux de neurones comme mécanisme pour réaliser ces spécifications va néanmoins entraîner plusieurs ruptures avec l'approche plus classique de la programmation logique.

1.1. Contraintes liées à l'utilisation de réseaux de neurones

On désire utiliser un réseau de neurone pour chercher dans un ensemble un élément vérifiant un certain nombre de contraintes. Il nous faut donc disposer d'un code associant à tout état du réseau l'un des points de l'espace de recherche. Un calcul consiste à laisser évoluer le réseau vers un état d'équilibre et la réponse ainsi obtenue est l'élément de l'espace de recherche que code ce point d'équilibre. Le réseau nous convient alors si sa réponse est une solution à notre problème. Notons que l'on ne peut plus chercher l'ensemble des solutions possibles mais uniquement l'une d'entre elles : la première a être atteinte. C'est donc une première rupture avec la programmation logique.

Si l'on utilise un réseau dont les neurones ont un espace d'états fini, comme nous allons le faire pour rester dans un cadre simple, ce réseau ne peut désigner par son état interne qu'un nombre fini de points dans l'ensemble de recherche. Un réseau de neurones isolé ne pourra donc travailler que sur une taille de données bornée a priori (par exemple, association d'un mot d'au plus dix lettres à une image de 20×200 pixels). Si l'on veut néanmoins réaliser un système travaillant sur des données de taille arbitraire, il faudra ajouter au système des mécanismes extérieurs (par exemple, déplacement de l'instrument de saisie de l'image ou support permettant de noter le texte lu). On ne s'intéressera pas ici à ce problème et on se restreindra à la réalisation de réseaux isolés. La seconde rupture avec la programmation logique est alors que l'espace de recherche doit être fini.

Regardons maintenant un point plus technique. Nous avons vu que contrôler l'état d'équilibre atteint à partir d'un état initial particulier est un problème difficile dès que l'on considère des réseaux récursifs. On ne cherchera donc pas à faire

dépendre le fonctionnement du réseau du choix d'un état initial. On considère ainsi comme réponse du réseau tout état d'équilibre atteint à partir d'un état initial quelconque. Tous les états d'équilibre du réseau codent donc des *réponses* possibles. Pour que ce réseau nous convienne il est donc nécessaire que tous ses points d'équilibre codent une *solution* à notre recherche. Enfin, on demande que toutes les solutions soit codées par au moins un état d'équilibre. Ce point nous sera utile lorsqu'on envisagera la combinaison de plusieurs réseaux. En effet, pour pouvoir aborder cette construction indépendamment des spécifications que ces réseaux réalisent, il est nécessaire que la donnée de ces réseaux rendent compte de tous les aspects de leur spécifications respectives.

Les propos ci-dessus supposaient implicitement qu'un point d'équilibre était effectivement atteint par le réseau. Pour être une réalisation d'une spécification comportant au moins une solution, un réseau devra être aussi au moins tel qu'un point d'équilibre soit toujours atteint et qu'une réponse soit ainsi donnée. Comme le point d'équilibre atteint à partir d'un état d'initialisation nous importe peu dès lors qu'il code une solution à notre recherche, on peut envisager une dynamique non déterministe. Avec cette hypothèse, il est alors plus facile de garantir la convergence vers un état d'équilibre (du moins presque sûrement : il suffit pour cela d'exhiber pour chaque état un chemin de probabilité non nul et menant vers un point d'équilibre).

Notons enfin une troisième et dernière rupture avec la programmation logique : l'absence de solution à une recherche ne peut être signalée. En effet, une telle absence ne peut être codée par un point d'équilibre particulier : elle serait alors toujours une réponse possible.

1.2. Définition

Rappelons les bases de travail que nous avons dégagées. Une spécification est définie à partir d'un espace de recherche fini et d'une propriété sur les points de cette espace. On appelle solution à cette spécification tout point de l'espace vérifiant la propriété associée. Un système (dans notre cas, un réseau de neurones) est un automate fini non-déterministe dont les trajectoires sont interprétées dans l'espace de recherche à l'aide d'un code. A tout instant un tel système désigne donc un point de l'espace de recherche et on considère comme la réponse du système le point de l'espace de recherche désigné par l'état terminal (sans successeur) éventuellement atteint par une trajectoire. Pour qu'un réseau soit une réalisation de la spécification, on lui demande que toutes ses réponses soient des solutions et ce quelque soit son état initial. On lui demande aussi de donner presque sûrement une réponse en un temps fini (si il existe une solution). On lui demande enfin que toute solution soit une réponse possible, dans le but de pouvoir utiliser ce système pour la construction de systèmes plus complexes.

On a alors trois ruptures essentielles avec la programmation logique: l'espace de recherche est fini ; une seule solution est proposée par le système isolé et une absence de solution ne peut être signalée. On retiendra la définition suivante:

DÉFINITION. *Une* spécification *sur un ensemble fini E est définie par la donnée d'une propriété $\mathcal{P}$ sur E. On notera « chercher x de E tel que $\mathcal{P}(x)$ » une telle spécification. On appelle* solution *tout élément de E vérifiant la propriété $\mathcal{P}$.*

Un système *interprété dans un ensemble fini E est un automate fini non-déterministe à chaque état duquel est associé un élément de E. On appelle* réponse *du système l'élément de E associé à l'état terminal éventuellement atteint par une trajectoire.*

Un système interprété dans E réalise *une spécification sur E si et seulement si les trois points suivants sont vérifiés.*

i) *Toutes les réponses du système sont des solutions de la spécification.*

ii) *Toute solution de la spécification est une réponse possible du système.*

iii) *Si il existe une solution pour la spécification alors le système donne presque sûrement une réponse en un temps fini.*

Dans cette définition la notion de donnée de calcul n'apparaît pas. On peut l'introduire en faisant dépendre l'ensemble des solutions de la donnée d'un élément d'un ensemble E de données de calcul. Il faut aussi pouvoir guider l'évolution d'un système en fonction d'un élément e de E, ce qui revient d'un point de vue descriptif à associer un graphe de transitions particulier à chaque élément e de E. La définition complétée est alors la suivante:

DÉFINITION. *Étant donnés deux ensembles finis E et F ainsi qu'une propriété $\mathcal{P}$ sur $E \times F$, on considère pour tout élément e de E la spécification sur F : « chercher x de F vérifiant la propriété $\mathcal{P}(e,x)$ ». On notera « étant donné e de E, chercher x de F vérifiant la propriété $\mathcal{P}(e,x)$ » cette famille de spécifications et on parlera de spécification sur F conditionnellement à E.*

Un système est dirigé *par un ensemble d'états P si son espace d'états est de la forme $P \times Q$ et si la projection sur P de l'état du système n'est pas modifiée par les transitions d'état du système :*

$$\begin{array}{ll} si & (p,q) \to (p',q') \quad \text{est une transition du système,} \\ alors & p = p' \quad . \end{array}$$

*Si à chaque élément de P est associé un élément d'un ensemble E, le système est aussi dit dirigé par E et le code correspondant est qualifié d'*interprétation de la commande. *Si le système est de plus interprété dans F, une réponse du système dirigé par un élément e de E est l'élément de F associé à l'état terminal éventuellement atteint par une trajectoire dont la projection sur P code e.*

Un système, dirigé par E et interprété dans F, réalise la spécification sur F conditionnellement à E définie par une propriété $\mathcal{P}$ si et seulement si pour tout élément e de E le système dirigé par e réalise la spécification « chercher x de F vérifiant la propriété $\mathcal{P}(e,x)$ ».

Montrons comment traduire, avec les termes que nous venons de définir, le système de lecture que nous avons spécifié en PROLOG. On doit tout d'abord se ramener à un espace de recherche fini. Notons I l'ensemble de toutes les images possibles sur un écran fini et A l'ensemble de toutes les lettres de l'alphabet. Traduire une image en un mot de n lettres c'est alors chercher une décomposition de l'image en n tranches consécutives pouvant chacune être associée à la lettre du même ordre dans le mot :

Étant donné i de I,
  chercher $(i_1, \ldots, i_n, a_1, \ldots, a_n)$ de $I^n \times A^n$
  vérifiant :
    $i = C(i_1, \ldots, i_n)$
    $i_k \mathcal{R} a_k$ pour tout k de $\{1, \ldots, n\}$

où C est une fonction définissant la mise bout à bout d'images et $\mathcal{R}$ une relation définissant l'adéquation entre une image et une lettre.

On cherchera à construire ce système à partir de deux réseaux réalisant les spécifications « Étant donné i de I, chercher $(i_1, \ldots, i_n)$ de I^n tq $i = C(i_1, \ldots, i_n)$. » et « Chercher (i, a) de $I \times A$ tel que $i\mathcal{R}a$. ». Le premier décompose donc une image en images partielles. Le second vérifie lui l'adéquation entre une image et une lettre. On prendra n copies de ce dernier réseau pour réaliser un système de lecture de mots d'au plus n lettres. En ce sens on pourra s'affranchir du fait que l'espace de recherche est nécessairement fini : on pourra avoir des spécifications non-bornées a priori qui complétées par différentes bornes aboutiront à une réalisation effective. De plus, comme pour le programme PROLOG, on souhaite pouvoir raffiner le système à l'aide d'un dictionnaire réalisant la spécification « Chercher $(a_1, \ldots, a_n)$ de A^n appartenant au dictionnaire. ». Il nous faudra donc exhiber un assemblage de systèmes réalisant la conjonction des spécifications des systèmes assemblés.

2. Choix d'une classe de réseaux de neurones

Il nous faut maintenant choisir une classe de réseaux parmi laquelle on cherchera une réalisation à une spécification. Nous nous proposons de construire ces réseaux à l'aide de neurones discrets pouvant choisir leur état parmi deux possibles, 0 et 1, et évoluant de manière non-déterministe. Contrairement aux machines de BOLTZMANN ce non-déterminisme ne porte pas sur la direction d'évolution de l'état x_i d'un neurone i : a tout instant , ce neurone reçoit un signal u_i qui *détermine* son évolution comme suit[2] :

$$x_i \leftarrow f(u_i)$$

où f désigne la fonction de seuillage définie par

$$f(u) = \begin{cases} 1 & \text{si } u > 0 \\ 0 & \text{sinon} \end{cases} .$$

Un neurone dont l'état vérifie $x_i = f(u_i)$ ne peut changer d'état et est donc à l'équilibre. Le non-déterminisme envisagé porte uniquement sur le temps nécessaire à une transformation, c'est-à-dire que seule est inconnue la durée d'une situation instable où $x_i \neq f(u_i)$. On suppose qu'une telle situation ne peut durer indéfiniment : au bout d'un temps fini soit l'état du neurone bascule soit son signal d'entrée devient tel que ce changement n'a plus lieu d'être. On suppose en outre que la durée d'une telle instabilité ne dépend pas de l'histoire du neurone.

Lorsqu'on considère un ensemble de n neurones on obtient un système ayant $\{0,1\}^n$ pour espace d'états. A un instant donné où les signaux reçus par les différents neurones sont résumés par le vecteur $(u_1, \ldots, u_n)$, toutes les transitions possibles de ce système sont de la forme :

$$(x_1, \ldots, x_n) \to (y_1, \ldots, y_n) \quad \text{avec} \quad \begin{cases} y_i & = f(u_i) \\ & \text{ou} \\ y_i & = x_i \end{cases} .$$

On obtient maintenant un réseau de neurones en assujétissant leurs évolutions : on les relie de telle façon que le signal reçu par un neurone soit une combinaison linéaire de l'état des neurones auxquels il est connecté. Un tel réseau est défini par une matrice de connexion $(\omega)_{ij}$ complétée par un vecteur de seuils ω_i. Le signal u_i reçu par un neurone i du réseau se trouvant dans l'état $(x_1, \ldots, x_n)$ est alors donné par :

$$u_i = \sum_j \omega_{ij} x_j + \omega_i$$

[2]On s'arrangera par la suite à ne jamais donner une valeur nulle au signal d'entrée d'un neurone de façon à ne pas dépendre de la valeur choisie pour la fonction de seuillage f en 0.

Le graphe de transitions du réseau est alors entièrement défini. Le non-déterminisme de la durée d'instabilité d'un neurone isolé introduit des bifurcations non-déterministes de la trajectoire suivie par un ensemble de neurones. Le choix de l'une ou l'autre direction est non relié à l'histoire du système: on peut donc modéliser le système à l'aide d'un processus de MARKOV. Les états terminaux de ce processus sont les points $x = (x_1, \ldots, x_n)$ où toutes les cellules sont à l'équilibre (*i.e.* pour tout neurone i, son état x_i est égal à 1 si et seulement si son signal d'entrée $u_i(x)$ est strictement positif).

Pour garantir que toute évolution du réseau aboutisse presque sûrement à un point d'équilibre au bout d'un temps fini, il suffit d'exhiber pour tout état x^o du réseau un chemin fini $x^o, \ldots, x^n$ aboutissant à un point d'équilibre x^n et accessible à la dynamique du réseau. Ce chemin doit donc être tel que l'on ait, pour tout indice t de $\{1, \ldots, n\}$ et tout neurone i:

$$x_i^{t+1} = \begin{cases} f(u_i(x^t)) \\ \quad \text{ou} \\ x_i \end{cases} .$$

Et le point x^n devant être un point d'équilibre, on doit aussi avoir pour tout neurone i:

$$x_i^n = f(u_i(x^n)) \quad .$$

Pour obtenir un système au sens de la section précédente il nous reste à interpréter les états du réseau à l'aide d'un code dans un espace de recherche. De façon à faciliter les assemblages de réseaux, nous allons donner différents rôles aux neurones d'un réseau. Nous distinguerons ainsi des neurones de contrôle, dont l'état sera invariant au cours de toute évolution du système et permettra de coder un élément d'information relatif à la commande extérieure du système. Nous distinguerons de même des neurones réponses, dont l'état codera un élément d'information relatif au point de l'espace de recherche désigné par le réseau. Nous serons aussi amenés à ajouter des neurones supplémentaires, ne jouant aucun rôle direct vis à vis de l'interprétation. Ces derniers sont nécessaires ne serait-ce qu'en raison du fait que toutes les fonctions booléennes ne sont pas réalisables à l'aide d'un seul neurone. Nous leur donnerons aussi un rôle plus essentiel de contrôle interne. Par exemple lorsqu'il faut « chercher x de E vérifiant $\mathcal{P}(x)$ ou $\mathcal{Q}(x)$ » un axe de recherche doit être choisi et l'on doit s'y tenir avec une certaine constance (du moins, si l'on veut être efficace). En raison de ce rôle, nous qualifierons de neurones mémoires ou de contrôle interne ces neurones supplémentaires (et non de neurones cachés comme il en est l'usage).

L'ensemble des neurones d'un réseau est donc divisé en trois parties: l'ensemble C des neurones de contrôle, celui R des neurones réponses et enfin M regroupant tous les neurones n'ayant qu'un rôle interne. L'état des neurones de contrôle

devant être constant au cours de toute évolution, ils n'ont pas de connexion afférente issue des autres neurones du réseau. La figure IV.13 donne une image d'un tel réseau.

Le code c utilisé pour l'interprétation de l'état du réseau est donc tel que $c(x)$ soit égal à $c(y)$ si les deux états x et y ne différent que par l'état de neurones ne faisant pas partie de l'ensemble R, appelé aussi support de la réponse. L'interprétation de la commande du réseau vérifie elle aussi cette propriété relativement à l'ensemble C, support de la commande. Lorsque l'espace de recherche (ou de commande) est de la forme $E \times F$, nous allons de même distinguer parmi les neurones de R (respectivement de C) des neurones support de l'information relative à E et de même relativement à F. Ainsi il suffira de connaître uniquement l'état de ces neurones support de E, pour connaître la projection sur E du point de l'espace de recherche désigné par le réseau (ou le point de l'espace de commande imposé par l'extérieur). L'intérêt d'un tel codage de l'information apparaît lors d'un assemblage : il est alors facile de rendre compte du fait que les deux réseaux travaillent sur des données communes (en d'autre termes : que leur spécifications portent sur des informations communes). L'assemblage de deux réseaux sera ainsi constitué des neurones des deux systèmes dont on ne prendra qu'une seule copie lorsqu'ils jouent un rôle identique dans les deux réseaux. On ajoutera aussi d'éventuels neurones de contrôle interne permettant de coordonner le travail des deux constituants de l'assemblage.

Donnons maintenant l'image d'un réseau de neurones candidat à la réalisation du système de lecture que nous avons initialement spécifié en PROLOG, puis en termes de recherche dans l'espace produit $I^n \times A^n$ des couples d'une décomposition d'une image en image partielles et d'une suite de lettres. A chaque élément d'information (lettre ou image partielle) est associé un ensemble de neurones (*cf* figure IV.14). La satisfaction des différentes contraintes est contrôlée par des

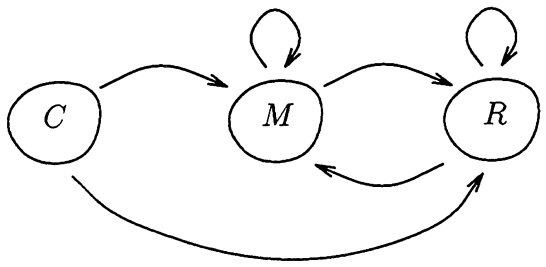

FIG. IV.13 - . Première ébauche de la structure d'un réseau. Les *flèches* indiquent l'existence possible de connections entre les neurones des différentes partie du réseau.

neurones supplémentaires représentés par des *flèches* exprimant l'influence qu'ils vont avoir sur les états du réseau. On souhaite pouvoir obtenir ce réseau par assemblages des réseaux de la figure IV.15.

3. Décompositions et assemblages

Regardons maintenant le problème de la décomposition d'une spécification en spécifications partielles. Notre but est d'obtenir un système réalisant cette spécification par assemblage de réseaux réalisant ces spécifications partielles. Ceci restreint notre champ d'investigation : en effet, une décomposition de spécification ne présente un intérêt dans ce cadre que si l'on dispose d'un assemblage de systèmes associé. Nous allons donc procéder indirectement, en regardant quels sont les assemblages de systèmes que l'on peut envisager et les constructions de spécifications induites. Il est temps de préciser ce que l'on désigne exactement par assemblage de systèmes. C'est l'obtention, par un procédé algorithmique, de la description d'un système à partir de la description de systèmes constituants.

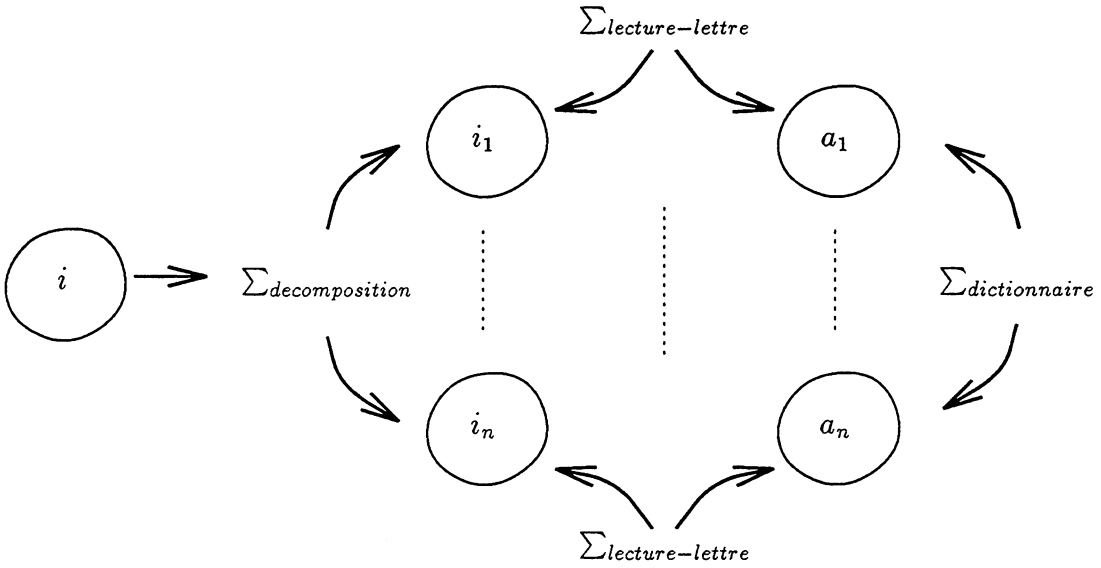

FIG. IV.14 - . Décomposition du problème de la lecture d'un mot. Un système *dictionnaire* vérifie la présence du mot lu dans le dictionnaire et le transforme si nécessaire. Un système de *décomposition* maintient l'adéquation entre l'image de référence et les n images partielles. Les systèmes de *lecture d'une lettre* sont eux chargés de trouver une image partielle ainsi qu'une lettre de l'alphabet pouvant être mis en correspondance.

Nous avons défini un système comme un automate fini dont les états sont interprétés à l'aide d'un code dans un espace de recherche. Il nous faut donc construire l'ensemble d'états du système assemblé, son graphe de transitions ainsi que son interprétation. La construction de l'espace d'états et celle de son interprétation vont de paire et vont induire la construction d'un espace de recherche. La construction du graphe de transitions va elle influer sur la propriété que doivent vérifier les réponses du système. Un assemblage de systèmes induit donc bien une construction de spécification. Quels sont les assemblages de systèmes que l'on peut réaliser, et par suite quelles sont les décompositions que l'on pourra envisager pour une spécification?

La difficulté lors de l'assemblage de systèmes est que l'on ne peut pas toujours manipuler directement le graphe de transitions. Ainsi dans le cadre des réseaux de neurones celui-ci est induit par un graphe de connections entre neurones et sa description complète pose des difficultés techniques ne serait-ce que pour un problème de taille : pour un réseau de n neurones binaires le système comporte 2^n états. Par ailleurs lorsque deux sous-systèmes partagent une information commune, on est confronté à un problème similaire à celui de l'accès simultané en écriture que l'on rencontre dans le domaine du calcul parallèle. Suivant les réponses apportées à ce problème les graphes de transitions obtenus vont différer. En outre, seuls les états terminaux nous intéressent réellement, la satisfaction d'une spécification par un système reposant sur eux. Nous ne nous attacherons donc pas à décrire la totalité du graphe de transitions mais uniquement la construction de l'espace d'états et des états terminaux. Les définitions d'assemblage que nous allons donner seront donc partielles, plusieurs réseaux pouvant y répondre, et non constructives. On parlera donc plutôt de formes d'assemblage. N'imposant pas un graphe de transition, elles auront l'avantage de pouvoir être réalisées de différentes manières suivant les choix faits pour les mécanismes sous-jacents aux systèmes.

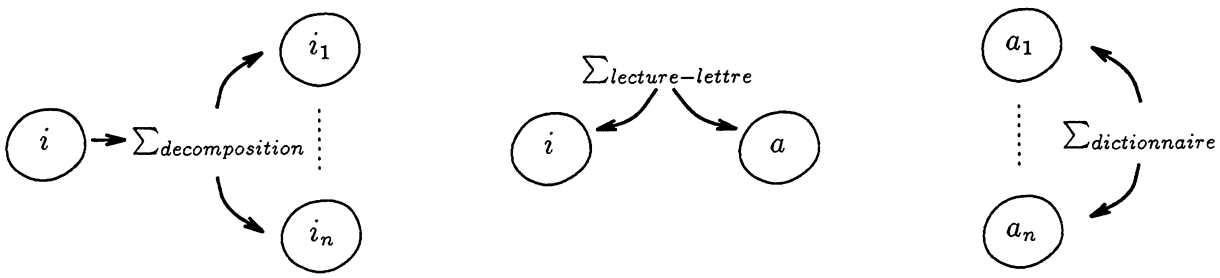

FIG. IV.15 - . Les trois éléments constituant le système de lecture. Chacun code une contrainte et l'on souhaite pouvoir obtenir par assemblage, un système vérifiant simultanément ces contraintes.

Outre la définition de chaque assemblage, nous allons nous attacher à l'interprétation que l'on peut donner aux états du système obtenu en fonction de l'interprétation des systèmes assemblés ainsi qu'à la combinaison des propriétés définissant leur spécifications. On obtiendra ainsi une description de la spécification à laquelle peut répondre l'assemblage, celle dont les solutions coïncident avec les réponses du système. Les définitions données pour chaque assemblage étant partielles, on ne pourra garantir qu'il réalise effectivement cette spécification candidate. En effet, ne connaissant pour l'instant pas son graphe de transitions on ne peut garantir qu'il donnera toujours une réponse.

3.1. Juxtaposition

Commençons par un assemblage, la juxtaposition de systèmes, permettant d'induire un espace de recherche produit des espaces de recherche partiels. Cette juxtaposition consiste uniquement à faire évoluer les différents systèmes en parallèle, sans synchronisation.

DÉFINITION. *Étant donnés deux systèmes ayant respectivement P et Q pour ensemble d'états,*

la juxtaposition *de ces deux systèmes est un système ayant $P \times Q$ pour ensemble d'états et dont l'ensemble des états terminaux est le produit des ensembles des états terminaux des systèmes juxtaposés.*

Cet assemblage est l'assemblage le plus simple que l'on puisse envisager. Il n'impose pas de contrainte de compatibilité aux systèmes assemblés (qui peuvent même utiliser des mécanismes différents) et ne demande l'ajout d'aucun mécanisme supplémentaire : sa réalisation demande uniquement de juxtaposer les deux systèmes. Son graphe de transition peut dès lors être décrit : il comporte toutes les transitions $(p, q) \to (p', q')$ telles que :

$$\begin{aligned} & p \to p' && \text{soit une transition du premier système} \\ \text{et} \quad & q \to q' && \text{soit une transition du second.} \end{aligned}$$

On remarque alors que pour qu'un état terminal de la juxtaposition soit toujours atteignable à partir de tout état initial il suffit que cette propriété soit vraie pour les deux systèmes juxtaposés.

De plus, si les deux systèmes juxtaposés sont respectivement interprétés dans les ensembles E et F à l'aide des codes c et d, alors les états de leur juxtaposition sont directement interprétables dans l'espace produit $E \times F$ à l'aide du code, noté (c, d), qui à (p, q) de $P \times Q$ associe $(c(p), d(q))$ de $E \times F$. On a donc la propriété

suivante :

PROPRIÉTÉ. *Si deux systèmes, respectivement interprétés dans les ensembles E et F à l'aide des codes c et d, réalisent respectivement les spécifications « chercher x de E tel que $\mathcal{P}(x)$ » et « chercher x de F tel que $\mathcal{Q}(x)$ »,*

alors leur juxtaposition, interprétée dans $E \times F$ à l'aide du code (c, d), réalise la spécification « chercher (x, y) de $E \times F$ tel que $\mathcal{P}(x)$ et $\mathcal{Q}(y)$ ».

3.2. Composition

Un autre assemblage ne faisant pas intervenir le mécanisme utilisé pour mettre en œuvre les systèmes est la composition. L'objectif est de réaliser la composition de deux fonctions ou plus généralement de deux relations. Le principe est d'obtenir d'un premier système les informations de contrôle d'un second système dirigé.

DÉFINITION. *Étant donnés un premier système, Σ_0, ayant P pour espace d'états ainsi qu'un second, Σ_1, ayant $P \times Q$ pour espace d'états et qui est dirigé par P,*

la composition *de Σ_0 avec Σ_1 est un système ayant $P \times Q$ comme espace d'états et dont les états terminaux sont de la forme (p, q) où p et (p, q) sont respectivement des états terminaux de Σ_0 et Σ_1.*

Le fait que cet assemblage ne dépende pas des mécanismes utilisés vient du fait que le second système est dirigé par P, c'est-à-dire, rappelons le, qu'il ne modifie pas la projection de son état sur P (ou , en termes plus mécaniques, qu'il n'agit pas sur ses constituants codant l'état relatif àP). Pour réaliser la composition de deux systèmes il suffit donc de réaliser les deux systèmes en ne prenant qu'un seul exemplaire des constituants relatifs à P. Il n'y a alors pas de conflit d'action sur ces constituants. Le graphe de transition de la composition comporte donc toutes les transitions $(p, q) \to (p', q')$ telles que :

$$p \to p' \quad \text{soit une transition du premier système}$$
$$\text{et} \quad (p, q) \to (p, q') \quad \text{soit une transition du second.}$$

On remarque alors que pour qu'un état terminal de la composition soit toujours atteignable à partir de tout état initial il suffit que cette propriété soit vraie pour les deux systèmes composés.

Pour que la composition des deux systèmes soit interprétable il est nécessaire que le premier système soit interprété dans le même espace et de la même façon que la commande du second. Soit c ce code de P dans un ensemble E. On peut

alors utiliser l'interprétation du second, donné par un code d de $P \times Q$ dans un ensemble F, pour coder un état de leur composition : il suffit de prendre le code (c, d) qui à (p, q) de $P \times Q$ associe $(c(p), d(p, q))$ de $E \times F$.

PROPRIÉTÉ. *Étant donné un premier système ayant P pour espace d'états et qui est interprété dans un ensemble E à l'aide d'un code c, étant donné un second système ayant $P \times Q$ pour espace d'états, dirigé par P, dont la commande est interprétée à l'aide du code c de P dans E et dont les réponses sont interprétées dans un ensemble F à l'aide d'un code d.*

Si ces deux systèmes réalisent respectivement les spécifications « chercher x de E tel que $\mathcal{P}(x)$ » et « étant donné x de E chercher y de F tel que $\mathcal{R}(x, y)$ »,

alors leur juxtaposition, interprétée à l'aide du code (c, d), réalise la spécification « chercher (x, y) de $E \times F$ tel que $\mathcal{P}(x)$ et $\mathcal{R}(x, y)$ ».

3.3. Abstraction

L'abstraction n'est pas réellement un assemblage car elle ne fait intervenir qu'un seul système dont on ne modifie que l'interprétation. L'objectif est de ne retenir de la réponse du système modifié qu'une information partielle.

PROPRIÉTÉ (DÉFINITION). *Soit un système ayant un espace d'états de la forme $P \times Q$ et interprété à l'aide d'un code de $P \times Q$ dans un ensemble de la forme $E \times F$. On suppose que ce code respecte la forme produit des deux espaces, d'états et de recherche, c'est-à-dire qu'à (p, q) il associe $(c(p), d(q))$ où c et d sont respectivement des codes de P dans E et de Q dans F.*

*L'*abstraction *sur E de ce système est le même système mais interprété dans E à l'aide du code qui à (p, q) de $P \times Q$ associe $c(p)$ de E.*

Si le premier système réalise la spécification « chercher (x, y) de $E \times F$ tel que $\mathcal{P}(x, y)$ » alors son abstraction sur E réalise la spécification « chercher x de E tel qu'il existe y de F avec $\mathcal{P}(x, y)$ »

3.4. Superposition conjonctive

La juxtaposition permet d'obtenir un parallélisme de calcul très simplifié : les deux systèmes n'interagissent aucunement. Lors d'une composition les deux systèmes partagent des informations mais un seul modifie cette information commune. La superposition conjonctive est une forme d'assemblage permettant d'exprimer que les sous-systèmes assemblés travaillent simultanément sur des données communes. La superposition de deux systèmes ayant un même ensemble d'états

Q, est ainsi un système ayant aussi Q pour ensemble d'états. L'objectif de cet assemblage est d'obtenir un système dont les réponses satisfassent les deux constituants. Un état ne doit donc être un état terminal d'une superposition conjonctive que si il est un état terminal pour les deux systèmes assemblés.

DÉFINITION. *Étant donnés deux systèmes ayant tous deux Q pour espace d'états, leur* superposition conjonctive *est un système ayant Q pour espace d'état et tel que l'ensemble de ses états terminaux soit égal à l'intersection des ensembles d'états terminaux des deux systèmes superposés.*

Pour qu'un tel assemblage puisse être interprété, il est nécessaire que les états des deux systèmes assemblés soient interprétés dans un même ensemble E à l'aide d'un même code c. On utilise alors le même code pour l'assemblage et l'on a le résultat immédiat suivant :

PROPRIÉTÉ. *Étant donnés deux systèmes ayant tout deux Q pour espace d'états, tout deux interprétés dans un même ensemble E à l'aide d'un code c et réalisant respectivement les spécifications « chercher x de E tel que $\mathcal{P}(x)$ » et « chercher x de E tel que $\mathcal{Q}(x)$ », on considère leur superposition conjonctive interprétée suivant le code c.*

Si celle-ci est telle qu'un état terminal soit accessible de tout point de départ,

alors cette superposition réalise la spécification « chercher x de E tel que $\mathcal{P}(x)$ et $\mathcal{Q}(x)$ ».

Notons que garantir que le système assemblé donnera toujours une réponse est un problème difficile. En effet, les graphes de transitions des systèmes assemblés ne donnent pas nécessairement suffisamment d'information. Il suffit de prendre l'exemple suivant pour s'en convaincre :

EXEMPLE. Considérons un système ayant $\{a, b, c\}$ comme ensemble d'états, dont l'unique transition est $a \to b$ et dont les états codant une réponse sont donc b et c. Le deuxième système considéré sur le même ensemble d'états a lui aussi une seule transition, $b \to a$, et ses états codant une réponse sont ainsi a et c. Toute l'information dont dispose le système obtenu par superposition des deux premiers est donc contenu dans les deux transitions $a \to b$ et $b \to a$ qui ne permettent pas de trouver l'unique réponse possible c, si le point de départ est a ou b.

Terminons par une remarque. Le fait d'imposer aux deux systèmes superposés un même ensemble d'états n'est pas restrictif en pratique. Par exemple, supposons que l'on veuille superposer un système ayant $P \times Q$ pour espace d'états (la

projection sur P codant un élément d'un ensemble E, celle sur Q un élément d'un ensemble F) à un système ayant $Q \times R$ pour espace d'états (et dont la projection sur Q est elle aussi interprétée dans F, et celle sur R dans un ensemble G). On souhaite donc exprimer par cette superposition que les deux systèmes partagent une information commune prenant ses valeurs dans F. Pour se ramener dans le cadre de notre définition, il suffit de transformer le premier système par juxtaposition avec un système ayant R comme espace d'états et dont tous les états sont terminaux. On obtient ainsi un système intermédiaire interprété dans $E \times F \times G$ dont les réponses projetées dans $E \times F$ correspondent à celles du premier système et n'imposant aucune restriction à la projection sur G. En prolongeant de manière similaire le second système, on obtient alors la superposition souhaitée.

3.5. Superposition alternative

La superposition alternative est une forme d'assemblage qui permet de coder l'union disjointe de deux ensembles ou de réaliser la disjonction de deux spécifications sur un même ensemble. Le principe est de construire, à partir de deux systèmes ayant un même ensemble d'états Q, un système ayant pour espace d'états l'ensemble $\{0,1\} \times Q$ et dont l'évolution au cours du temps reflète l'un ou l'autre des systèmes superposés, cette alternative étant contrôlée par l'information supplémentaire prenant la valeur 0 ou 1. La valeur de ce bit de contrôle peut être une donnée du système et on parle alors d'alternative dirigée par l'extérieur. La valeur de ce bit peut être aussi contrôlée par le système-même et on parle alors d'alternative libre. Le but est d'obtenir un système dont les réponses sont des réponses du premier système si le bit de contrôle est à 0 et des réponses du second sinon ; d'où la définition suivante :

DÉFINITION. *Étant donnés deux systèmes,* Σ_0 *et* Σ_1*, ayant tous deux* Q *pour espace d'états, leur* superposition alternative *est un système ayant* $\{0,1\} \times Q$ *pour espace d'états et tel que l'ensemble de ses états terminaux soit égal à l'union disjointe des ensembles d'états terminaux des deux systèmes superposés, c'est à dire l'ensemble des couples de la forme :*

$$\begin{cases} (0,p) & \text{où } p \text{ est un état terminal de } \Sigma_0 \\ (1,p) & \text{où } p \text{ est un état terminal de } \Sigma_1 \end{cases}$$

On distingue deux formes de superposition alternative, celle dirigée par l'extérieur *et la forme* libre. *Une superposition alternative est dite dirigée par l'extérieur si le bit de contrôle a un état invariant au cours des transformations d'état du système, c'est-à-dire si :*

$$\text{pour toute transition d'états } (c,x) \to (d,y) \qquad d = c \quad ;$$

elle sera dite libre dans le cas contraire.

Un état d'une superposition peut être interprété de différentes façons suivant les interprétations des systèmes assemblés. Si les deux constituants sont interprétés à l'aide d'un même code de Q dans un ensemble E, il suffit de reprendre ce code pour le système obtenu par assemblage. Dans un cas plus général, si les deux systèmes assemblés sont interprétés dans un même ensemble E mais à l'aide de deux codes différents c et d, il est nécessaire d'utiliser le code $c \cup d$ de $\{0,1\} \times Q$ dans E défini par :

$$(c \cup d)(f,q) = \begin{cases} c(q) & \text{si } f = 0 \\ d(q) & \text{si } f = 1 \end{cases},$$

et par lequel un élément de E peut ainsi avoir plusieurs représentations.

Enfin, si un état du premier système est interprété dans un espace E à l'aide d'un code c, et un état du second dans un ensemble F à l'aide d'un code d, alors les codes c et d induisent directement un code $c + d$ de $\{0,1\} \times Q$ dans l'union disjointe $E + F$ de E et F. En notant $(0,e)$ (respectivement $(1,f)$) l'élément de $E + F$ obtenu à partir d'un élément e de E (respectivement f de F), ce code $c + d$ est le suivant :

$$(c+d)(f,q) = \begin{cases} (0,c(q)) & \text{si } f = 0 \\ (1,d(q)) & \text{si } f = 1 \end{cases}.$$

Enfin lorsque le système est libre il est nécessaire de pouvoir interpréter la commande : on utilise pour cela le code qui à (f,q) de $\{0,1\} \times Q$ associe f.

Suivant ces deux interprétations différentes et suivant que la superposition soit libre ou dirigée de l'extérieur, on a les propriétés de réalisation suivantes :

PROPRIÉTÉ. *Étant donnés deux systèmes, Σ_0 et Σ_1, ayant tous deux Q pour espace d'états, interprétés dans les ensemble E et F, à l'aide des codes respectifs c et d, et réalisant respectivement les spécifications « chercher x de E tel que $\mathcal{P}(x)$ » et « chercher x de F tel que $\mathcal{Q}(x)$ », on considère leur superposition alternative dirigée de l'extérieur et interprétée suivant le code $c \cup d$ si E et F sont égaux (ou $c + d$ sinon).*

Si celle-ci est telle qu'un état terminal soit accessible de tout point de départ,

alors cette superposition réalise la spécification « Étant donné f de $\{0,1\}$, si $f = 0$ chercher x de E tel que $\mathcal{P}(x)$, sinon chercher x de F tel que $\mathcal{Q}(x)$ ».

PROPRIÉTÉ. *Étant donnés deux systèmes, Σ_0 et Σ_1, ayant tous deux Q pour espace d'états, tous deux interprétés dans un même ensemble E à l'aide des codes respectifs c et d et réalisant respectivement les spécifications « chercher x de E tel*

que $\mathcal{P}(x)$ » et « chercher x de E tel que $\mathcal{Q}(x)$ », on considère leur superposition alternative libre et interprétée suivant le code $c \cup d$ si E et F sont égaux (ou $c+d$ sinon).

Si celle-ci est telle qu'un état terminal soit accessible de tout point de départ,

alors cette superposition réalise la spécification « chercher x de E tel que $\mathcal{P}(x)$ ou x de F tel que $\mathcal{Q}(x)$ » que l'on peut mettre sous la forme plus simple « chercher x de E tel que $\mathcal{P}(x)$ ou $\mathcal{Q}(x)$ », lorsque E et F sont égaux.

4. Réalisation de la superposition conjonctive

La réalisation de la superposition conjonctive est certainement le point le plus complexe de notre démarche : il nous faut coordonner deux systèmes travaillant simultanément sur des données communes. Commençons donc par cet assemblage : il en découlera les principaux choix techniques relatifs aux connexions entre neurones.

Soient deux réseaux, Σ_1 et Σ_2, dont on se propose de construire un assemblage par superposition conjonctive. On suppose donc que ces deux réseaux sont interprétés dans un même espace de recherche (qui ne nous importe pas à ce point du débat) et ce à l'aide d'un même code (qui ne nous intéresse pas plus). Cette hypothèse se traduit comme suit : l'ensemble des neurones support de la réponse du premier réseau peut être mis en correspondance point à point avec celui codant la réponse du second. On ne fera donc pas de distinction entre ces deux ensembles : soit R l'ensemble de neurones support de l'interprétation commune aux deux réseaux. On distingue maintenant l'ensemble M_1 (respectivement l'ensemble M_2) des neurones du premier système (respectivement du second) ne faisant pas partie de R. Ces derniers neurones peuvent être des neurones de contrôle interne ou externe, peu importe : ils ne codent pas d'information commune aux deux réseaux.

Il suffit alors de prendre pour ensemble des neurones de l'assemblage l'union de ces trois ensemble M_1, R et M_2. En effet, les états de cet ensemble de neurone codent alors *toute* l'information relative aux deux réseaux en assurant une cohérence de l'information commune[3] : à un état x de l'assemblage correspond l'état du premier réseau donné par les neurones de M_1 et R (état partiel que l'on notera $x|_{M_1 \cup R}$) et la situation est identique avec le second réseau.

[3]Pour être complet il faudrait aussi confondre les paires de neurones de contrôle externe codant une information de contrôle identique d'un réseau à l'autre. Ce point est nécessaire en pratique pour ne pas avoir à gérer une cohérence d'information dupliquée, mais n'introduit aucune difficulté supplémentaire car ces neurones n'ont pas de connexions afférentes : il suffit de prendre pour l'assemblage une seule copie de chacun d'eux. Nous ignorerons donc ce point.

Il nous reste maintenant à décrire les connexions, entre les différents neurones de $M_1 \cup R \cup R2$, pour aboutir à un réseau $\Sigma_1 \| \Sigma_2$ réalisant une superposition conjonctive de Σ_1 et Σ_2, c'est-à-dire tel que l'on ait pour tout état x :

$$x \text{ est un état d'équilibre de } \Sigma_1 \| \Sigma_2 \text{ ssi } \begin{cases} x|_{M_1 \cup R} \text{ est un état d'équilibre de } \Sigma_1 \\ \quad \text{et} \\ x|_{M_2 \cup R} \text{ est un état d'équilibre de } \Sigma_2 \end{cases}$$

Étant amené à comparer le signal reçu par un neurone dans différents réseaux ainsi que son caractère d'équilibre ou d'instabilité, on adoptera les notations suivantes relatives à un réseau Σ. Étant donné un état x de ce réseau ainsi qu'un de ces neurones i, on note $u_{i,\Sigma}(x)$ le signal reçu par ce neurone lorsque le réseau se trouve dans l'état x. Étant donné un état x du réseau, on note $\mathcal{E}_\Sigma(x)$ l'ensemble des neurones se trouvant à l'équilibre lorsque le réseau se trouve dans l'état x :

$$\mathcal{E}_\Sigma(x) = \{\text{neurone } i \text{ de } \Sigma \text{ tq} \quad x_i = f(u_{i,\Sigma}(x))\}$$

En particulier, on peut reformuler la réalisation par $\Sigma_1 \| \Sigma_2$ d'une superposition conjonctive de Σ_1 et Σ_2 comme suit :

Pour tout état x de $\Sigma_1 \| \Sigma_2$,

$$\mathcal{E}_{\Sigma_1 \| \Sigma_2}(x) = M_1 \cup R \cup M2 \quad \text{ssi} \quad \begin{cases} \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R}) & = M_1 \cup R \\ & \text{et} \\ \mathcal{E}_{\Sigma_2}(x|_{M_2 \cup R}) & = M_2 \cup R \end{cases}$$

Commençons par les connexions afférentes aux neurones de M_1 et M_2. Comme nous ne changeons pas l'interprétation des états du réseau, le comportement de ces neurones peut être le même avant et après assemblage. Il suffit donc de prendre les mêmes connexions : la connexion dans $\Sigma_1 \| \Sigma_2$ d'un neurone j de $M_1 \cup R$ (respectivement de $M_2 \cup R$) vers un neurone i de M_1 est prise identique à celle liant le neurone j à i dans Σ_1 (respectivement dans Σ_2). De plus on n'ajoute aucune connexion issue de neurones de M_1 vers des neurones de M_2 et réciproquement ni de M_2 vers M_1. Ainsi lorsque le réseau $\Sigma_1 \| \Sigma_2$ est dans un état x, un neurone de M_1 reçoit un signal identique à celui qu'il aurait reçu avec l'état $x|_{M_1 \cup R}$ dans Σ_1 (et réciproquement, il en est de même pour un neurone de M_2) :

$$\begin{aligned} u_{i,\Sigma_1 \| \Sigma_2}(x) &= u_{i,\Sigma_1}(x|_{M_1 \cup R}) \quad \text{pour tout } i \text{ de } M_1 \\ u_{i,\Sigma_1 \| \Sigma_2}(x) &= u_{i,\Sigma_2}(x|_{M_2 \cup R}) \quad \text{pour tout } i \text{ de } M_2 \quad . \end{aligned}$$

Le comportement obtenu pour les neurones de M_1 et M_2 est donc bien similaire à celui donné par les deux systèmes assemblés. En particulier on a donc, pour tout état x de $\Sigma_1 \| \Sigma_2$:

$$\begin{gathered} M_1 \cap \mathcal{E}_{\Sigma_1 \| \Sigma_2}(x) = M_1 \cap \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R}) \\ \text{et} \\ M_2 \cap \mathcal{E}_{\Sigma_1 \| \Sigma_2}(x) = M_2 \cap \mathcal{E}_{\Sigma_2}(x|_{M_2 \cup R}) \quad . \end{gathered}$$

Occupons nous maintenant des connexions afférentes aux neurones de R. Il est suffisant qu'elles soient telles qu'un neurone i de R soit à l'équilibre pour un état x de $\Sigma_1||\Sigma_2$ si et seulement si ce même neurone est à l'équilibre pour l'état $x|_{M_1 \cup R}$ du premier réseau et à l'équilibre pour l'état $x|_{M_2 \cup R}$ du second :

$$i \in \mathcal{E}_{\Sigma_1||\Sigma_2}(x) \quad \text{ssi} \begin{cases} i \in & \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R}) \\ & \text{et} \\ i \in & \mathcal{E}_{\Sigma_2}(x_{M_2 \cup R}) \end{cases} ,$$

ou en d'autres termes :

$$f(u_{i,\Sigma_1||\Sigma_2}(x)) = x_i$$
$$\text{ssi}$$
$$f(u_{i,\Sigma_1}(x|_{M_1 \cup R})) = f(u_{i,\Sigma_2}(x|_{M_2 \cup R})) = x_i \quad .$$

Cette simplication a l'avantage considérable d'aborder la construction de façon indépendante sur chacun des neurones de R. Mais si il est facile d'imposer l'équilibre à un neurone de l'assemblage dès lors que ce neurone est à l'équilibre pour les deux réseaux de départ, la réciproque n'est pas évidente à obtenir. En effet, donnons au neurone i toutes les connexions lui afférant dans le premier et le second réseau, de sorte que l'on ait :

$$u_{i,\Sigma_1||\Sigma_2}(x) = u_{i,\Sigma_1}(x|_{M_1 \cup R}) + u_{i,\Sigma_2}(x|_{M_2 \cup R}) \quad .$$

On a alors bien l'implication recherchée : si $u_{i,\Sigma_1}(x|_{M_1 \cup R})$ et $u_{i,\Sigma_2}(x|_{M_2 \cup R})$ sont tous deux positifs (respectivement tous deux négatifs) alors $u_{i,\Sigma_1||\Sigma_2}(x)$ est positif (respectivement négatif). Mais la réciproque n'est pas vraie : le neurone i peut être à l'équilibre pour l'assemblage sans être à l'équilibre pour l'un des constituants. En utilisant une autre combinaison linéaire des poids de connexion initiaux [4] le problème reste identique : le signal émis par l'un des deux réseaux peut masquer le signal émis par l'autre.

Pour échapper à cette difficulté, qui n'apparaît que lorsqu'il y désaccord entre les deux réseaux, il serait souhaitable que les deux réseaux constituants soient tels que l'on puisse écrire :

$$f(u_{i,\Sigma_1||\Sigma_2}(x)) = f(u_{i,\Sigma_1}(x|_{M_1 \cup R})) \quad \text{si} \quad i \in \mathcal{E}_{\Sigma_2}(x|_{M_2 \cup R})$$
$$\text{et de même :}$$
$$f(u_{i,\Sigma_1||\Sigma_2}(x)) = f(u_{i,\Sigma_2}(x|_{M_2 \cup R})) \quad \text{si} \quad i \in \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R}) \quad .$$

Cela revient à dire que si un réseau est satisfait de l'état d'un neurone alors le signal qu'il lui envoie est suffisamment faible pour ne pas lui masquer d'éventuels signaux demandant un changement d'état. Pour imposer un état à un neurone, support de la réponse, un réseau doit donc mettre en œuvre un mécanisme ayant

[4] Utiliser une combinaison linéaire des poids induit une combinaison linéaire des signaux reçus et facilite donc l'étude du réseau obtenu. Nous n'avons pas essayé d'autres formes de combinaison.

la forme suivante : si le neurone réponse n'est pas dans un état adéquat lui envoyer un signal de forte intensité l'incitant à basculer, puis, lorsque ce changement a eu lieu, lui émettre un signal le plus faible possible de façon à ne pas le bloquer dans ce nouvel état. Nous qualifierons de propriété d'action nulle à l'équilibre ce mécanisme pour lequel nous proposons la définition plus formelle suivante :

DÉFINITION. *Un réseau de neurones Σ, ayant R comme ensemble de neurones supports de la réponse, a la propriété d'*action nulle à l'équilibre *si il est tel que l'on ait pour tout état x du réseau et pour tout neurone i de R :*

$$\bar{R} \subset \mathcal{E}_\Sigma(x) \Rightarrow \begin{cases} u_{i,\Sigma}(x|_{\bar{R}}) = 0 & \text{si } i \in \mathcal{E}_\Sigma(x) \\ f(u_{i,\Sigma}(x|_{\bar{R}})) = f(u_{i,\Sigma}(x)) & \text{sinon} \end{cases} ,$$

où $u_i(x|_{\bar{R}})$ désigne le signal reçu par le neurone i et émis par les neurones ne faisant pas partie de R :

$$u_i(x|_{\bar{R}}) = \sum_{j \notin R} \omega_{ij} x_j \quad .$$

D'une manière générale on notera $u_{i,\Sigma}(x|_I)$ le signal partiel reçu par un neurone i d'un réseau Σ et émis par un sous-ensemble I des neurones du réseau, lorsque celui-ci se trouve dans un état x. Si le réseau est défini par des connexions ω_{ij} et des seuils ω_i, ce signal partiel s'exprime comme suit :

$$u_{i,\Sigma}(x|_I) = \sum_{j \in I} \omega_{ij} x_j \quad (+ \quad \omega_i \quad \text{si } i \in I) \quad .$$

En particulier, si x est un état d'équilibre d'un réseau vérifiant la propriété d'action nulle à l'équilibre, alors le signal partiel $u_i(x|_{\bar{R}})$ est nul pour tout neurone i du support R de la réponse du réseau. Ceci implique que les connexions internes à R doivent suffirent à elles seules pour maintenir un état d'équilibre des neurones supports de la réponse. Les connexions afférentes aux neurones de R et issues de neurones extérieurs à R ont donc pour rôle unique de choisir un de ces points d'équilibre. En d'autres termes, si le réseau est une réalisation d'une spécification « chercher x de E tel que $\mathcal{P}(x)$ », les connexions internes à son support de réponse R ont pour rôle de sélectionner dans l'ensemble des états $\{0,1\}^R$ de ce support ceux codant un élément x de E et les autres connexions du réseau ont pour rôle de choisir l'un de ces élément x vérifiant la propriété $\mathcal{P}$. Cette répartition des rôles va nous permettre de proposer un assemblage réalisant la superposition conjonctive de deux réseaux. Remarquons d'abord que pour que deux réseaux soient candidats à une telle superposition leurs supports de réponse doivent être identiques, ce qui se traduit maintenant par l'existence d'une correspondance bijective entre les neurones de ces deux supports respectant non seulement l'interprétation des états mais aussi les connexions entre ces neurones : deux neurones réponses du premier

réseau doivent être connectés par un lien de même poids que leurs homologues dans le second réseau. On peut alors proposer l'assemblage suivant :

DÉFINITION. *Soient deux réseaux,* Σ_1 *et* Σ_2*, ayant respectivement* $M_1 \cup R$ *et* $M_2 \cup R$ *comme ensemble de neurones. L'ensemble de neurones* R *est leur support commun de réponses. Les connexions du premier réseau (respectivement du second) étant définies par une matrice de connexions* ω^1_{ij} *(respectivement* ω^2_{ij}*) et un vecteur de seuils* ω^1_i *(respectivement* ω^2_i*), on suppose que l'on a les égalités de connexion suivantes :*

$$\begin{aligned} &\forall i,j \in R \qquad \omega^1_{ij} = \omega^2_{ij} \\ &\forall i \in R \qquad\quad \omega^1_i = \omega^2_i \quad , \end{aligned}$$

exprimant que les deux réseaux travaillent dans un même espace.

On définit alors le réseau $\Sigma_1 \| \Sigma_2$, parallélisation *de* Σ_1 *et* Σ_2*, par l'ensemble de neurones* $M_1 \cup R \cup M_2$ *et les connexions* ω_{ij} *ainsi que les seuils* ω_i *donnés par :*

$$\begin{aligned}
&\forall i \in M_1 \quad \forall j \in M_1 \cup R \qquad \omega_{ij} = \omega^1_{ij} \\
&\forall i \in M_2 \quad \forall j \in M_2 \cup R \qquad \omega_{ij} = \omega^2_{ij} \\
&\forall i \in R \quad \forall j \in M_1 \qquad \omega_{ij} = \omega^1_{ij} \\
&\forall i \in R \quad \forall j \in M_2 \qquad \omega_{ij} = \omega^2_{ij} \\
&\forall i \in R \quad \forall j \in R \qquad \omega_{ij} = \omega^1_{ij} = \omega^2_{ij} \\
&\forall i \in M_1 \qquad\qquad\qquad \omega_i = \omega^1_i \\
&\forall i \in M_2 \qquad\qquad\qquad \omega_i = \omega^2_i \\
&\forall i \in M_1 \qquad\qquad\qquad \omega_i = \omega^1_i = \omega^2_i \quad .
\end{aligned}$$

Cette parallélisation consiste donc simplement à superposer les deux réseaux sans dupliquer les matériaux communs (neurones ou connexions). On a alors la propriété suivante :

PROPRIÉTÉ. *Soient deux réseaux,* Σ_1 *et* Σ_2*, tels que leur parallélisation* $\Sigma_1 \| \Sigma_2$ *soit définie.*

Si ces deux réseaux ont la propriété d'action nulle à l'équilibre,

alors $\Sigma_1 \| \Sigma_2$ *est une superposition conjonctive de* Σ_1 *et* Σ_2*. De plus cette superposition a la propriété d'action nulle à l'équilibre.*

La preuve de cette propriété est relativement longue mais simple : l'assemblage

a été choisi de manière à pouvoir faire la preuve presque indépendamment sur chaque neurone. Commençons par traduire les liens entre les connexions des différents réseaux, avant et après assemblage, en termes de liens entre les signaux émis dans ces réseaux. On a pour tout état x du réseau $\Sigma_1 \| \Sigma_2$:

$$\begin{array}{lll}
\forall i \in M_1 & u_{i,\Sigma_1\|\Sigma_2}(x) & = u_{i,\Sigma_1}(x|_{M_1 \cup R}) \\
\forall i \in M_2 & u_{i,\Sigma_1\|\Sigma_2}(x) & = u_{i,\Sigma_2}(x|_{M_2 \cup R}) \\
\forall i \in R & u_{i,\Sigma_1\|\Sigma_2}(x|_{M_1}) & = u_{i,\Sigma_1}(x|_{M_1}) \\
& u_{i,\Sigma_1\|\Sigma_2}(x|_R) & = u_{i,\Sigma_1}(x|_R) \\
& & = u_{i,\Sigma_2}(x|_R) \\
& u_{i,\Sigma_1\|\Sigma_2}(x|_{M_2}) & = u_{i,\Sigma_2}(x|_{M_2}) \quad ,
\end{array}$$

ces derniers signaux partiels vérifiant par définition :

$$\begin{array}{lll}
\forall i \in R & u_{i,\Sigma_1\|\Sigma_2}(x) & = u_{i,\Sigma_1\|\Sigma_2}(x|_{M_1}) \\
& & + u_{i,\Sigma_1\|\Sigma_2}(x|_R) \\
& & + u_{i,\Sigma_1\|\Sigma_2}(x|_{M_2}) \\
& u_{i,\Sigma_1}(x|_{M_1 \cup R}) & = u_{i,\Sigma_1}(x|_{M_1}) \\
& & + u_{i,\Sigma_1}(x|_R) \\
& u_{i,\Sigma_2}(x|_{M_2 \cup R}) & = u_{i,\Sigma_2}(x|_{M_2}) \\
& & + u_{i,\Sigma_2}(x|_R)
\end{array}$$

Si un état x du réseau $\Sigma_1 \| \Sigma_2$ est tel que $x|_{M_1 \cup R}$ et $x|_{M_1 \cup R}$ soient respectivement des états d'équilibre de Σ_1 et Σ_2 on a alors directement pour tout neurone i de M_1 :

$$\begin{aligned}
f(u_{i,\Sigma_1\|\Sigma_2}(x)) &= f(u_{i,\Sigma_1}(x|_{M_1 \cup R})) \\
&= x_i \quad ,
\end{aligned}$$

et de même pour tout neurone i de M_2 :

$$\begin{aligned}
f(u_{i,\Sigma_1\|\Sigma_2}(x)) &= f(u_{i,\Sigma_2}(x|_{M_2 \cup R})) \\
&= x_i \quad .
\end{aligned}$$

En utilisant l'hypothèse d'action nulle à l'équilibre pour Σ_1, on a pour tout neurone i de R :

$$\begin{aligned}
f(u_{i,\Sigma_1\|\Sigma_2}(x)) &= f(u_{i,\Sigma_1}(x|_{M_1}) + u_{i,\Sigma_2}(x|_{R \cup M_2})) \\
&= f(u_{i,\Sigma_2}(x|_{R \cup M_2})) \\
&= x_i \quad .
\end{aligned}$$

Le point x est donc un état d'équilibre du réseau global $\Sigma_1 \| \Sigma_2$.

Réciproquement, si un état x du réseau $\Sigma_1 \| \Sigma_2$ est un état d'équilibre, on a dans un premier temps pour tout neurone i de M_1 :

$$\begin{aligned} f(u_{i,\Sigma_1}(x|_{M_1 \cup R})) &= f(u_{i,\Sigma_1 \| \Sigma_2}(x)) \\ &= x_i \quad , \end{aligned}$$

et de même pour tout neurone i de M_2 :

$$\begin{aligned} f(u_{i,\Sigma_2}(x|_{M_2 \cup R})) &= f(u_{i,\Sigma_1 \| \Sigma_2}(x)) \\ &= x_i \quad . \end{aligned}$$

Ayant donc $\bar{R} \subset \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R})$, on peut utiliser l'hypothèse d'action nulle à l'équilibre de Σ_1. On a ainsi pour tout neurone i de R :

$$i \in \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R}) \Rightarrow u_{i,\Sigma_1}(x|_{M_1}) = 0 \quad .$$

Ce qui entraîne :

$$i \in \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R}) \Rightarrow u_{i,\Sigma_2}(x|_{M_2 \cup R}) = u_{i,\Sigma_1 \| \Sigma_2}(x) \quad ,$$

c'est-à-dire :

$$i \in \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R}) \Rightarrow i \in \mathcal{E}_{\Sigma_2}(x|_{M_2 \cup R}) \quad .$$

Par symétrie des hypothèses, on a de même la réciproque :

$$i \in \mathcal{E}_{\Sigma_2}(x|_{M_2 \cup R}) \Rightarrow i \in \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R}) \quad .$$

Autrement dit, étant donné un état d'équilibre de l'assemblage, si un neurone du support de la réponse est à l'équilibre pour l'un des réseaux assemblés il est aussi à l'équilibre pour l'autre réseau. Il nous reste donc à montrer que l'hypothèse d'équilibre de l'état x pour $\Sigma_1 \| \Sigma_2$ entraîne que tout neurone i de R soit à l'équilibre pour au moins l'un des deux réseaux Σ_1 et Σ_2. Supposons que cela ne soit pas le cas et soit un neurone i qui n'est à l'équilibre ni pour Σ_1 ni pour Σ_2, malgré le fait que x soit un état d'équilibre pour $\Sigma_1 \| \Sigma_2$. Supposons de plus que l'état x_i de ce neurone soit égal à 1 (Le cas $x_i = 0$ donne lieu à une discussion identique.) On a alors la contradiction suivante :

$$u_{i,\Sigma_1 \| \Sigma_2}(x) \quad = u_{i,\Sigma_1}(x|_{M_1}) + u_{i,\Sigma_2}(x|_{M_2 \cup R}) \quad > 0 \quad ,$$

avec

$$\begin{aligned} u_{i,\Sigma_2}(x|_{M_2 \cup R}) \quad & < 0 \quad , \\ u_{i,\Sigma_1}(x|_{M_1}) \quad & < 0 \end{aligned}$$

La dernière inégalité étant obtenue par hypothèse d'action nulle à l'équilibre de Σ_1 qui entraîne :

$$f(u_{i,\Sigma_1}(x|_{M_1})) = f(u_{i,\Sigma_1}(x|_{M_1 \cup R})) \quad ,$$

le fait $\bar{R} \subset \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R})$ étant bien vérifié. Les deux états $x|_{M_1 \cup R}$ et $x|_{M_2 \cup R}$ sont donc respectivement des états d'équilibre de Σ_1 et Σ_2.

Il nous reste à prouver la propriété d'action nulle à l'équilibre de l'assemblage. Soit un état x du réseau $\Sigma_1 \| \Sigma_2$ tel que tous les neurones ne faisant pas partie du support de la réponse soient dans un état d'équilibre :

$$M_1 \cup M_2 \subset \mathcal{E}_{\Sigma_1 \| \Sigma_2}(x) \quad .$$

Ceci induit directement :

$$\begin{aligned} M_1 &\subset \mathcal{E}_{\Sigma_1}(x|_{M_1 \cup R}) \quad , \\ M_2 &\subset \mathcal{E}_{\Sigma_2}(x|_{M_2 \cup R}) \end{aligned}$$

et on peut donc utiliser les hypothèses d'action nulle à l'équilibre de Σ_1 et Σ_2. Soit maintenant un neurone i du support de la réponse R. Si i est à l'équilibre pour $\Sigma_1 \| \Sigma_2$, on déjà vu qu'alors i est aussi à l'équilibre pour Σ_1 et Σ_2 (les neurones de M_1 et M_2 étant à l'équilibre) et l'on a donc :

$$u_{i,\Sigma_1 \| \Sigma_2}(x|_{M_1 \cup M_2}) = u_{i,\Sigma_1}(x|_{M_1}) + u_{i,\Sigma_2}(x|_{M_2}) = 0 \quad .$$

De même si i n'est pas à l'équilibre pour $\Sigma_1 \| \Sigma_2$, on a vu que i est dans un état instable soit pour Σ_1 soit pour Σ_2. On est alors dans l'un des deux cas suivant :

i) équilibre pour l'un (disons Σ_1) et instabilité pour l'autre:

$$\begin{aligned} u_{i,\Sigma_1}(x|_{M_1}) &= 0 \quad , \\ f(u_{i,\Sigma_2}(x|_{M_2})) &= f(u_{i,\Sigma_2}(x|_{M_2 \cup R})) \quad , \end{aligned}$$

d'où :

$$\begin{aligned} f(u_{i,\Sigma_1 \| \Sigma_2}(x|_{M_1 \cup M_2})) &= f(u_{i,\Sigma_1}(x|_{M_1}) + u_{i,\Sigma_2}(x|_{M_2})) \\ &= f(u_{i,\Sigma_1}(x|_{M_1}) + u_{i,\Sigma_2}(x|_{M_2 \cup R})) \\ &= f(u_{i,\Sigma_1 \| \Sigma_2}(x|_{M_1 \cup R \cup M_2})) \quad ; \end{aligned}$$

ii) instabilité pour les deux réseaux Σ_1 et Σ_2 (ce qui entraîne que la fonction de seuillage f a même valeur en $u_{i,\Sigma_1}(x|_{M_1 \cup R})$ et $u_{i,\Sigma_2}(x|_{M_2 \cup R})$:

$$\begin{aligned} f(u_{i,\Sigma_1}(x|_{M_1})) &= f(u_{i,\Sigma_1}(x|_{M_1 \cup R})) \\ &= f(u_{i,\Sigma_2}(x|_{M_2 \cup R})) \\ &= f(u_{i,\Sigma_2}(x|_{M_2})) \end{aligned}$$

d'où, les signaux $u_{i,\Sigma_1}(x|_{M_1})$ et $u_{i,\Sigma_2}(x|_{M_2 \cup R})$ ayant même signe :

$$f(u_{i,\Sigma_1 \| \Sigma_2}(x|_{M_1 \cup M_2})) = f(u_{i,\Sigma_1 \| \Sigma_2}(x|_{M_1 \cup R \cup M_2})) \quad .$$

On a donc bien la propriété d'action nulle à l'équilibre de l'assemblage. □

Le fait que la superposition soit aussi à action nulle à l'équilibre est un point essentiel de notre démarche : on cherche une classe de réseaux stables par construction. On remarquera aussi que les connexions internes à l'ensemble des neurones support de la réponse sont identiques avant et après assemblage. Cet ensemble de neurones ainsi que ses connexions matérialise donc bien le support de l'interprétation du réseau dans un espace de recherche E : on peut utiliser ce sous-réseau pour toutes les spécifications sur cet espace E. On complète alors la réalisation d'une spécification par un sous-réseau ayant une action nulle lorsqu'il est à l'équilibre sur le sous-réseau support de l'information. Les points d'équilibre de ce second sous-réseau peuvent être interprétés en termes de transition dans l'espace de recherche. Plus formellement, l'image d'un réseau devient donc la suivante.

Soit un réseau Σ réalisant une spécification de la forme « chercher x de E vérifiant $\mathcal{P}(x)$ ». On suppose que ce réseau est à action nulle à l'équilibre. Parmi les neurones de ce réseau on distingue deux ensembles de neurones R et M. On distingue aussi le sous-réseau Σ_R constitué des neurones de R et de toutes les connexions de Σ ne portant que sur des neurones de R. Ce sous réseau est le support de la réponse : à chacun des points d'équilibre de Σ_R est associé un point de E. Regardons maintenant le rôle des neurones de M en considérant une évolution du réseau ne portant que sur les neurones de M (Ce qui constitue bien une évolution possible du réseau, bien qu'elle soit artificielle : les neurones évoluent de manière asynchrone.). Partant d'un état quelconque du réseau, on fait ainsi évoluer les neurones de M jusqu'à obtenir un état d'équilibre y de ces neurones. On peut alors utiliser l'hypothèse d'action nulle à l'équilibre au point y : les neurones de M induisent un signal nul sur les neurones de R qui sont à l'équilibre et un signal masquant ceux internes à Σ_R si ces neurones ne sont pas à l'équilibre. En d'autres termes l'état des neurones de M codent alors une transformation à faire subir à l'état des neurones de R. Les différents points d'équilibre atteignables au cours de cette expérience (faire évoluer uniquement les neurones de M) codent donc autant d'alternatives de transformation pour atteindre un point codant une solution à la spécification.

5. Toute spécification est réalisable

Vérifions dés maintenant que l'hypothèse d'action nulle à l'équilibre n'est pas trop restrictive. Il suffit pour cela de montrer que toute spécification admet au moins une réalisation vérifiant cette hypothèse. Le principe de cette démonstration s'appuie sur le fait que toute spécification sur un ensemble fini peut être réduite à une formule booléenne mise sous forme normale conjonctive (CNF). En effet, soit une spécification définie sur un ensemble fini E par une propriété $\mathcal{P}$. E étant fini, il existe un entier n ainsi qu'une fonction surjective c de $\{0,1\}^n$ dans E. On peut alors utiliser un réseau de n neurones pour coder tous les éléments de E à l'aide de ce code c. Par ailleurs, toute partie P de $\{0,1\}^n$ peut être défini

par une application booléenne f de $\{0,1\}^n$ dans $\{0,1\}$ par:

$$(x_1,\ldots,x_n) \in P \Leftrightarrow f(x_1,\ldots,x_n) = 1 \quad .$$

Soit donc f la fonction booléenne définissant les éléments de $\{0,1\}^n$ pour lesquels est associé un élément de E vérifiant la propriété $\mathcal{P}$. Cette fonction f peut être mise sous forme normal conjonctive[5]. Pour réaliser notre spécification, il suffit alors de réaliser dans un premier temps toutes les disjonctions de littéraux qui la définissent, puis de les assembler par superposition conjonctive. On obtient alors un réseau dont les réponses coïncident avec les solutions souhaitées (à toute réponse correspond une solution *via* le code c et réciproquement toute solution est codée par au moins une réponse). Il restera alors a vérifier qu'une réponse sera toujours donnée au bout d'un temps fini (du moins presque sûrement). Le réseau que nous proposons pour réaliser une disjonction de littéraux est le suivant.

PROPRIÉTÉ. *Soit f une disjonction de littéraux sur $\{0,1\}^n$. Cette disjonction est définie, à l'aide de deux parties disjointes P et N de l'ensemble des indices $\{1,\ldots,n\}$, par:*

$$f(x_1,\ldots,x_n) = \sum_{i\in P} x_i + \sum_{i\in N} \bar{x}_i \quad .$$

On associe un réseau à cette disjonction comme suit. L'ensemble des neurones est $\{1,\ldots,n\} \cup c$ où c est un neurone de contrôle et les autres neurones constituent le support de la réponse. Les connexions sont les suivantes:

$$\begin{array}{lll} \forall i \in \{1,\ldots,n\} & \omega_{ii} = 1 & \omega_i = -\dfrac{1}{2} \\ \forall i \in P & \omega_{ic} = -1 & \omega_{ci} = 1 \\ \forall i \in N & \omega_{ic} = 1 & \omega_{ci} = -1 \\ et & \omega_c = \dfrac{1}{2} - |N| & , \end{array}$$

où $|N|$ désigne le cardinal de l'ensemble N.

Ce réseau a alors la propriété d'action nulle à l'équilibre et réalise la spécification « chercher $(x_1,\ldots,x_n)$ tel que $f(x_1,\ldots,x_n) = 1$ ».

La spécification revient à chercher un état $(x_1,\ldots,x_n)$ des neurones réponses tel que l'on ait pour au moins un neurone i:

$$\begin{cases} x_i = 1 & \text{si } i \in P \\ x_i = 0 & \text{si } i \in N \end{cases} \quad .$$

[5] *i.e.* exprimée comme une conjonction de disjonctions de littéraux, un littéral étant une application de la forme $(x_1,\ldots,x_n) \to x_i$ ou de la forme $(x_1,\ldots,x_n) \to \bar{x}_i$

Les connexions du réseau ont donc été choisies de sorte que l'on ait :

i) Si le neurone de contrôle C ce trouve dans l'état nul, alors les neurones de $\{1,\ldots,n\}$ peuvent indifféremment être dans les états 0 et 1 qui sont tous deux stables. De plus si un neurone de P est à 1 ou si un neurone de N est à 0 alors le seul état stable pour le neurone c est 0. Par suite, toute solution est une réponse possible.

ii) Si aucun des neurones de P ne se trouve dans l'état 1 ni aucun neurone de N ne se trouve dans l'état nul, alors le seul état possible pour le neurone de contrôle c est 1. De plus, si le neurone c est à 1, tous les neurones de P ont pour seul état stable 1 et tous les neurones de N ont pour seul état stable 0. Par suite, tout état qui n'est pas une solution, n'est pas une réponse. On remarquera aussi que si un état ne code pas une solution, un chemin existe vers une réponse : le neurone de contrôle passe dans l'état 1, puis à 0 dès qu'au moins un neurone de P est passé en l'état 1 (ou un neurone de N à 0). Le réseau réalise donc bien la spécification citée.

iii) Enfin on remarque que si le neurone de contrôle est à l'équilibre, alors, si l'état du réseau code une réponse, le neurone c (qui se trouve donc en l'état nul) émet un signal nul à tous les neurones du support de la réponse qui sont tous à l'équilibre. Dans le cas contraire, le neurone c est à l'état 1 et émet donc à tout neurone i du support de la réponse un signal nul s'il est à l'équilibre (*i.e.* $i \notin P \cup N$) et un signal déterminant à lui seul l'évolution du neurone si celui si n'est pas à l'équilibre (*i.e.* $i \in P \cup N$). Le réseau a donc bien la propriété d'action nulle à l'équilibre. □

Considérons maintenant le réseau obtenu par superposition conjonctive des m réseaux réalisant, comme plus haut, autant de disjonctions de littéraux f_i sur $\{0,1\}^n$. Les réponses de ce réseau coïncident alors avec les solutions de la conjonction $\prod_i f_i$. Il nous reste a démontrer qu'une réponse est accessible de tout état du réseau, dès lors qu'il existe une solution à la conjonction. On remarquera qu'un tel réseau comprend $n+m$ neurones : un par variable et un par disjonction.

PROPRIÉTÉ. *Soit un réseau obtenu par superposition conjonctive de m réseaux, associés chacun à une disjonction de littéraux sur $\{0,1\}^n$, et dont les réponses coïncident donc avec les solutions d'une CNF f. On suppose que la conjonction obtenue a au moins une solution $(s_1,\ldots,s_n)$. Soit un état $(x_1,\ldots,x_n,c_1,\ldots,c_m)$ du réseau et l'on considère le chemin décrit dans l'espace d'état du système par*

l'algorithme suivant :

pour toute disjonction i satisfaite
$c_i \leftarrow 0$
tant que l'état $(x_1, \ldots, x_n)$ n'est pas une solution de f
{
pour toute disjonction i non satisfaite
$c_i \leftarrow 1$
pour toute variable i littéral d'une disjonction non satisfaite
$x_i \leftarrow s_i$
pour toute disjonction i satisfaite
$c_i \leftarrow 0$
}
pour toute disjonction i
$c_i \leftarrow 0$

Ce chemin est fini, accessible à la dynamique du réseau et aboutit à un état d'équilibre. Par suite le réseau en question réalise la CNF f.

Le chemin décrit est fini : La distance entre l'état courant $(x_1, \ldots, x_n)$ et la solution $(s_1, \ldots, s_n)$ diminue strictement à chaque passage dans la boucle. Le point atteint est évidemment un point d'équilibre. Il reste donc à vérifier que toutes les transitions sont franchissables par le réseau.

i) la transition $c_i \leftarrow 0$ est invoquée uniquement pour des disjonctions satisfaites. Par construction des réseaux réalisant les disjonctions, les neurones c_i associés à ces disjonctions reçoivent alors bien un signal négatif.

ii) la transition $c_i \leftarrow 1$ est invoquée uniquement pour des disjonctions non satisfaites : le signal reçu par le neurone associé est alors bien positif.

iii) la transition $x_i \leftarrow s_i$ est invoquée uniquement pour des variables i qui sont littéraux d'une disjonction non satisfaite. De plus, lors de ces appels, tous les neurones de contrôle c_j associés à des disjonctions satisfaites sont dans l'état 0 et émettent donc un signal nul au neurone i. Considérons toutes les disjonctions non satisfaites et dont la variable x_i est un littéral. Les neurones de contrôle associés à ces disjonctions étant tous dans l'état 1, ils émettent tous un signal au neurone i l'incitant à changer d'état. Si x_i est différent de s_i, la transition est donc franchissable. Dans le cas contraire, le neurone i se trouve dans un état instable dans lequel il doit rester : par hypothèse de non synchronisation entre les neurones, cela est possible. □

Toute spécification admet donc bien au moins une réalisation ayant la propriété

d'action nulle à l'équilibre. Imposer aux réseaux cette propriété d'action nulle, n'est donc pas restrictif. Néanmoins on peut se poser le problème de l'efficacité des systèmes ainsi obtenus : si ils donnent presque sûrement une réponse au bout d'un temps fini, quelle est la durée moyenne d'un calcul? La longueur du chemin vers une réponse, explicitée plus haut, est proportionnelle au nombre de variables mais est dirigée par une information dont le réseau ne dispose pas, à savoir une solution. En pratique le chemin suivi par le réseau sera certainement bien plus sinueux. En effet, si une disjonction, n'est pas satisfaite tous les neurones associés aux variables de cette disjonction peuvent changer d'état et il est fort peu probable que seules les transformations amenant à une solution aient cours.

En d'autres termes, la seule coordination, qu'il y a entre les différents sous-réseaux qui ont été assemblés pour obtenir le système souhaité, se fait par un partage d'information portant uniquement sur un *état* de la recherche et nullement sur les *directions* de recherche prises par chacun. Les sous-réseaux peuvent ainsi arriver à trouver un consensus sur la réponse mais un choix fait de concert sur la direction à prendre dans telle ou telle situation est nécessaire pour être efficace. Pour limiter ce problème il serait souhaitable que chacun des sous-réseaux induise un minimum de transformations à un état ne les satisfaisant pas. Par exemple, il serait souhaitable qu'un réseau réalisant une disjonction de littéraux n'incite qu'un seul neurone, codant une variable, à changer d'état. Avec le réseau proposé plus haut, ce choix est entièrement laissé au non-déterminisme du système. Il serait souhaitable de *guider* ce choix, au moins partiellement. En d'autres termes, lorsque diverses alternatives de transformation se proposent à un réseau il est souhaitable que celui-ci en choisisse une. Nous allons donc aborder le problème de la réalisation d'une superposition alternative avec ce souci.

6. Réalisation de la superposition alternative

Dans le but de pouvoir réaliser facilement une superposition alternative de deux systèmes, nous allons imposer aux réseaux à assembler une forme particulière. On devra bien entendu tenir compte des contraintes déjà imposées par la superposition conjonctive et obtenir après assemblage un réseau ayant la même forme.

Soient deux réseaux, Σ_1 et Σ_2, dont on se propose de construire un assemblage par superposition alternative. On suppose donc que ces deux réseaux ont un même ensemble de neurones supports de la réponse R. On distingue maintenant l'ensemble M_1 (respectivement l'ensemble M_2) des neurones du premier système (respectivement du second) ne faisant pas partie de R. L'ensemble des neurones de M_1, R et M_2 suffit alors pour coder toute l'information relative au deux réseaux. On suppose aussi que ces deux réseaux sont à action nulle à l'équilibre : si les neurones de M_1 ou M_1 sont tous à l'équilibre ils émettent un signal nul à tout

neurone de R qui se trouve à l'équilibre. Cette contrainte nous est imposée par la superposition conjonctive, mais va nous être aussi utile pour la réalisation d'une superposition alternative. Il nous faut, en effet, obtenir un réseau dont les points d'équilibre correspondent à l'un ou l'autre des systèmes assemblés et l'hypothèse d'action nulle à l'équilibre nous donne un moyen pour annuler l'effet sur les neurones réponses de l'un ou l'autre des deux sous-réseaux.

Pour pouvoir facilement annuler l'effet sur la réponse du à un réseau, on lui demande de comporter un neurone de contrôle ayant le double rôle de détecter si il y a lieu ou non de transformer l'état du réseau et de mettre en marche les éventuels mécanismes nécessaires . En inhibant ce seul neurone, on annule alors l'effet de tout le réseau.

DÉFINITION. *Soit un réseau Σ ayant pour ensemble de neurones $R \cup M \cup \{c\}$, R, M et $\{c\}$ étant disjoints. On suppose que R est le support de la réponse de ce réseau. Le neurone c est alors dit* neurone contrôle de l'action *du réseau si et seulement si on a pour tout état x du système :*

$$\begin{cases} x_c = 0 \\ et \\ M \subset \mathcal{E}_\Sigma(x) \end{cases} \Rightarrow \quad \forall i \in R \quad u_{i,\Sigma}(x|_{M\cup\{c\}}) = 0$$

$$x_c = 1 \quad \Rightarrow \quad \mathcal{E}_\Sigma(x) \neq R \cup M \cup \{c\}$$

Si le neurone de contrôle est actif le réseau n'est donc pas à l'équilibre. Si il est inactif et si les neurones de M agissant sur l'état du réseau sont à l'équilibre, alors les neurones de M émettent un signal nul aux neurones de R.

Pour réaliser une superposition alternative de deux réseaux à action libre à l'équilibre et disposant d'un neurone contrôle de leur action, il suffit donc d'ajouter un neurone inhibant l'un ou l'autre des deux neurones de contrôle :

PROPRIÉTÉ. *Soient deux réseaux Σ_1 et Σ_2, ayant respectivement $M_1 \cup R \cup \{c_1\}$ et $M_2 \cup R \cup \{c_2\}$ comme ensemble de neurones. L'ensemble de neurones R est leur support commun de réponses. Les neurones c_1 et c_2 sont des neurones contrôlant l'action respectivement de Σ_1 et Σ_2. Les connexions du premier réseau (respectivement du second) étant définies par une matrice de connexions ω^1_{ij} (respectivement ω^2_{ij}) et un vecteur de seuils ω^1_i (respectivement ω^2_i), on suppose de plus que l'on a les égalités de connexion suivantes :*

$$\begin{aligned} \forall i,j \in R \qquad & \omega^1_{ij} = \omega^2_{ij} \\ \forall i \in R \qquad & \omega^1_i = \omega^2_i \quad , \end{aligned}$$

exprimant que les deux réseaux travaillent dans un même espace.

On considère alors le réseau Σ construit à partir de $\Sigma_1 \| \Sigma_2$ en ajoutant un neurone c dirigeant l'alternative et connecté aux deux neurones c_1 et c_2 comme suit :

$$\omega_{c_1 c} = U_1 < -[\sum_{i \ tq \ \omega^1_{ic_1} > 0} \omega^1_{ic_1} + \omega^1_{c_1}] \quad ,$$

$$\omega_{c_2 c} = U_2 > \sum_{i \ tq \ \omega^2_{ic_2} > 0} \omega^2_{ic_2} + \omega^2_{c_2} \quad .$$

On modifie aussi le seuil du neurone c_2 :

$$\omega_{c_2} = \omega^2_{c_2} - U_2 \quad .$$

Ce réseau est alors une superposition alternative dirigée par l'état du neurone c des réseaux Σ_1 et Σ_2.

Les connexions supplémentaires ont été choisies de sorte que l'on ait pour tout état x tel que $x_c = 0$:

$$u_{i,\Sigma}(x) = u_{i,\Sigma_1}(x|_{M_1 \cup R \cup \{c_1\}})$$
$$c_2 \in \mathcal{E}_\Sigma(x) \Rightarrow x_{c_2} = 0 \quad ;$$

le rôle des deux systèmes étant inversé lorsque $x_c = 1$.

Si un état x est un état d'équilibre pour Σ, alors l'un des deux neurones c_1 et c_2 est dans un état nul, et par suite le réseau correspondant a une action nulle sur les neurones de R. L'autre réseau est alors libre : toutes les connexions supplémentaires lui afférant émettent des signaux nuls. L'état d'équilibre du réseau global est donc un état d'équilibre de ce sous-réseau. Réciproquement, à tout état d'équilibre d'un des deux sous-réseaux correspond un état d'équilibre de l'assemblage. □

Pour obtenir un réseau assemblage ayant même forme que les réseaux assemblés il reste à ajouter au réseau précédent une cellule de contrôle de l'action. Ceci ne pose pas de problème. Nous ne retiendrons néanmoins pas cet assemblage, la raison étant qu'il y est difficile de contrôler l'alternative : en inhibant l'un des deux sous-réseaux, on ne peut plus s'apercevoir si ce réseau est à l'équilibre ou non. Cela n'est pas gênant pour une alternative dirigée de l'extérieur : dans ce cas seules ont un intérêt les réponses du sous-réseau désigné par l'extérieur. Mais dans le cas d'une alternative libre, les réponses des deux sous-réseaux conviennent et le système doit pouvoir reconnaître la présence d'une réponse du premier sous-réseau même si les actions menées avaient pour but l'obtention d'une réponse du second sous-réseau. Il faut pour cela dissocier les fonctions de détection et

de déclenchement d'une action qui sont confondues dans le neurone de contrôle. Nous proposons donc l'assemblage suivant :

DÉFINITION. *Soient deux réseaux Σ_1 et Σ_2, ayant respectivement $M_1 \cup R \cup \{c_1\}$ et $M_2 \cup R \cup \{c_2\}$ comme ensemble de neurones. L'ensemble de neurones R est leur support commun de réponses. Les neurones c_1 et c_2 sont des neurones contrôlant l'action respectivement de Σ_1 et Σ_2. Les connexions du premier réseau (respectivement du second) étant définies par une matrice de connexions ω^1_{ij} (respectivement ω^2_{ij}) et un vecteur de seuils ω^1_i (respectivement ω^2_i), on suppose de plus que l'on a les égalités de connexions suivantes :*

$$\begin{aligned} \forall i,j \in R \qquad & \omega^1_{ij} = \omega^2_{ij} \\ \forall i \in R \qquad & \omega^1_i = \omega^2_i \quad , \end{aligned}$$

exprimant que les deux réseaux travaillent dans un même espace.

On considère alors le réseau $\Sigma_1 \vee \Sigma_2$ construit à partir de $\Sigma_1 \| \Sigma_2$ comme suit. Le neurone c_1 (respectivement c_2) est dissocié en deux neurones d_1 et a_1 (respectivement d_2 et a_2). On ajoute aussi deux neurones m et c, ce dernier étant destiné a être le neurone contrôlant l'action de l'assemblage $\Sigma_1 \vee \Sigma_2$. Les connexions supplémentaires ou modifiées par rapport à l'assemblage $\Sigma_1 \| \Sigma_2$ sont les suivantes :

$$\begin{aligned} \forall i \in M_1 \cup R \quad & \omega_{d_1 i} = \omega^1_{c_1 i} \quad \omega_{i a_1} = \omega^1_{i c_1} \\ \forall i \in M_2 \cup R \quad & \omega_{d_2 i} = \omega^2_{c_2 i} \quad \omega_{i a_2} = \omega^2_{i c_2} \\ & \omega_{d_1} = \omega^1_{c_1} \quad \omega_{d_2} = \omega^2_{c_2} \end{aligned}$$

ainsi que :

$$\begin{array}{llll} \omega_{cd_1} = 1 & \omega_{cd_2} = 1 & \omega_c = -\dfrac{3}{2} & \\ \omega_{md_1} = -1 & \omega_{md_2} = 1 & \omega_{mm} = 1 & \omega_m = -\dfrac{1}{2} \\ \omega_{a_1 c} = 1 & \omega_{a_1 m} = -1 & \omega_{a_1} = -\dfrac{1}{2} & \\ \omega_{a_2 c} = 1 & \omega_{a_2 m} = 1 & \omega_{a_2} = -\dfrac{3}{2} & \end{array}$$

Les connexions supplémentaires ont été choisies de sorte que l'on ait :

i) le neurone d_1 a un *comportement* dans $\Sigma_1 \vee \Sigma_2$ similaire à celui du neurone c_1 dans Σ_1. Le neurone a_1 a un *effet* dans $\Sigma_1 \vee \Sigma_2$ similaire à celui du neurone c_1 dans Σ_1. De même pour d_2 et a_2 vis à vis de Σ_2.

ii) le neurone c calcule la conjonction des états de c_1 et c_2 et détecte donc s'il y a lieu on non d'entreprendre une action. Si son état est nul, les deux neurones acteurs a_1 et a_2 sont alors inhibés : ils ont pour seul équilibre 0. L'action induite par le réseau sur les neurones réponses est alors nulle. Si son état vaut 1, alors le neurone c inhibe l'un des deux neurones acteurs de sorte que les actions correspondant à un seul des deux sous réseaux soient engagées.

iii) le neurone m choisit l'action à entreprendre si le neurone c détecte la nécessité d'une action. Le mécanisme proposé permet de ne pas proposer deux fois de suite la même réponse si le système est perturbé. Le principe est le suivant : si une réponse est trouvée par l'assemblage et que cette réponse correspond uniquement à un des deux sous-réseaux, alors le neurone m change d'état de sorte qu'en cas d'une perturbation la nouvelle solution recherchée corresponde à l'autre sous-réseau.

On a alors la propriété suivante :

PROPRIÉTÉ. *Soient deux réseaux, Σ_1 et Σ_2, ayant la propriété d'action nulle à l'équilibre et tels que le réseau $\Sigma_1 \vee \Sigma_2$ soit défini.*

alors $\Sigma_1 \vee \Sigma_2$ est une superposition alternative de Σ_1 et Σ_2. Cette superposition a la propriété d'action nulle à l'équilibre et son action est contrôlée par le neurone c. Enfin, si les deux réseaux donnent presque sûrement une réponse en un temps fini, il en est de même pour leur assemblage.

Nous ne considérerons maintenant que des réseaux à action nulle à l'équilibre et disposant d'un neurone de contrôle. Pour que cette classe de réseaux soit stable pour les assemblages que nous avons proposés, il nous faut ajouter un neurone c contrôlant une superposition conjonctive de deux réseaux . Pour cela on duplique les neurones de contrôle des réseaux assemblés en un neurone de détection d_i et un neurone d'action a_i. L'état du neurone de contrôle c calcule la disjonction des états des neurones de détection d_i et chaque neurone d'action a_i calcule la conjonction de l'état de la cellule c et de celui du neurone de détection associé a_i : il y a une action a entreprendre dès lors qu'un des deux réseaux n'est pas à l'équilibre.

6.1. Une autre forme de réalisation pour une CNF

On peut obtenir maintenant une construction réalisant une conjonction d'une disjonction de littéraux par superposition conjonctive de réseaux eux-mêmes obtenus par superposition alternative.

Comme réalisation d'un littéral $(x_1, \ldots, x_n) \to x_i$ on prend le réseau défini sur

l'ensemble de neurones $\{1,\ldots,n\} \cup c$ par les connexions :

$$\forall j \in \{1,\ldots,n\} \quad \omega_{jj} = 1 \quad \omega_j = -\frac{1}{2}$$
$$\omega_{ic} = 1 \qquad \omega_{ci} = -1 \; \omega_c = \frac{1}{2} \quad .$$

Comme réalisation d'un littéral $(x_1,\ldots,x_n) \to \bar{x}_i$ on prend le réseau défini sur l'ensemble de neurones $\{1,\ldots,n\} \cup c$ par les connexions :

$$\forall j \in \{1,\ldots,n\} \quad \omega_{jj} = 1 \quad \omega_j = -\frac{1}{2}$$
$$\omega_{ic} = -1 \qquad \omega_{ci} = 1 \quad \omega_c = \frac{1}{2} \quad .$$

Ces deux réseaux sont à action nulle à l'équilibre et ont c comme neurone de contrôle. On réalise alors une disjonction de m littéraux à l'aide de $(m-1)$ superpositions alternatives. Enfin, on réalise une conjonction de l disjonctions de littéraux par superposition conjonctive des l réseaux réalisant les disjonctions. On a alors la propriété de convergence suivante :

PROPRIÉTÉ. *Un réseau obtenu par superpositions conjonctives de réseaux, eux-mêmes obtenus par superpositions alternatives de littéraux, donne presque sûrement une réponse en un temps fini, si il existe une solution.*

La preuve se fait par induction dans l'ensemble des réseaux ainsi obtenus par superpositions alternatives puis conjonctives de littéraux. On note $\mathcal{R}_{CNF}$ cet ensemble de réseaux. On distingue le sous-ensemble $\mathcal{R}_{DL}$ des réseaux obtenus par superposition alternative de littéraux et le sous-ensemble $\mathcal{R}_{CL}$ des réseaux obtenus par par superposition conjonctive de littéraux. Soit un réseau Σ de $\mathcal{R}_{CNF}$. On est alors dans l'un des trois cas suivant :

$$\Sigma \in \mathcal{R}_{DL}$$
$$\Sigma \in \mathcal{R}_{CL}$$

ou peut être mis sous la forme :

$$\Sigma = \Sigma_0 || (\Sigma_1 \vee \Sigma_2)$$

où

$$\Sigma_0 \in \mathcal{R}_{CNF}$$

et Σ_0 est construit avec une superposition conjonctive de moins que Σ. Et de plus :

$$\Sigma_1 \in \mathcal{R}_{DL}$$
$$\Sigma_2 \in \mathcal{R}_{DL}$$

et sont construits sans littéraux communs.

Remarquons que dans le troisième cas, si Σ admet au moins une solution alors Σ_0 admet une solution et au moins un des deux réseaux $\Sigma_0||\Sigma_1$ ou $\Sigma_0||\Sigma_2$ admet aussi une solution. De plus, en réitérant ce processus de décomposition sur les réseaux Σ_0, $\Sigma_0||\Sigma_1$ et $\Sigma_0||\Sigma_2$, on finit par aboutir uniquement à des réseaux de $\mathcal{R}_{DL}$ ou de $\mathcal{R}_{CL}$. Des trois lemmes suivants, on déduit donc par induction que tout réseau de $\mathcal{R}_{CNF}$ donne presque sûrement une réponse en un temps fini, si il existe une solution.

LEMME. *(germe d'induction)*

Un réseau de $\mathcal{R}_{DL}$ donne presque sûrement une réponse en un temps fini.

LEMME. *(germe d'induction)*

Un réseau de $\mathcal{R}_{CL}$ donne presque sûrement une réponse en un temps fini, si il existe une solution.

LEMME. *(d'induction)*

Étant donnés un réseau Σ_0 de $\mathcal{R}_{CNF}$ ainsi que deux réseaux Σ_1 et Σ_2 de $\mathcal{R}_{DL}$ construits sans littéraux communs.

Si les trois réseaux Σ_0, $\Sigma_0||\Sigma_1$ et $\Sigma_0||\Sigma_2$ donnent presque sûrement une réponse en un temps fini, dès lors qu'il existe une solution,

alors le réseau $\Sigma = \Sigma_0||(\Sigma_1 \vee \Sigma_2)$ donne presque sûrement une réponse en un temps fini, si il existe une solution.

Les deux premiers lemmes sont immédiats. Il reste donc à prouver le troisième. On suppose donc qu'il existe au moins une solution et que cette solution est une réponse de $\Sigma_0||\Sigma_1$. Partant d'un état de Σ il faut exhiber un chemin fini, accessible à la dynamique du réseau et aboutissant à un point d'équilibre. Soit m le neurone de $\Sigma_1 \vee \Sigma_2$ contrôlant le choix de l'une des deux alternatives réalisées par ces réseaux. Si initialement ce neurone m désigne le réseau Σ_1, on fait évoluer

les neurones acteurs de Σ_2 jusqu'à obtenir leur équilibre. En ce point, l'action des neurones acteurs de Σ_2 sur les neurones réponses est nulle. Le réseau Σ est alors équivalent au réseau $\Sigma_0||\Sigma_1$ et, par hypothèse, il existe donc un chemin qui aboutit à un point d'équilibre.

Si le neurone m désigne initialement le réseau Σ_2, il n'est cette fois-ci plus garanti qu'il existe une solution commune à Σ_0 et Σ_2. On cherche donc à changer d'alternative pour se ramener au cas précédent. Pour cela, on fait évoluer les neurones acteurs de Σ_1 jusqu'à obtenir leur équilibre; puis l'on fait de même avec les neurones acteurs de Σ_0. En ce point, l'action des neurones acteurs de Σ_1 sur les neurones réponses est nulle. Et l'action des neurones acteurs de Σ_0 sur les neurones réponses est nulle, si ceux-ci sont à l'équilibre. Toute les transitions du réseau Σ_2 sont alors accessibles à la dynamique du réseau Σ. On fait donc évoluer Σ_2 vers un point d'équilibre. Une fois ce point atteint, les actions des neurones acteurs de Σ_1 et Σ_2 sur les neurones réponses sont nulles le réseau Σ_0 peut donc alors agir librement. On lui fait donc atteindre une de ses solutions. Si celle-ci est aussi une solution de Σ_1 ou Σ_2 on a atteint un point d'équilibre. Si ce n'est pas le cas le neurone m contrôlant l'alternative de la superposition alternative $\Sigma_0||\Sigma_1$ peut changer d'état et on se retrouve ainsi dans le premier cas étudié. Dans tous les cas on obtient donc un point d'équilibre. □

7. Récapitulation

Récapitulons les différents choix que nous avons dû faire pour proposer des assemblages de réseaux qui correspondent aux superpositions conjonctive et alternative.

Nous avons dû nous restreindre à une classe particulière de réseaux: à action nulle à l'équilibre et disposant d'un neurone de contrôle. De plus, pour pouvoir assembler deux réseaux nous avons aussi dû supposer qu'ils aient un support de réponses commun R et des connections entre ces neurones de R identiques. Pour réaliser une spécification sur un ensemble E il faut donc d'abord choisir un réseau

$$\sum_{R \to E}$$

ayant un ensemble R de neurones et dont les points d'équilibre codent les points de E. A ce niveau, il est préférable de s'assurer qu'un point d'équilibre est atteignable depuis tout état initial, de sorte que ce réseau

$$\sum_{R \to E}$$

réalise la spécification élémentaire « chercher x de E ».

Pour construire un réseau réalisant une spécification sur E de la forme « cher-

cher x de E tel que $\mathcal{P}(x)$ », on ajoute au réseau de base

$$\sum_{R \to E}$$

un ensemble de neurones M, un neurone particulier c ainsi qu'un ensemble de connexions dont aucune n'est simultanément issue du support de réponse R et afférente à R. Le réseau obtenu

$$c \to M \sum_{R \to E} \mathcal{P}_E$$

doit être à action nulle sur R et être contrôlé par le neurone c. Le rôle des neurones de M est de choisir, parmi tous les états d'équilibre du réseau support

$$\sum_{R \to E}$$

un état codant un élément de E vérifiant la propriété $\mathcal{P}$. Le neurone c a le double rôle de détecter si il y a lieu ou non de modifier l'état des cellules de R et d'inhiber si nécessaire l'action des neurones de M sur ceux de R. On peut éventuellement ajouter un ensemble C des neurones dont l'état ne dépend pas des autres neurones du réseau et servant ainsi à coder une commande extérieure au réseau interprétée dans un ensemble F. On notera

$$c \to M \overset{C \to F}{\sum_{R \to E}} \mathcal{P}_{F \times E}$$

un tel réseau dirigé par l'extérieur, lorsqu'il réalise la spécification « étant donné y de F, chercher x de E vérifiant $\mathcal{P}(y, x)$ ».

Reprenons maintenant les formes d'assemblage que nous avons proposées et donnons les constructions correspondantes. fflOn note x_i l'état d'un neurone i.

CONSTRUCTION. (juxtaposition)

Étant donnés deux réseaux

$$\Sigma_1 = c \to M \sum_{R \to E} \mathcal{P}_E \quad \text{et} \quad \Sigma_2 = d \to N \sum_{S \to F} \mathcal{Q}_F$$

on construit un réseau noté

$$(\Sigma_1 || \Sigma_2)$$

comme suit :

- On prend tout le matériel des deux réseaux (neurones et connexions).

- On ajoute trois neurones c', d' et α. On donne à c' toutes les connexions issues de c. De même on donne à d' toutes les connexions issues de d.

- On ajoute des connexions de sorte que les neurones c', d' et α calculent les fonctions suivantes :

$$x_{c'} \leftarrow x_c \wedge x_\alpha$$
$$x_{d'} \leftarrow x_d \wedge x_\alpha$$
$$x_\alpha \leftarrow x_c \vee x_d$$

On a alors :

$$(\Sigma_1||\Sigma_2) = \alpha \rightarrow M \cup N \cup \{c, c', d, d'\} \sum_{R \cup S \rightarrow E \times F} \mathcal{P}_E \wedge \mathcal{Q}_F$$

CONSTRUCTION. (composition)

Étant donnés deux réseaux

$$\Sigma_1 = c \rightarrow M \sum_{R \rightarrow E} \mathcal{P}_E \quad \text{et} \quad \Sigma_2 = d \rightarrow N \sum_{S \rightarrow F}^{R \rightarrow E} \mathcal{Q}_{E \times F}$$

on construit un réseau noté

$$(\Sigma_1 > \Sigma_2)$$

comme suit :

- On prend tout le matériel du second réseau (neurones et connexions). Du premier on retient tout le matériel sauf les neurones de R déjà donnés par le second réseau (les connexions afférentes à R premier réseau sont donc connectées aux neurones du second).

- On ajoute trois neurones c', d' et α. On donne à c' toutes les connexions issues de c. De même on donne à d' toutes les connexions issues de d.

- On ajoute des connexions de sorte que les neurones c', d' et α calculent les fonctions suivantes :

$$x_{c'} \leftarrow x_c \wedge x_\alpha$$
$$x_{d'} \leftarrow x_d \wedge x_\alpha$$
$$x_\alpha \leftarrow x_c \vee x_d$$

On a alors :

$$(\Sigma_1 > \Sigma_2) = \alpha \rightarrow M \cup N \cup \{c, c', d, d'\} \sum_{R \cup S \rightarrow E \times F} \mathcal{R}_{E \times F}$$

où $\mathcal{R}_{E \times F}$ est la propriété définie par :

$$\mathcal{R}_{E \times F}(x, y) \quad \text{ssi} \quad \mathcal{P}_E(x) \wedge \mathcal{Q}_{E \times F}(x, y)$$

CONSTRUCTION. (abstraction)

Étant donné un réseau

$$c \to M \sum_{R \cup S \to E \times F} \mathcal{P}_{E \times F} \quad .$$

En ne retenant que l'information donnée par les neurones de R on obtient le réseau

$$c \to M \cup S \sum_{R \to E} \mathcal{R}_E \quad ,$$

où $\mathcal{R}_E$ est la propriété définie par :

$$\mathcal{R}_E(x) \quad \text{ssi} \quad \exists y \in F \quad \mathcal{P}_{E \times F}(x, y)$$

CONSTRUCTION. (superposition conjonctive)

Étant donné deux réseaux

$$\Sigma_1 = c \to M \sum_{R \to E} \mathcal{P}_E \quad \text{et} \quad \Sigma_2 = d \to N \sum_{R \to E} \mathcal{Q}_E$$

on construit un réseau noté

$$(\Sigma_1 || \Sigma_2)$$

comme suit :

- On prend tout le matériel des deux réseaux (neurones et connexions) sans dupliquer les neurones de R et les connexions internes à R.

- On ajoute trois neurones c', d' et α. On donne à c' toutes les connexions issues de c. De même on donne à d' toutes les connexions issues de d.

- On ajoute des connexions de sorte que les neurones c', d' et α calculent les fonctions suivantes :

$$x_{c'} \leftarrow x_c \wedge x_\alpha$$
$$x_{d'} \leftarrow x_d \wedge x_\alpha$$
$$x_\alpha \leftarrow x_c \vee x_d$$

si ce réseau donne presque sûrement une réponse en un temps fini, on a alors :

$$(\Sigma_1 || \Sigma_2) = \alpha \to M \cup N \cup \{c, c', d, d'\} \sum_{R \to E} \mathcal{P}_E \wedge \mathcal{Q}_E$$

CONSTRUCTION. (superposition alternative)

Étant donnés deux réseaux

$$\Sigma_1 = c \to M \sum_{R \to E} \mathcal{P}_E \quad \text{et} \quad \Sigma_2 = d \to N \sum_{R \to E} \mathcal{Q}_E$$

on construit un réseau noté

$$(\Sigma_1 \vee \Sigma_2)$$

comme suit :

- On prend tout le matériel des deux réseaux (neurones et connexions) sans dupliquer les neurones de R et les connexions internes à R.

- On ajoute quatre neurones c', d', α et β. On donne à c' toutes les connexions issues de c. De même, on donne à d' toutes les connexions issues de d.

- On ajoute des connexions de sorte que les neurones c', d', α et β calculent les fonctions suivantes :

$$\begin{aligned} x_{c'} &\leftarrow x_\alpha \wedge x_\beta \\ x_{d'} &\leftarrow x_\alpha \wedge \bar{x_\beta} \\ x_\alpha &\leftarrow x_c \wedge x_d \\ x_\beta &\leftarrow x_\beta \wedge (x_c = x_d) \vee \bar{x_\beta} \wedge (x_c \neq x_d) \end{aligned}$$

On a alors :

$$(\Sigma_1 \vee \Sigma_2) = \alpha \to M \cup N \cup \{c, c', d, d', \beta\} \sum_{R \to E} \mathcal{P}_E \vee \mathcal{Q}_E$$

8. Lien avec les mémoires associatives

La description d'un système et de sa construction par assemblage que nous venons de donner peut être reprise en soulignant un lien avec la notion de mémoire associative. Rappelons qu'une mémoire associative est mécanisme mémorisant des mots ou unités d'information, pouvant compléter une information partielle en exhibant un des mots mémorisés compatibles et pouvant aussi corriger une information incohérente relativement aux mots mémorisés. Par exemple, un dictionnaire associatif est un mécanisme mémorisant l'ensemble des mots d'un dictionnaire et pouvant répondre à une requête de la forme : « Je cherche un mot finissant par la syllabe *tion* » ou « Le mot que je cherche ressemble au mot *ckat* ».

La réalisation de mémoire associative avec un réseau de neurones se fait usuellement de la manière suivante. Les éléments d'information mémorisés sont codés par les points d'équilibre d'un réseau. Une entrée (*i.e.* une information incomplète ou incohérente) est codée par un état initial à partir duquel le réseau va

évoluer vers un état d'équilibre (codant donc l'information complétée voir corrigée). Nous pensons que ce genre de réalisation ne peut être une réponse définitive à la construction de mémoires associatives. Premièrement, une telle mémoire doit pouvoir être utilisée par d'autres parties d'un système. Une requête étant codée par un état initial, un réseau extérieur doit alors pouvoir imposer un état initial au réseau associatif. Ce mécanisme doit donc être décrit, ce qui n'est généralement pas le cas[6].

Deuxièmement, il faut pouvoir coder à partir de l'ensemble des états possibles du réseau toutes les requêtes (*i.e.* informations incohérentes mais aussi informations incomplètes). En pratique, les seules requêtes envisagées sont pour cela toujours complètes : une valeur est supposée pour chacun des éléments d'informations et la mémoire associative a pour rôle de les corriger. Diverses alternatives de correction sont alors possibles mais le choix est fait de manière arbitraire vis-à-vis des réseaux faisant la requête si il n'y a pas retour d'information. Un mécanisme supplémentaire est donc nécessaire.

En d'autres termes, l'approche usuelle pour réaliser une mémoire associative avec un réseau de neurones est incomplète : il lui manque la description de la mise en œuvre d'une requête ainsi que des mécanismes permettant à la mémoire de choisir une des alternatives cohérentes en fonction des souhaits du requêteur. Nous arguons que la définition que nous avons donnée plus haut de la réalisation d'une spécification par un système permet de répondre correctement à ces questions. Le principe est de ne pas chercher à coder la requête mais d'identifier celle-ci à un système ou, plus rigoureusement, à l'assemblage du système réalisant la mémoire asssociative à un autre système restreignant l'espace de recherche. L'assemblage dont il est question ici est un système réalisant la conjonction des spécifications des systèmes constituant.

Ainsi, un sous-système dictionnaire mémorise une liste de mots et, utilisé seul, un tel système fournit un exemple de mots pris dans cette liste. Assemblé avec un autre système, imposant par exemple une lettre particulière, le système dictionnaire complète cette information en donnant un mot tenant compte de la lettre imposée (s'il en existe un). On a donc un mécanisme d'adressage par le contenu. Assemblé avec un système de lecture d'une image en une suite de lettres, ce module dictionnaire cherche un mot compatible avec l'image. On a donc un mécanisme de contrôle de cohérence de l'information. D'une manière plus générale, un système réalisant une spécification S peut être vu comme une mémoire associative dont la requête est précisée par assemblage avec d'autres systèmes. Isolée, cette mémoire répond à la requête très générale : « Tout mot convient » et fait son choix en ne tenant compte que de ses propres contraintes. Assemblée par su-

[6]On peut imaginer le processus de synchronisation suivant supposant que les neurones du réseau associatifé sont aussi des neurones du réseau faisant la requête : calcul du réseau faisant la requête, puis une fois son état d'équilibre atteint et *détecté*, calcul du réseau associatif.

perposition conjonctive avec un autre système réalisant une spécification R, cette mémoire répond à la requête : « Le mot cherché est une solution à R ».

Conclusion

Nous venons de présenter un cadre de construction de réseaux de neurones. Celui-ci permet de réaliser toute spécification sur un ensemble fini E de la forme « chercher x de E vérifiant la propriété $\mathcal{P}(x)$ ». Les réseaux obtenus peuvent être assemblés pour construire des réseaux réalisant des combinaisons de spécifications. La correction des réponses données par ces réseaux est garantie par construction.

Un point gênant est que pour l'un de ces assemblages il n'y a pas induction de la convergence du système obtenu à partir de l'hypothèse de convergence vers une solution des réseaux assemblés. Cet assemblage est la superposition conjonctive dont le but est d'obtenir un système dont les réponses conviennent simultanément aux réseaux assemblés. Pour obtenir une preuve de terminaison par construction, il semble qu'il faille imposer des contraintes trop fortes aux réseaux,c'est-à-dire pénalisant trop l'efficacité du calcul. On peut, par exemple, imposer que toute solution soit atteignable de tout point non-terminal. A quelques détails près cette propriété (qui garantit la terminaison de tout calcul) est conservée par assemblage. De plus, toute spécification admet au moins une spécification de cette forme. Mais un tel système est totalement inefficace. Le problème de la terminaison est-il un problème qu'il faut traiter indépendamment de la preuve de correction d'un système? Il en est déjà ainsi pour les preuves de programmes à la HOARE.

Enfin il se pose la question suivante : les assemblages proposés et les décompositions de spécifications qu'ils induisent sont-il suffisamment puissants pour que l'on puisse décomposer puis réaliser toutes les spécifications que l'on souhaite faire? On garantit que toute spécification admet au moins une réalisation, mais peut-on l'obtenir par décomposition puis assemblage? Par exemple, dans le cadre qui vient d'être défini, on voit mal comment construire un système réalisant la fermeture transitive d'une relation, à partir d'un réseau réalisant cette relation.

Conclusion

Le principe des réseaux de neurones est simple : on conçoit un système comme l'agencement de systèmes élémentaires, les neurones. Ces derniers ont un état résumé par une quantité scalaire et qui peut être modulée par des signaux activateurs et inhibiteurs. Ces neurones sont alors connectés en un réseau de sorte que chacun puisse moduler l'activité des autres. Cette modulation est contrôlée par l'état du neurone et pondérée par des coefficients caractérisant le réseau. En termes plus précis, cette interaction entre neurones est décrite par l'ensemble des états d'équilibre : un réseau de neurones est défini par une équation implicite entre les états d'équilibre et les coefficients caractérisant le réseau.

Cette définition doit être complétée par un mécanisme permettant d'atteindre un point d'équilibre. On peut alors utiliser le système obtenu pour résoudre divers problèmes : on se fixe une méthode nous permettant de choisir un état initial du réseau en fonction d'une instance du problème. Le point d'équilibre éventuellement atteint par la trajectoire ainsi choisie est considéré comme une réponse au calcul. Il faut donc aussi se fixer un code permettant d'interpréter dans l'espace des solutions les points d'équilibre. Le problème est alors de construire un réseau pour une application particulière et de s'assurer de l'adéquation entre le réseau obtenu et le comportement souhaité.

La première difficulté est de choisir un mécanisme permettant effectivement d'obtenir une réponse à partir de tout point initial. A ce niveau on retiendra les dynamiques asynchrones pour lesquelles on dispose d'un critère efficace pour garantir la convergence de tout calcul : il suffit d'exhiber, pour tout état initial, un chemin aboutissant à un point d'équilibre.

A ce niveau, la réponse classique au problème de l'obtention d'un réseau ayant un comportement particulier s'appuie sur le fait qu'un même réseau peut avoir une gamme de comportements très riche suivant les valeurs choisies pour les divers coefficients le caractérisant. On prend donc un réseau dont la gamme de comportements semble suffisamment vaste pour contenir le comportement souhaité. Les coefficients de contrôle de ce réseau sont alors jugés par comparaison,

entre le comportement qu'ils induisent et celui souhaité, puis modifiés par retouches successives : on parle d'algorithmes d'apprentissage.

Ces algorithmes utilisent le lien implicite entre les états d'équilibre d'un réseau et les coefficients décrivant ce réseau. Le comportement souhaité est décrit à l'aide d'une experience sur le réseau, dont le résultat est jugé par une fonction d'évaluation associant à toute réponse une qualité plus ou moins bonne. Il apparaît alors que l'on ne peut pas ainsi obtenir tout comportement choisi *a priori*: il est nécessaire que ce comportement ne répose que sur le lieu des points d'équilibre et nullement sur les conditions pour obtenir un point d'équilibre particulier.

Le cadre de construction de réseaux de neurones que nous avons présenté tient compte de ces remarques. On retient une dynamique asynchrone et on ne fait reposer l'interprétation d'un réseau que sur le lieu des points d'équilibre. On obtient un cadre permettant de réaliser toute spécification sur un ensemble fini E de la forme « chercher x de E vérifiant la propriété $\mathcal{P}(x)$ ». Les réseaux obtenus peuvent être assemblés pour construire des réseaux réalisant des combinaisons de spécifications. La correction des réponses données par ces réseaux est garantie par construction. Par contre les propriétés de convergence ne sont pas obtenues par construction.

ANNEXE A

Implémentation sur machine parallèle

Nous nous proposons de simuler un réseau de neurones sur une machine parallèle à mémoire distribuée. Une telle machine est constituée d'un ensemble d'unités de calcul disposant chacune d'un espace de travail auquel elles ont un accès exclusif. Afin de pouvoir coordonner les calculs effectués sur les différents sites, ceux-ci sont liés les uns aux autres par un réseau d'interconnexions. La programmation d'une telle machine recouvre donc trois aspects : il faut distribuer les informations (attribuer un processeur à chaque information initiale, finale ou intermédiaire) , contrôler et coordonner l'ensemble des calculs et enfin il faut organiser les échanges d'information. Ces trois questions inter-agissent : se simplifier la tâche pour l'un de ces points entraîne des complications ou une perte d'efficacité pour les autres.

Pour programmer sur une telle machine un algorithme donné, on peut procéder comme suit. On choisit *a priori* une distribution des informations : telle donnée du problème sera initialement connue par tel processeur, telle opération sera effectuée sur tel autre processeur et tel résultat sera finalement obtenu en tel lieu. De ce placement, on va donc déduire un schéma de communications : telle information devra être acheminée de tel processeur à tel autre. Toute l'efficacité du programme obtenu va donc essentiellement reposer sur le choix du placement (il faut choisir un placement réduisant au maximum le volume d'échanges d'informations et permettant un maximum de calculs simultanés) et la programmation des schémas de communications (il faut coordonner au mieux les différents échanges d'informations pouvant avoir lieu simultanément).

L'organisation adoptée pour cette annexe est la suivante : nous allons dans un premier temps rechercher dans les algorithmes de réseaux de neurones la source essentielle de difficultés vis-à-vis de leur programmation sur une machine parallèle distribuée. On se ramènera ainsi à un objectif plus simple : la programmation du produit itéré d'une matrice par un vecteur. Pour ce calcul nous proposerons un placement des informations qui, comparé aux solutions classiques, réduit de beaucoup le volume d'échanges d'informations. Ce travail est présenté dans le rapport [9] (avec D. DELESSALLE et D. TRYSTRAM). Citons aussi la

thèse d'H. FRYDLENDER [13] consacrée à l'implémentation de réseaux de neurones sur machines parallèles à mémoire distribuée. Partant d'un travail commun, H. FRYDLENDER en donne une présentation détaillée et le généralise à diverses structures de la matrice. Nous proposerons enfin une réalisation des schémas de communications ainsi induits en nous plaçant dans le cadre de réseaux de communications décrits par un graphe de CAYLEY. Le cas particulier de l'hypercube est présenté dans l'article [10] (avec D. DELESSALLE et D. TRYSTRAM).

1. Communications induites par les algorithmes de réseaux de neurones

Il y a essentiellement deux types d'algorithmes à écrire pour simuler un réseau de neurones : un algorithme d'utilisation permettant de rechercher une solution dans une situation donnée et un algorithme d'apprentissage permettant de trouver les coefficients définissant un réseau le plus approprié possible à l'application visée. Comme algorithme d'apprentissage, nous retiendrons celui dit de la rétropropagation du gradient sous sa forme générale due à F. PINEDA [40, 41] et L. ALMEIDA [3, 4]. Outre le fait que cet algorithme peut *a priori* être utilisé avec un réseau de neurones quelconque (non nécessairement organisé en couches), son intérêt du point de vue de la programmation sur une machine parallèle est qu'il recouvre tout les types d'échange d'information que l'on peut rencontrer avec les réseaux de neurones (en particulier, on y retrouve l'algorithme permettant d'utiliser le réseau). Rappelons la forme de cet algorithme pour un réseau défini sur un ensemble de neurones $\{1, \ldots, n\}$ par la donnée de coefficients ω_{ij} et ω_i pour tout neurone i et j :

i) recherche, suivant une méthode itérative propre au réseau, d'un vecteur réponse x vérifiant :

$$x_i = f(u_i) \quad \text{avec} \quad u_i = \sum_j \omega_{ij} x_j + \omega_i \quad .$$

ii) recherche, suivant la même méthode itérative, d'un vecteur y vérifiant :

$$y_i = f'(u_i) \,.(c_i + \sum_j y_j \omega_{ji})$$

avec

$$c_i = \begin{cases} x_i - d_i & \text{si la réponse désirée pour le neurone } i \text{ est } d_i \\ 0 & \text{si la réponse du neurone } i \text{ ne nous importe pas} \end{cases} .$$

iii) pour toute connexion d'une cellule j vers un neurone i, augmenter le coefficient synaptique ω_{ij} de la quantité :

$$-\lambda x_j y_i \quad .$$

Quelles sont les communications induites par cet algorithme si l'on veut l'implanter sur une machine parallèle à mémoire distribuée? Plus précisément quelles sont les informations qui doivent se rencontrer et comment donc distribuer ces informations aux différents processeurs pour minimiser le volume d'échanges entre processeurs? Les différentes rencontres d'informations sont résumées par des opérations ayant la forme suivante (Les indices en t dénotent les diverses occurrences d'une variable lors d'un calcul itératif dont la solution est annotée d'un indice ∞.):

$$u_i^t \leftarrow \sum_j \omega_{ij} x_j^t + e_i$$

$$(*) \qquad x_i^{t+1} \leftarrow F(x_i^t, u_i^t)$$

$$v_i^t \leftarrow \sum_j \omega_{ji} y_j^t$$

$$(*) \qquad y_i^{t+1} \leftarrow G(x_i^\infty, u_i^\infty, v_i^t)$$

$$\omega_{ij} \leftarrow \omega_{ij} - \lambda y_i^\infty x_j^\infty$$

Étant donné un neurone i il est donc naturel de confier à un même processeur les informations relatives aux diverses occurrences de x_i, u_i, y_i et v_i de façon à ce que les calculs marqués d'une astérisque $*$ puissent être fait sans aucun échange d'information. Notons $P(i)$ ce processeur. De même notons $P(i,j)$ le processeur mémorisant le coefficient ω_{ij}. Il ne reste alors plus que trois formes d'échanges d'informations à organiser:

i) (figure A.16) La quantité x_j^t doit passer du processeur $P(j)$ au processeur $P(i)$ pour le calcul de u_i^t. Sur le chemin x_j^t doit être multiplié par ω_{ij} et le message doit donc passer par le processeur $P(i,j)$ détenant cette information. Ce mouvement d'information doit avoir lieu pour tout couple (i,j) de neurones. Le volume de messages véhiculés peut être réduit en remarquant qu'un message x_j issu d'un processeur $P(j)$ peut être dupliqué au cours de sa diffusion à tous les processeurs de la forme $P(k,j)$. De même tous les messages de la forme $\omega_{ik}x_k^t$, émis à destination d'un même processeur $P(i)$ par les processeurs $P(i,k)$, peuvent être réduits par addition avant de rejoindre le processeur $P(i)$. Le volume de calcul est essentiellement dû aux multiplications $\omega_{ij}x_j^t$ et est donc proportionnel au nombre de coefficients ω_{ij} contenu dans un processeur.

ii) (figure A.17) De même, lors du calcul des quantités v_j, pour tout couple (i,j) de neurones, un message x_i doit être acheminé d'abord jusqu'au processeur $P(i,j)$ pour y être multiplié par le coefficient ω_{ij} puis jusqu'au processeur $P(j)$ chargé de l'information v_j. Ce schéma de communication peut être déduit du précédent en remarquant que les messages circulent en sens inverse et doivent être dupliqués là où ils étaient réduits par addition (réciproquement réduits par ad-

dition là où ils étaient dupliqués). De même, le temps de calcul est proportionnel au nombre de coefficients ω_{ij} contenus dans un processeur.

iii) (figure A.18) Enfin, pour la mise à jour des coefficients ω_{ij}, pour tout couple (i,j) de neurones, les messages y_i et x_j respectivement issus des processeurs $P(i)$ et $P(j)$ doivent se rejoindre dans le processeur $P(i,j)$ pour y être combinés. Là aussi on remarque que ce schéma de communication est similaire au premier :

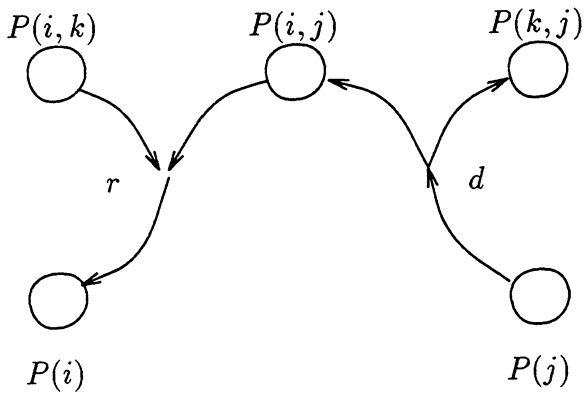

FIG. A.16 - . Mouvements d'informations lors de la recherche d'une réponse du réseau. Le symbole d indique qu'un message peut être dupliqué. Le fait que deux messages destinés à un même processeur peuvent être réduits en un seul par addition est signalé par le symbole r.

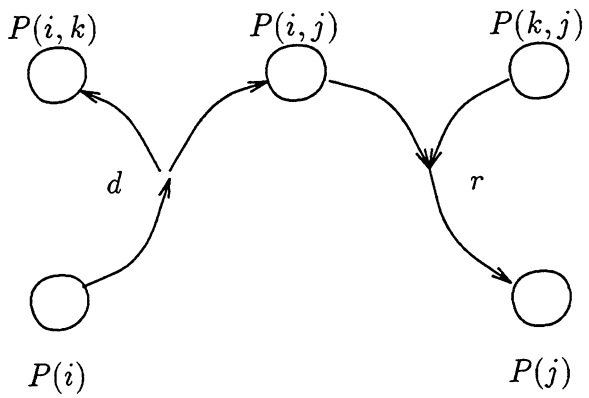

FIG. A.17 - . Mouvements d'informations lors du calcul du vecteur intermédiaire.

il suffit d'inverser le sens de parcours des messages entre un processeur $P(i)$ et un processeur $P(i,k)$ et de remplacer les accumulations par des duplications. Là encore le temps de calcul est proportionnel au nombre de coefficients ω_{ij} contenu dans un processeur.

Les trois schémas de communication se déduisant les uns des autres, il suffit de trouver une distribution des informations (les coefficients ω_{ij} ainsi que les diverses quantités x_i, u_i, y_i et v_i) optimisant le premier schéma pour optimiser simultanément les trois étapes de l'algorithme. En conclusion, pour simuler un réseau de neurones sur une machine à mémoire distribuée, il faut essentiellement s'attacher à l'implémentation du calcul itéré d'un vecteur par une matrice :

$$x_i^{t+1} \leftarrow \sum_j \omega_{ij} x_j^t \quad .$$

2. Le produit itéré d'un vecteur par une matrice

Nous nous proposons de programmer sur une machine parallèle distribuée le calcul d'un produit itéré d'un vecteur par une matrice :

$$x_i^{t+1} \leftarrow \sum_j \omega_{ij} x_j^t \quad .$$

Nous insistons sur le qualificatif d'*itéré*, car on souhaite que ce soit le même processeur qui soit responsable des différentes occurrences d'une même coordonnée du vecteur x de façon à pouvoir utiliser le même programme de multiplication, d'itération en itération. En d'autres termes, sans cette contrainte de pouvoir itérer

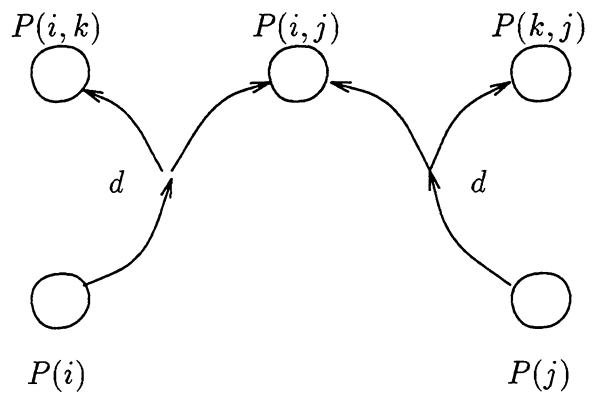

FIG. A.18 - . Mouvements d'informations lors de la mise à jour des coefficients de contrôle ω.

le produit, on pourrait envisager une programmation telle que la $i^{\text{ème}}$ coordonnée du résultat ne soit pas obtenue dans le processeur possédant la $i^{\text{ème}}$ coordonnée du vecteur initial.

Pour écrire notre programme il nous faut donc désigner pour chaque coordonnée du vecteur un processeur responsable. Ce processeur détiendra la valeur initiale de cette coordonnée et devra finalement recevoir la même coordonnée du produit. Notons $P(i)$ le processeur ainsi responsable de la $i^{\text{ème}}$ coordonnée. De même notons $P(i,j)$ le processeur mémorisant le coefficient ω_{ij}. C'est aussi en ce processeur qu'aura lieu le produit $\omega_{ij}x_j$. Le programme obtenu a alors la forme suivante :

Pour toute coordonnée j
    envoi de x_j du processeur $P(j)$ à tous les processeurs $P(i,j)$.
Pour tout couple de coordonnées (i,j)
    calcul dans le processeur $P(i,j)$ du produit $\omega_{ij}x_j$
Pour toute coordonnée i
    réception par le processeur $P(i)$ de la somme des produits
        $\omega_{ij}x_j$ émis par les processeurs $P(i,j)$.

Il n'y a que deux types de contraintes de précédence : un processeur $P(i,j)$ ne peut effectuer le produit dont il a la charge qu'une fois qu'il a reçu la coordonnée détenue par le processeur $P(j)$. Une fois ce calcul fait, le processeur $P(i,j)$ peut alors émettre son résultat au processeur $P(i)$. On peut donc *a priori* commencer la phase de calcul des produits avant que la première phase de communication soit achevée et l'on peut même commencer la seconde phase de communication dès l'obtention des premiers résultats partiaux. Néanmoins nous ne tiendrons pas compte de ces deux remarques en les considérant comme des améliorations à apporter au dernier moment. Notre programme comporte alors trois phases distinctes : la première consistant à diffuser partiellement les coordonnées du vecteur initial de sorte qu'un processeur $P = P(i,j_1) = \cdots = P(i,j_k)$ reçoive toutes les coordonnées $j_1, \ldots, j_k$. Au cours de la seconde phase, sont calculés des produits scalaires partiels (le processeur $P = P(i,j_1) = \cdots = P(i,j_k)$ calcule le produit de $(\omega_{i,j_1}, \ldots, \omega_{i,j_k})$ par $(x_{j_1}, \ldots, x_{j_k})$). Enfin les dernières sommations sont effectuées au cours de la troisième et dernière phase.

Comment choisir maintenant les deux placements $P(i)$ et $P(i,j)$ pour obtenir un programme efficace?

2.1. Les solutions classiques

La solution la plus classique consiste à stocker entièrement la $i^{\text{ème}}$ ligne de la matrice dans le processeur devant obtenir la $i^{\text{ème}}$ coordonnée du résultat. On

parle de placement en lignes de la matrice et cela revient à prendre :

$$P(i,j) = P(i) \quad .$$

Avec ce placement, le programme se simplifie en une phase de diffusion suivie uniquement du calcul des produits scalaires. La phase de diffusion est alors totale : chaque processeur doit recevoir la totalité du vecteur initial. On parle d'*échange total* : chaque processeur détient initialement une information qui lui est propre (ici, un ensemble de coordonnées) et chaque processeur doit recevoir la totalité des informations (ici, le vecteur).

Une autre solution classique est le placement en colonnes de la matrice :

$$P(i,j) = P(j) \quad .$$

Ici, c'est la phase de diffusion qui disparaît et il suffit de regrouper dans chaque processeur la somme de tous les produits partiaux calculés au préalable. On qualifiera ce schéma de communication d'*échange total, personnalisé et avec accumulation* : chaque processeur détient initialement un terme pour chaque processeur de la machine et chaque processeur doit recevoir la somme des termes qui lui sont destinés. Ce schéma de communication est en fait le dual du précèdent : l'échange total (Il suffit de faire circuler les messages en sens inverse et de sommer les messages qui se rencontrent là où ils étaient précédemment dupliqués).

Il nous reste donc à programmer en fonction du réseau liant les processeurs un échange total ou son dual l'échange total, personnalisé et avec accumulation. Le volume de messages à échanger au cours de ces deux schémas de communication est important : chaque processeur a un message à recevoir de chaque autre (respectivement à émettre). Notons p le nombre de processeurs et τ le nombre maximum de messages qu'un processeur peut émettre ou recevoir par unité de temps. En gérant au mieux les transits des différents messages dans le réseau, on peut donc espérer au mieux que ces deux schémas prennent un temps d'exécution de l'ordre de $\tau(p-1)$ secondes. Nous verrons que l'on arrive effectivement à atteindre cette borne pour certains réseaux de communications.

Les deux placements de la matrice, en lignes et en colonnes, qui tous deux suppriment une phase de communication du programme de référence, induisent un volume important de communications. On peut alors se demander si il existe un placement de la matrice permettant de réduire ce volume de communication quitte à le faire en deux phases : recueil de coordonnées disponibles dans un voisinage restreint du processeur puis, de même, calcul et diffusion des termes partiels relatifs à des coordonnés se trouvant dans des processeurs proches. La section suivante propose un tel placement [9] sur une idée développée en commun avec H. FRYDLENDER [13].

2.2. Une autre solution plus efficace

Supposons arbitrairement que l'ensemble des processeurs a une structure d'ensemble produit $L \times C$. Cette structure va nous permettre d'organiser facilement le produit en deux phases de communications : diffusion puis accumulation. L'idée est de choisir un placement du vecteur et de la matrice de telle sorte qu'au cours de la première phase de diffusion, un processeur (l, c) ne communique qu'avec des processeurs se trouvant sur la même ligne que lui (*i.e.* de la forme $(l, *)$). De même, au cours de la phase d'accumulation des résultats partiaux, on souhaite qu'un processeur (l, c) ne communique qu'avec des processeurs se trouvant sur la même colonne que lui (*i.e.* de la forme $(*, c)$). Donnons d'abord le programme que l'on souhaite utiliser. Nous chercherons seulement ultérieurement le placement des données adéquat : permettant d'obtenir ainsi effectivement le produit d'un vecteur par une matrice.

En parallèle sur toutes les lignes de processeurs
  échange total entre tous les processeurs d'une même ligne
  des coordonnées connues par chacun.
En parallèle sur tous les processeurs
  calcul de produits scalaires partiaux
En parallèle sur toutes les colonnes de processeurs
  accumulation par tous les processeurs d'une même colonne
  des termes associés calculés par les processeurs de la colonne.

On souhaite donc d'abord effectuer un échange total entre les processeurs d'une même ligne, puis un échange total personnalisé et avec accumulation entre tous les processeurs d'une même colonne. On a alors le résultat suivant :

PROPRIÉTÉ. *Etant donné un placement $P(i) = (l_i, c_i)$ des coordonnées x_i d'un vecteur x sur un ensemble de processeurs $L \times C$, alors le programme ci-dessus correspond au calcul du produit du vecteur x par la matrice (ω_{ij}) si le placement $P(i, j)$ des éléments ω_{ij} vérifie :*

$$P(i, j) = (l_j, c_i) \quad .$$

En d'autres termes le coefficient ω_{ij} doit se trouver sur la même ligne de processeurs que la coordonnée x_j et la même colonne de processeurs que la coordonnée x_i.

Il faut vérifier que le calcul $\omega_{ij}x_j$ puisse effectivement être calculé puis acheminé au processeur $P(i)$. L'information x_j, issue du processeur $P(j) = (l_j, c_j)$, étant diffusée uniquement sur la ligne $(l_j, *)$ de processeurs, seuls ces derniers peuvent effectuer le calcul $\omega_{ij}x_j$. Le processeur $P(i, j)$ se trouve donc sur la ligne de processeurs $(l_j, *)$. De même le processeur $P(i) = (l_i, c_i)$ ne peut accumuler que

des termes qui ont été calculés sur la colonne de processeurs $(*, c_i)$. Le processeur $P(i,j)$ ce trouve donc sur cette colonne $(*, c_i)$. En conclusion, si $P(i) = (l_i, c_i)$ désigne le processeur responsable de la $i^{\text{ème}}$ coordonnée d'un vecteur, il faut placer le coefficient ω_{ij} de la matrice dans le processeur :

$$P(i,j) = (l_j, c_i)$$

se trouvant sur la même ligne que le processeur $P(j)$ et la même colonne que le processeur $P(i)$. □

Si l'on veut maintenant équilibrer la charge de calcul des différents processeurs, ce qui revient ici à équilibrer entre les divers processeurs le nombre de coefficients de la matrice stockés par chacun, il suffit de choisir un stockage équilibré du vecteur. On obtient alors un parallélisme de calcul optimal : le temps de calcul (ici proportionnel au nombre de coefficients de la matrice) est effectivement divisé par le nombre de processeurs. A-t-on amélioré le temps de communication, par rapport aux solutions classiques ?

Pour répondre à cette question il est nécessaire de prendre en compte un point jusqu'ici ecarté : le graphe de connexion des processeurs. Si l'on veut effectuer simultanément les différents échanges totaux relatifs à chaque ligne de processeurs, il est nécessaire que soient connexes les sous-graphes de connexions relatifs à chaque ligne de processeurs. Le plus simple est de prendre un réseau de processeurs décrit par un produit de deux graphes (le premier décrivant les interconnexions entre les processeurs d'une même ligne et le second décrivant les interconnexions entre les processeurs d'une même colonne). Dans le cas d'un tel graphe de connexions produits de deux graphes, notre solution, comparée aux solutions classiques, permet donc de passer d'un échange total sur le graphe produit à deux échanges totaux sur les graphes facteurs de ce produit. On diminue ainsi sensiblement le volume de communications.

En effet, notons $|L|$ et $|C|$ le nombre de processeurs que l'on trouve respectivement sur une ligne et une colonne de processeurs. Notons comme précédemment τ le nombre maximum de messages qu'un processeur peut émettre ou recevoir en une une unité de temps et en prenant comme unité de message le nombre de coordonnées stockées par un processeur. Chaque processeur doit d'abord recevoir $(|L| - 1)$ messages puis en émettre $(|C| - 1)$. On ne peut donc espérer un temps de communication inférieur à $\tau.(L + C - 2)$ (Nous donnerons ultérieurement une réalisation de ce schéma de communications permettant d'atteindre effectivement cette borne sur des graphes particuliers.). Obtenant précédemment un temps de $\tau.(|L|.|C| - 1)$, notre placement induit donc bien un programme plus rapide (d'autant plus que $|L|$ et $|C|$ sont voisins). On peut espérer ainsi un gain de l'ordre de 5 pour une machine de 144 processeurs. Pour une étude des gains effectifs sur un réseau de TRANSPUTER, on se reportera à la thèse d'H. FRYDLENDER [13]. On pourra aussi se reporter à l'article [9] relatif à la CONNECTION MACHINE.

L'objection la plus importante que l'on peut faire à ce placement est qu'il est spécifiquement adapté au produit itéré d'un vecteur par une matrice : inséré dans un programme plus vaste, ce calcul d'un produit risque d'induire d'importants mouvements d'information pour obtenir à la bonne place les coefficients de la matrice. Ce phénomène n'a pas lieu dans notre cas : celui des réseaux de neurones.

En effet, il nous reste essentiellement deux point à programmer, le calcul d'un vecteur intermédiaire et la mise à jour des coefficients de la matrice de connexions, et nous avons vu que ces deux calculs entraînent les mêmes mouvements d'information que le calcul du produit d'un vecteur par une matrice. Dans le premier cas, celui du calcul d'un vecteur intermédiaire y, nous devons essentiellement effectuer un calcul de la forme :

$$v_i \leftarrow \sum_j \omega_{ji} y_j \quad ,$$

qui invoque la transposée de la matrice de connexions, et que l'on réalise avec le même placement que précédemment comme suit :

En parallèle sur toutes les colonnes de processeurs
  échange total entre tous les processeurs d'une même colonne
  des coordonnées connues par chacun.
En parallèle sur tous les processeurs
  calcul de produits scalaires partiaux
En parallèle sur toutes les lignes de processeurs
  accumulation par tous les processeurs d'une même ligne
  des termes associés calculés par les processeurs de la ligne.

On a vu en effet qu'il suffisait d'inverser le sens de parcours des messages et de substituer l'une à l'autre les opérations de diffusion et d'accumulation (Ce qui revient ici, à inverser le rôle des lignes et des colonnes de processeurs.). Ainsi, la coordonnée y_j est diffusée du processeur $P(j) = (l_j, c_j)$ à tous ceux de la colonne $(*, c_j)$. Le processeur $P(j,i) = (l_i, c_j)$ peut alors effectuer le produit $\omega_{ji} y_j$. Ce terme sera alors accumulé dans le processeur $P(i) = (l_i, c_i)$ lors de l'échange total avec accumulation suivant les lignes.

Finalement, pour acheminer les informations nécessaires à la mise à jour des coefficients de la matrice :

$$\omega_{ij} \leftarrow \omega_{ij} - \lambda y_i x_j$$

il suffit d'effectuer en parallèle un échange total des coordonnées de y sur chaque colonne de processeurs et un échange total des coordonnées de x sur chaque ligne de processeurs. Ainsi, la coordonnée y_i est diffusée du processeur $P(i) = (l_i, c_i)$ à tous ceux de la colonne $(*, c_i)$. Parallèlement, la coordonnée x_j est diffusée du processeur $P(j) = (l_j, c_j)$ à tous ceux de la ligne $(l_j, *)$. Le processeur responsable

du coefficient ω_{ij}, à savoir le processeur $P(i,j) = (l_j, c_i)$, dispose alors de toutes les informations nécessaires à son calcul de mise à jour.

En conclusion, nous venons de présenter un placement sur un réseau de processeurs des diverses informations relatives à un réseau de neurones, permettant d'obtenir un gain de rapidité pour les différentes phases de calcul invoquées par le réseau, lorsque le réseau d'interconnexion des processeurs est décrit par un graphe produit. Il reste essentiellement à écrire un programme assurant un échange total d'information sur un réseau de processeurs ainsi que le schéma de communication dual : l'échange total personnalisé avec accumulation. Le réseau de processeurs que l'on utilisera *in fine* pour les calculs sur le réseau de neurones sera construit suivant le produit de deux graphes et les communications invoquées seront uniquement des échanges globaux, avec ou sans accumulation, suivant l'une ou l'autre direction du réseau produit. Le réseau le plus simple ayant une telle structure est le tore, produit de deux anneaux. Outre un gain en temps, le placement proposé apporte donc un gain du point de vue de la facilité de la programmation : comme nous allons le voir il est beaucoup plus facile de décrire un échange total sur un anneau de processeurs que sur un tore.

3. Réalisations des communications

Étant donné un ensemble P de processeurs, on décrit un réseau de communications à l'aide d'un graphe donnant l'ensemble des canaux permettant d'acheminer des informations d'un processeur sur un autre. Nous supposons que ces canaux sont bidirectionnels : si un processeur peut émettre des messages vers un autre processeur, alors il peut aussi en recevoir de ce même processeur. Nous supposons aussi qu'il existe un mécanisme permettant à tout processeur d'émettre simultanément un message distinct sur chacun des liens dont il dispose et de se mettre en position de réception d'un message sur chacun de ces liens. De sorte que si tous les processeurs exécutent simultanément cette opération, chacun d'eux échange une information avec tous ses voisins dans le réseau de communications. Plus formellement, préalablement à cette opération un message $m_{(p,q)}$ est choisi par le processeur source p de chaque lien de communications (p,q) liant deux processeurs p et q. A la fin de cette opération (après émission puis réception de tous les messages) chaque processeur p dispose des messages $m_{(q,p)}$ émis par les processeurs q auxquels il est lié. Notons un point important : on suppose que cette instruction ne prend fin que lorsque un message a été reçu sur *chaque* lien. Cette instruction que nous nommerons *échange local* constituera pour nous la brique de

base pour l'élaboration d'algorithmes de communication[1].

Le schéma de communication que nous nous proposons de réaliser est l'échange total : initialement chaque processeur possède un élément d'information et l'on souhaite que chaque processeur récupère finalement l'ensemble des informations disponibles sur le réseau (en connaissant l'origine de chaque élément d'information). On appellera message du processeur p l'élément d'information détenu initialement par le processeur p.

L'approche classique à ce problème consiste à décrire pour chaque processeur l'arbre de diffusion que va devoir suivre son message : cet arbre décrit les liens entre processeurs que le message va devoir franchir pour inonder le réseau. Ces arbres doivent donc recouvrir le réseau : on garantit ainsi que le message est reçu par tous les processeurs et ce une seule fois. Les différents arbres relatifs aux différents messages doivent être compatibles : un seul message peut être émis à un instant donné sur un lien. Il faut alors, en pratique, retarder l'émission de certains messages. On se retrouve donc face au problème dit de la recherche d'une famille d'arbres recouvrants et disjoints dans le temps. La difficulté de cette approche est qu'il faut finalement déduire un programme en chaque processeur du réseau à partir du chemin suivi par chaque message. En d'autres termes, il faut passer d'une description relative aux messages et globale au réseau à une description locale aux processeurs et ordonnançant les ordres d'arrivée et d'émission des messages. Nous allons donc nous écarter de cette approche classique et aborder directement la description d'un échange total localement à chaque processeur.

Pour cela nous allons adopter deux principes : premièrement un style de programmation dit SPMD[2] (un même programme est utilisé par tous les processeurs) et deuxièmement une désignation des messages *relative* à chaque processeur (Un processeur p ne désigne plus un message par son processeur q d'origine mais par la situation relative de q par rapport à p.). Utilisés conjointement, ces principes trouvent toute leur force lorsqu'on les applique à un réseau de communications décrit par un graphe régulier (informellement, la description d'un tel réseau à partir d'un processeur est uniforme sur l'ensemble des processeurs). En effet, connaissant les messages émis par un processeur a un instant donné et sachant

[1]Cette instruction d'échange local est par exemple disponible sur la CONNECTION MACHINE 2. On peut aussi l'écrire à la OCCAM comme suit :

PAR pour tout couple (p, q) de processeurs liés
  lien(p, q) ! $m_(p, q)$
  lien(q, p) ! $m_(q, p)$
  lien(p, q) ? $m_(q, p)$
  lien(q, p) ? $m_(p, q)$

[2]Single Program Multiple Data

donc que ses voisins vont émettre des messages en position relative identique, on va pouvoir déduire l'origine des messages reçus par un processeur. Si l'on note T cette opération qui va nous permettre de déduire l'origine relative $T(r)$ des messages reçus sur un lien par un processeur en échange de l'émission de messages d'origine relative r, un échange total va prendre la forme d'une succession d'échanges locaux :

Pour tout processeur p
pour i variant de 0 jusqu'à $i_{\max}$
émission à chaque processeur voisin
d'un message d'origine relative r_i
réception en échange de chaque processeur voisin
d'un message d'origine relative $T(r_i)$

Pour que ce programme soit correct il faudra s'assurer que les messages émis ont bien été reçus au préalable. Il faudra de même s'assurer que tous les messages sont bien reçus lors de la terminaison de l'algorithme. Pour des raisons d'efficacité, on pourra aussi vérifier qu'un message n'est pas reçu plus d'une fois par un même processeur. A partir d'un choix des origines relatives des messages émis à chaque instant, ces vérifications se font en utilisant l'opérateur T permettant de déduire ce qui est reçu en fonction de ce qui est émis : on construit ainsi de manière itérative l'ensemble des origines relatives des messages connus par un processeur au cours du temps.

Une fois établie une relation entre l'origine des messages émis et celle des messages reçus en échange, la conception d'un programme d'échange total repose donc essentiellement sur l'ordonnancement des messages à émettre par un processeur (ou d'une manière duale : à recevoir). Avant de donner une classe de graphes (les graphes de CAYLEY) où cette relation entre réception et émission puisse être décrite, présentons au préalable un modèle permettant de mesurer le coût en temps d'une communication et de cerner un critère d'optimalité pour un algorithme d'échange total. Nous nous placerons ensuite dans le cas simple d'un anneau de processeurs. Nous conclurons par un ordonnancement des émissions dans les cas particuliers du tore et de l'hypercube.

3.1. Coût des communications

Le modèle couramment admis pour mesurer le temps d'une communication entre deux processeurs tient compte d'un temps d'initialisation, au cours duquel la communication est établie entre les deux processeurs et qui ne dépend donc que des processeurs choisis, et d'une durée de communication proprement dite, qui est proportionnelle au volume d'informations échangées. Avec des unités de temps et de volume d'information arbitraire, la durée $t(L)$ de l'émission d'un

processeur à un autre d'une quantité L d'information est de la forme :

$$t(L) = \beta + \tau \,.\, L \quad .$$

La quantité $\frac{1}{\tau}$ représente donc la bande passante d'un lien de communication : le volume d'information qu'il est possible de recevoir en une unité de temps.

Pour un échange total, nous allons utiliser parallèlement tous les liens disponibles pour un processeur. Le temps minimal pour recevoir un volume L d'information en un échange prend toujours la même forme :

$$t(L) = \beta + \tau \,.\, L \quad .$$

La durée β représentant cette fois-ci le temps d'initialisation simultanée de tous les canaux d'un processeurs. La quantité $\frac{1}{\tau}$ représente maintenant la bande passante d'un processeur : le volume d'information qu'il lui est possible de recevoir en une unité de temps en utilisant simultanément tous ses liens. Pour que cette durée minimale de réception d'un volume d'information L soit effectivement atteint, il sera nécessaire que la même quantité d'information soit reçue sur chacun des liens, un échange local ne prenant fin qu'avec la terminaison des toutes les communications.

Peut-on maintenant apprécier le temps que va prendre un échange total sur un réseau de processeurs ? On suppose que les messages à diffuser par les processeurs ont tous un même volume que nous prendrons donc comme unité de volume d'information. On a alors le résultat suivant :

PROPRIÉTÉ. *Le temps pris pour un échange total concu comme une succession d'échanges locaux, sur un réseau de p processeurs et dont le diamètre est d, est au moins égal à :*

$$d \,.\, \beta + (p-1) \,.\, \tau \quad ,$$

où $\frac{1}{\tau}$ est la bande passante d'un processeur.

En effet, au moins d échanges de messages doivent avoir lieu pour que le message le plus éloigné puisse parvenir jusqu'à un processeur. De plus, chaque processeur doit recevoir $p-1$ messages. En supposant que ceux-ci ont pu être regroupés au mieux de façon à pouvoir être émis au cours des d d'échanges déjà requis, on obtient donc la borne inférieure citée plus haut. □

Pour pouvoir affirmer l'optimalité d'un échange total *parmi ceux construits à partir d'échanges locaux*, sur un réseau de p processeurs et dont le diamètre est d, il suffit donc de s'assurer qu'il comporte uniquement d échanges chacun d'eux invoquant des volumes d'informations identiques sur chacun des liens. L'hypothèse de construction de l'échange total à l'aide d'une succession d'échanges locaux est

essentielle : sans cette contrainte il n'est pas impossible de masquer le temps d'initialisation sur un lien (il suffit de commencer la réception d'un message sur ce lien avant de finir les réceptions sur les autres liens). Dans le cas général donner une borne inférieure du temps pris par un échange total est un problème actuellement sans réponse. Ainsi l'on ne connait pas d'exemple d'algorithme d'échange total prenant un temps inférieur à celui cité plus haut. Se restreindre à des échanges totaux construits à partir d'échanges locaux est donc une restriction acceptable.

3.2. Le cas de l'anneau

Considérons un anneau de $n = 2d + 1$ processeurs (n est supposé impair pour des raisons de simplicité lors de la description de l'algorithme) : les processeurs sont étiquetés de 0 à $n-1$ et le processeur p est relié aux deux processeurs $(p-1)$ *modulo* n et $(p+1)$ *modulo* n. La position relative d'un processeur q par rapport à un processeur p est $(q-p)$ *modulo* n.

Comment déduire l'origine relative des messages reçus au cours d'un échange par un processeur, connaissant l'origine relative des messages émis ? Supposons qu'au cours d'un échange le processeur p envoie à son voisin supérieur $(p+1)$ un message d'origine relative r (issu donc de $p+r$). Au cours de ce même échange le processeur $p-1$ envoie lui aussi à son voisin supérieur (à savoir le processeur $(p-1)+1 = p$ que nous avons pris comme référence) le message d'origine relative r (issu donc du processeur $((p-1)+r)$). En échange du message d'origine relative r émis à son voisin supérieur $p+1$, le processeur p reçoit de son voisin inférieur $p-1$ le message d'origine relative $r-1$. Réciproquement, en échange d'un message d'origine relative r émis à son voisin inférieur $p-1$, le processeur p reçoit de son voisin supérieur $p+1$ le message d'origine relative $r+1$. L'échange total prend donc la forme suivante :

Pour tout processeur p
pour i variant de 0 jusqu'à $i_{\max}$
émission d'un message $(p + r_i^-)$ au processeur voisin $(p-1)$
émission d'un message $(p + r_i^+)$ au processeur voisin $(p+1)$
réception en échange du message $(p + r_i^+ - 1)$ du processeur voisin $(p-1)$
réception en échange du message $(p + r_i^- + 1)$ du processeur voisin $(p+1)$

Comment choisir les messages à émettre ? Le plus simple est de procéder de manière itérative : ayant décrit les i premières étapes et connaissant donc l'ensemble des messages reçus par un processeur au cours de ces étapes, on peut regarder les messages connus par un processeur et non par son voisin et qu'il est donc possible de lui transmettre. Initialement, chaque processeur ne connaît qu'un seul message et celui-ci est inconnu des deux voisins. Après i échanges, chaque processeur a reçu tous les messages se trouvant à une distance inférieure ou égale à i. Les

messages à échanger se trouvent alors à l'extrémité de ce segment et le nombre d'échanges à effectuer est égal au diamètre d de l'anneau. Un échange total sur un anneau de processeurs peut donc être décrit par l'algorithme :

Pour tout processeur p
pour i variant de 0 jusqu'à $d-1$
émission du message $(p+i)$ au processeur voisin $(p-1)$
émission du message $(p-i)$ au processeur voisin $(p+1)$
réception du message $(p-i-1)$ du processeur voisin $(p-1)$
réception du message $(p+i+1)$ du processeur voisin $(p+1)$

Le point essentiel permettant la vérification de ce programme est que, connaissant l'ensemble $P(p)$ des processeurs dont le processeur p a reçu les messages à un instant t, on peut déduire par *translation*, et suivant l'hypothèse de comportement similaire des autres processeurs, l'ensemble des messages connus par ces derniers :

$$P(p+k) = P(p) + k = \{q+k \quad \text{où } q \in P(p)\} \quad .$$

On peut aussi vérifier que cet échange total est optimal : il comporte un nombre d'échanges égal au diamètre de l'anneau et tous ces échanges sont équilibrés : le même nombre de messages est reçu sur chacun des liens (la bande passante des processeurs est utilisée au mieux). Si le nombre de processeurs de l'anneau est pair, alors à la dernière étape de l'algorithme il ne reste plus qu'un seul message à recevoir. Pour obtenir un échange total optimal, on peut remarquer que les deux voisins du processeur de référence détiennent ce dernier message : il suffit donc de le partager en deux parties égales, on peut ainsi toujours utiliser au mieux la bande passante des processeurs.

3.3. Les graphes de Cayley

L'algorithme d'échange total présenté ci-dessus repose essentiellement sur une opération de translation dans le réseau de processeurs. Cette opération permet de décrire le réseau relativement à chaque processeur et d'obtenir ainsi une description SPMD de l'échange total. Les graphes où cette opération de translation peut être généralisée sont les graphes de CAYLEY :

DÉFINITION. *Soit $(G,.)$ un groupe dont l'élément neutre est noté 1. Soit $S \subset G$ une famille génératrice de ce groupe :*

$$\forall g \in G \quad \exists s_1, \ldots, s_n \in S \quad tq \quad g = \prod_{i=1}^{n} s_i \quad .$$

Le graphe de CAYLEY *sur G et généré par S est le graphe (G, U) où $U \subset G \times G$ est défini par :*

$$U = \{(g, s.g) \quad g \in G, \quad s \in S\} \quad .$$

On appelle position relative d'un élément g de G par rapport à une origine o de G l'unique élément r de G tel que :

$$g = r \,.\, o \quad .$$

Un sommet a donc autant de voisins qu'il y a de générateurs dans S. En pratique on choisit l'ensemble des générateurs S de telle sorte que le graphe soit non dirigé (on exprime ainsi que l'on utilise des canaux de communication bi-directionnels) et sans boucle (il n'y a pas lieu de relier un processeur à lui même) :

$$1 \notin S \quad \text{et} \quad \forall s \in S \quad s^{-1} \in S \quad .$$

Comme exemples de tel graphe, on peut citer le tore et l'hypercube :

Un tore de dimension 2, de l lignes et c colonnes est le graphe de CAYLEY sur le groupe $G = (\frac{\mathbb{Z}}{l\mathbb{Z}} \times \frac{\mathbb{Z}}{c\mathbb{Z}}, +)$ généré par $S = \{(1,0), (-1,0), (0,1), (0,-1)\}$. Un hypercube de dimension n est le graphe de CAYLEY sur le groupe $G = (\frac{\mathbb{Z}}{2\mathbb{Z}}^n, +)$ généré par $S = \{(x_1, \ldots, x_n) \in G \quad \text{tq} \quad \exists! i \quad x_i = 1\}$.

Abordons maintenant le problème de l'écriture d'un échange total sur un graphe de CAYLEY sur un ensemble de processeurs G et généré par un ensemble de directions S. L'intérêt des graphes de CAYLEY combinés avec un style de programmation SPMD est résumé par le résultat suivant :

PROPRIÉTÉ. *Pour qu'au cours d'un échange tous les processeurs recoivent sur le lien s un message d'origine relative r il est suffissant que tout processeur ait déjà reçu le message d'origine relative $r.s^{-1}$ et l'émette dans la direction opposée s^{-1}. L'échange prend alors la forme suivante :*

Pour tout processeur p

émission du message $r.s^{-1}.p$ au processeur voisin $s^{-1}.p$

réception en échange du message $r.p$ du processeur voisin $s.p$

En effet, au cours de l'échange le processeur p de G envoie à son voisin $s^{-1}.p$ un message d'origine relative $e = r.s^{-1}$ (issu donc de $e.p$). Par hypothèse de comportement SPMD au cours de ce même échange le processeur $s.p$ envoie lui aussi à son voisin se trouvant dans la direction s^{-1} (à savoir le processeur $s^{-1}.(s.p) = p$ que nous avons pris comme référence) le message d'origine relative e (issu donc du processeur $e.(s.p) = (e.s).p = r.p$). En échange du message d'origine relative

$r.s^{-1}$ émis à son voisin $s^{-1}.p$, le processeur p reçoit donc de son voisin $s.p$ le message d'origine relative r. □

Ce résultat va nous permettre d'écrire un échange total comme une succession d'échanges locaux et en ordonnançant pour un seul processeur (le processeur unité) la reception des messages. Se donner un tel ordonnancement des réceptions, revient à fixer pour tout processeur $p \in G$ un instant de réception par le processeur de référence du message issu de p. On notera $T(p)$ cet instant. Il faut aussi fixer une direction de réception $R(p) \in S$, direction suivant laquelle le message issu de p sera reçu par le processeur unité. Si on considère que plusieurs messages peuvent être concaténés en un seul de façon à pouvoir être émis au cours d'un même échange et sur un même lien, on a alors le résultat suivant :

PROPRIÉTÉ. *Soit un ordonnancement $(T(p), R(p))$ qui, à tout processeur $p \in G$, associe un instant $T(p) \in \mathbb{N}$ et une direction $R(p) \in S$ de réception par le processeur unité. On note R_i^s l'ensemble des processeurs dont l'instant et la direction de réception par le processeur unité est donné par le couple $(i, s) \in \mathbb{N} \times S$. Si cet ordonnancement vérifie :*

$$T(1) = 0$$
$$\forall p \in G \qquad T(p.s^{-1}) < T(p) \quad \text{où } s = R(p)$$

alors, le programme suivant correspond à un échange total :

Pour tout processeur p
pour i variant de 1 jusqu'à $max_{p \in G}(T(p))$
pour tout générateur s
émission des messages de $R_i^s.s^{-1}.p$ au processeur voisin $s^{-1}.p$
réception en échange des messages de $R_i^s.p$ sur le lien s

On remarquera tout d'abord que l'hypothèse sur l'ordonnancement $(T(p), R(p))$, implique que tous les éléments de R_i^s sont au moins à une distance i du processeur unité. En particulier on a $R_0^s = \{1\}$. Soit $P_i(p)$ l'ensemble des processeurs dont les messages sont parvenus au processeur p après i échanges locaux. Nous allons montrer par récurrence sur le nombre d'itérations que :

$$P_i(p) = P_i(1).p$$
$$P_i(1) = \bigcup_{s \in S, j \leq i} R_j^s$$

On a initialement :

$$P_0(p) = \{p\} = P_0(1).p \quad \text{et} \quad P_0(1) = \bigcup_{s \in S} R_0^s$$

Si après i échanges on a l'hypothèse de récurrence, alors tout processeur p a reçu les messages issus de $R^s_{i+1}.s^{-1}.p$ pour toute direction $s \in S$. Ce résultat est d'abord établi pour le processeur unité en utilisant l'hypothèse sur l'ordonnancement :

$$R^s_{i+1}.s^{-1} \subset \bigcup_{d \in S, j \leq i} R^d_j = P_i(1) \quad ,$$

puis pour tout processeur p par hypothèse de récurrence :

$$R^s_{i+1}.s^{-1}.p \subset P_i(1).p = P_i(p)$$

Les messages de $R^s_{i+1}.s^{-1}.p$ peuvent être émis par le processeur p suivant la direction s^{-1} pour recevoir en échange les messages de R^s_{i+1} sur le lien s. On a alors :

$$\begin{aligned} P_{i+1}(p) &= P_i(p) \cup \bigcup_{s \in S} R^s_{i+1}.p \\ &= \bigcup_{s \in S, j \leq i+1} R^s_j.p \end{aligned}$$

L'ensemble des messages reçus par un processeur se déduit donc bien par translation de l'ensemble des messages reçus par le processeur unité. En particulier, tous les processeurs finissent par recevoir tous les messages. □

Il nous reste maintenant à exhiber un ordonnancement de la réception des messages par le processeur unité. On peut ainsi utiliser le programme suivant qui utilise une variable R pour noter les processeurs déjà pris en compte :

$T(1) = 0$
$R \leftarrow \{1\}$
pour $i = 1$ jusqu' au diamètre du graphe
    $V \leftarrow$ { processeurs à une distance i de celui de reférence }
    $U \leftarrow V$
    tant que $V \neq \emptyset$
        pour $s = s_1$ jusqu'à s_n
            si $\exists p$ tq $p.s^{-1} \in R$
                $T(p) \leftarrow i$
                $R(p) \leftarrow s$
                enlever p à V
    ajouter U à R

Avec cet ordonnancement tous les messages distants de i seront reçu à la $i^{\text{ème}}$ étape. Le nombre d'étapes est ainsi minimal mais la quantité d'informations reçues suivant les différentes directions n'est pas nécessairement équilibrée. Équilibrer la quantité d'informations reçues sur chacun des liens et à chaque étape est

un problème difficile en général car il est nécessaire de prendre alors en compte les symétries du réseau. Nous aborderons donc ce problème uniquement dans le cas particulier du tore puis celui de l'hypercube.

3.4. Le cas du tore

Notre but n'est pas de donner un algorithme d'échange total pour un tore quelconque (nous verrons en effet que de nombreuses discussions ont lieu suivant les dimensions du tore) mais de montrer comment aboutir à un algorithme adapté à un tore donné. Nous ne regarderons donc que quelque cas simples et illustratifs.

Commençons par donner une description d'un tore de l lignes et c colonnes, à partir de l'ensemble de générateurs $S = \{n, e, w, s\}$. Ce tore est le graphe de CAYLEY construit sur le plus grand groupe commutatif G généré par S et vérifiant les identités suivantes :

$$n^l = e^c = ns = ew = 1 \quad .$$

Considérons le cas d'un tore carré, en supposant de plus que le nombre de lignes est impair : $(l = c = 2k + 1)$. On peut alors partitionner l'ensemble $G - \{1\}$, des processeurs autre que celui pris comme référence, en quatres parties disjointes :

$$\begin{aligned}
RN &= \{n^i e^j \quad \text{avec} \quad 1 \leq i \leq k \quad \text{et} \quad 0 \leq j \leq k\} \\
RE &= \{e^i s^j \quad \text{avec} \quad 1 \leq i \leq k \quad \text{et} \quad 0 \leq j \leq k\} \\
RS &= \{s^i w^j \quad \text{avec} \quad 1 \leq i \leq k \quad \text{et} \quad 0 \leq j \leq k\} \\
RW &= \{w^i n^j \quad \text{avec} \quad 1 \leq i \leq k \quad \text{et} \quad 0 \leq j \leq k\}
\end{aligned}$$

Ces quatres parties RN, RE, RS et RW ont le même nombre d'éléments et comportent de plus le même nombre d'éléments se trouvant à une distance donnée de l'origine 1. On peut donc les utiliser pour ordonnancer les réceptions du processeur origine : elles vont respectivement correspondre à l'ensemble des processeurs dont le message va parvenir par les liens n, e, s et w. On obtient donc l'échange total suivant pour un tore carré de coté impair $(l = c = 2k + 1)$ (les opérations d'addition sont à comprendre *modulo* le nombre de lignes ou de

colonnes et les processeurs sont référencés à l'aide d'un couple $(ligne, colonne)$) :

Pour tout processeur p
pour δ variant de 1 jusqu'à $2k$

$$(min, max) = \begin{cases} (1, \delta) & \text{si } \delta \leq k \\ (\delta - 1, k) & \text{sinon} \end{cases}$$

pour tout i compris entre min et max
émission des messages
$p - (i - 1, \delta - i)$ au processeur voisin $+(1, 0)$
$p - (\delta - i, i - 1)$ au processeur voisin $+(0, 1)$
$p + (i - 1, \delta - i)$ au processeur voisin $-(1, 0)$
$p + (\delta - i, i - 1)$ au processeur voisin $-(0, 1)$
réception en échange des messages
$p + (i, \delta - i)$ du processeur voisin $+(1, 0)$
$p + (\delta - i, i)$ du processeur voisin $+(0, 1)$
$p - (i, \delta - i)$ du processeur voisin $-(1, 0)$
$p - (\delta - i, i)$ du processeur voisin $-(0, 1)$

Cet échange total est optimal car il est constitué d'un nombre d'échanges égal au diamètre du tore et qu'au cours de chacun de ces échanges le même nombre de messages est reçu depuis chaque lien. Regardons maintenant le cas d'un tore comportant plus de colonnes que de lignes et en supposant toujours impairs les nombres de lignes et de colonnes ($l = 2k + 1$ et $c = 2b + 1$ avec $b > k$). Les k premières étapes de l'algorithme précédent conviennent toujours, mais au delà les voisins e et w auront plus d'informations à fournir à chaque échange que les voisins s et n : il faut donc rétablir l'équilibre pour obtenir un échange total optimal. Pour cela nous allons partager l'émission de certains messages. Plus précisément, le tore est maintenant divisé en six parties disjointes :

$$\begin{aligned} RN &= \{n^i e^j \quad \text{avec} \quad 1 \leq i \leq k \quad \text{et} \quad 0 \leq j \leq b\} \\ RS &= \{s^i w^j \quad \text{avec} \quad 1 \leq i \leq k \quad \text{et} \quad 0 \leq j \leq b\} \\ RSE &= \{s^k e^i \quad \text{avec} \quad 1 \leq i \leq b - k\} \\ RNW &= \{n^k w^i \quad \text{avec} \quad 1 \leq i \leq b - k\} \\ RE &= \{e^i s^j \quad \text{avec} \quad 1 \leq i \leq b \quad \text{et} \quad 0 \leq j \leq k\} - RSE \\ RW &= \{w^i n^j \quad \text{avec} \quad 1 \leq i \leq b \quad \text{et} \quad 0 \leq j \leq k\} - RNW \end{aligned}$$

Les quatres parties RN, RE, RS et RW représentent respectivement les messages que l'on va recevoir suivant les directions n, e, s et w. La partie RSE (respectivement RNW) contient les messages que l'on va recevoir simultanément des directions s et e (respectivement n et w). La réception des messages se fait

dès que possible et les messages à recevoir suivant deux directions sont découpés de sorte que les premières moitiés soient échangées suivant l'axe $n - s$ (les secondes suivant $e - w$). On obtient alors un échange total optimal prenant la forme suivante :

Pour tout processeur p
pour i variant de 1 jusqu'à k
    réception des messages à une distance i
        de RN, RE, RS et RW
pour i variant de $k+1$ jusqu'à b
    réception des messages à une distance i
        de RN, RE, RS, RW, RSE et RNW
pour i variant de $b+1$ jusqu'à $k+b$
    réception des messages à une distance i
        de RN, RE, RS et RW

3.5. Le cas de l'hypercube

Commençons par donner une description d'un hypercube de dimensions n, à partir de l'ensemble de n générateurs $S = \{d_1, \ldots, d_n\}$. Un hypercube de dimension n est le graphe de CAYLEY construit sur le plus grand groupe commutatif G généré par S et vérifiant les identités suivantes :

$$d_i^2 = 1 \qquad \forall i \in \{1, \ldots, n\} \quad .$$

Il y a alors bijection entre G et les parties de S : à $\{s_1, \ldots, s_k\}$ de $\mathcal{P}(S)$ est associé $\prod_{i=1}^{k} s_i$ de G. Ainsi,

si à p de $\mathcal{P}(S)$ est associé g de G
alors à $p\Delta\{s\}$ est associé $g.s = g.s^{-1}$

le symbole Δ désignant la différence symétrique de deux ensembles.

Nous identifierons G à $\mathcal{P}(S)$. Pour obtenir un échange total sur un hypercube, il nous faut donc exhiber un ordonnancement $(T(p), R(p)) \in \mathbb{N} \times S$ pour tout p de $\mathcal{P}(S)$ vérifiant :

$$\forall p \subset S \qquad T(p\Delta R(p)) < T(p) \quad .$$

Pour obtenir un tel partitionnement, considérons la permutation circulaire $\sigma = (d_1 \to d_2 \to \cdots \to d_n \to d_1)$ de S. On appelle alors orbite d'un élément p de $\mathcal{P}(S)$, l'ensemble des éléments de $\mathcal{P}(S)$ obtenu par application successive de σ à p. On définit alors la période d'un élément p de $\mathcal{P}(S)$, comme le plus petit entier

non nul i tel que $\sigma^i(p)$ soit égal à p. Cette période est alors au moins égale à n : $\sigma^n(p) = p$. On a de plus les deux propriété suivantes :

PROPRIÉTÉ. *Si deux parties de S ne diffèrent que d'un élément alors, alors au moins une des deux a pour période le cardinal n de S.*

Soit p une partie de S et s n'appartenant pas à p. On suppose que p et $p \cup \{s\}$ on pour période respectives k et l, toutes deux stritement inférieures àn. On a alors $\sigma^k(s) \neq s$ et $\sigma^l(s) \neq s$. Par hypothèse de périodicité de p et de $p \cup \{s\}$, on a alors:

$$s \notin p \Rightarrow \sigma^k(s) \notin p \cup \{s\} \Rightarrow \sigma^{k+l}(s) \notin p \cup \{s\}$$

or on a aussi :

$$s \in p \cup \{s\} \Rightarrow \sigma^l(s) \in p \Rightarrow \sigma^{k+l}(s) \in p$$

En conclusion au moins une des deux périodes est égal à n. □

PROPRIÉTÉ. *Si une partie p de S a au moins deux éléments alors au plus un élément s de p peut être tel que l'ensemble $p - \{s\}$ n'ait pas pour période le cardinal n de S.*

Soit p une partie de S tel $\{s,t\} \subset p$. On suppose $s \neq t$ et que $p - \{s\}$ et $p - \{t\}$ ont pour période respectives k et l, toutes deux stritement inférieures àn. Soient maintenant ik de $k\mathbb{Z} - n\mathbb{Z}$ qui est non vide car $k < l$ (respectivement $jl \in l\mathbb{Z} - n\mathbb{Z} \neq \emptyset$). On a alors $\sigma^{ik}(s) \neq s$ et $\sigma^{jl}(s) \neq s$. Par hypothèse de périodicité de $p - \{s\}$ et $p - \{t\}$, on a alors:

$$s \in p - \{t\} \Rightarrow \sigma^{ik}(s) \in p - \{s,t\} \Rightarrow \sigma^{ik+jl}(s) \in p - \{s\}$$

or on a aussi :

$$s \notin p - \{s\} \Rightarrow \sigma^{jl}(s) \notin p \Rightarrow \sigma^{ik+jl}(s) \notin p - \{t\}$$

On a alors $\sigma^{ik+jl}(s) = t$. Or ceci doit être vrai pour tout ik de $k\mathbb{Z} - n\mathbb{Z}$ et jl de $l\mathbb{Z} - n\mathbb{Z}$. D'où $k = l$ et $n = 2k$. D'où l'on obtient la contradiction $s = t$.

Le programme suivant permet alors d'obtenir un ordonnancement correct en équilibrant de plus le nombre de messages qui vont être reçus à une étape sur les

différents liens :

$T(\emptyset) = 0$
pour tout $p \neq \emptyset \in \mathcal{P}(S)$

$$T(p) = \begin{cases} |p| & \text{si la période de } p \text{ est } n \\ n & \text{sinon} \end{cases}$$

$R \leftarrow \emptyset$
pour i variant de 1 jusqu'à n
  $V \leftarrow \{p \in \mathcal{P}(S) \quad \text{tq} \quad |p| = i\}$
  tant que $V \neq \emptyset$
    choisir $p \in V$
    si la période de p est n alors
      choisir s de S dans p telle que
        la période de $T(p - \{s\}) = T(p) - 1$
      pour j variant de 0 jusqu'à $n - 1$
        $R(\sigma^j(p)) = \sigma^j(s)$
    sinon
      ajouter l'orbite de p à R
    enlever l'orbite de p à V
tant que $R \neq \emptyset$
  pour s variant de s_1 jusqu'à s_n
    si $R \neq \emptyset$, choisir $p \in R$
    $R(p) = s$
    enlever p à R

La correction de ce programme s'appuie sur plusieurs points :

1) Pour tout p de période n, on peut effectivement choisir un élément s de p tel que $T(p - \{s\}) = T(p) - 1$. Si p ne comporte qu'un seul élément s alors $T(p) = 1 > T(\emptyset)$. Si p comporte plus d'un élément il faut choisir s tel que $p - \{s\}$ soit de période n : comme nous l'avons vu, au plus un choix ne convient pas.

2) Si s vérifie la propriété précédente vis-à-vis d'un élément p dont la période est n, alors il en est de même pour $\sigma^i(s)$ vis-à-vis de $\sigma^i(p)$. Tous les messages issus d'un processeur dont la période est n, peuvent donc être reçus au plus tôt.

3) Tous les messages issus d'un processeur dont la période est différente de n, peuvent être reçus depuis tout lien, pour peu que cette réception se fasse au dernier moment. En effet, tous les voisins d'un nœud de période différente de n

sont de période n et les messages associés ont donc été reçus.

Conclusion

Les deux points essentiels de cette annexe sont, premièrement, la présentation d'un placement original des éléments de la matrice intervenant dans le produit itéré d'un vecteur par une matrice, et deuxièmement, la présentation d'une nouvelle approche pour la réalisation d'un échange total sur une machine à mémoire distribuée.

Le placement proposé pour les coefficients d'une matrice intervenant dans un produit itéré permet de réduire sensiblement le volume des échanges d'informations et par là d'augmenter l'efficacité de l'implémentation obtenue sur une machine à mémoire distribuée. Ce placement a été obtenu en étudiant dans un premier temps les échanges d'informations imposés par un placement quelconque. En s'affranchissant ainsi de contraintes faites *a priori* (comme considérer comme un tout insécable les lignes ou les colonnes de la matrice), nous avons exhibé un placement adapté à la structure du réseau de processeurs et non à la structure de la matrice. On obtient ainsi un gain conséquent par rapport aux placements classiques en ligne et en colonne.

L'intérêt de l'approche proposée pour la réalisation d'un échange total, est surtout de faciliter la conception d'un tel algorithme : les contraintes de précedence à respecter sont plus faciles à vérifier que celles imposées par la recherche d'arbres de recouvrement disjoints dans le temps.

Bibliographie

[1] Emile H.L. Aarts et Jan H.M. Korst, *Boltzmann machines for travelling salesman problems*, European journal of Operational research (1989).

[2] ———, *Computations in massively parallel networks based on the boltzmann machine : a review*, Parallel Computing vol. 9 (1989), p. 129.

[3] Luis B. Almeida, *Backpropagation in perceptrons with feedback*, Neural Computation (1987), p. 199.

[4] ———, *A learning rule for asynchronous perceptrons with feedback in a combinatorial environment*, IEEE international conference on neural networks (San Diego), vol. 2, 1987, p. 609.

[5] CESTA, *Cognitiva 85: A la frontière de l'intelligence artificielle, des sciences de la connaissances et des neurosciences*, France, 1985.

[6] Y. Le Cun, *Une procédure d'apprentissage pour réseau à seuil assymétrique*, [5], 1985, p. 599.

[7] ———, *Generalization and networks design strategies*, Connectionism in perspective (North-Holland) (R. Pfeifer, Z. Schreter, F. Fogelman-Soulie, et L. Steels, editeurs.), 1989, pp. 143–156.

[8] Y. Le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, et L. Jackel, *Back-propagation applied to handwritten zip code recognition*, Neural Computation vol. 1 (1989), p. 541.

[9] D. Delessalle, D. Trystram, et D. Wenzek, *Communication on the connection machine*, Rapport , LMC INPG, 1990.

[10] ———, *Optimal total exchange on a simd distributed memory hypercube*, Distributed memory computing conference, 1991, p. 279.

[11] Olivier François, *Ergodicité des processus neuronaux*, Comptes rendus de l'Académie des Sciences vol. 310 (1990), p. 435.

[12] P. Friedel et D. Zwierski, *Introduction to neural networks*, Rapport , LEP, 1991.

[13] Hervé Frydlender, *Implantation de réseaux de neurones artificiels sur multi-processeurs à mémoire distribuée*, Ph.D. thesis, Institut National Polytechnique de Grenoble, 1992.

[14] Catherine Fuchs et Bernard Victorri, *Modéliser la levée d'ambiguïtés à l'aide d'un réseau connexionniste*, Technique et Science informatiques vol. 11 (1992), p. 93.

[15] Kunihito Fukushima, *Neocognitron : a self organizing neural networks model for a mechanism of pattern recognition unaffected by shift in position*, Biological Cybernetics vol. 36 (1980), p. 193.

[16] ———, *A neural network for visual pattern recognition*, Computer vol. 21 (1988), p. 65.

[17] D.O. Hebb, *The organization of behavior*, J. Wiley and Sons, 1949.

[18] Benjamin J. Hellstrom et Laven N. Kanal, *Asymetric mean-field neural networks for multiprocessor scheduling*, Neural Networks vol. 5 (1992), p. 671.

[19] Laurent Hérault, *Réseaux de neurones récursifs pour l'optimisation combinatoire*, Ph.D. thesis, Institut National Polytechnique de Grenoble, 1991.

[20] Laurent Hérault et Jean Jacques Niez, *Neural networks and graph k-partionnement*, Complex Systems vol. 3 (1989), p. 531.

[21] G. Hinton et T. Sejnowski, *Optimal perceptual inference*, IEEE conference on computer vision and pattern recognition, 1983, p. 448.

[22] ———, *Learning in Boltzmann machines*, [5], 1985, p. 283.

[23] ———, *Learning and relearning in Boltzmann machines*, [46], vol. 1, 1986, p. 282.

[24] G. Hinton, T. Sejnowski, et D.Ackley, *Boltzmann machines: constraint satisfaction networks that learn*, Rapport CS-84-119, Carnegie-Mellon University, Pittsburg,USA, 1984.

[25] J. Hopfield, *Neural networks and physical systems with emergent collective computationel abilities*, Proc. Nat. Acad. Sci. USA vol. 79 (1982), p. 2554.

[26] ———, *Neurons with graded response have collective computational properties like those of two-state neurons*, Proc. Nat. Acad. Sci. USA vol. 81 (1984), p. 3088.

[27] Ken ichi Funahashi, *On the approximate realization of continuous mappings by neural networks*, Neural Networks vol. 2 (1989), p. 183.

[28] Bronislaw Jakubczyk, *Stochastic stability of nonsymmetric threshold networks*, Complex Systems vol. 3 (1989), p. 457.

[29] J.Hopfield et D.Tank, *Neural computations of decision in optimization problems like those of two-state neurons*, Biol. Cyber. vol. 52 (1985), p. 141.

[30] ———, *Computing with neural circuits: a model*, Science vol. 233 (1986), p. 625.

[31] Yves Kamp et Martin Hasler, *Réseaux de neurones récursifs pour mémoires associatives*, Presses Polytechniques et Universitaires Romandes, 1990.

[32] Y. Lecun, *Modèles connexionnistes de l'apprentissage*, Ph.D. thesis, Universiteé de Paris VI, 1987.

[33] W.A. Little, *The existence of persistent states in the brain*, Math. Biosc. vol. 19 (1974), p. 101.

[34] W.S. McCulloch et W.A. Pitts, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics vol. 5 (1943), p. 115.

[35] M. Minsky et S. Papert, *Perceptrons*, MIT Press, 1988, Première édition en 1969.

[36] D. Mitra, F. Romeo, et A. Sangiovanni-Vincentelli, *Convergence and finite-time behavior of simulated annealing*, Advances in Applied Probability vol. 18 (1986), p. 747.

[37] P. Parent et S.W. Zucker, *Trace Inference, Curvature Consistency and Curve Detection*, IEEE Transactions on Pattern Analysis and machine intelligence vol. 11 (1989), no. 8.

[38] D. Parker, *Learning logic*, Rapport TR-47, MIT, 1985.

[39] P. Peretto, *Collective properties of neural networks: a statiscal physics approach*, Biologycal Cybernetics vol. 50 (1984), p. 51.

[40] Fernando J. Pineda, *Generalization of back-propagation to recurrent and higher order neural networks*, IEEE conference on neural information processings systems, 1987, p. 602.

[41] ———, *Generalization of back-propagation to recurrent neural networks*, Physical Review Letters vol. 59 (1987), p. 2229.

[42] ———, *Dynamics and architecture for neural computation*, Journal of Complexity vol. 4 (1988), p. 216.

[43] ———, *Recurent backpropagation and the dynamical approach to adaptative neural computation*, Neural Computation vol. 1 (1989), p. 161.

[44] Jean Pierre Raysz, *Algorithmes d'apprentissage pour réseaux connexionnistes récurrents, application à la modélisation de processus d'interprétation*, Ph.D. thesis, Université de Caen, 1991.

[45] F. Rosenblatt, *Principles of neurodynamics*, Spartan Books, New York, 1962.

[46] D. Rumelhart et J. Mc Clelland (editeurs.), *Parallel Distributed Processing: explorations in the micro structure of cognition*, MIT Press, 1986.

[47] D. Rumelhart, G. Hinton, et R. Williams, *Learning internal representations by back-propagating errors*, Nature vol. 322 (1986), p. 533.

[48] ———, *Learning internal representations by error propagation*, [46], vol. 1, 1986, p. 318.

[49] T. Sejnowski et C. Rosenberg, *NETtalk: a parallel network that learns to read aloud*, Rapport 86-01, Johns Hopkins University, USA, 1986.

[50] ———, *Parallel networks that learn to prononce english text*, Complex Systems vol. 1 (1987), p. 145.

[51] B. Victorri, J.P. Raysz, et A. Konfe, *Un modèle connexionniste de la polysémie*, Actes de la conférence Neuro-Nimes, 1988.

Table des matières

« Construction de réseaux de neurones »

Résumé :

La dénomination de *réseaux de neurones* recouvre tout un ensemble de méthodes de calcul dont le point commun est de décrire le calcul d'une solution à un problème comme la recherche d'un état d'équilibre par un ensemble de cellules simples inter-agissant entre elles via un réseau de connections paramétrées. L'approche usuelle, pour obtenir un réseau de neurones ayant un comportement souhaité, consiste à tester sur des exemples un réseau choisi *a priori* et à modifier ses paramètres de contrôle jusqu'à ce que l'on obtienne un comportement satisfaisant. La difficulté de ces méthodes est que leur succès ou leur échec reposent sur le choix d'un premier réseau et que l'on ne dispose pas de règles permettant de déduire ce choix de la structure du problème.

La motivation de cette thèse a donc été de décrire des méthodes de synthèse permettant une construction modulaire de réseaux de neurones. Aussi, cette thèse propose une classe de réseaux de neurones parmi lesquels toute spécification de la forme « chercher un élément de E (fini) vérifiant la propriété P » admet au moins une réalisation. En outre, les réseaux de cette classe peuvent être combinés pour obtenir un réseau réalisant une combinaison des spécifications des réseaux combinés.

« A neural networks construction set »

Abstract :

The main point of the miscellaneous neural networks models is to describe a computation as the search of an equilibrum state by a set of simple cells which interact through a network of weighed connexions. The usual way to get a neural system which performs a desired computation is to test a first network with examples and to change the various weights in a direction which improves the results and to proceed until a correct network is reached. The main problem of this learning comes from the choice of a first network on which the success of the approach relies : how can this choice be inferred from the target computation structure?

To give an answer to this problem, the purpose of this thesis is a neural construction set which allows the construction step by step of more and more complex neural systems. So this thesis gives a subset of neural networks in which any specification like « find an element of the finite set E which verifies the property P » can be implemented. Futhermore those networks can be combined to get a system which implements a combination of the combined networks specification.