Titre: Title:	Conception d'un algorithme d'apprentissage pour un réseau de quantrons
Auteur: Author:	Jean Pepga Bissou
Date:	2007
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Pepga Bissou, J. (2007). Conception d'un algorithme d'apprentissage pour un réseau de quantrons [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8064/

UNIVERSITÉ DE MONTRÉAL

CONCEPTION D'UN ALGORITHME D'APPRENTISSAGE POUR UN RÉSEAU DE QUANTRONS

JEAN PEPGA BISSOU

DÉPARTEMENT DE MATHÉMATIQUES
ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
D'UN DIPLÔME DE PHILOSOPHIAE DOCTOR (Ph. D.)
(MATHÉMATIQUES DE L'INGÉNIEUR)
AOÛT 2007

Your file Votre référence
ISBN: 978-0-494-35518-3
Our file Notre référence
ISBN: 978-0-494-35518-3

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

CONCEPTION D'UN ALGORITHME D'APPRENTISSAGE POUR UN RÉSEAU DE QUANTRONS

présentée par : PEPGA BISSOU Jean

en vue de l'obtention d'un diplôme de : Philosophiae Doctor

a été dûment acceptée par le jury d'examen constitué de :

PIERRE Samuel, ing, Ph.D., président

LABIB Richard, Ph.D., membre et directeur de recherche

BRAULT Jean-Jules, Ph.D., membre

CADOTTE Patrick, Ph.D., examinateur externe

REMERCIEMENTS

Je souhaite tout d'abord remercier mon directeur de recherche, Dr Richard Labib, d'une part, pour son soutien financier, mais surtout, pour ses conseils toujours avisés. Nos discussions ont toujours été très enrichissantes et m'ont permis de mener ce travail à terme.

Je remercie Messieurs Jean-Jules Brault et Samuel Pierre, tous deux professeurs à l'École Polytechnique, pour la considération et l'attention accordée à mon travail.

J'aimerais également remercier l'ensemble des chercheurs qui ont contribué de manière significative à l'avancement de la conception de réseaux de neurones artificiels.

Enfin, je remercie ma famille, particulièrement mon père et ma feue mère, Bissou Bilong Jonas et Victorine Pepga, car sans eux, je ne serais jamais parvenu à cette étape.

RÉSUMÉ

Le besoin d'obtenir des modèles mathématiques de fonctionnement neuronal se rapprochant du neurone biologique donne lieu à la définition de modèle de calculs originaux. En prenant en considération les propriétés importantes du fonctionnement cérébral et en s'appuyant sur le mouvement brownien, Labib a défini un nouveau modèle mathématique de neurone artificiel connu sous le nom de quantron. Ce dernier se situe parmi les neurones artificiels de la future génération à cause de sa capacité à résoudre des problèmes non-linéaires. Cependant, il faudrait déterminer dans quelle mesure le quantron peut être utilisé dans une conception de réseau.

Partant d'une étude des principales étapes de conception d'un réseau de neurones artificiels, une analyse détaillée du quantron est effectuée. Ce nouveau neurone est comparé à certains neurones usuels. Ces étapes mènent à la conception d'un nouvel algorithme de rétropropagation de l'erreur adapté à un réseau de quantrons multicouches (MLQ). Il est implémenté et testé pour une application de reconnaissance de caractères prédéfinis.

Le besoin d'optimiser les performances des réseaux de neurones a conduit à une nouvelle forme de réseau appelée machine à état liquide (LSM). Dans ces machines, le perceptron joue le rôle de filtre de sortie. Or, l'utilisation du quantron comme nouveau filtre de sortie (de la LSM) est envisagé. Pour y arriver, une amélioration du modèle de quantron standard est réalisée. Ce modèle est utilisé dans la conception d'un nouveau réseau de quantrons parallèles possédant une seule couche. Un nouvel algorithme d'apprentissage adapté à cette nouvelle structure est développé. Il est implémenté et testé à l'aide de la même application de reconnaissance de caractères utilisée pour le MLQ. Le fonctionnement de cette structure prouve qu'elle n'est pas

seulement utilisable en filtre de sortie dans une LSM, mais peut être aussi utilisée comme un réseau de neurones.

ABSTRACT

Recent interests in developing mathematical models that mimic the way biological neurons operate led to the creation or discovery of a new type of artificial neuron called the quantron. The latter is the work of Labib who used the important properties of the brain and Brownian motion to obtain his results. The quantron can be regarded as the artificial neuron of the future because of its capacity to solve non-linear problems. However, the limit of its usefulness or application in network design should be evaluated.

Beginning with a study of the main steps in network design of artificial neurons, a detailed analysis of the quantron is performed. This current neuron is compared with classical ones. From this study, a new backpropagation algorithm that fits the Multi-Layer Quantron neural network (MLQ) is created. This algorithm is implemented and tested to recognize predefined characters.

To optimize the performance of the network, its link with a Liquid State Machine (LSM) is studied. Those machines use the perceptron for its output filter module. The use of a quantron as output filter for the LSM was considered. An improvement of the standard quantron is thus carried out. This new model is used in the design of a new parallel quantron network. A new learning algorithm that fits this new structure is developed. This algorithm was implemented and tested with the characters recognition application used for the MLQ. As a result, this new structure works independently from the liquid module of the LSM proving that it can be used as an output filter as well as an artificial neural network.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES NOTATIONS ET SYMBOLES

ASIC	: Circuit intégré spécialisé
CAM	: Mémoire adressable par contenu
FPGA	: Réseau de portes logiques programmables
LSM	: Machine à état liquide
MLP	: Perceptron multicouche
MLQ	: Quantron multicouche
MSE	: Erreur quadratique moyenne
RAM	: Mémoire vive
RBF	: Fonction à base radiale
RISC	: Processeur à simple instruction
RNA	: Réseau de Neurones Artificiels
RPROP	: Algorithme de rétropropagation 'Resilient Propagation'
SoC	: Système embarqué sur une puce
SOM	: Carte auto-organisatrice
SRAM	: Mémoire vive statique
TCAM	: CAM ternaires
XOR	: Ou-exclusif
VHDL	: Langage programmation pour la conception de circuits intégrés

CHAPITRE 1

INTRODUCTION

L'utilisation d'images numériques dans divers domaines de hautes technologies conduit à différents types d'applications. La nécessité de transmettre en temps réel des images numériques a permis de développer de nouvelles techniques de reconnaissance [14]. L'obtention d'un système adéquat pour le traitement d'images demande une considération de plusieurs caractéristiques définissant l'image. Cette prise en compte dans le traitement augmente le temps d'exécution et affecte la qualité du traitement.

Pour pallier à cette faille, plusieurs méthodes algorithmiques ont été élaborées afin de faciliter une reconstruction dynamique des caractéristiques d'une image, tout en améliorant les performances d'un tel système. Ces principales techniques sont basées sur la modélisation mathématique du cerveau humain connue sous le nom de Réseau de Neurones Artificiels (RNA) ou de machines spécialisées.

Deux principales approches peuvent être considérées :

- **La description de l'objet en fonction de sa structure.**

Elle se base sur les algorithmes de reconnaissance et les tâches d'appariement permettant la construction d'une structure d'invariants. Cette dernière est difficilement détectable dans les

images. Elle permet de décrire l'objet sous sa forme géométrique et non de l'identifier. Elle est surtout utilisée dans la reconnaissance par composantes [39].

La description des caractéristiques de l'objet.

Elle consiste à modéliser l'objet à partir de ses images réelles et non sous sa forme géométrique. Elle permet également de faire une reconnaissance d'images d'un objet versus son modèle où l'image modèle appartient à une famille définissant l'objet à identifier. La principale application de cette approche est la reconnaissance par apprentissage de réseaux de neurones [42].

L'approche de reconnaissance par apprentissage de réseaux de neurones est une méthode dynamique qui nécessite une structure d'implémentation flexible. Elle tient compte surtout de la reconnaissance des positions, de l'orientation et des échelles constituant l'image d'un objet donné. Elle procède soit par l'attribution des pixels de l'image à des classes connues à priori (on parle, dans ce cas, de classification ou de catégorisation supervisée) ; soit par leur attribution à des classes inconnues (il s'agit alors de classification non supervisée).

Le perceptron multicouche (MLP) est le réseau de neurones généralement utilisé pour les applications de classification. Diverses améliorations de ce réseau ont été proposées afin d'augmenter sa capacité sans trop alourdir sa complexité. Mais, bien que ces solutions soient performantes, elles ont tendance à s'éloigner progressivement du caractère omnipotent d'un neurone biologique [17].

Dans le cadre de cette recherche, un MLP est conçu pour une problématique de reconnaissance de caractères. Cette étude servira à comprendre le fonctionnement des algorithmes d'apprentissage associés à cette famille de réseaux. L'analyse du MLP permettra d'obtenir des outils d'évaluation d'une nouveau type de neurones artificiels connu sous le nom de quantron

[24] [23]. Ce dernier sera étudié et implémenté afin de mieux exploiter ses principales caractéristiques dans la conception d'un nouveau modèle de réseau de neurones appelé réseau de quantrons multicouches (MLQ). Il nécessitera la conception de nouveaux algorithmes d'apprentissage adaptés à la fonction discriminante du quantron.

Cependant, l'implémentation matérielle de grands réseaux de neurones multicouches implique plus de calculs, en points flottants et de la rétropropagation, lors de l'apprentissage [48]. Cette profondeur de pipeline cause de sérieux problèmes, vu la nécessité de disposer de grands circuits simples ou bidimensionnels pour les calculs de rétropropagation des fonctions de coûts dans l'algorithme [4]. L'exigence de calculer de façon précise les fonctions de dérivée nécessitant la mise en cascade de plusieurs multiplexeurs, pour la mise à jour des valeurs, peut devenir un goulot d'étranglement pour l'amélioration de la performance des calculs [2].

Cette nécessité d'augmenter les performances et de minimiser le coût global du système conduit à une proposition de plate-forme [11] [13] définie grâce à un ensemble d'outils de conception logiciels/matériels, facilitant l'ordonnancement et l'obtention d'une solution proche de l'optimum. Cependant, dans le cas du quantron, l'utilisation d'une machine à état liquide (LSM) constituée d'un réseau de quantrons parallèles à couche unique comme composante de son filtre de sortie, est une alternative permettant de contourner les difficultés d'une implémentation de MLQ. Ce nouveau réseau de quantrons parallèles nécessitera une autre innovation algorithmique adaptée à la fonctionnalité du quantron.

1.1 Problématique

Les réseaux de neurones formels sont à l'origine d'une tentative de modélisation mathématique du cerveau humain. Les premiers travaux datent de 1943 et sont l'oeuvre de Mc Culloch et Pitts [32], respectivement biologiste et logicien. Ils ont présenté un modèle assez simple pour les neurones et ont exploré ses possibilités. L'idée principale des réseaux de neurones "modernes" est la suivante : à partir d'une unité simple, le neurone est capable de réaliser quelques calculs élémentaires. Par la suite, certaines de ces unités sont reliées entre elles afin d'augmenter la puissance de calcul du réseau ainsi obtenu. Il est important de noter que ces neurones manipulent des données numériques et non symboliques.

Deux visions s'affrontent alors. D'un côté, les tenants de la modélisation biologique veulent respecter un certain nombre de contraintes liées à la nature du cerveau. De l'autre côté, les tenants de la puissance de calcul s'intéressent au modèle lui-même sans aucun lien avec la réalité biologique. Le premier modèle de réseau de neurones artificiels basé sur la modélisation biologique du cerveau humain est le perceptron multicouche. Cependant, il ne représente pas le comportement réel du cerveau et a une capacité limitée pour résoudre les problèmes complexes.

Plusieurs études ont été réalisées afin de définir de nouvelles structures mathématiques ayant un comportement similaire au cerveau humain. De ces études, découle une innovation de Labib [23] qui définit un neurone artificiel, le quantron, dont la définition mathématique présente une grande similitude avec le neurone formel.

Le quantron est un neurone artificiel qui présente l'avantage de résoudre des problèmes non linéaires à l'antipode du perceptron qui se limite à une résolution linéaire. Compte tenu de cet

avantage, peut-on envisager de concevoir des réseaux de quantrons ayant des niveaux de performance supérieurs à ceux des réseaux de perceptrons dans les domaines d'applications de classification ? Ceci se résume à savoir s'il est possible de concevoir des modèles d'algorithmes à apprentissage supervisé optimisés pour les réseaux de quantrons implémentés de façon logicielle ou matérielle avec le minimum de contraintes d'optimisation de performance.

1.2 Objectifs

Dans cette thèse, nous cherchons à concevoir un nouvel algorithme d'apprentissage adapté à une nouvelle forme de réseau de neurones artificiels : le réseau de quantrons multicouches (MLQ). Pour résoudre cette problématique, nous devons tenir compte du cadre d'utilisation de cet algorithme. C'est dans cette optique qu'un modèle d'application de classification adapté aux spécifications mathématiques du quantron doit être élaboré.

En partant de l'étude du perceptron et du fonctionnement du réseau de perceptrons multicouches (MLP), l'implémentation de ce type de réseau pour une application de reconnaissance de caractères (définie dans une matrice bidimensionnelle) permettra de mieux analyser le fonctionnement de l'algorithme d'apprentissage de rétropropagation de l'erreur par la descente du gradient. Par la suite, une analyse du fonctionnement du quantron qui est l'élément fondamental du nouveau réseau sera nécessaire. Pour y arriver, il faudra faire une étude comparative du quantron et des réseaux de neurones existants. De cette étude doit découler les avantages et les inconvénients du quantron qui peuvent faciliter l'intégration future de certaines méthodes conceptuelles associées aux familles de neurones artificiels.

L'expression de la fonction de sortie du modèle mathématique du quantron n'est pas fonction des données d'entrée mais plutôt de la valeur de son seuil de conduction préétabli. Ce manque de relation directe entre les paramètres d'entrée et la sortie du quantron conduit à la recherche d'une nouvelle définition de l'expression de sortie en fonction des données d'entrée. Cette approximation de la fonction mathématique de la résultante du quantron et une étude d'un ensemble d'algorithmes d'apprentissage usuels permettront de simplifier l'implémentation de l'algorithme de rétropropagation du MLQ. La possibilité d'amélioration de performance de cet algorithme sera analysée par l'intégration du concept du « Resilient Propagation » (RPROP) pour certains paramètres du réseau.

Une proposition d'une approche de conception optimisée pour l'implémentation matérielle mènera à étudier l'intégration du quantron dans une machine à état liquide (LSM). À cet effet, il faut concevoir un nouvel algorithme d'apprentissage inspiré du modèle p-delta retrouvé dans une LSM à base de perceptrons.

1.3 Contributions

L'innovation de cette recherche se trouve dans la conception d'un nouvel algorithme d'apprentissage par rétropropagation de l'erreur pour un réseau de quantrons multicouches. Ainsi, implémenter le quantron, nouveau modèle de neurone artificiel, sous une architecture de réseau de neurones artificiels, pour résoudre une famille d'applications de classification, a été un des principaux objectifs. Ce travail dresse au niveau analytique les caractéristiques du quantron comparativement à certains modèles de neurones existant. L'étude des réseaux usuels amène à la présentation d'une nouvelle famille d'architectures de réseaux de neurones artificiels

multicouches dont l'élément fondamental est le quantron. Cette famille de réseaux est vouée à la résolution des problèmes de classification tels que les applications de reconnaissance de caractères. Cependant, la non proportionnalité de la fonction de sortie du quantron par rapport à ses données d'entrée augmente la complexité de l'application directe d'un algorithme d'apprentissage supervisé dans un réseau de quantrons multicouches (MLQ). Par conséquent, l'expression de la fonction résultante du quantron n'est pas continuellement dérivable autour de sa valeur de sortie. Pour contourner cette contrainte, une approximation quadratique de la fonction résultante du quantron a été définie. Elle garantit que la fonction résultante soit dérivable au voisinage de sa valeur de sortie. Cet apport a pour but de faciliter l'implantation de l'apprentissage du réseau MLQ. Un modèle quadratique de la résultante du quantron est proposé.

La définition d'une nouvelle fonction de sortie du quantron a permis de construire une architecture de réseau de quantrons multicouches approximés. Celle-ci a été soumise à un apprentissage supervisé des données des échantillons d'une application de reconnaissance du caractère A définie dans un espace matriciel bidimensionnel. Les outils d'apprentissage adaptés au concept mathématique du quantron qui sont indispensables au bon fonctionnement du nouveau réseau ont été identifiés. Ils ont conduit à la construction d'un algorithme modifié de rétropropagation de l'erreur par la descente du gradient du MLQ. Cette autre innovation a été implémentée et testée grâce à un exemple d'application de reconnaissance du caractère A développée. Cette dernière est définie par une matrice 5x7 comme domaine de représentation des caractères où chaque unité de celle-ci représente un pixel. Une amélioration de cet algorithme à été faite sur la mise à jour de certains paramètres du quantron en se basant sur le modèle algorithmique du RPROP.

Le besoin d'amélioration de performance, dans le cas des grands réseaux de perceptrons multicouches, exige pour l'une des méthodes d'optimisation un partitionnement logiciel/matériel du réseau. Ce dernier est effectué dans le but d'implémenter matériellement les modules lents afin qu'ils deviennent des accélérateurs [41]. Cette latence est due aux mécanismes de l'algorithme de rétropropagation de l'erreur qui, lorsque le circuit envisagé devient de dimension considérable, augmente le temps d'exécution du processus de calcul. Pour contrer cette lacune, une nouvelle approche de conception connue sous le nom de machine à état liquide (LSM) a été développée par Maass et son équipe [28]. Elle utilise comme neurone actif, pour fin de validation de la sortie, le perceptron. L'exploration de cette méthode a permis de définir un nouvel algorithme d'apprentissage pour un réseau de quantrons parallèles à une seule couche. Elle a nécessité au préalable la définition d'une nouvelle forme de quantron simplifié. Cette dernière a pour avantage de réduire considérablement le temps de calcul de la valeur résultante de la sortie du quantron contrairement au quantron standard. Le nouvel algorithme du quantron parallèle à été implémenté et testé à l'aide du modèle d'application de reconnaissance de caractères initialement utilisé dans le cas du MLQ. Les résultats de cette implémentation ont montré qu'il est possible d'utiliser un réseau de quantrons parallèles comme filtre de sortie d'une LSM.

1.4 Organisation de la thèse

Ce document est organisé en six chapitres. La présentation des motivations de cette recherche, le cadre et la répartition de ce travail viennent d'être relatés dans ce chapitre.

Le deuxième chapitre se résume à une présentation des étapes importantes de la méthodologie de conception de réseau de neurones artificiels éventuellement applicable au quantron.

Le troisième chapitre est consacré à l'analyse du concept mathématique du quantron. Il compare la fonctionnalité du quantron à celle du perceptron et des neurones à impulsions, tout en faisant ressortir la versatilité de ce nouveau modèle.

Le quatrième chapitre est consacré à la modélisation d'un nouveau réseau de quantrons multicouches (MLQ) conçu pour une application de reconnaissance de caractères définie sous forme de matrice bidimensionnelle représentant des pixels.

Le cinquième chapitre présente les étapes de conception d'une machine à état liquide (LSM). Le filtre de sortie de la LSM est étudié dans le cas de l'utilisation du quantron comme élément fondamental. Ceci a conduit à la définition d'une nouvelle forme simplifiée de quantron. Cette dernière est utilisée dans la conception du réseau de quantrons parallèles à une seule couche. Un nouvel algorithme d'apprentissage adapté à ce réseau est développé, implémenté et testé en utilisant une application de reconnaissance de caractères définie dans le chapitre précédent.

La dernière partie du document dresse une discussion et une conclusion de l'ensemble des travaux réalisés et des perspectives futures.

CHAPITRE 2

OUTILS NÉCESSAIRES À L'ANALYSE D'UN RÉSEAU

Dans la conception des réseaux informatiques, l'essentiel du travail consiste à trouver des topologies satisfaisantes à meilleurs coûts possible et respectant les contraintes de service et d'exécution. Cette problématique est traditionnellement formulée comme un problème de programmation en nombres entiers. Elle est considérée comme une optimisation très difficile à cause du risque élevé d'explosion combinatoire.

Afin de suggérer la solution d'une topologie d'un réseau, l'évaluation des différentes alternatives possibles ainsi que celle de leur complexité doivent être exécutées. Celles-ci peuvent être réalisées au moyen de méthodes heuristiques qui essayent de réduire les choix topologiques, même si cela mène probablement à des solutions non optimales [36]. Pour cette raison, beaucoup de chercheurs ont opté pour les métaheuristiques telles que le recuit simulé, l'algorithme génétique, la logique floue et les réseaux de neurones artificiels.

Les réseaux de neurones artificiels peuvent être définis comme des modèles d'interprétation mathématique du neurone humain. En général, ces modèles décrivent un neurone par un système recevant des signaux d'entrée à l'aide de connexions multiples. Chaque connexion a un coefficient de pondération multiplicatif appelé coefficient synaptique. Le système émet un signal en sortie grâce à une fonction de transfert (d'activation) définie [17]. Le changement des conditions expérimentales ne doit pas affecter leurs mécanismes d'apprentissage. Ils ont pour particularité la capacité de traiter des problèmes complexes. Leur structure interne masque la

recherche du chemin aboutissant au résultat. Seules leurs sorties fournissent les solutions recherchées. Ce sont des machines autonomes capables de communiquer leur expertise.

2.1 Familles des réseaux de neurones artificiels.

Les réseaux se classent en deux grandes familles : les réseaux de neurones non compétitifs et les réseaux de neurones compétitifs. Chacune de ces familles peut être soumise à un apprentissage supervisé ou non supervisé. Le tableau suivant résume les types de famille de réseaux de neurones artificiels.

Tableau 2-1 : Classification des réseaux de neurones artificiels

Classe de réseau neurones artificiels	**Méthode d'apprentissage**	**Exemple de réseaux**
Réseaux de neurones non compétitifs	Non supervisé	Réseau à base radiale (RBF) [17]
	Supervisé	Perceptrons multicouches (MLP) [17]
Réseaux de neurones compétitifs	Non supervisé	Self Organisation Map (SOM) [25]
	Supervisé	Réseau de Hopfield [17][20]

Dans le cas des réseaux de neurones non compétitifs, chaque neurone porte une partie incomplète du message. La restitution de l'intégralité de l'information nécessite le regroupement

de l'ensemble des éléments contenus dans chaque neurone. Ces réseaux contiennent autant de neurones émetteurs que de neurones récepteurs [17].

Les réseaux de neurones compétitifs sont caractérisés par une structure dans laquelle une information transite d'un point à l'autre par différents chemins en même temps. Il s'agit d'une architecture de réseau en dérivation. Les neurones sont simultanément concurrents dans le transport de l'information. Si l'un d'eux disparaît, l'information est malgré tout intégralement transmise, ce qui constitue une sécurité similaire à celle de nombreux réseaux, comme par exemple la structure en anneau du réseau informatique. Ce réseau peut être supervisé ou non supervisé. Dans un réseau de neurones compétitifs supervisé, il existe des nœuds qui concentrent temporairement la totalité des messages transmis en sortie de boucle, à l'entrée du circuit ou au cours des échanges [17]. Lorsque le réseau compétitif est non supervisé, aucun neurone ne concentre l'intégralité de l'information transmise, même de façon temporaire. Cette information est plutôt diffusée sur des nœuds du système [17].

Une fois la structure du réseau définie, il faut à présent énoncer les règles d'apprentissage qui permettent d'expliquer des phénomènes donnés. L'application de reconnaissance de caractères retenue dans le cadre de cette recherche s'applique aux modèles de réseaux a apprentissage supervisé.

2.2 Conception de réseaux à apprentissage supervisé.

Les principales étapes de conception d'un réseau de neurones artificiels à apprentissage supervisé sont présentées dans la figure 2.1.

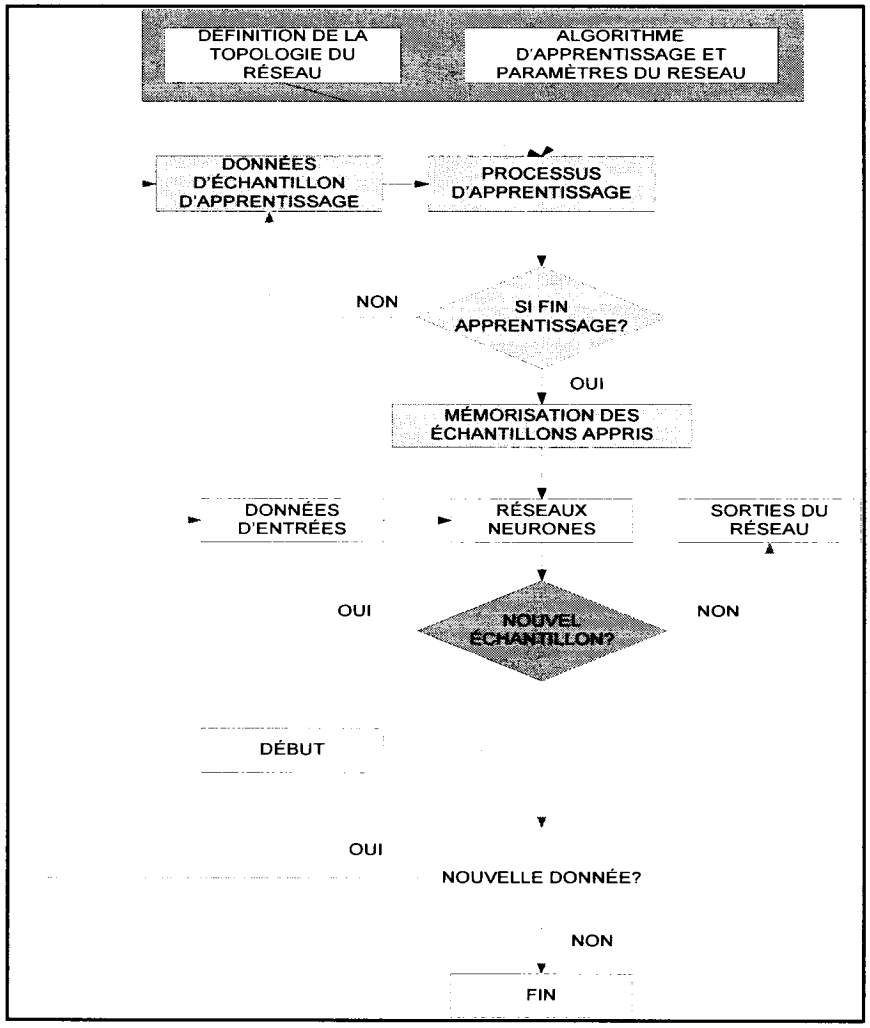


Figure 2-1 : Structure de conception de réseau de neurones artificiels

La première étape de conception est le choix de la topologie et celui du type d'apprentissage. Ils sont fonction du type de neurones et de la problématique à résoudre par le réseau. Pour le quantron, le choix topologique du réseau dépendra surtout du modèle d'application envisagée. C'est dû au fait que, comme il sera présenté dans le chapitre suivant, il est possible de l'utiliser tant comme neurone statique que dynamique à cause de sa versatilité et de la flexibilité

d'interprétation de la valeur de sa fonction discriminante. La seconde étape est la conception de l'algorithme d'apprentissage associé au modèle du réseau de neurones.

2.2.1 **Apprentissage du réseau de neurones artificiels**

L'apprentissage est la méthode qui permet de déterminer, pour une donnée d'entrée, une réponse valable en sortie du réseau. La question revient à trouver un ensemble de valeurs pour les paramètres du réseau. Ils permettent de résoudre un problème donné; il s'agit de déboucher sur un concept qui facilite la détection de bons paramètres du réseau pour résoudre le problème. La méthode de recherche des valeurs des paramètres du réseau la plus utilisée est l'algorithme de rétropropagation de l'erreur par la descente du gradient.

- **L'algorithme de rétropropagation par la descente du gradient**

Cette forme d'apprentissage est utilisée dans le cas des réseaux de neurones multicouches à apprentissage supervisé [17]. Le réseau de perceptrons multicouches qui est implémenté dans la seconde partie du chapitre emploie ce concept. Cet apprentissage se caractérise par le fait qu'il s'effectue dans la direction opposée au flux de transmission des données d'entrée dans le réseau. Son principe consiste à parvenir à ajuster les paramètres du réseau en fonction d'un seuil d'apprentissage limitant la valeur tolérée de la fonction d'erreur. Cette erreur est le résultat du calcul de comparaison d'une valeur désirée à la valeur de sortie du réseau.

L'utilisation de ces règles engendre plusieurs interprétations d'algorithmes optimisés selon les besoins des applications envisagées [22] [38]. Une méthode d'optimisation de la technique de rétropropagation est l'algorithme RPROP [21].

- **L'algorithme de 'Resilient Propagation' (RPROP)**

L'algorithme RPROP est un modèle d'apprentissage par rétropropagation du gradient de l'erreur. Il se distingue du modèle classique de rétropropagation de la descente du gradient au niveau de l'expression de la mise à jour des paramètres du réseau [21]. Sa méthode d'actualisation des paramètres tient seulement compte de l'information locale du gradient de l'erreur du paramètre considéré et non du résultat du calcul de son gradient. Il existe plusieurs dérivés de cet algorithme [43]. Il est utilisé généralement pour l'amélioration de la performance des réseaux de perceptrons multicouches [38]. Son application est envisageable dans l'amélioration des performances du futur algorithme de rétropropagation du MLQ (chapitre 4).

L'information locale considérée dans cet algorithme est la variation du signe du gradient de l'erreur entre deux itérations successives. Elle introduit donc une nouvelle forme de mise à jour $\Delta_{i,k}$ (i indique le neurone et k la couche) pour chaque paramètre du réseau comme suit :

$$\Delta_{i,k} = \begin{cases} \eta^{+}\Delta_{i,k}(compt-1), & si \quad \dfrac{\delta E(compt-1)}{\delta Parametre_{i,k}} \dfrac{\delta E(compt)}{\delta Parametre_{i,k}} > 0 \\ \eta^{-}\Delta_{i,k}(compt-1), & si \quad \dfrac{\delta E(compt-1)}{\delta Parametre_{i,k}} \dfrac{\delta E(compt)}{\delta Parametre_{i,k}} < 0 \\ \Delta_{i,k}(compt-1), & autrement \end{cases} \tag{1.1}$$

Avec : $0 < \eta^- < 1 < \eta+$. Le paramètre *compt* représente le nombre d'itérations possibles, η^- étant un facteur d'ajustement. E représente la fonction de l'erreur et $parametre_{i,k}$ le paramètre du neurone i de la couche k.

Dans le cas du MLQ, on pourra augmenter sa performance en tenant compte de cette actualisation pour le paramètre S (représentant le noyau) dans son algorithme d'apprentissage tel que présenté dans le chapitre 4.

Ainsi, chaque changement de signe du gradient de l'erreur d'un paramètre donné indique que la dernière actualisation de ce dernier était, soit trop grande ou trop petite. Quand elle est trop grande, la valeur de mise à jour est diminuée par le facteur d'ajustement η^-. Si au contraire elle est trop petite, la valeur de mise à jour est alors augmentée par le facteur d'ajustement η^+. Autrement, la valeur d'ajustement ne change pas.

L'ajustement des paramètres se fait selon les règles suivantes :

$$\Delta Parametre_{i,k}(compt) = \begin{cases} -\Delta_{i,k}(compt), & si \quad \dfrac{\delta E(compt)}{\delta Parametre_{i,k}} > 0 \\ +\Delta_{i,k}(compt), & si \quad \dfrac{\delta E(compt)}{\delta Parametre_{i,k}} < 0 \\ 0, & autrement \end{cases} \tag{1.2}$$

$$Parametre_{i,k}(compt+1) = Parametre_{i,k}(compt) + \Delta Parametre_{i,k}(compt) \tag{1.3}$$

Une exception s'impose pour cette méthode. Lorsqu'à l'étape de l'itération précédente la valeur de l'erreur est trop grande et que le minimum n'est pas atteint, ce qui s'exprime par le changement de signe du gradient du paramètre, il est ajusté de la manière suivante :

$$\Delta Parametre_{i,k}(compt-1) = -\Delta Parametre_{i,k}(compt-1), \quad si \quad \frac{\delta E(compt-1)}{\delta Parametre_{i,k}} \frac{\delta E(compt)}{\delta Parametre_{i,k}} < 0 \qquad (1.4)$$

On initialise tout d'abord les valeurs des paramètres ($\Delta_{i,k} = \Delta_0$) Cette valeur d'initialisation n'est pas critique. Cependant, il faut définir des valeurs $\Delta \max$ et $\Delta \min$. Les facteurs η^+ et η^- sont désignés de façon à minimiser le saut de la valeur minimale de l'erreur durant le processus de convergence. L'adaptation de cet algorithme pour le réseau MLQ nécessite certaines modifications. Cette question sera abordée plus en détails au chapitre 4.

Ces différents algorithmes nécessitent un processus itératif au cours duquel on améliore l'apprentissage. L'erreur commise sur la base de l'apprentissage est réduite à chaque itération. Si l'on procède, à chaque itération, à une validation croisée sur une base d'essai non utilisée par l'apprentissage, il peut arriver un moment où l'amélioration de l'erreur sur la base d'apprentissage conduit à une augmentation de l'erreur sur la base d'essai. Ce phénomène est bien connu en modélisation. Il correspond schématiquement au cas où trop de variables explicatives sont introduites dans un modèle. Il arrive alors que des explications ne puissent être données sur le comportement global du système, mais sur les aléas spécifiques aux données de l'apprentissage : les résidus sont donc modélisés. Il y a alors "sur-apprentissage" ou "apprentissage par-coeur". Le modèle perd sa capacité de généralisation [1]. Dans l'implémentation du réseau de quantrons multicouches présentée au chapitre 4, il serait question de considérer la méthode décrite précédemment.

2.3 Approche d'implémentation de réseau de neurones artificiels

Les principales approches d'implémentation sont les suivantes:

2.3.1 **Le calcul parallèle en utilisant des systèmes à multiprocesseurs**

Dans cette approche, un réseau peut être émulé par un grand nombre de processeurs. Contrairement aux états transitoires des réseaux de neurones qui sont asynchrones, le principal désavantage du traitement parallèle est le maintient d'une synchronisation des processus à l'aide de l'horloge du système [18]. Cette absence de synchronisation des réseaux de neurones résulte du codage de l'information qui est différent du codage des ordinateurs numériques.

2.3.2 **La conception logicielle/matérielle**

Il s'agit d'une approche mixte permettant au concepteur de décider de l'exécution logicielle ou matérielle [37][44] implantable sur une plate-forme embarquée ou une plate-forme sur puce (SoC) [40]. Le concept de plate-forme SoC est d'actualité. Cependant, plusieurs définitions du concept de plate-forme ont été utilisées dans le contexte de la conception des produits électroniques. Ces définitions dépendent du domaine d'application. Ainsi, il existe plusieurs classes de plate-forme: les plates-formes matérielles, les plates-formes logicielles, les plates-formes d'ordinateurs, des produits dérivés et des interfaces standards de plates-formes [41].

Dans le domaine des circuits intégrés, une plate-forme se définit comme étant un circuit flexible dédié à un ensemble d'applications prédéfinies [49]. La configuration ou la reconfiguration est réalisable par programmation et/ou modification des composantes de la puce (SoC) [11][1]. L'aspect programmation implique une modification électrique ou l'utilisation de logiciels exécutés sur un microprocesseur [44] [45]. Cette approche est moins optimisée et peut provoquer une baisse de performance ou une augmentation de la surface physique de la puce [40].

En somme, le choix entre l'exécution analogue et numérique dépend surtout du type de réseaux de neurones et de l'application à implémenter. Dans le cadre de cette recherche, la priorité est portée sur l'élaboration d'un nouvel algorithme d'apprentissage pour un réseau de quantrons. L'application de reconnaissance de caractères utilisée dans l'implémentation de l'algorithme d'apprentissage (chapitre 4), ne nécessite pas un besoin d'implémentation mixte. Dans la suite de cette recherche et dépendamment du type d'applications futures, l'implémentation d'une plate-forme mixte pourra être une alternative envisageable.

2.4 Test de performance des réseaux de neurones

Dans la problématique de classification, la mesure la plus courante de l'estimation des performances d'un réseau de neurones est le taux d'erreur défini comme suit :

$$Taux\ d'erreur = \frac{Nombre\ des\ éléments\ mal\ classés}{Nombre\ des\ éléments\ classés} = 100\% - Taux\ de\ bonne\ classification \qquad (1.5)$$

Il s'agit d'une mesure globale de la performance. Dans le cas d'une application à plusieurs classes, il faut définir une mesure d'erreur locale c'est-à-dire une mesure d'erreur pour chacune des classes.

Dans le cadre de l'implémentation du réseau de neurones, dans les prochains chapitres, l'application de reconnaissance de caractères envisagée est de petite dimension. De ce fait, le nombre d'échantillons d'exemples généré est inférieur à 100. La méthode de test utilisée est de sous-échantillonner les données en 2 classes à savoir : les données d'entrainement et les données de test. Ce qui permet d'estimer le taux d'erreur de performance du réseau. Cette méthode produit une bonne approximation de test de performance.

Cependant, la difficulté réside dans la recherche de la détermination du nombre exact d'échantillons nécessaire pour les exemples et surtout des données à tester. Pour ce faire, un cas particulier de la validation croisée connu sous le nom de « Leave-one out » [15] est considéré. Elle consiste à retirer un exemple de l'ensemble des échantillons supervisés. Ensuite, le reste des échantillons d'exemples est entraîné. Cette opération est réitérée pour chaque exemple de l'échantillon. Elle est lourde du point de vue temps machine. Dans l'implémentation du réseau de quantrons multicouches (MLQ), il s'agira de tirer au hasard un ensemble de vecteurs de test parmi l'ensemble des exemples de l'échantillon. Cette opération sera réitérée un certain nombre de fois de façon à obtenir une estimation moyenne stable.

2.5 Implémentation et analyse d'un MLP pour une application de classification

L'application de reconnaissance de caractères permettra d'analyser par simulations logicielle, le comportement d'une architecture de réseau de neurones de perceptrons à trois couches. Il a été démontré que cette configuration du MLP est suffisante pour la résolution des applications de classification [17].

Les résultats de ces analyses serviront plus tard à faire des études de comparaison de cette architecture et de celle d'un réseau de quantrons pour le même type d'application.

2.5.1 Modèle théorique

Le réseau de neurones artificiels multicouches est constitué d'un ensemble de perceptrons groupés sous différentes couches et interconnectés entre eux. En considérant que l'application est une représentation de caractères sous forme de pixels, un ensemble de matrices d'échantillons pour l'apprentissage du réseau à vingt six caractères a donc été élaboré.

Il faut cependant noter que l'application de reconnaissance de caractères est une application très répandue. Il existe un grand ensemble de ce type d'applications [46]. L'un des algorithmes les plus performants sur le marché est LeNET d'AT&T développé par LeCun [26]. Mais, le but de l'implémentation est d'obtenir un modèle de référence qui permettra d'analyser les performances du quantron par rapport aux méthodes de classification utilisant une architecture de MLP.

Pour chaque modèle de données, l'introduction des erreurs dans un pourcentage variant de 0 à 20% est amorcée. Elle est nécessaire dans l'étape d'apprentissage afin que le réseau augmente son taux de bonne classification.

Le nombre de neurones de la couche d'entrée est égal à la dimension du vecteur d'entrée. Le nombre de neurones de la couche cachée sera supérieur au nombre de neurones de la couche d'entrée. Ce choix permet d'augmenter la possibilité de séparation des classes. Chaque modèle de classe est associé à un neurone de la couche de sortie. De ce fait, le nombre de neurones de la couche de sortie est égal au nombre de modèles de classe à reconnaître.

2.5.2 **Simulations du MLP**

La programmation de ce réseau de MLP a été faite à l'aide du langage haut niveau le C++. L'application supportée est la reconnaissance de lettres préalablement entraînées. Celles-ci sont présentées sous forme d'une matrice paramétrique (*a, b*) où a *et b* représentent l'espace de coordonnées du vecteur d'entrée.

Le nombre de neurones de la couche d'entrée est égal au produit de *a* et de *b*. Un espace de dimension 7x5 est déterminé. Il en résulte 35 neurones pour la couche d'entrée. Chaque élément de la matrice (vecteur d'entrée) représente un pixel qui peut prendre une valeur entre 0 et 1. La valeur 1 est associée à un pixel activé. Dans le cas contraire, la valeur 0 l'est.

Le nombre de neurones de la couche cachée est donné par l'équation suivante :

$$Nbre\_neurones\_cachée \geq Nbre\_neurones\_entrée + Nbre\_neurones\_sortie \quad (2.1)$$

Le modèle du MLP à trois couches ainsi proposé est constitué d'une couche d'entrée ayant 35 neurones qui représentent les pixels de la matrice de données, d'une couche cachée de 61 neurones (somme des neurones d'entrée et de sortie) et d'une couche de sortie de 26 neurones qui représentent les lettres de l'alphabet.

Dans la définition de l'algorithme de programmation par objets, deux classes ont été considérées dont la classe perceptron et la classe MLP. La classe perceptron représente l'élément de base de la structure et celle de MLP inclut en elle les fonctions des neurones et les couches.

La phase d'apprentissage a été faite selon l'algorithme de rétropropagation de l'erreur dans lequel l'erreur quadratique moyenne (MSE) est minimisée :

Tableau 2-2 : Partie du code de l'algorithme de rétropropagation du MLP

```
-----
IF (MSE < MSE_max) break;
// Ensuite: calcul de l'erreur quadratique de la couche cachée ainsi que de la variation de poids
// des nœuds de cette couche.
                    FOR (j = 0; j < taille_cache; j++) {
                                FOR (k = 0, somme = 0; k < taille_sortie; k++)
                                somme += delta_poid_sortie [k] * noeud_de_sortie -> nœud [k].poids [j];
  // variation de poids de la couche cachée formule 4.34 (neural network, Haykin, 2nd)
                                delta_poid_milieu [j] = somme * noeud_cache -> nœud [j].sortie * (1 -
noeud_cache -> nœud [j].sortie);
                    }
// Mise à jour des poids de la couche de sortie.
                    FOR (k = 0; k < taille sortie; k++)
                                FOR (j = 0; j < taille cache; j++)
                                            noeud_de_sortie -> nœud [k].poids [j] += eta * delta_poid_sortie [k] *
noeud_cache -> noeud [j].sortie;
// Mise a jour des poids de la couche cachée (milieu).
                    FOR (j = 0; j < taille_cache; j++)
                                FOR (i = 0; i < taille_entree; i++)
                                            noeud_cache -> nœud [j].poids [i] += eta * delta_poid_milieu [j] *
donne [i];
-----
```

Dans chaque étape d'apprentissage, les matrices initiales des lettres sont modifiées en introduisant aléatoirement un pourcentage d'erreur afin de présenter un nombre raisonnable d'échantillons au réseau. Un total de 2600 échantillons soit 100 échantillons par caractère a été utilisé. Les figures suivantes montrent un exemple de simulation du caractère H.

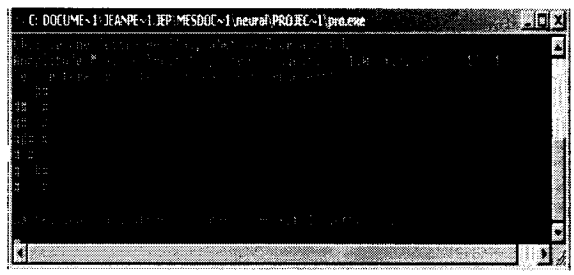


Figure 2-2 : Simulation du caractère H bruité à 20%. Le réseau donne comme résultat le caractère E.

Dans le cas de l'exemple de la figure 2-2, le MLP commet une erreur de résultat de classification. Il présente la lettre E en sortie contrairement à la lettre H présentée en entrée.

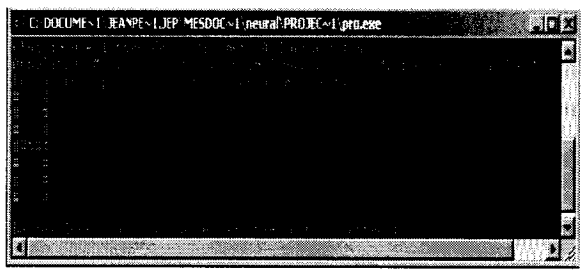


Figure 2-3 : Simulation du caractère H bruité à 2%. Le résultat obtenu est le caractère H

Dans le cas de l'exemple de la figure 2-3, le MLP donne le vrai résultat de classification. Il présente la lettre H en sortie similaire à la lettre présentée en entrée.

2.5.3 **Analyse des résultats d'implémentation du MLP**

La figure 2-5 représente le taux de réussite de classification en fonction du pourcentage d'erreur ajouté et du nombre d'échantillons présenté lors de l'apprentissage.

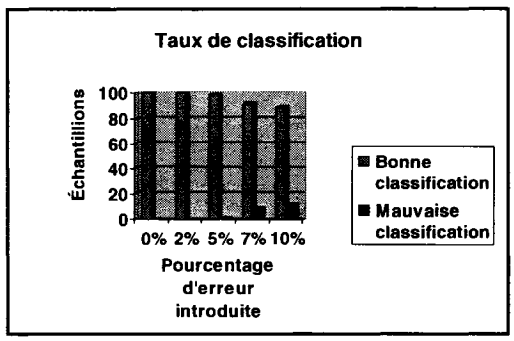


Figure 2-4 : Taux de classification du MLP en fonction du pourcentage d'erreur introduit dans les données d'entrées.

On constate que lorsque le pourcentage de l'erreur dépasse 5%, le système ne parvient plus à bien classifier les données présentées. Ce qui est compréhensible compte tenu la taille restreinte de la matrice de données d'entrée. Pour pallier à cette situation, il faudrait augmenter sa taille et par conséquent, le nombre des neurones des couches d'entrée et cachées.

D'une façon générale, utiliser un perceptron pour faire une séparation non linéaire aboutit à des résultats assez décevants. Les possibilités de ce module sont extrêmement limitées. Alors, imiter plus fidèlement la structure du cerveau en faisant travailler ensemble plusieurs neurones semble-t-elle envisageable ? Supposons que le but soit de séparer deux ensembles de points arbitrairement choisis en utilisant seulement un nombre fini de séparations linéaires. En dimension n, sachant qu'on peut utiliser $2n$ hyperplans afin d'enfermer un point dans une petite boîte, en choisissant bien les coefficients de $2n$ neurones, il est possible de s'arranger pour que seuls les points d'un petit cube de l'espace donnent une réponse égale à 1 pour chacun des neurones. Un point extérieur donnera toujours au moins un 0. S'il existe p points, $2np$ neurones qui permettent de les reconnaître peuvent être associés à ces points. L'opération logique de reconnaissance sera en fait un *ET* logique effectué sur tous les neurones correspondant à une "boîte" donnée. En utilisant ainsi $2np$ neurones suivis de p portes logiques, une porte peut être jumelée à chaque point. Il suffit ensuite de faire un *OU* logique sur les portes afin de séparer certains points des autres. Or, il se trouve que quelle que soit la dimension de l'espace, les opérations *ET* et *OU* sont linéairement séparables, c'est-à-dire qu'il y a pour chacune d'entre elles un neurone permettant de la calculer. En utilisant un ensemble de neurones possédant une structure particulière, la séparation des deux ensembles de points choisis arbitrairement sera plausible. Une fois l'analyse des simulations effectuée, l'évaluation de la performance de cette structure peut être faite en l'implémentant matériellement.

2.5.4 **Implémentation matérielle du MLP**

Quand l'implémentation logicielle du réseau MLP n'est pas satisfaisante du point de vue de la vitesse d'exécution, une alternative est l'implémentation matérielle de ce réseau dans un Field Programmable Gate Array (FPGA).

Un banc d'essai post-synthétisé est réalisé pour évaluer le comportement de ce modèle de réseaux multicouches [12] [8] [9]. Cette conception est faite en utilisant un FPGA MAXII. Le diagramme bloc suivant représente la structure logique de cette implémentation.

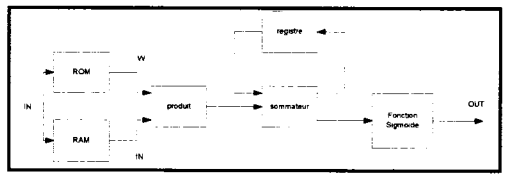


Figure 2-5 : Architecture matérielle du perceptron.

Cette implémentation devient plus complexe lorsqu' il s'agit d'un réseau de perceptrons multicouches. Le diagramme suivant montre la structure d'un réseau.

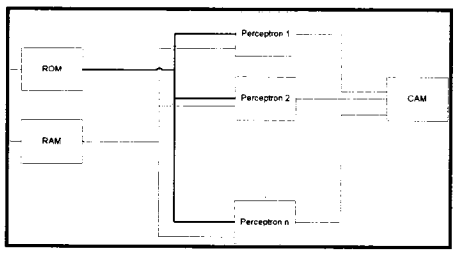


Figure 2-6 : Architecture matérielle du MLP.

Dans l'implantation des réseaux de neurones en circuits intégrés analogiques, la non linéarité est habituellement implicite dans une des autres opérations mises en œuvre dans le neurone. En guise d'exemple, la multiplication ou l'addition analogique dans laquelle l'amplificateur introduit la notion de saturation; par conséquent, la nature de la fonction non linéaire est plutôt non contrôlée.

L'approximation linéaire par morceaux de la fonction sigmoïdale, fonction de transfert du perceptron, est nécessaire. Une méthode serait de procéder comme dans [3] à une segmentation linéaire de cette fonction avec une mémorisation des points critiques (qui sont les extrêmes de chaque segment de droite). Il s'agit de concevoir une table de valeurs combinées avec une approximation linéaire. La division de la fonction sigmoïdale en 16 segments est proposée. Ainsi, la valeur du vecteur d'entrée peut être comprise dans un intervalle [-15, 15] et la sortie Y dans l'intervalle [0, 1].

Quant à la sortie du réseau, l'utilisation d'une CAM (Content Adressable Memory), une mémoire à contenu adressable, est possible. Il s'agit d'une mémoire qui permet d'effectuer une recherche en parallèle dans un tableau de cellules de mémoires selon une valeur de données d'entrée. Elle permet de savoir si l'élément à déterminer se trouve dans le tableau. Mais elle n'autorise pas une recherche sur un sous-groupe contrairement à une TCAM (Ternary Content Addressable Memory). Cette dernière supporte la notion d'entrée indifférente.

L'implémentation matérielle des réseaux multicouches de grande dimension cause souvent des problèmes de complexité de circuits. Ainsi, une contrainte au niveau de la performance de ceux-ci peut survenir [20].

2.6 Conclusion

Les systèmes séquentiels s'approchent de leur limite d'où provient l'intérêt à présent des processeurs en parallèle. Cette dernière nécessite la recherche des algorithmes pouvant les porter à leur pleine capacité d'utilisation. Cependant, les réseaux de neurones artificiels, par définition, permettent ce parallélisme car, ils sont des ensembles d'automates élémentaires capables de travailler simultanément.

Les réseaux de neurones artificiels utilisent des algorithmes d'apprentissage pour solutionner une application spécifique. Tel que présenté dans ce chapitre, il existe une variété d'algorithmes d'apprentissage pour les diverses familles de réseaux de neurones artificiels. Ces algorithmes sont conçus dans le but d'optimiser la performance d'un modèle de réseau donné, tout en essayant de maintenir la fonctionnalité de celui-ci très proche de celle du modèle du cerveau humain. Son implémentation peut être faite selon plusieurs approches ; à savoir totalement logicielle, en utilisant des langages de programmation évoluée, mixte (logicielle/matérielle) ou totalement matérielle.

Cependant, l'implémentation matérielle de larges réseaux, comme dans le cas des perceptrons multicouches, cause souvent des problèmes de complexité de circuits. Ceci peut devenir une contrainte au niveau de performance due au possible latence créée par exemple, par l'augmentation des multiplexeurs en cascade. Pour contourner ces limites, la conception d'une nouvelle machine à état liquide (LSM) à base du quantron est envisagée. Elle devra réagir comme un approximateur universel. La méthode

d'apprentissage, dans ce système, sera inspirée de la règle de p-delta [4]. Le chapitre 5 est entièrement dédié à cette nouvelle machine.

CHAPITRE 3

IMPLÉMENTATION ET ANALYSE DU QUANTRON

Le quantron, basé sur la forme analytique du mouvement brownien géométrique, est un nouveau neurone artificiel qui s'avère présenté de grandes similitudes avec le neurone humain. Cette nouvelle formulation mathématique du neurone à été conçue par Labib [23]. Son existence a engendré une nouvelle vision de la modélisation du neurone biologique par sa façon de véhiculer l'information et d'interpréter à sa sortie une donnée d'entrée. Plus précisément, l'idée consiste à transmettre une information reçue en entrée du neurone sous forme d'impulsions dont la destination est la sortie du quantron considéré.

Dans la première partie du chapitre, le quantron est expliqué. Une analyse de l'implémentation de sa structure mathématique y est développée. La seconde partie du chapitre est consacrée à des comparaisons de ce nouveau modèle de neurone et des modèles usuels existant afin d'optimiser la conception d'un nouveau réseau de quantrons pour la résolution des problèmes de classification.

3.1 Le modèle mathématique du Quantron

L'interprétation mathématique du comportement du neurone biologique consiste à imiter son fonctionnement. Cette approche a pour but de créer des systèmes déterminant des relations

complexes et non linéaires entre des données d'entrée d'une application prédéfinie et ses résultats interprétés par les sorties du système.

Comme dans le cas des neurones artificiels usuels, le quantron se compose d'un ensemble d'entrées qui sont associées chacune à trois paramètres (W, S, θ) internes et un paramètre externe (n). Ils sont tous interprétés par une fonction de sortie définissant la conduction ou la non conduction du système. La structure suivante représente le modèle d'un quantron à deux entrées.

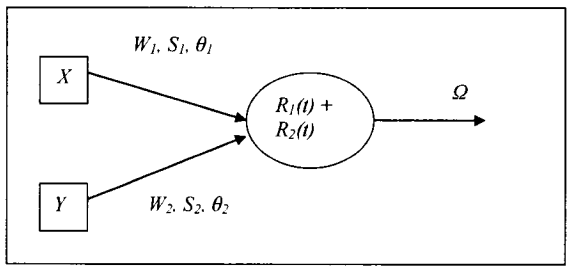


Figure 3-1 : Modèle formel d'un quantron à 2 entrées X et Y

Les paramètres internes W_i, S_i et θ_i, où $i = \{1,2\}$ sont nécessaires à l'indication du potentiel d'excitation ou d'inhibition du signal d'entrée associé. Le paramètre Ω est la sortie du quantron. Dans le tableau 3-1, les paramètres d'un quantron sont définis.

Tableau 3-1 : Définition des paramètres d'une entrée d'un quantron.

Paramètres	**Définitions**
W	Il s'agit d'une constante représentant le poids du raccordement des impulsions du potentiel post-synaptique. Elle intervient entre une entrée du quantron et sa résultante. Elle peut être vue comme l'équivalent d'un poids synaptique dans un perceptron. Le signe de sa valeur servira d'indication au type de potentiel d'action d'une donnée d'entrée.
S	C'est le cycle de communication entre une membrane émettrice et une membrane réceptrice. Il représente le temps nécessaire à la transmission d'une information entre deux neurones.
θ	Elle détermine la latence ou le déphasage avant le début de la transmission.
n	Nombre de cycles de communication d'une donnée d'entrée. Cette constante est associée directement au potentiel post-synaptique.

Le fonctionnement du quantron repose sur le mécanisme de transmission d'une information du neurone biologique comme le montre la figure 3-2.

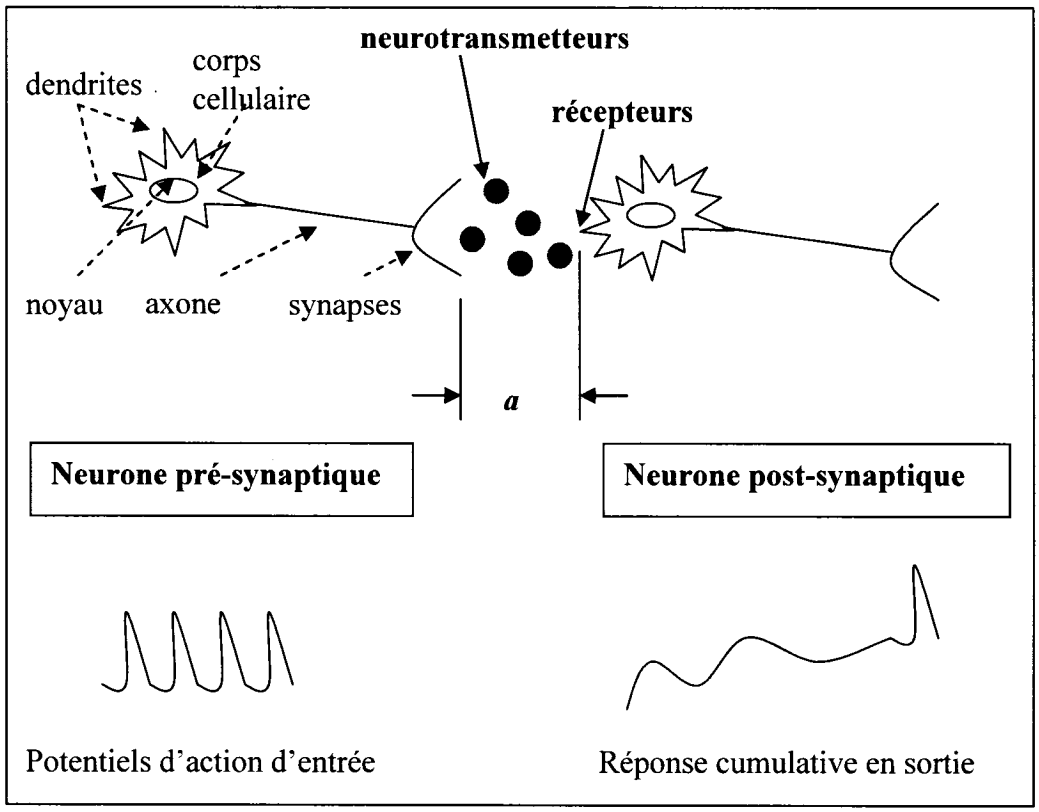


Figure 3-2 : Illustration d'une transmission d'informations entre deux neurones biologiques.

En effet, lors de la transmission d'un signal, il provient du potentiel de la membrane pré-synaptique par des impulsions d'action qui sont interprétées comme des charges électriques. Cette énergie circulant dans un canal de communication de très petite dimension est conduite vers une cellule réceptrice ou membrane post-synaptique. L'arrimage de ces transmetteurs sur la membrane post-synaptique génère une action qui modifiera le potentiel d'action de cette réceptrice en élément d'excitation ou d'inhibition. Le calcul de ce potentiel est défini par la formule suivante :

$$V(t) = \frac{2w}{\sqrt{2\pi t}} \int_{\ln a}^{\infty} e^{-\frac{x^2}{2t}} dx = 2w\left(1 - \phi\left(\frac{\ln a}{\sqrt{t}}\right)\right) \tag{3.1}$$

Les variables de l'équation (3.1) sont :

- W, le poids synaptique du quantron
- a, la distance entre le point d'émission du potentiel et le point de réception de celui-ci
- t, le temps
- $\phi(\bullet)$, la fonction de répartition de la variable aléatoire gaussienne centrée réduite.

La définition de la fonction de densité est donnée par la formule suivante :

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{3.2}$$

La variation du potentiel d'action est limitée par une mesure du temps S qui est un paramètre du quantron. Il définit la période nécessaire que prendra le neurotransmetteur d'un signal d'entrée pour agir sur la membrane réceptrice (figure 3-2). Cette mesure permet de calculer la variation du potentiel V_0 de la membrane post-synaptique. On a :

$$V_0 = V(t) \times (u(t) - u(t - S)) \tag{3.3}$$

Où u $(\bullet)$ représente la fonction échelon (encore appelée fonction Heaviside)

L'émission d'un potentiel d'action crée un potentiel d'action final V_F sur la membrane post-synaptique tel que :

$$V_F = -V(t-S)\times(u(t-S)-u(t-2S))+V(S)\times(u(t-S)-u(t-2S)) \qquad (3.4)$$

Ces définitions de variation de voltages initial et final permettent d'exprimer la variation globale du niveau de polarisation dans le neurone post-synaptique. Cette fonction notée $\varphi(t)$ est donnée par :

$$\varphi(t) = V(t)\times u(t)+(V(S)-V(t)-V(t-S))\times u(t-S)+(V(t-S)-V(S))\times u(t-2S) \quad (3.5)$$

Cette fonction détermine la variation du niveau de polarisation dans une synapse. Elle est similaire à la fonction d'activation du perceptron usuel.

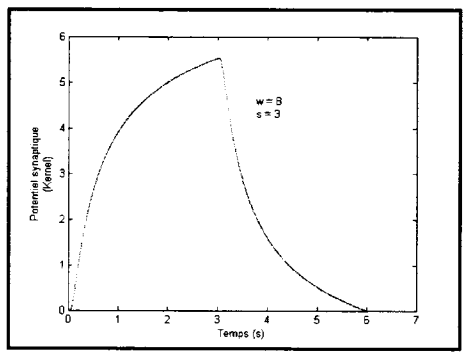


Figure 3-3 : Exemple de variation du niveau de polarisation dans une synapse excitatrice (W est positif) résultant d'un seul potentiel d'action.

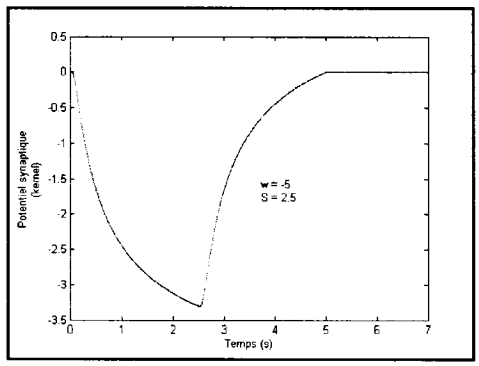


Figure 3-4 : Exemple de variation du niveau de polarisation dans une synapse inhibitrice (*W* est négatif) résultant d'un seul potentiel d'action.

La propagation d'une réponse potentielle, d'un neurone vers un autre, se fait par la libération des neurotransmetteurs à travers la fente synaptique *a* (figure 3-2). Cette libération est due à la présence d'un potentiel d'action du neurone émetteur. Cependant, son amplitude, dans le cas du quantron, n'est pas proportionnelle au pourcentage des neurotransmetteurs relâchés. C'est plutôt la fréquence d'arrivée des potentiels d'action qui détermine une éventuelle réponse sous forme de dépolarisation ou d'hyperpolarisation du neurone post-synaptique. Dans ce cas, le contrôle de l'accumulation des potentiels d'action dans le temps détermine s'il y a déclenchement ou non du potentiel d'action post-synaptique. Mathématiquement, ceci revient à calculer la sommation temporelle de la fonction $\varphi(t)$.

Sachant que x est le temps s'écoulant entre deux arrivées successives du potentiel d'action et est strictement supérieur à zéro, x^{-1} définit la fréquence pré-synaptique des signaux d'entrée. Les

entrées du quantron sont des fréquences d'arrivée des signaux émis contrairement au perceptron qui a pour entrées les amplitudes des signaux émis.

La variation de voltage cumulative V_c survenant à l'intérieur de la membrane en fonction du temps est donnée par :

$$V_c = \sum_{i=0}^{+\infty} \varphi(t - ix) \tag{3.6}$$

En considérant le délai θ représentant le temps écoulé avant l'arrivée du premier potentiel d'action de la membrane pré-synaptique, la réponse totale *R(t)* du neurone post-synaptique est définie par:

$$R(t) = \sum_{i=0}^{+\infty} \varphi(t - \theta - ix) \tag{3.7}$$

Le seuil Γ est le niveau à atteindre en sortie pour que le quantron soit stimulé. Si la réponse totale en sortie est inférieure au seuil, il y aura absence de transmission donc impossibilité de propagation du message. La sortie du quantron est déterminée par la fonction suivante :

$$\Omega = \begin{cases} 0 & Si \sum_{k=1}^{m} R(t) < \Gamma, \quad \forall t \in [0, \infty) \\ \alpha & Sinon \end{cases} \tag{3.8}$$

où :

- m = nombre d'entrées synaptiques.

- $\alpha = \inf \{t > 0 : \sum_{k=1}^{m} R(t) = \Gamma\}$ (c'est le premier instant où la valeur du seuil est atteinte dans le temps).

Pour déterminer les paramètres du quantron (W, S et θ) dans un problème donné, il faut préalablement fixer les valeurs des constantes a et Γ qui représentent respectivement la longueur de la fente synaptique et le seuil critique à atteindre.

Les figures 3-5 à 3-8 montrent un exemple de résolution d'un cas de classification du problème du XOR à l'aide d'un quantron à 2 entrées.

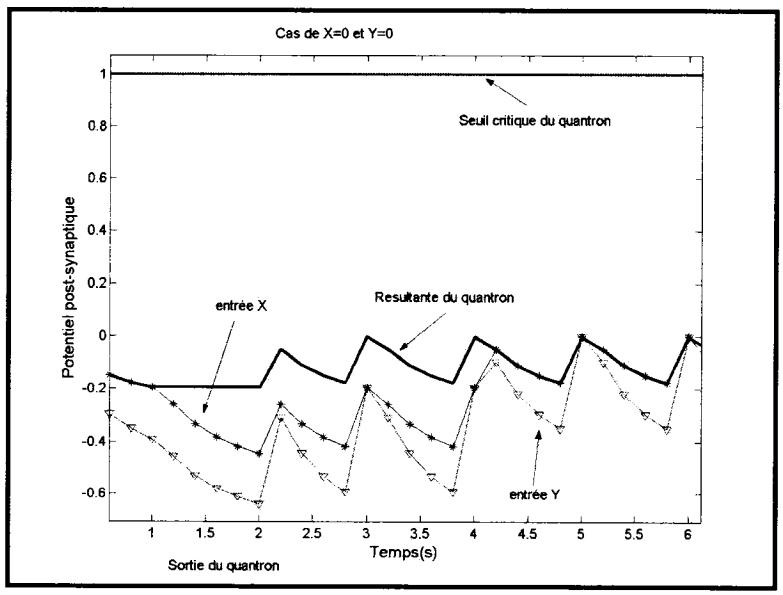


Figure 3-5 : Cas de classification du XOR par un quantron à 2 entrées X=0 et Y=0

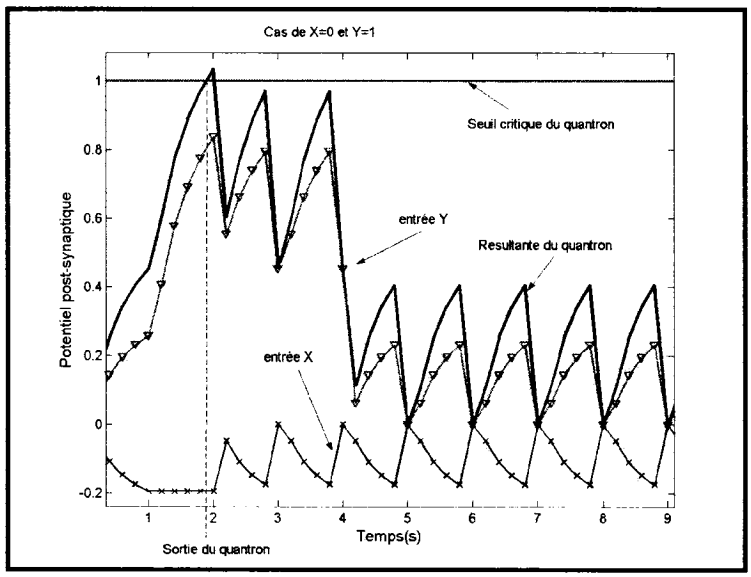


Figure 3-6 : Cas de classification du XOR par un quantron à 2 entrées X=0 et Y=1

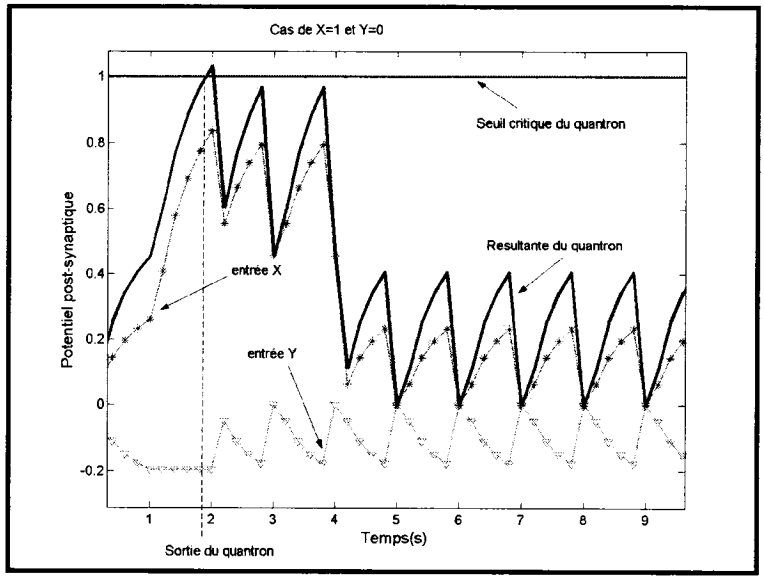


Figure 3-7 : Cas de classification du XOR par un quantron à 2 entrées X=1et Y=0

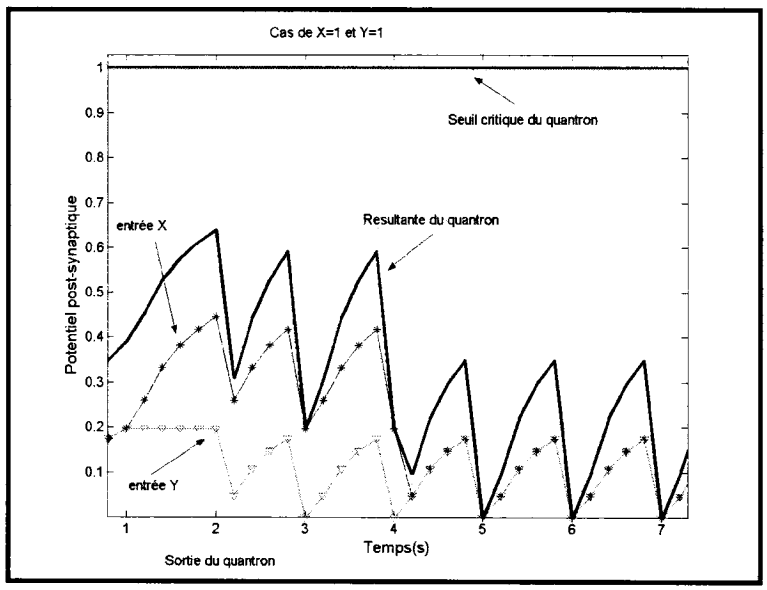


Figure 3-8 : Cas de classification du XOR par un quantron à 2 entrées X=1 et Y=1

Afin de développer pour la première fois un algorithme d'apprentissage efficace, il est essentiel de définir de façon approchée le mécanisme de calcul du quantron par similitude au comportement du perceptron. Ce qui permet d'obtenir une formulation mathématique plus simple du niveau de la polarisation globale $\rho(t)$ dans une synapse représentée par :

$$\rho(t) = w \times (u(t) - u(t - S)) \tag{3.9}$$

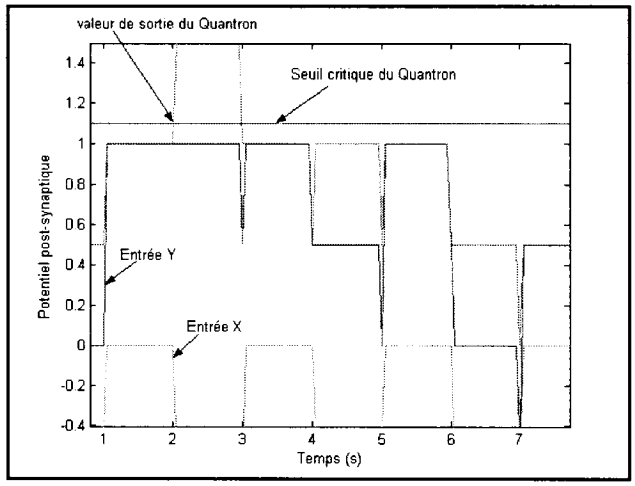


Figure 3-9 : Cas de classification du XOR par un quantron à 2 entrées $X=1$ et $Y=0$ utilisant l'expression d'approximation du niveau de polarisation de la synapse.

L'efficacité de cette nouvelle structure a été démontrée dans la résolution du problème non linéaire de la classification d'un OU-EXCLUSIF (XOR). Cette analyse [23] a montré qu'un seul quantron à deux entrées est capable de séparer en deux classes les points d'un XOR en utilisant six paramètres. Par contre, un seul perceptron ne peut résoudre un tel problème. Il s'agit d'un cas non linéairement séparable. L'utilisation d'un réseau de perceptrons nécessite au moins sept paramètres et une couche cachée pour solutionner le problème du XOR.

Le fait de pouvoir utiliser moins de paramètres pour la résolution des problèmes plus complexes contrairement aux méthodes standards, tout en minimisant la complexité topologique du réseau, donne lieu d'explorer plus en détails les possibilités qu'offre l'utilisation du quantron.

Aucune application n'a encore été testée en utilisant un réseau de quantrons. Ceci est dû au fait que la conception d'un tel réseau nécessite encore du travail au niveau de la compréhension du

mécanisme de définition de certains paramètres d'initialisation et de son algorithme d'apprentissage qui demeure encore indéfini. Mais cette nouvelle innovation est-elle vraiment plus performante que le perceptron ? Afin de répondre à ces questions, il importe de comparer le quantron au perceptron.

3.2 Étude comparative entre le quantron et le perceptron

Le perceptron est un modèle de neurone artificiel du à Mc Culloch et Pitts [32] dont la fonction mathématique est une simple dérivée d'une analyse du neurone biologique [17]. La figure suivante montre le diagramme bloc d'un perceptron.

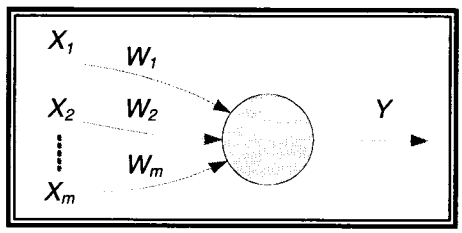


Figure 3-10 : Diagramme bloc du perceptron à m entrées.

X_i {i=1....m} représente le vecteur d'entrée, b est le biais, W_i les poids synaptiques et Y la sortie du perceptron.

L'équation de la sortie du perceptron est donnée par :

$$Y = S\left[bW_1 + \sum_{i=2}^{m}(W_i X_i)\right] \tag{3.10}$$

Ou *S(R)* la fonction d'activation. Dans le cas d'une sigmoïde, on obtient :

$$S(R) = \frac{1}{1+\exp(-R)} \tag{3.11}$$

Contrairement au quantron, l'équation (3.10) montre que le perceptron fait abstraction de la notion temporelle du neurone biologique. De ce fait, il remplace l'intégration temporelle par une simple sommation des signaux arrivant (entrées) au neurone. Par la suite, la somme obtenue est comparée à un seuil afin de déduire la sortie du neurone. L'obtention de ce comportement nécessite la soustraction du seuil considéré à la somme des entrées. Alors, le résultat obtenu est transféré par une fonction de type sigmoïdal. Le résultat après transfert représente la sortie du neurone. Cet enchaînement "sommation" puis "non-linéarité" représente finalement les propriétés "physiques" du neurone.

Les équations (3.3) et (3.4) montrent que la quantron tient compte des potentiels d'actions respectivement initial et final d'un stimulus provenant de la membrane pré-synaptique. Le quantron évalue la variation globale de ces potentiels selon l'équation (3.5). Le temps d'émission et la période d'activation des stimuli ainsi émis sont pris en compte. Dans le cas ou le quantron conduit, il produit en sortie une valeur du temps correspondant au premier instant où la somme des variations des potentiels émis (3.7) atteint sa valeur de seuil critique.

Le quantron se sert donc d'une valeur de seuil pour activer sa sortie contrairement au perceptron qui emploie une fonction de seuil généralement de la forme sigmoïdale. La flexibilité de la

structure du quantron, grâce à ces paramètres de déphasage θ et de temps de transmission nécessaire (*S)* pour un stimulus, lui permet de pouvoir résoudre des problèmes non-linaires comme le cas du *XOR*.

Le tableau 3-2 résume les caractéristiques d'un perceptron versus celles du quantron.

Tableau 3-2 : Comparaison du perceptron et du quantron.

Perceptron	**Quantron**
▪ Ne fait pas intervenir la notion du temps (équation 3.10)	▪ Simulé par un potentiel d'action produisant une onde variante dans le temps à chacune des synapses (équation 3.3 et 3.4)
▪ Simple sommation des données d'entrée (équation 3.10)	▪ La somme de ces ondes produit en sortie l'activité du quantron (équation 3.7).
▪ Résultat de la sommation est comparé à un seuil et on en déduit la sortie du perceptron (équation 3.10) est déduite.	▪ La sortie émet un signal lorsque le potentiel de sortie atteint ou dépasse le seuil critique prédéfini (figure 3.5.).
▪ Utilise une fonction de transfert de type sigmoïdal (équation 3.11)	▪ Utilise une fonction de seuil pour activer la sortie (équation 3.8)
▪ Fonction discriminante linéaire	▪ Résout des problèmes linéaires ou non linéaires (figure 3.4)
▪ Utilise une fonction linéaire, Heaviside ou sigmoïdale comme fonction d'activation	▪ La fonction d'activation est une fonction qui est définie selon une valeur de seuil critique Γ (équation 3.5).
▪ Manque de similitudes avec le neurone biologique	▪ Grande similitude avec le neurone biologique

Comme nous l'avons analysé dans le chapitre précédent, le perceptron est généralement utilisé dans un réseau multicouche. Le principe consiste à regrouper les perceptrons par des couches placées de bout en bout et complètement connectées entre deux couches adjacentes. En d'autres termes les entrées des neurones de la seconde couche sont les sorties des neurones de la première couche. Les neurones de la première couche sont reliés aux données d'entrée du réseau. Ils calculent leurs sorties qui sont transmises aux neurones de la couche suivante et ainsi de suite.

Ainsi, le MLP calcule une fonction vectorielle. Il permet l'ajustement de ses paramètres grâce à un apprentissage supervisé appliqué à l'algorithme de rétropropagation de l'erreur. Les valeurs des connexions synaptiques et des seuils sont ajustées afin de modifier la fonction évaluée.

Par contre, le réseau de quantrons n'admet pas toujours une circulation de l'information vers la sortie. Si la conception d'un réseau de quantrons multicouches est envisagée, il importe de prendre en considération l'aspect expliqué dans la figure 3-11 :

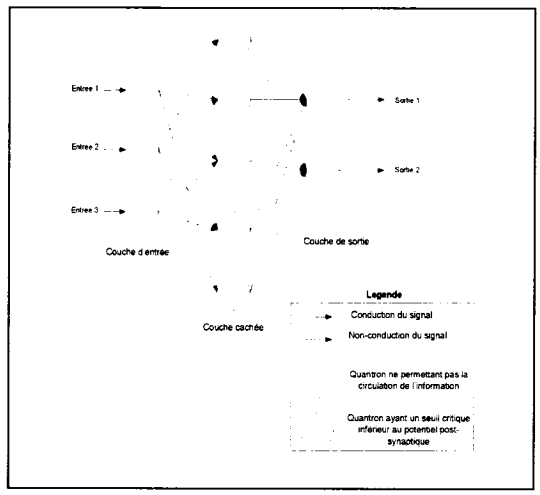


Figure 3-11 : Exemple de topologie d'un réseau de quantrons multicouches ayant des cas de non conduction de l'information vers la sortie.

Le mécanisme de conduction du quantron nécessite d'atteindre au minimum l'amplitude de sa fonction de seuil. Cette problématique demande une modification de la méthode standard d'apprentissage supervisé par rétropropagation. Cette modification consiste à forcer, dans certains cas, la circulation de l'information afin de pouvoir toujours rétropropager l'erreur. D'où la nécessité de définir un algorithme spécifique au réseau de quantrons multicouches.

3.3 Comparaison du quantron aux neurones à impulsions

Le transport de l'information dans le neurone s'effectue par les «spikes» (potentiel d'action) des dendrites aux synapses sous forme d'influx nerveux le long de l'axone. Cela correspond à une dépolarisation électrique de la membrane pré-synaptique. Le potentiel d'action, en pratique, se propage à une vitesse constante sans dégradation, ni atténuation du signal aux synapses. Il peut être affecté par la présence de bruit causée par plusieurs facteurs dont l'écart entre des potentiels d'action, la présence d'autres potentiels d'action, la modification de la fréquence. L'émission et le transport des potentiels d'action ont été largement étudiés par Hodgkin et Huxley [19]. Cette étude a débouché sur le développement d'une famille de réseaux de neurones appelés réseaux à impulsions (spiking neurones) dont les plus populaires sont décrits par Gerstner et Kistler [16].

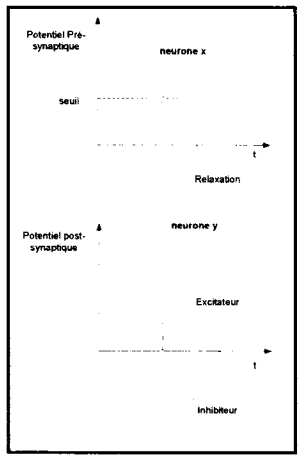


Figure 3-12 : Exemple de transmission d'impulsions d'un neurone x vers un neurone y.

Le neurone x génère un potentiel d'action quand le potentiel de sa membrane dépasse le seuil. Après cette génération d'impulsions, le potentiel de la membrane du neurone x est réinitialisé à son état de repos. Pourtant, le quantron ne considère pas le temps de réinitialisation comme hypothèse dans sa formule mathématique. Cependant, il tient compte de la distance a entre la longueur de la fente post-synaptique et pré-synaptique (équation 3.1). Ceci laisse au concepteur la liberté d'introduire ou non le temps de relaxation selon les besoins de l'application.

Dans le neurone à impulsions, l'impulsion d'entrée crée une réponse au niveau du neurone y (neurone post-synaptique). Le signe du voltage de cette réponse permettra de déterminer s'il s'agit d'un potentiel excitateur ou inhibiteur. Pour le quantron une émission d'impulsions vers la membrane post-synaptique est excitatrice, si la variation du niveau de polarisation dans la synapse augmente (Figure 3-3). Elle est inhibitrice dans le cas contraire (Figure 3-4).

Dans un réseau de neurones à impulsions, le temps est la métrique primordiale [6]. Ainsi, la méthode de codage d'impulsions varie en fonction du type de neurones à impulsions exploité.

Par exemple, le codage par intervalle entre deux potentiels d'action successifs (ISI) présente plusieurs avantages non négligeables par rapport au codage par taux moyen d'émission d'impulsions (codage fréquentiel). L'un de ces avantages est la réduction du temps de réception du potentiel pré-synaptique. Le codage fréquentiel est simplement scalaire contrairement au codage temporel par ISI qui peut être vectoriel [10]. Maass et Bishop [29] décrivent deux familles de codes à savoir le codage à impulsions (pulse code) et le codage de taux d'émission (rate code). Ils constatent que l'apprentissage de réseaux de neurones à impulsions est généralement un apprentissage non supervisé.

Cette différence fait constater que les neurones à impulsions sont moins optimisés pour des applications de classification ou de régression [7]. Même si le principe de base du quantron est similaire à celui d'un réseau à impulsions, sa sortie génère une seule unité temporelle. Cette dernière définit le moment où la résultante atteint la valeur seuil du quantron. Tandis qu'un neurone à impulsions émet une série d'impulsions en sortie de la fonction des impulsions d'entrée.

3.4 Conclusion

Le quantron est un modèle mathématique de représentation du neurone formel présentant de la flexibilité dans son implémentation. Il est adaptable à plusieurs modèles de réseaux. Un autre avantage de ce modèle réside dans sa capacité à résoudre des problèmes complexes, contrairement au modèle du perceptron qui se limite à la résolution des problèmes linéaires uniquement.

La difficulté majeure du quantron est l'implémentation d'algorithmes d'apprentissage adaptés à la fonction mathématique de sa résultante. Pour un réseau de quantrons multicouches, la propagation du signal d'entrée doit être garantie afin d'obtenir une valeur de sortie qui sera interprétée de façon inconditionnelle. Si toute valeur d'entrée produit une sortie comparable à une valeur désirée, le mécanisme d'ajustement des paramètres doit être défini en fonction des paramètres du quantron. Cette étape d'ajustement de paramètres ne peut être accomplie en utilisant la formule directe de la fonction de sortie du réseau car cette dernière n'a pas de lien direct avec les entrées d'une part. Sa valeur de sortie n'a pas une période prédéfinie d'autre part. Elle exigera une analyse mathématique. Cette analyse consistera à trouver une approximation de la fonction de la résultante du quantron (fonction de sortie) de telle sorte que celle-ci soit fonction des entrées. Cette approximation permettra de faciliter l'implémentation du calcul de dérivées de la fonction des coûts (fonctions d'erreurs). Ceci facilitera l'ajustement des paramètres des quantrons du réseau dans le processus de rétropropagation.

CHAPITRE 4

ALGORITHME D'APPRENTISSAGE DU RÉSEAU DE QUANTRONS MULTICOUCHES

Les réseaux de neurones artificiels sont généralement utilisés dans la résolution des problèmes complexes de classification ou de prédiction. Dans le cas de la reconnaissance de formes, on se sert de réseaux de neurones à apprentissage supervisé [30] [27]. Cependant, ces réseaux usuels dont les neurones ont des sorties définies en fonction des entrées peuvent requérir un apprentissage de rétropropagation de l'erreur par la méthode de la descente du gradient.

Le quantron, grâce à sa similitude avec les neurones biologiques, contrairement au perceptron, motive l'élaboration d'un réseau à base d'un apprentissage supervisé. Cette première conception du MLQ servira de jalon à l'amélioration de la performance des mécanismes d'apprentissage des applications de classification.

L'équation de sortie des réseaux de quantrons multicouches (MLQ) ne dépend pas directement de ses entrées mais plutôt des valeurs des seuils critiques des quantrons intrinsèques. L'application utilisée pour cette implémentation est la reconnaissance de caractères qui sont représentés par une matrice bidimensionnelle de coordonnées cartésiennes (X, Y). Étant donné que, par définition, les entrées d'un quantron sont fonction d'impulsions fréquentielles des potentiels pré-synaptiques, il importe de définir les coordonnées de cette matrice dans un nouveau référentiel en coordonnées temporelles (x_1, x_2). Chaque élément de la matrice exprime le nombre d'impulsions émis par la membrane pré-synaptique aux entrées des neurones.

Suite au changement du système de coordonnées de la matrice de représentation des caractères, une approximation de la fonction de sortie du quantron est matérialisée afin de simplifier l'apprentissage du réseau. Cette fonction devra relier directement les sorties des quantrons cachés à leurs entrées, ce qui permettra une liaison implicite entre les entrées et les sorties du réseau.

Cette transformation de la fonction de sortie du quantron conduit à un nouveau MLQ qui est employé dans l'implémentation de l'algorithme d'apprentissage. Ce dernier a été inspiré de la rétropropagation de l'erreur par la méthode de la descente du gradient telle qu'utilisée dans le cas des perceptrons multicouches (MLP). Cependant, certaines modifications ont été nécessaires afin de permettre une convergence de la fonction d'erreur. Cette minimisation de l'erreur obtenue à la sortie du réseau a été améliorée par l'intégration des principes du «resilient propagation» (RPROP) adaptée aux paramètres des quantrons. Les résultats de l'ensemble de ces analyses sont testés dans le réseau MLQ initial (non approximé).

4.1 Algorithme d'apprentissage du MLQ

4.1.1 Apprentissage du MLQ par la rétropropagation de l'erreur

Les algorithmes d'optimisation de fonctions efficaces utilisent, en général, la différentielle de la fonction considérée, c'est-à-dire, son gradient car il est à valeur réelle. Comme les fonctions de transfert ou résultantes utilisées pour les neurones sont différentiables, l'erreur commise par un réseau donné est une fonction dérivable. L'algorithme de rétropropagation permet justement de

calculer le gradient de cette erreur de façon efficace : le nombre d'opérations (multiplications et additions) à faire est en effet proportionnel au nombre de connexions du réseau comme dans le cas du calcul de la sortie de celui-ci. Cet algorithme rend ainsi possible l'apprentissage d'un MLP [17] et sera adapté à l'apprentissage d'un MLQ.

Les performances de cette notion sont parfois très limitées car des problèmes de lenteur sur la vitesse de convergence peuvent survenir. Ces derniers tirent leur origine dans le calcul de la dérivée de la fonction d'activation ou des difficultés à trouver une solution optimale lorsqu'on est bloqué dans un minimum local lors du processus itératif. Plusieurs formes d'amélioration sont proposées au modèle d'apprentissage dont certaines sont intéressantes pour la conception d'algorithmes de réseaux de quantrons telle que la méthode du RPROP.

La principale difficulté de l'apprentissage d'un MLQ, contrairement au MLP, est due au fait que la sortie du quantron est égale au temps correspondant au premier moment où la résultante du quantron, $R(t)$, atteint un seuil critique fixé préalablement. L'implémentation simplifiée d'un algorithme de rétropropagation de l'erreur nécessite une transformation de l'expression de sortie du quantron en fonction des entrées; il existe différentes méthodes d'approximation de cette transformation.

Considérons, par hypothèse, que tous les seuils critiques des quantrons du MLQ sont positifs, l'observation de la réponse à la sortie du réseau, $R(t)$, permet de constater que la courbe de cette fonction est approximée par une parabole. La figure ci-dessous montre un exemple d'allure de la courbe de la résultante du MLQ.

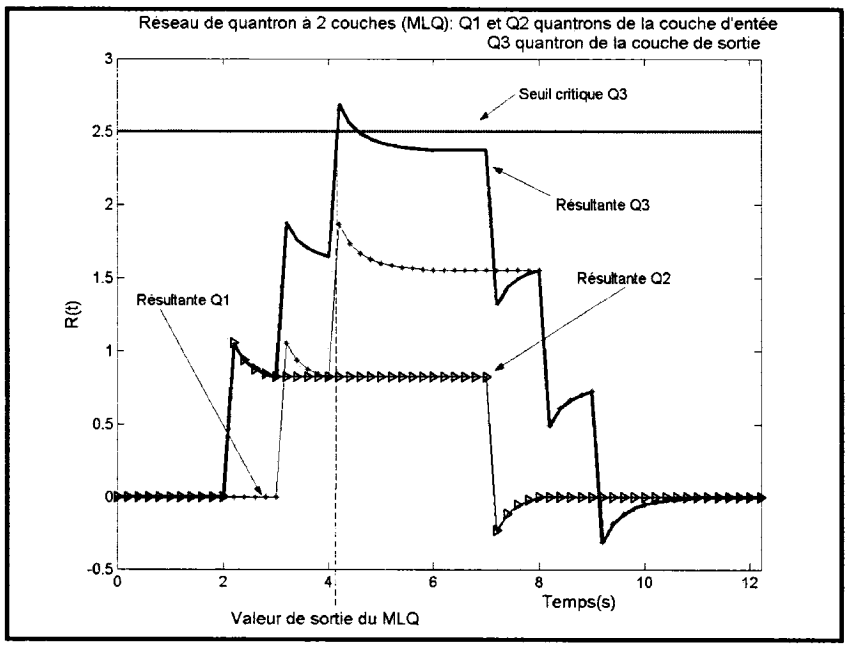


Figure 4-1 : Exemple du comportement de la résultante d'une sortie du MLQ

Pour cet exemple, la valeur de sortie du MLQ est Q3 = 4.2 s. Le seuil critique de la fonction du quantron de sortie est $\Gamma = 2.5$.

La fonction de sortie du quantron peut être approximée de façon analytique par des méthodes nécessitant des calculs plus complexes et ne garantissant pas réellement une meilleure approximation de sa valeur de sortie [33]. L'une des méthodes usuelles est l'approximation par la transformée de Fourier. La complexité de l'équation de cette méthode augmente en fonction de la minimisation de l'erreur de troncature. L'hypothèse utilisée dans le cadre de cette implantation est une approximation de la fonction de sortie du quantron par une parabole.

4.1.2 **Approximation du quantron par une parabole**

L'analyse du fonctionnement du quantron pour un ensemble de données de tests a permis de constater, en considérant la valeur du seuil critique de transmission de la sortie du quantron positif lorsque ce dernier conduit, que sa fonction résultante aura l'allure d'une parabole dans le voisinage du temps définissant la valeur de sa sortie. Cette première observation permet de considérer l'hypothèse selon laquelle la fonction résultante du quantron aurait un comportement quadratique. La figure suivante représente l'approximation de la fonction résultante du quantron sous forme d'une parabole.

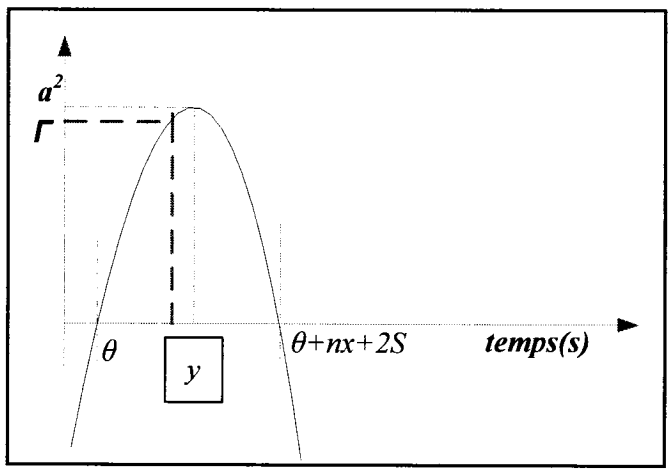


Figure 4-2 : Hypothèse d'approximation du quantron sous forme de parabole.

La parabole pour cette approximation aura toujours une concavité vers le bas. La valeur absolue du maximum de cette parabole correspond au poids synaptique *W*. *y* représente le premier instant où le seuil critique de la fonction résultante est atteint comme le montre la figure ci-dessus.

La seconde hypothèse est la définition du paramètre de déphasage θ qui représente le premier moment où la valeur de la résultante du quantron *R(t)* devient supérieure à zéro. Dans le cas d'un quantron à une seule entrée, il n'y a qu'un seul paramètre de déphasage θ. Cependant, lorsqu'il s'agit d'un quantron à plusieurs entrées, la valeur considérée de θ pour l'approximation de la parabole est le minimum de l'ensemble des valeurs de θ.

On remarque également que cette parabole s'annule à un second point. Il correspond à la valeur du noyau du quantron (*2S*), plus le nombre d'impulsions de fréquence de données d'entrée, le tout décalé de θ. L'équation de cette parabole est définie de la manière suivante :

$$R(t) = -a^2(t-\theta)(t-\theta-nx-2S) \tag{4.1}$$

où a est une constante positive, θ le temps écoulé avant que le premier potentiel ne survienne au neurone post-synaptique, S l'unité de temps écoulé ou la durée de vie de l'association neurotransmetteur - récepteur, n le nombre d'impulsions du signal d'entrée et x le signal d'entrée.

Comme première hypothèse, le maximum de cette parabole correspond au poids synaptique :

$$R(t)_{\max} = W \tag{4.2}$$

On a :

$$\frac{dR(t)}{dt} = -a^2(t-\theta-nx-2S+t-\theta) \tag{4.3}$$

Ce qui permet l'optimisation :

$$-a^2(t-\theta-nx-2S+t-\theta)=0 \tag{4.4}$$

De cette expression, on tire la valeur du temps :

$$t=\frac{2\theta+nx+2S}{2} \tag{4.5}$$

En remplaçant (4.5) dans (4.2), on obtient :

$$-a^2(\frac{2\theta+nx+2S}{2}-\theta)(\frac{2\theta+nx+2S}{2}-\theta-nx-2S)=W \tag{4.6}$$

D'où :

$$a^2=\frac{4W}{(nx+2S)^2} \tag{4.7}$$

En remplaçant (4.7) dans (4.1) et lorsque le quantron conduit où $R(t)=\Gamma$, on a :

$$\frac{4W}{(nx+2S)^2}(t-\theta)(t-\theta-nx-2S)=\Gamma \tag{4.8}$$

La résolution de cette dernière équation permet de déterminer la sortie t du quantron en fonction des entrées et de ses paramètres.

$$t_{1,2} = \frac{2WS + 2\phi W + nxW \pm (nx + 2S)\sqrt{w(w-\Gamma)}}{2W} \tag{4.9}$$

La sortie à considérer se révèle être le minimum des valeurs positives entre t_1 et t_2. Ce choix s'explique par la définition de la sortie d'un quantron. En effet, la valeur de sortie du quantron correspond au premier moment où la résultante dépasse le seuil critique prédéfini. La formule de celle retenue est :

$$t = \frac{2WS + 2\theta W + nxW - (nx + 2S)\sqrt{w(w-\Gamma)}}{2W} \tag{4.10}$$

La figure 4-3 représente le quantron approximé par une parabole.

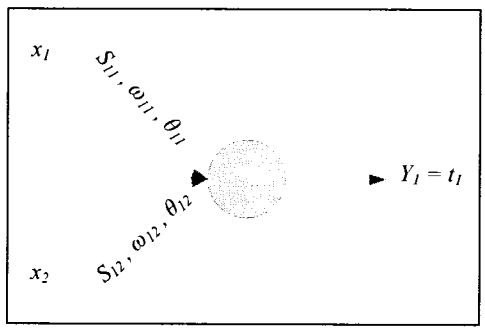


Figure 4-3 : Topologie du quantron approximé par une parabole.

Ce quantron contient deux entrées x_1 et x_2 et une sortie Y_1 qui est égale à t_1. Cette dernière est définie par l'équation (4.10).

La figure 4-4 représente la topologie du MLQ implémenté. Il est constitué des quantrons approximés par une parabole.

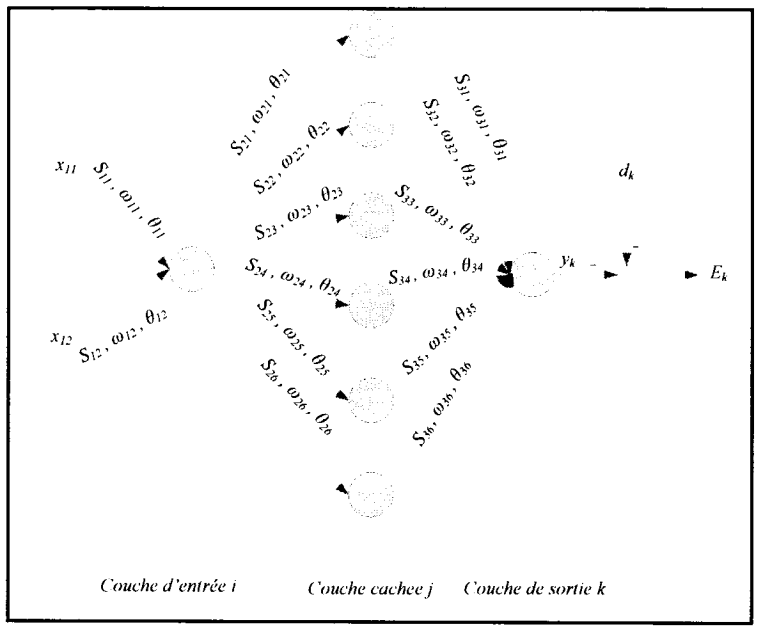


Figure 4-4 : Topologie du MLQ où les quantrons sont des approximations paraboliques.

Le réseau considéré dans le cadre de cette implémentation est constitué de 3 couches dont une couche cachée contenant 6 neurones. Le nombre de neurones de la couche cachée est déterminé à la suite d'essais qui ont eu comme objectif de favoriser la transmission du signal d'entrée vers le quantron de la couche de sortie.

Cette transformation permet à présent de calculer la rétropropagation par la méthode de la descente du gradient en fonction des paramètres W, S et θ. Le tableau 4-1 représente le calcul des dérivées de la sortie t en fonction de ses paramètres.

Tableau 4-1 : Dérivées de la sortie t en fonction des différents paramètres.

Paramètre	$\frac{dt_1}{paramètre}$
x	$\frac{n(W-\sqrt{W(W-\Gamma)}}{2W}$
W	$\frac{-\Gamma(nx+2S)}{4W\sqrt{W(W-\Gamma)}}$
θ	1
S	$\frac{W-\sqrt{W(W-\Gamma)}}{W}$

Ce calcul servira à implémenter un algorithme de rétropropagation de l'erreur par la méthode de descente du gradient. Il faut cependant préciser que l'expression obtenue pour la valeur de t n'est valide que lorsque le domaine de définition de W est restreint à une valeur positive et supérieure au seuil critique Γ, ce qui permet d'éliminer les valeurs complexes et les cas de non conduction du quantron. Le dénominateur de l'amplitude maximum de la parabole a^2 doit être différent de 0, ce qui est toujours vrai car S et ix sont toujours positifs.

L'expression de la sortie du quantron est continue par intervalle donc dérivable sur ces intervalles pour certains de ses paramètres. La rétropropagation de l'erreur par la méthode de la descente du gradient peut donc être introduite dans l'algorithme d'apprentissage en tenant compte des contraintes considérées lors de la dérivation.

4.1.3 **Apprentissage par descente du gradient appliqué au MLQ**

Une initialisation des paramètres du réseau de façon aléatoire précède la présentation des exemples au MLQ représenté à la figure 4-4. Le calcul des sorties correspondantes est effectué. L'erreur est évaluée et une correction des paramètres est appliquée.

L'algorithme de rétropropagation de l'erreur par descente du gradient repose sur la minimisation de l'erreur quadratique entre les valeurs de sortie calculées et les valeurs de sortie désirées.

Soit l'expression de l'erreur quadratique moyenne :

$$E = \frac{1}{2}\sum (d_k - t_k)^2 \tag{4.11}$$

où t_k représente la valeur de sortie calculée et d_k la valeur de sortie désirée. L'indice k représente le $k_{ième}$ nœud (quantron) de la couche de sortie.

Pour minimiser E, on calcule son gradient par rapport aux paramètres W, S et θ. On modifie ensuite ces paramètres dans le sens inverse du gradient de la sortie du réseau vers les entrées. Le gradient de l'erreur est obtenu en effectuant une dérivation en chaîne. Ainsi, l'expression des gradients d'erreur du réseau MLQ dans laquelle les quantrons sont approximés par une parabole (figure 4-2) est calculable pour chacune des couches d'après les équations suivantes :

- **Pour la couche de sortie :**

$$\frac{\delta E}{\delta parametre_k} = -(d_k - t_k)\frac{\delta t_k}{\delta parametre_k} = \delta_k \frac{\delta t_k}{\delta parametre_k} \tag{4.12}$$

L'expression des dérivées de la fonction de sortie en fonction des paramètres du quantron a été calculée dans le tableau 4-1. En remplaçant ces expressions pour chacun des paramètres, on obtient :

$$\frac{\delta E}{\delta t_k} = -(d_k - t_k) = \delta_k \tag{4.13}$$

$$\frac{\delta E}{\delta W_k} = -(d_k - t_k)\frac{\delta t_k}{\delta W_k} = \delta_k \frac{-\Gamma_k (n_k x_k + 2S_k)}{4W_k \sqrt{W_k (W_k - \Gamma_k)}} \tag{4.14}$$

$$\frac{\delta E}{\delta \phi_k} = -(d_k - t_k)\frac{\delta t_1}{\delta \phi_k} = \delta_k \tag{4.15}$$

$$\frac{\delta E}{\delta S_k} = -(d_k - t_k)\frac{\delta t_1}{\delta S_k} = \delta_k \frac{W_k - \sqrt{W_k (W_k - \Gamma_k)}}{W_k} \tag{4.16}$$

- **Pour la couche cachée**

$$\frac{\delta E}{\delta parametre_j} = -\sum_k^1 \left[-(d_k - t_k) \frac{\delta t_k}{\delta x_k} \right] \frac{\delta t_j}{\delta parametre_j} \tag{4.17}$$

En remplaçant dans cette expression chaque paramètre, on obtient :

$$\frac{\delta E}{\delta W_j} = -\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \frac{-\Gamma_j (n_j x_j + 2S_j)}{4W_j \sqrt{W_j (W_j - \Gamma_j)}} \tag{4.18}$$

$$\frac{\delta E}{\delta \phi_j} = -\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \tag{4.19}$$

$$\frac{\delta E}{\delta S_j} = -\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \left(\frac{W_j - \sqrt{W_j (W_j - \Gamma_j)}}{W_j} \right) \tag{4.20}$$

- **Pour la couche d'entrée**

$$\frac{\delta E}{\delta parametre_i} = -\sum_j^1 \left(\sum_k^1 \left[-(d_k - t_k) \frac{\delta t_k}{\delta x_k} \right] \frac{\delta t_j}{\delta x_j} \right) \frac{\delta t_i}{\delta parametre_i} \tag{4.21}$$

En remplaçant dans cette expression chaque paramètre, on obtient :

$$\frac{\delta E}{\delta W_i} = -\sum_j^1 (\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \frac{n_j (W_j - \sqrt{W_j (W_j - \Gamma_j)})}{2W_j}) \frac{-\Gamma_i (n_i x_i + 2S_i)}{4W_i \sqrt{W_i (W_i - \Gamma_i)}} \quad (4.22)$$

$$\frac{\delta E}{\delta \phi_i} = -\sum_j^1 (\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \frac{n_j (W_j - \sqrt{W_j (W_j - \Gamma_j)})}{2W_j}) \quad (4.23)$$

$$\frac{\delta E}{\delta S_i} = -\sum_j^1 (\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \frac{n_j (W_j - \sqrt{W_j (W_j - \Gamma_j)})}{2W_j}) \frac{W_i \sqrt{W_i (W_i - \Gamma_i)}}{W_i} \quad (4.24)$$

La mise à jour des connexions synaptiques est faite selon un processus itératif. La formule de mise à jour des paramètres du réseau est donnée par :

$$Parametre_n(compt) = Parametre_n(compt-1) - \eta \frac{\delta E}{\delta Parametre_n(compt-1)} \quad (4.25)$$

où $Parametre_n$ symbolise l'un des paramètres du quantron (W, S et θ), η une constante choisie aléatoirement appelée taux d'apprentissage, et *compt* représente le nombre d'itérations.

Le critère d'apprentissage est atteint pour un stimulus donné lorsque la fonction d'erreur est inférieure à la valeur de tolérance γ préétablie. Celui-ci peut également être établi par la méthode de cross-validation sur un ensemble de test.

4.2 Application de l'algorithme d'apprentissage du MLQ

L'algorithme d'apprentissage adapté au MLQ, lors d'une simulation, modifie les paramètres du réseau tels les poids W, les déphasages θ et les seuils S afin de fournir à ses neurones de sortie les valeurs désirées en sortie en fonction des échantillons d'entrée. Il est donc question d'apprentissage supervisé du MLQ.

Pour réaliser un apprentissage supervisé du MLQ, il est nécessaire de disposer au préalable d'échantillons. Ces derniers constituent un ensemble de données de n couples (X_i, Y_i) avec i =1, 2,... n. où X_i est le vecteur d'entrée et Y_i, le vecteur de sortie désiré. Ils permettent de caractériser les vecteurs des paramètres W, S, θ du MLQ capables de mettre ces informations en correspondance. La fonction de transfert $F(X_i)$ du réseau MLQ est définie par :

$$F(X_i) = Y_i \qquad \text{pour } i = 1,2,...,n. \tag{4.26}$$

4.2.1 Modèle d'application de reconnaissance de caractères

L'application retenue dans le cadre de l'implantation de ce premier algorithme d'apprentissage du MLQ est la reconnaissance de caractères. Une matrice 5x7 est définie en tant que domaine de représentation des caractères ou données d'entrée du réseau.

Chaque couple de coordonnées temporelles de cette matrice représente un pixel qui prend la valeur 1 quand il émet un signal ou 0 dans le cas d'une non conduction. La construction d'un

regroupement d'ensembles de caractères qui serviront d'échantillons d'apprentissage devient donc possible.

La figure 4-5 fait ressortir un exemple de valeurs de données d'entrée qui est la représentation de la lettre A.

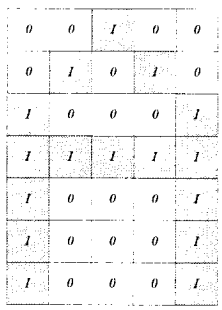


Figure 4-5 : Matrice de données d'entrée; exemple de représentation de la lettre A

Cette matrice contient 35 éléments. La valeur 1 dans la case d'un élément symbolise une émission de signal. Cette dernière indique la présence d'une impulsion fréquentielle tandis que la valeur 0 symbolise une non conduction donc, l'absence d'émission d'impulsions fréquentielles.

Il faut définir pour chaque élément une unité de temps correspondant à la valeur désirée en sortie de celui-ci. Comme l'explique la figure 4-8, chaque caractère d'entrée sera associé à une valeur temporelle au moins supérieure à la valeur maximum du déphasage du neurone de la couche de sortie. Cette valeur désirée est aléatoirement choisie pour chaque caractère. Une méthode consisterait à faire correspondre à chaque caractère une valeur de temps proportionnelle à l'ordre de ce caractère dans l'alphabet. Par exemple, le caractère A, première lettre de l'alphabet, sera identifié par une valeur de sortie désirée, le temps d_A. Le caractère B est identifié en sortie par le

temps d_B avec $d_A < d_B$. Ceci revient à dire que la valeur désirée de chaque caractère sera donnée par la formule suivante :

$$d_i = \alpha_i(rang_i) \tag{4.27}$$

où d_i est la valeur désirée du caractère i, α_i une constante définie en fonction du déphasage du neurone de la couche de sortie et $rang_i$ qui définit la position du caractère i dans l'alphabet.

4.2.2 **Caractéristiques du réseau MLQ**

La conception d'un nouveau réseau de neurones de quantrons multicouches doit absolument garantir l'existence de l'un des principaux éléments le distinguant d'une structure quelconque dotée d'une forme d'intelligence. De ce fait, ce réseau doit disposer d'une capacité de reconfiguration de sa structure en temps réel. Ce dynamisme se traduit par l'introduction du concept d'apprentissage par rétropropagation permettant un ajustement itératif des paramètres des quantrons du réseau.

Cet apprentissage supervisé est caractérisé par la présence d'un professeur ayant des connaissances de l'environnement dans lequel évolue le réseau de neurones. Ces connaissances portent sur des ensembles de couples de vecteurs d'entrée et de sortie désirés du réseau. Chacun de ces couples de vecteurs correspond à un stimulus qui devrait permettre au réseau de produire une sortie égale ou proche de la valeur de sortie désirée. A cet effet, l'on parle aussi d'apprentissage par des exemples. La figure 4-6 présente le mécanisme de fonctionnement de cet d'apprentissage.

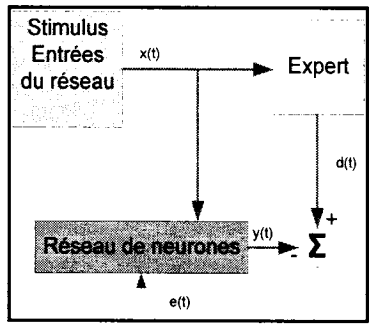


Figure 4-6 : structure de l'apprentissage supervisé

Comme l'indique la figure 4-6, le stimulus *x(t)* produit par l'environnement inconnu au réseau est acheminé au réseau et à l'expert. Ce dernier utilise ses connaissances intrinsèques pour produire une sortie désirée *d(t)*. Celle-ci est comparée à la sortie du réseau *y(t)* afin de déterminer l'erreur produite, *e(t)*, qui sera réinjectée dans le réseau.

Pour qu'une sortie soit validée, la valeur de l'erreur doit se situer dans un intervalle dont la borne supérieure sera une constante, le seuil d'apprentissage, définie positivement proche de 0 et la borne inférieure, son opposé. Dans le cas d'une erreur quadratique, la borne inférieure de l'intervalle sera 0. On peut donc dire que la connaissance des stimuli par l'expert est graduellement transférée vers le réseau jusqu'à l'atteinte d'un critère d'arrêt prédéfini qu'on a appelé le seuil d'apprentissage. Cette erreur permettra de modifier le comportement du réseau selon une procédure itérative afin qu'il simule une réponse égale à celle désirée. Par la suite, on pourra éliminer l'expert et laisser le réseau fonctionner de façon autonome.

Le quantron se veut être un élément présentant un comportement autonome. De ce fait, les quantrons d'un réseau doivent être indépendants les uns des autres. Comme il a été présenté au chapitre 2, le quantron est un neurone doté de certains paramètres d'ajustement indépendants et dont la fonction de transfert est une sommation dynamique de la variation des potentiels

synaptiques de noyau S pondérés par un facteur de poids W et dont la latence avant émission est définie par le facteur θ. Le seul critère de conduction est que la résultante calculée doit produire un signal de sortie si et seulement si elle dépasse la valeur de seuil Γ prédéfinie. Cet état stabilisera la sortie du quantron à une valeur du temps correspondant au premier instant où la valeur de la résultante dépasse le seuil de conduction Γ du quantron.

Il importe que le nouveau réseau soit capable de s'auto-organiser. Ainsi, il ne devrait pas être contrôlé par un quantron central mais le pouvoir de décision doit être plutôt distribué à l'ensemble des quantrons des réseaux. Ceci sera rendu possible grâce au choix topologique du réseau de quantrons multicouches. Chaque neurone d'une couche est connecté à tous les neurones de la couche adjacente.

La méthode d'apprentissage par rétropropagation doit garantir une transmission de l'information de façon non-linéaire. Cette condition est prise en considération par l'utilisation du principe de la descente de gradient dans l'approche conceptuelle du nouvel algorithme du MLQ.

En tenant compte de ces propriétés, différents types de configuration architecturale du quantron multicouche peuvent être élaborés. Mais nous allons nous limiter à un cas de configuration simple qui facilitera l'analyse de la conception de cette nouvelle structure.

4.2.3 **Topologie du réseau MLQ**

Dans le réseau de quantrons multicouches (MLQ), chaque quantron d'une couche est totalement connecté aux quantrons de la couche suivante. La topologie utilisée, dans le cadre de cette application de reconnaissance de caractères, est formée de 3 couches de neurones sans

connexions à l'intérieur d'une même couche. Dans le cadre de cette première implémentation, le réseau est limité à la reconnaissance d'un seul caractère à la fois (un quantron de sortie) contrairement à une approche de reconnaissance des 26 caractères simultanément.

La figure 4-7 illustre ce modèle du réseau de quantrons multicouches.

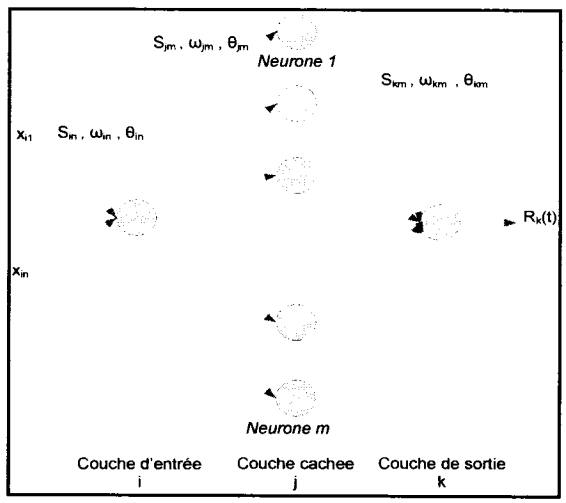


Figure 4-7 : Topologie du MLQ utilisée pour une application de reconnaissance de caractères bidimensionnels.

La couche d'entrée, *i*, reçoit les informations à traiter provenant de la matrice des données. La couche cachée, *j*, effectue le traitement spécifique des données présentées en entrée. La couche de sortie *k* permet de quantifier les résultats.

Ce type de réseau utilise des neurones complémentaires pour assurer la transmission de l'information. Comme le présente la couche cachée de la figure 4-7, Ils sont souvent parallèles

car ils représentent une configuration d'un ensemble d'influx nerveux disjoints. On parle, dans ce cas, d'un réseau de neurones non compétitifs [17].

Les réseaux non compétitifs supervisés sont reliés à un même neurone de sortie comme le montre la couche de sortie de la figure 4.7. Celui-ci possède une importante fonction de relais et centralise temporairement l'information transitant dans le réseau. Si l'un des chemins en parallèle disparaissait, le neurone central perdrait une partie de l'information de manière diffuse et non de manière radicale [17].

Puisque l'apprentissage est supervisé, chaque échantillon présenté au réseau est accompagné d'une valeur désirée comparable à la sortie du réseau. La détermination des valeurs désirées de chaque pixel de la matrice s'effectue au préalable par propagation lorsqu'il y a émission d'impulsions. Dans ce cas, une sortie réelle est chiffrée pour chaque donnée d'entrée. Ce calcul est fait de proche en proche de l'entrée jusqu'à la sortie du réseau MLQ. On procède ainsi par simulations d'essai-erreur en présentant des entrées au MLQ et en variant aléatoirement certains de ses paramètres. La complexité de cette approche s'intensifie en fonction de la dimension du réseau et de la variation de ses paramètres.

La méthode de détermination de valeurs de sortie désirées du réseau en question consiste à définir des valeurs désirées pour chaque vecteur d'entrée. Ces valeurs choisies tiennent compte du temps de calcul de chaque automate (quantron), du seuil critique du quantron de la couche de sortie, de la valeur du vecteur présentée en entrée et surtout, du temps de propagation avant du signal durant son passage dans le réseau.

Après avoir établi l'architecture du MLQ, les valeurs de sortie désirées sont déterminées pour chaque échantillon comme l'indique l'équation (4.27).

4.3 Simulation de l'algorithme de la descente du gradient du MLQ

L'expérimentation de la descente du gradient dans cette application prouve que cette technique est suffisante dans l'utilisation d'un réseau de MLP mais non concluante pour le réseau MLQ. Cette non-convergence a permis de modifier l'algorithme de descente de gradient du MLQ. Elle se justifie par le fait que le domaine de définition du paramètre des poids synaptiques, W, est limité aux valeurs positives et supérieures au seuil critique du quantron.

Toutefois, la limitation du domaine de définition est nécessaire car elle permet d'éliminer les résultats complexes des valeurs des paramètres et de la sortie approximative du réseau. Lorsque le réseau s'introduit dans le domaine complexe, cela se traduit par le fait qu'au moins une connexion du réseau ne conduit pas.

4.3.1 **Cas de non conduction d'un quantron du réseau MLQ**

Selon l'hypothèse d'approximation du quantron sous forme d'une parabole comme l'indique l'équation 4.10, la non conduction du quantron est équivalent à poser :

$$w(w-\Gamma)<0 \tag{4.28}$$

Avec w et Γ représentant respectivement le poids synaptique et le seuil du quantron. Dans ce cas, la fonction de sortie du quantron approximé t, est une expression complexe et non souhaitable. Une solution à ce problème est de fixer l'amplitude de la parabole à la valeur 0 lorsque le

quantron se trouve dans une phase de non-conduction. Cette amplitude, définie par l'équation 4.7, donnera:

$$\frac{4w}{(nx+2S)^2}=0 \tag{4.29}$$

Le dénominateur de cette expression est supposé non nul selon les hypothèses de départ. Ainsi, l'équation 4.29 revient à fixer :

$$w=0 \tag{4.30}$$

En plus, un quantron qui ne conduit pas équivaut à l'inexistence de variation de potentiel d'action dans ces entrées. Ceci est traduit par l'égalité suivante :

$$nx=0 \tag{4.31}$$

Cet ensemble de considération permet d'éviter l'influence des signaux non désirés dans le calcul de la fonction résultante du quantron. La mise à jour du paramètre w à 0, dans la phase de non-conduction, permet d'annuler l'amplitude du signal de sortie du quantron durant cette période. L'obtention d'une valeur complexe à la sortie du quantron devient donc impossible. La considération de cette modification dans l'algorithme d'apprentissage du MLQ induira une amélioration de la convergence.

4.4 Adaptation de l'algorithme d'apprentissage par descente du gradient au MLQ

L'adaptation de l'algorithme de rétropropagation du gradient s'effectue en fonction des paramètres W, S et θ du MLQ. Le domaine de définition des paramètres W reste inchangé, c'est-à-dire que pour chaque quantron du réseau, W restera positif et supérieur au seuil critique de celui-ci. La mise à jour des paramètres W du MLQ est alors effective, comme dans le cas de l'algorithme de rétropropagation par la méthode de la descente du gradient du MLQ (équations (4.14), (4.18) et (4.22)).

Quant aux paramètres θ, on définit les minima des déphasages de quantrons en fonction de leurs appartenances aux couches du réseau. Le schéma suivant représente les paraboles d'approximation du réseau.

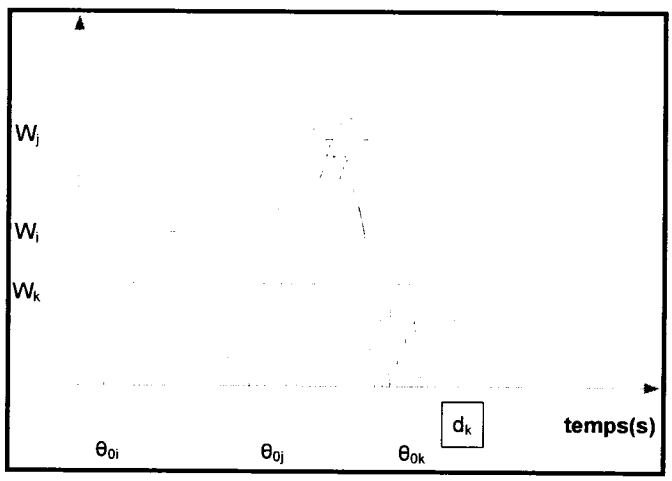


Figure 4-8 : Description du comportement des résultantes des quantrons selon les couches du MLQ.

θ_{0i}, θ_{0j}, et θ_{0k} sont respectivement les déphasages maximum des quantrons des couches d'entrée, cachée et de sortie. Les valeurs de ces déphasages sont fixées au préalable en fonction des valeurs désirées d_k de la sortie du réseau. On définit un taux d'ajustement $\Delta\theta$ représentant la variation de déphasage. La mise à jour des paramètres θ des quantrons du réseau est établie selon l'équation suivante :

$$\theta_{couche}(compt) = \theta_{couche}(compt-1) + \Delta\theta_{couche} \tag{4.32}$$

Par exemple, on a, pour la couche de sortie $\Delta\theta_k$ = 0.01, la couche cachée $\Delta\theta_j$ = 0.005 et la couche d'entrée $\Delta\theta_i$ = 0.0002, *compt* représente le nombre d'itérations.

Enfin, pour les paramètres de durée de vie de l'association neurotransmetteur – récepteur, S, des quantrons du réseau sont initialisés aléatoirement et mis à jour selon les équations de rétropropagation par la méthode de la descente du gradient (équations (4.16), (4.20) et (4.24)).

4.5 Simulation de l'algorithme de la descente du gradient adapté au MLQ

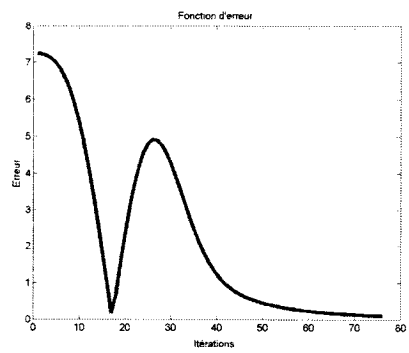
L'implémentation de cet algorithme réalisée à l'aide du logiciel MATLAB [31] permet de constater que l'apprentissage par rétropropagation de l'erreur en tenant compte de la descente du gradient converge. Les figures suivantes présentent des exemples de résultats de cet algorithme. La première modification apportée à l'algorithme est le contrôle de pas des variations du déphasage de chaque neurone. Car, si l'on veut être cohérent avec l'hypothèse d'approximation parabolique, la valeur de déphasage doit correspondre au premier temps où la courbe de sortie du neurone considéré commence à croître.

La valeur de sortie et la valeur désirée doivent être toujours supérieures au déphasage pour qu'il y ait conduction, ce qui se justifie par le fait que dans l'hypothèse, le quantron conduit lorsque la sortie atteint le maximum de la parabole. Il est surtout fonction du paramètre W, poids synaptique du quantron approximé. La valeur de seuil de conduction du quantron est considérée positive et inférieure au maximum de la parabole approximant le quantron.

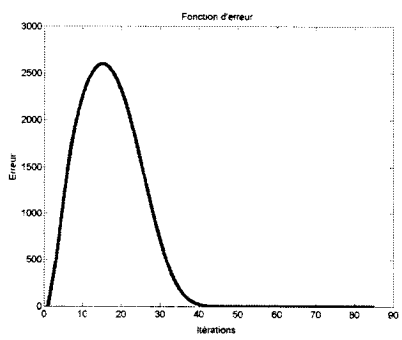
En tenant compte de cet ensemble d'éléments dans la modification de l'implémentation de notre algorithme, les résultats de simulations de plusieurs échantillons différents montrent diverses formes de courbes de convergence.

L'application de reconnaissance de caractères définie permet de tester l'implémentation de la rétropropagation de l'erreur par la descente du gradient.

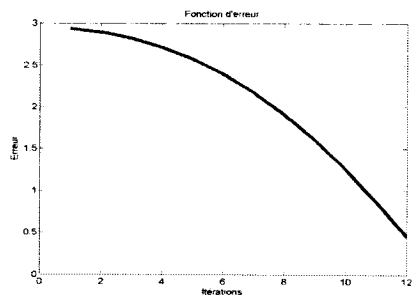
La figure suivante représente l'apprentissage de la lettre A. Les paramètres α_i et $rang_i$ de l'équation de la valeur désirée en sortie (4.27) ont respectivement pour valeur 10 et 1. Un ensemble de 35 échantillons distincts de la lettre A bruités sur un pixel est généré. La lettre A et les 33 échantillons de A bruités sur un pixel sont présents lors du processus d'apprentissage. Les 2 autres échantillons non appris serviront au test de validation.



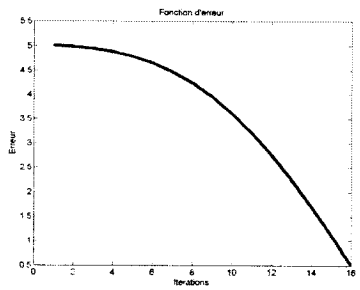
Simulation 3 : L'erreur croit par moment avant d'atteindre la convergence souhaitée.



Simulation1 : Courbe de l'erreur converge vers le seuil 0.01 après moins de 80 itérations.



Simulation 5 : moins de 13 itérations avant la convergence de l'erreur.



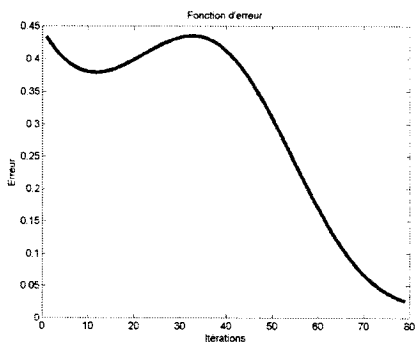
Simulation 12 : la courbe d'erreur atteint le seuil de convergence après 16 itérations.

Figure 4-9 : Comportement des courbes d'erreur quadratique de l'algorithme du MLQ de rétropropagation de l'erreur par descente de gradient modifié avec γ=0.1 du caractère A.

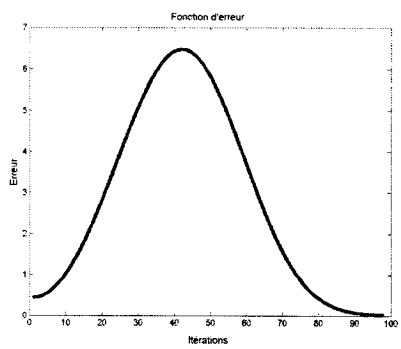
D'une façon générale, cet algorithme converge rapidement contrairement à celui de la descente du gradient standard. Dans certains cas, telle la simulation 3, la courbe de convergence épouse

une allure en dents de scie au lieu de la forme habituelle décroissante (simulation 12). Il convient de noter que dans la figure 4-9; le critère d'arrêt γ est égal à 10%. Les valeurs d'erreur augmentent considérablement dans les zones pré-convergentes, ce qui se traduit par le fait que la limitation du domaine de définition du paramètre de poids synaptique, W, entraîne une définition de continuité par morceaux de la fonction de sortie. Lorsqu'une itération des calculs d'ajustement dépasse les bornes des domaines définis, il se produit une nouvelle initialisation des paramètres, ce qui peut parfois induire une augmentation de l'erreur au lieu de la diminution souhaitée. Dans ce cas, la courbe prend l'allure des simulations 1 ou 3 comme représenté dans la figure 4-9.

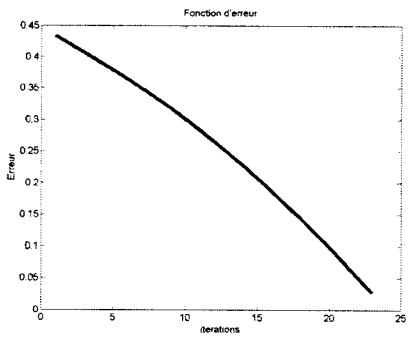
La diminution de la valeur de l'erreur acceptable γ à 2.5% montre une variation du comportement de la courbe d'erreur. La figure 4-10 illustre quelques exemples de ces courbes d'erreurs. Dans ce cas, l'erreur s'apparente à une parabole (simulation 1 et 3) ou demi-parabole (simulation 5 et 12) avec une concavité orientée vers le bas. Le nombre d'itérations moyen pour atteindre la convergence augmente inversement proportionnel à γ.



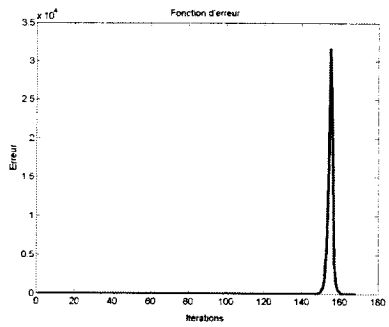
Simulation 1 : Courbe de l'erreur converge vers le seuil 0.025 après moins de 80 itérations.



Simulation 3 : Courbe de l'erreur converge vers le seuil 0.025 après moins de 100 itérations.



Simulation 5 : Courbe de l'erreur converge vers le seuil 0.025 après moins de 25 itérations.



Simulation 12 : Courbe de l'erreur converge vers le seuil 0.025 après moins de 180 itérations.

Figure 4-10 : Comportement des courbes d'erreur quadratique de l'algorithme du MLQ de rétropropagation de l'erreur par descente de gradient modifié avec γ=0.025 du caractère A.

Une amélioration de cet algorithme est envisageable afin d'accélérer la convergence et aussi faciliter une minimisation des augmentations brusques de l'erreur avant sa convergence.

La mise à jour des paramètres du réseau $\Delta Parametre_n$ dépend non seulement du taux d'apprentissage du réseau mais aussi du gradient de l'erreur du paramètre considéré $\frac{\delta E}{\delta Parametre_n}$. Ainsi, l'effet du taux d'apprentissage sur un paramètre donné peut être appliqué par le changement du signe de son gradient d'erreur, ce qui conduit à une amélioration de l'algorithme basée sur la méthode RPROP [21]. Cette dernière a pour principal avantage de pouvoir effectuer une mise à jour directe du paramètre en tenant compte uniquement du signe de son gradient et non de sa valeur.

4.6 Validation de l'apprentissage du caractère A

La méthode de validation utilisée a été testée pour la lettre A. Le principe consiste à définir un échantillon de lettres A. Cet ensemble de lettres est constitué d'une lettre A non bruitée et de 35 lettres A bruitées dont chacune sur un pixel différent. Deux sous-ensembles disjoints sont formés à partir de ces 36 éléments. L'un de ces sous-ensembles sert à l'entrainement du réseau et le second aux tests de reconnaissance. L'ensemble de reconnaissance est composé des lettres non entrainées et l'ensemble d'apprentissage est celui comportant les lettres entraînées.

L'ensemble destiné à l'entrainement du réseau contiendra le plus grand nombre de lettres. Des 36 lettres, la lettre A non bruitée et un ensemble de lettres A bruitées seront entrainées. Le reste des lettres A bruitées forme l'ensemble de validation. Ainsi, Plusieurs validations seront

effectuées en faisant varier de manière croissante le nombre d'éléments du sous-ensemble de reconnaissance. Dans un premier temps, le processus débutera avec 35 éléments du sous-ensemble d'apprentissage et 1 élément de l'ensemble de reconnaissance. À l'itération suivante, le nombre d'éléments à entraîner passera à 34 et 2 éléments constitueront l'ensemble de reconnaissance. Cette même logique sera adoptée pour les prochaines itérations et le nombre d'éléments du sous-ensemble de reconnaissance est limité à 5. Chaque itération de validation est répétée 3 fois.

Le tableau suivant représente le résultat de tests de validation de la lettre A .

Tableau 4-2 : Test de validation du réseau MLQ pour la lettre A

Taux d'apprentissage ($0<\eta<1$)	**Critère d'arrêt ($0<\gamma<1$)**	**Nombre d'éléments de l'ensemble d'entraine ment**	**Nombre d'éléments de l'ensemble de reconnais-sance**	**Taux moyen d'apprentis-sage**	**Taux moyen de reconnais-sance**
0.3	2.5%	35	1	100%	100%
		34	2	100%	99.33%
		33	3	99.96%	55.44%
		32	4	99.96%	58.33%
		31	5	99.86%	33.33%

Le facteur d'apprentissage choisi est de 0.3 et le critère d'arrêt retenu après les simulations d'apprentissage est de 2.5%. Lorsque le nombre d'éléments de l'ensemble de reconnaissance augmente, le taux de reconnaissance diminue.

Le taux d'apprentissage varie en fonction du facteur d'apprentissage η considéré. Le MLQ se comporte mieux lorsque la valeur de son facteur d'apprentissage est comprise entre 0 et 0.4. Le nombre d'itérations nécessaires à la convergence augmente proportionnellement en fonction de la valeur du facteur d'apprentissage. Ce facteur d'apprentissage a été choisi de façon optimale après un certain nombre de tests pour chacun des réseaux. Le tableau montre que le modèle d'apprentissage de l'algorithme de rétropropagation du MLQ approximé est acceptable. Néamoins, il est possible d'améliorer ses performances. A cet effet, il faudra augmenter les dimensions de la matrice des caractères.

4.7 Résumé des étapes de l'algorithme du MLQ

Le tableau suivant résume les étapes de l'algorithme d'apprentissage du MLQ

1- Fixer les valeurs de déphasage maximum de chaque couche du réseau par un processus d'essaie erreur tel que le déphasage de la couche de sortie soit strictement inferieur à la valeur désirée en sortie du réseau.

Dans un réseau à 3 couches on aura respectivement : θ_{0i}, θ_{0j}, et θ_{0k}

avec $\theta_{0i} \leq \theta_{0j} \leq \theta_{0k} < d_k$

2- Fixer le seuil de conduction de chaque quantron selon les résultats des tests de conduction préétablis sur chacun.

Γ_i, $i=1,\ldots$n. (n est le nombre de quantrons du réseau)

3- Déterminer la valeur désirée en sortie pour chaque donnée de l'ensemble des vecteurs d'apprentissage

d_k, $k=1,\ldots$m. (m est le nombre de valeurs désirées)

4- Initialiser les paramètres des quantrons du réseau avec des valeurs aléatoires.

Les valeurs des W, au voisinage des seuils de conduction

Les valeurs des S à 1

Les valeurs des θ inférieures au seuil maximum de déphasage selon l'appartenance à la couche de quantron en question

5- Définir les conditions de non conduction.

$$\text{Si } nx=0 \mapsto W=0.$$

6- Chaque échantillon possède ses valeurs cibles que le réseau doit prédire. Soit un échantillon $\vec{x}$ présenté à l'entrée du réseau. Il se propage vers l'avant dans le réseau selon la formule suivante :

$\vec{x}_k^{(n-1)} \mapsto \vec{x}_j^{(n)}$ (où k et j les couches du réseau adjacente respectivement de la gauche vers la droite). la valeur $\vec{x}_j^{(n)}$ est calculée par l'équation 4.10 de la résultante du quantron approximé.

$$t = \frac{2WS + 2\theta W + nxW - (nx + 2S)\sqrt{w(w-\Gamma)}}{2W} \quad \text{où } \vec{x}_j^{(n)} = t$$

7- Limiter la plage de variation des paramètres W du quantron pour les données d'entrées actifs.

$W_{\min} < W < W_{\max}$ avec $W_{\max} > \Gamma$

8- Limiter la plage de variation des déphasages de quantron en fonction des couches auxquels elles appartiennent. (figure 4-8).

9- Le calcul de l'erreur entre la sortie donnée du réseau et les vecteurs désirés à la sortie de l'échantillon courant est donné par l'équation 4.11:

$$E = \frac{1}{2}\sum (d_k - t_k)^2$$

10- L'erreur est propagée vers l'arrière grâce à la formule suivante :

$$Parametre_n(compt) = Parametre_n(compt-1) - \eta \frac{\delta E}{\delta Parametre_n(compt-1)}$$

11- La mise à jour des paramètres est effectuée en fonction des couches, de l'arrière vers l'avant :

- **Pour la couche de sortie**

$$\frac{\delta E}{\delta \alpha_k} = -(d_k - t_k) = \delta_k \tag{4.13}$$

$$\frac{\delta E}{\delta W_k} = -(d_k - t_k)\frac{\delta \alpha_k}{\delta W_k} = \delta_k \frac{-\Gamma_k (n_k x_k + 2S_k)}{4W_k \sqrt{W_k (W_k - \Gamma_k)}} \tag{4.14}$$

$$\frac{\delta E}{\delta \phi_k} = -(d_k - t_k)\frac{\delta \alpha_1}{\delta \phi_k} = \delta_k \tag{4.15}$$

$$\frac{\delta E}{\delta S_k} = -(d_k - t_k)\frac{\delta \alpha_1}{\delta S_k} = \delta_k \frac{W_k - \sqrt{W_k (W_k - \Gamma_k)}}{W_k} \tag{4.16}$$

- **Pour la couche cachée**

$$\frac{\delta E}{\delta parametre_j} = -\sum_k^1 \left[-(d_k - t_k)\frac{\delta \alpha_k}{\delta x_k} \right] \frac{\delta \alpha_j}{\delta parametre_j} \tag{4.17}$$

Pour chaque paramètre, on obtient :

$$\frac{\delta E}{\delta W_j} = -\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \frac{-\Gamma_j (n_j x_j + 2S_j)}{4W_j \sqrt{W_j (W_j - \Gamma_j)}} \quad (4.18)$$

$$\frac{\delta E}{\delta \phi_j} = -\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \quad (4.19)$$

$$\frac{\delta E}{\delta S_j} = -\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \left(\frac{W_j - \sqrt{W_j (W_j - \Gamma_j)}}{W_j} \right) \quad (4.20)$$

- **Pour la couche d'entrée**

$$\frac{\delta E}{\delta parametre_i} = -\sum_j^1 \left(\sum_k^1 \left[-(d_k - t_k) \frac{\delta t_k}{\delta x_k} \right] \frac{\delta t_j}{\delta x_j} \right) \frac{\delta t_i}{\delta parametre_i} \quad (4.21)$$

Pour chaque paramètre, on obtient :

$$\frac{\delta E}{\delta W_i} = -\sum_j^1 \left(\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \frac{n_j (W_j - \sqrt{W_j (W_j - \Gamma_j)})}{2W_j} \right) \frac{-\Gamma_i (n_i x_i + 2S_i)}{4W_i \sqrt{W_i (W_i - \Gamma_i)}} \quad (4.22)$$

$$\frac{\delta E}{\delta \phi_i} = -\sum_j^1 \left(\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \frac{n_j (W_j - \sqrt{W_j (W_j - \Gamma_j)})}{2W_j} \right) \quad (4.23)$$

$$\frac{\delta E}{\delta S_i} = -\sum_j^1 \left(\sum_k^1 \left[-(d_k - t_k) \frac{n_k (W_k - \sqrt{W_k (W_k - \Gamma_k)})}{2W_k} \right] \frac{n_j (W_j - \sqrt{W_j (W_j - \Gamma_j)})}{2W_j} \right) \frac{W_i \sqrt{W_i (W_i - \Gamma_i)}}{W_i} \quad (4.24)$$

12- Sitôt que les poids de chaque couche sont mis à jour, l'on procède à l'itération suivante jusqu'à ce que l'erreur soit inférieure ou égale à la valeur de tolérance prédéfinie.

13- L'ensemble des opérations est répété pour chaque échantillon d'apprentissage.

4.8 Conclusion

La modélisation des réseaux de quantrons multicouches est une nouvelle réalisation basée sur l'utilisation des quantrons dans une application de reconnaissance de caractères. Ce modèle de réseau a été défini dans une première tentative de simulation du comportement d'un nouvel algorithme de rétropropagation de l'erreur.

En ce qui concerne l'application de reconnaissance, la définition d'une matrice de représentation de caractères dotée d'un ensemble d'informations, à savoir le nombre de fréquences impulsionnelles et le temps d'écoulement de chaque pixel, s'est avéré nécessaire à la compatibilité des métriques des entrées d'un quantron. Cette définition a été établie en tenant compte de la réalité de la transmission d'un signal sous forme de pixel. La dimension de la matrice de données a été déterminée de façon aléatoire car celle-ci ne constitue pas un facteur important au bon fonctionnement de l'algorithme.

L'implantation d'un algorithme d'apprentissage pour résoudre la problématique de reconnaissance de caractères a nécessité plusieurs analyses. Tout d'abord, la fonction de transfert

de la sortie de quantron indirectement liée aux données d'entrée a été approximée. Plusieurs possibilités d'approximation ont été plausibles et le modèle le plus rapide et surtout simple pour l'implémentation de l'algorithme d'apprentissage a été une approximation parabolique du comportement de la résultante du quantron en fonction des rôles de ses paramètres. Elle a permis la définition d'un réseau de MLQ dont les quantrons sont approximés en parabole. Ce réseau implanté dans un algorithme de rétropropagation a facilité l'apprentissage des échantillons de caractères.

La première version de l'algorithme fonctionnel implémenté est une approche de rétropropagation par la méthode de la descente de gradient modifiée en tenant compte des contraintes, des domaines de dérivabilité, des paramètres des quantrons. Cette modification de l'algorithme standard de descente du gradient a permis d'observer l'allure des courbes de simulation de convergence de l'erreur dont certaines convergent en dents de scie. Cette non conformité a débouché sur des améliorations du modèle algorithmique permettant l'accélération de la convergence de la fonction d'erreur et minimisant les augmentations des valeurs de cette fonction dans les zones de pré-convergence.

L'amélioration de l'algorithme du MLQ est inspirée du RPROP. En tenant compte de la prédétermination des limites de variation de certains paramètres, à savoir θ et S des quantrons dans l'algorithme de rétropropagation, l'introduction du mécanisme du RPROP a favorisé la réduction du nombre d'itérations de convergence et par conséquent, celle du temps de simulation. Cette mise à jour des valeurs des paramètres des quantrons dans cette méthode a permis de valider une nouvelle version de l'algorithme de rétropropagation du MLQ. Les résultats des simulations de cette nouvelle version d'algorithmes ont montré une rapidité de convergence et une réduction des erreurs des données bruitées.

Une fois que les données des échantillons d'apprentissage ont été simulées, les valeurs des paramètres des quantrons obtenues sont intégrées dans le réseau de quantrons multicouches non approximés. Ce dernier est testé pour la lettre A par un nouvel ensemble de données afin d'analyser les résultats obtenus. La détection ou la reconnaissance de caractères non bruités donne des résultats totalement conformes aux attentes. Mais, lorsque du bruit est introduit dans les données d'entrée, en modifiant le nombre d'impulsions des fréquences d'entrée, les cas obtenus pour lesquels les résultantes donnent des valeurs erronées varient en fonction du pourcentage de valeurs bruitées de la matrice de données. Cependant, ces cas problématiques sont acceptables compte tenu de la dimension de la matrice de données ainsi que des hypothèses émises lors de l'approximation des quantrons. Est-ce pourquoi qu'il faudrait apporter une nouvelle vision d'implémentation mixte (logicielle - matérielle) basée sur le fonctionnement approximatif du quantron? Cette nouvelle approche peut être vue sous deux angles. Soit que les erreurs d'approximation sont au niveau du degré de troncature de la fonction de transfert du quantron ou alors dans une autre forme d'approximation du quantron dans laquelle les variations des valeurs de ses paramètres sont moins limitées. Cette vision plus complexe permettra de bénéficier des atouts du quantron non exploités tels que les cas de non conduction qui apporteront plus de flexibilité dans la reconnaissance ou l'ajout des nouveaux éléments de données dans l'algorithme d'apprentissage.

CHAPITRE 5

ALGORITHME D'APPRENTISAGE DU RÉSEAU DE QUANTRONS PARALLÈLES

Le perceptron est utilisé dans la machine à état liquide pour approximer des fonctions continues grâce à sa capacité d'extraction de l'information. Il y joue le rôle de filtre de sortie du système. Cependant, ce neurone ne peut pas résoudre des problèmes non linéairement séparables.

Une machine à état liquide conserve des informations des entrées antérieures dans l'état courant du liquide grâce à sa récursivité. Ces machines transforment la complexité du problème de petite dimension à de grande dimension, facilitant ainsi l'identification spatiale des données. Cette transformation des données d'entrée amène à l'obtention, à la sortie, d'un résultat différent pour chacun des stimuli d'entrée.

Les machines à état liquide ont été développées à partir de deux types de neurones, les neurones à impulsions pour le module liquide et le perceptron comme filtre de sortie. Dans ce chapitre, nous nous attelons à étudier les possibilités offertes par l'exploitation du quantron dans les machines à état liquide. Il sera étudié, d'une part, en tant que filtre de sortie, et par la suite, la possibilité de l'intégrer comme microcircuit sera évaluée. Mais, l'utilisation du quantron dans la conception du LSM est-elle utile compte tenu de sa capacité de résoudre des problèmes non linéairement séparables ? Afin de mieux aborder cette interrogation, il faudra évoquer tout d'abord le concept de machine à état liquide.

5.1 Concept de machine à état liquide

La machine à état liquide est une approche du mécanisme d'analyse des données d'entrée continues qui s'avère plus réaliste. Cette modélisation a été développée par Maass et son équipe [28]. Le principe consiste à effectuer un calcul, en temps réel, en intégrant des perturbations continues dans un système dynamique et homogène. Il en résulte la transformation des vecteurs d'entrée de petite dimension à une grande, incluant des perturbations aléatoires et continues. Ainsi, le processus de calcul devient plus efficace lorsque la dynamique du système est complexe, plus précisément, lorsque la dimension de la structure de l'état liquide est grande. Dans ce cas, le système devient un approximateur universel capable, grâce à un ensemble de filtres de sortie, d'approximer n'importe quel type de données d'entrée.

La structure liquide est la partie complexe du système en termes de conception. Il s'agit d'une modélisation du comportement d'un liquide stable, au départ, mais susceptible d'être perturbé par la présence d'une entrée quelconque. En théorie, tous les systèmes peuvent être utilisés pour modéliser l'état liquide. Par contre, il faut trouver ceux qui fourniront une meilleure performance durant l'exécution des tâches ciblées.

En ce qui a trait au modèle proposé par Maass, l'état liquide retenu est un réseau de neurones à impulsions [29]; il sert d'entrée à la fonction d'affichage. Il ne produit aucun rendement mais favorise la transformation des entrées sous forme de données de plus grande dimension. Le filtre de sortie analyse l'état du liquide pour une entrée donnée et calcule le rendement du système.

Comme pour les liquides, l'état liquide de la LSM se compose d'un ensemble d'éléments dans lequel chacun d'eux exerce une influence sur son voisin immédiat en cas de perturbation du

liquide ou en présence d'un stimulus externe. La particularité de la LSM réside dans le fait que, contrairement au réseau de neurones conventionnels, elle ne requiert pas la construction des états du liquide en fonction du type d'application. Elle produit une série de vecteurs de sortie proportionnelle à la série de vecteurs de données d'entrée. Ces vecteurs de sortie seront dynamiquement analysés par les filtres de sortie adéquatement configurés pour le type d'application envisagé.

La figure 5.1 représente le diagramme bloc d'une machine à état liquide.

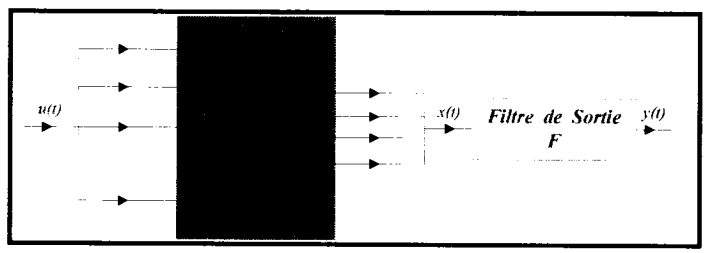


Figure 5-1 : Schéma bloc d'une machine à état liquide

Elle est constituée de trois composantes : une couche d'entrée, un module liquide représentant une structure de grande dimension et un filtre de sortie.

5.1.1 **La composante liquide de la LSM**

La composante liquide L est un microcircuit défini par un réseau de neurones interconnectés de manière aléatoire. Cette structure a pour but de transformer les données d'entrée du système en

données de grande dimension. Elle transforme les entrées *u(t)* en états liquides *x(t)*, d'après la formule :

$$x(t) = (Lu)(t) \tag{5.1}$$

x(t) passant par un grand nombre d'états sous l'influence des entrées. Il ne dépend pas seulement de l'entrée *u(t)*, mais aussi de façon non linéaire des valeurs antérieures *u(t)*. Le vecteur *x(t)* caractérisant l'état liquide du système est de très grande dimension en comparaison au vecteur d'entrée *u(t)*.

La figure 5-2 représente la conception d'un microcircuit du module Liquide *L* d'une LSM.

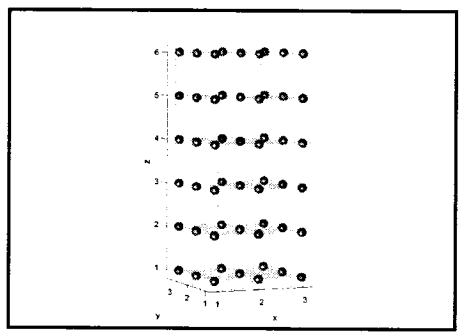


Figure 5-2 : Structure de microcircuit du module Liquide d'une LSM

Cette structure est constituée de 54 neurones à impulsions dont 43 excitateurs (en gris) et de 11 neurones inhibiteurs (en noir) ; ce sont des neurones à impulsions de la forme «integrated and fire».

Cette modélisation est réalisée par un réseau de neurones à impulsions interconnectés. Le réseau produit est aléatoirement distribué par des neurones excitateurs ou inhibiteurs. Selon le modèle présenté par Maass, un rapport d'environ 20% de neurones inhibiteurs devrait être respecté.

La caractéristique la plus importante de la structure liquide consiste à pouvoir réagir de façon différente selon les séquences d'entrée présentées [34]. Cette propriété du liquide explique sa capacité de créer les trajectoires internes différentes pour chacune des classes d'entrée. Elle est donc directement liée à la complexité du liquide.

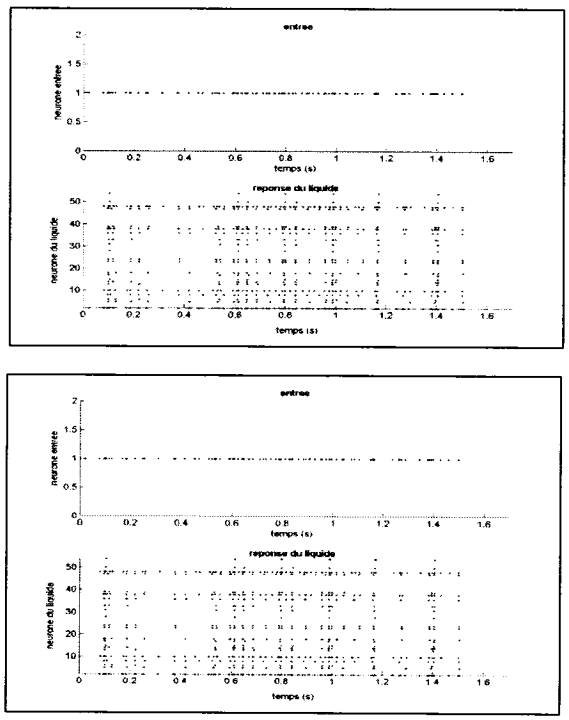
5.1.2 **Simulation de la composante liquide de la LSM**

Le modèle de liquide présenté à la figure 5-2 a été simulé avec quatre échantillons de signaux à impulsions. Les résultats de cette simulation sont présentés à la figure 5-3. Le modèle de neurone à impulsions utilisé pour cette implémentation est du type statique [35]. Il est ainsi défini car l'amplitude de chaque réponse post-synaptique est constante. La formule de la réponse synaptique de chaque impulsion est donnée par :

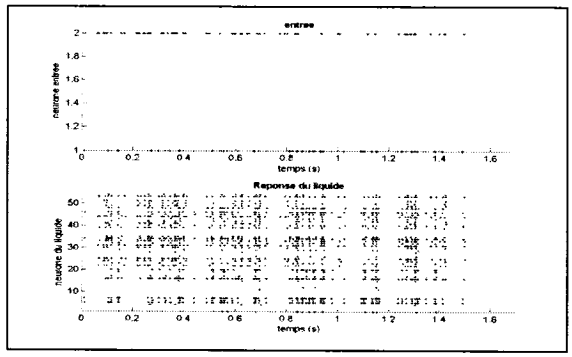
$$x(t) = w \times e^{-\frac{t}{\tau}} \tag{5.2}$$

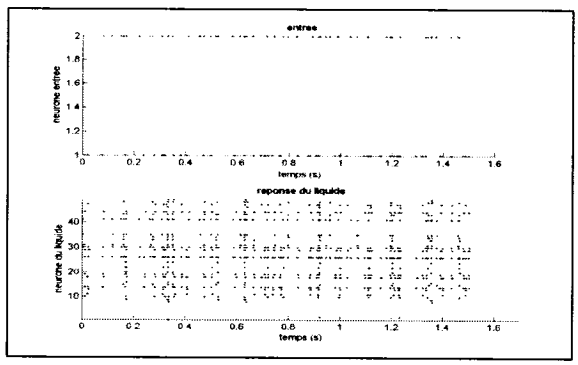
Où w est l'amplitude de réponse encore appelée poids synaptique et τ la constante de temps synaptique.

Tableau 5-1 : Simulations du microcircuit du modèle liquide (figure 5-2) de neurones à impulsions statiques

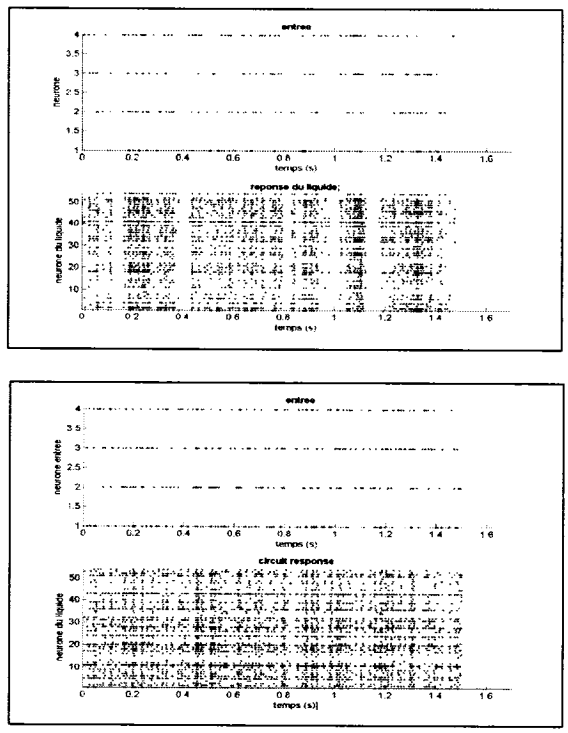


Réponse de stimulus du liquide avec un neurone à impulsion d'entrée. Deux types de signaux à impulsion presque semblables génèrent chacune une réponse totalement différente





Réponse de stimulus du liquide avec deux neurones d'entrée à impulsion. Deux entrées de signaux à impulsions légèrement différents génèrent chacune une réponse totalement différente



Réponse de stimulus du liquide avec quatre neurones d'entrée à impulsion. Deux fonctions d'entrée distinctes génèrent chacune une réponse totalement différente du liquide

Les résultats de simulations prouvent que l'une des caractéristiques les plus importantes de la structure liquide est de pouvoir réagir de façon différente selon les séquences d'entrée présentées. Cette propriété de séparation du liquide explique sa capacité de créer des trajectoires internes différentes pour chacune des classes d'entrée. Une fois la conception du liquide terminée, il faut pouvoir interpréter les résultats ou réponses de ce microcircuit. Cette partie est gérée par le filtre de sortie de l'élément du liquide [35]. Cependant, dans cette partie, l'objectif est d'arriver à générer une implémentation qui sera analysée par un filtre de sortie conçu à l'aide du quantron.

5.1.3 **Le filtre de sortie de la LSM**

Le filtre de sortie F de la LSM, connecté à la sortie du module liquide, produit une sortie selon la formule suivante :

$$y(t) = F(x(t)) \tag{5.3}$$

La capacité de ce module de distinguer chaque trajectoire et de les relier aux sorties de chaque cible est la propriété d'approximation. Il calcule en temps réel la sortie $y(t)$. Il peut être sans mémoire et un neurone comme le perceptron est utilisé pour chaque unité de sortie caractérisée par une règle d'apprentissage appelée p-delta [4]. Le quantron pourrait être aussi un exemple de modèle de filtre adéquat pour l'interprétation de données de sortie.

5.1.4 **La règle d'apprentissage p-delta**

La règle p-delta est un algorithme défini pour un réseau de perceptrons en parallèle à une seule couche [5]. La règle p-delta permet à un perceptron d'approximer toute fonction mathématique d'excitation à pulses. Elle génère pour chacune des données d'entrée une activation de sortie. Elle actualise aussi les poids synaptiques des entrées en fonction des valeurs désirées, en tenant compte de la marge d'erreur prédéfinie, assurant ainsi la convergence de l'algorithme.

Son principe revient à calculer le nombre de pulses émis en sortie pendant un intervalle de temps bien déterminé; un ajustement des poids synaptiques des sorties parallèles de neurones est donné dans l'équation suivante :

$$\vec{W} \leftarrow \vec{W} - \eta(\|\vec{W}\|^2 - 1)\vec{W} + \eta\,\vec{k} \tag{5.4}$$

où :

$$\vec{k} = \begin{cases} -\vec{x} \; si \; \hat{o} > o + \varepsilon \; et \; 0 \le \vec{W}.\vec{x} \\ \vec{x} \; si \; \hat{o} < o - \varepsilon \; et \; \vec{W}.\vec{x} \le 0 \\ \mu\vec{x} \; si \; \hat{o} \le o + \varepsilon \; et \; 0 \le \vec{W}.\vec{x} < \gamma \\ -\mu\vec{x} \; si \; \hat{o} \ge o - \varepsilon \; et \; -\gamma < \vec{W}.\vec{x} \le 0 \\ \vec{0} \quad \text{sinon} \end{cases} \tag{5.5}$$

avec :

$\vec{W}$: vecteur des poids synaptiques des entrées ;

$\vec{x}$: vecteur des entrées du filtre de sortie ;

$\hat{o}$: nombre de pulses obtenu en sortie ;

o : objectif à atteindre ;

ε : erreur tolérée ;

γ : marge à respecter pour assurer la convergence lors de l'entraînement ;

μ : facteur entre 0 et 1 relié à l'importance de respecter la marge (généralement fixée à 0,5) ;

η : facteur d'apprentissage.

Cependant, il importe de choisir adéquatement les paramètres d'entraînement des neurones d'extraction car ils ont une influence sur les densités de pulses approximables en sortie. Dans le cas où les neurones d'extraction sont des quantrons, un nouvel algorithme d'apprentissage adapté à ce type de filtre de sortie est envisageable.

5.2 Filtre de sortie du liquide à l'aide du quantron

Comme nous l'avons présentée dans le chapitre 3, l'équation de la résultante du quantron est donnée par la formule suivante :

$$R(t) = \sum_{i=0}^{+\infty} \varphi(t - \Theta - ix) \tag{5.6}$$

La sortie du quantron étant égale à l'instant où la somme des impulsions d'entrée atteint le seuil prédéfini selon l'équation 3.8.

Le module liquide de la LSM engendre en sortie des impulsions de signaux pour lesquelles les amplitudes sont constantes si le liquide est constitué de neurones à impulsions statiques. La résultante de chaque unité de vecteur d'entrée du quantron peut être normalisée. Donc, chaque canal qui conduit sera affecté de la valeur 1. Pour les filtres de sortie formés de perceptrons, l'algorithme d'apprentissage P-delta procède à la sommation des impulsions de signaux émises par les unités de sortie du liquide. La valeur de la sortie du quantron peut donc être redéfinie comme correspondant à l'instant où la valeur de sa fonction résultante atteint la somme des canaux d'entrée du quantron émettant un signal. Ainsi, le seuil de sortie du quantron est fixé en fonction de la somme des impulsions de signaux émises par l'ensemble des canaux d'entrée. Cette considération se traduit par une relation linéaire entre le vecteur d'entrée du quantron et sa résultante.

Une nouvelle forme simplifiée du quantron est donnée par cette expression :

$$\varphi(t) = f(w)(u(t) - u(t - S)) \tag{5.7}$$

Où *f(w)* est une fonction dépendant du paramètre de poids synaptique et du vecteur d'entrée du quantron. Cette fonction est définie comme suit :

$$f(w) = wix \tag{5.8}$$

où

w: représente le poids synaptique ;

i : le nombre d'impulsions fréquentielles émises dans une période de communication ;

x : l'entrée du quantron.

On définira une période d'apprentissage correspondant au cycle de temps nécessaire à la transmission et au traitement (apprentissage) d'un stimulus (information) d'entrée du quantron. Le facteur temps réel, bien qu'il soit considéré, dans ce cas, n'est pas une contrainte car la période de traitement du filtre doit être largement supérieure à la période maximum nécessaire à la transmission d'une donnée.

La figure suivante explique le mécanisme de validation de la résultante de sortie du quantron.

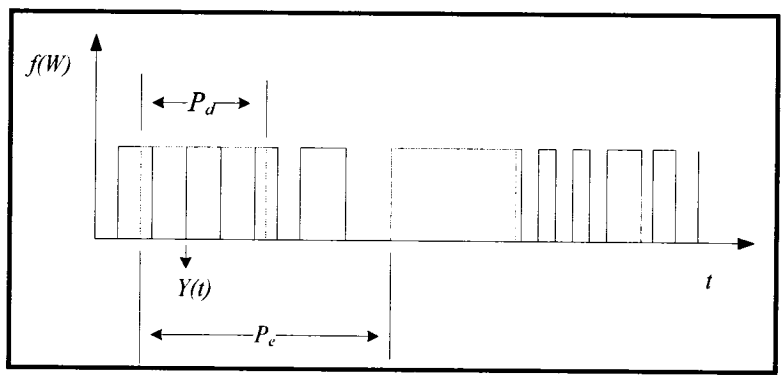


Figure 5-3 : Exemple de validation de la sortie *Y(t)* du quantron parallèle durant un cycle d'entrainement.

La période d'émission d'une impulsion en entrée doit être inférieure à la période d'entraînement. C'est-à-dire :

$$P_e > P_d \tag{5.9}$$

Où P_e et P_d sont respectivement la période de traitement d'un stimulus et la période de transmission d'une entrée.

Pour l'application de reconnaissance de caractères relatée au chapitre 4, le but est d'arriver à définir un algorithme d'apprentissage permettant au réseau de quantrons de procéder à une reconnaissance de caractères spatiale par période d'émission de stimulus. En considérant cette application pour la définition de l'algorithme d'apprentissage du réseau de quantrons parallèles à une seule couche, la résultante de la nouvelle forme simplifiée de quantron est définie par :

$$R(t) = \sum_j \varphi(t-\theta) = \sum_j wix(u(t-\theta) - u(t-\theta-S) \tag{5.10}$$

Où j représente la dimension du vecteur d'entrée.

Comme dans le cas de l'algorithme p-delta du perceptron, la normalisation de la fonction résultante (5.10) du quantron sera effectuée. Ainsi, le paramètre i définissant la répétition de la fréquence de l'unité du vecteur d'entrée n'aura aucune influence sur le résultat de sortie du quantron. Ceci s'explique par le fait que nous considérons que le traitement d'un stimulus d'entrée se fait dans une plage de temps d'apprentissage constant fixée au départ. Dans cette plage, l'évaluation de l'émission d'un signal pour chaque canal d'entrée est faite. Pour cette

évaluation, nous faisons abstraction de la répétition du paramètre *i* durant cette même période. La valeur de *i* est fixée à 1. Considérant l'hypothèse qui imposait de fixer le seuil de transmission du quantron en fonction de la somme des signaux à impulsions émis par l'ensemble des canaux lors d'une communication, nous pouvons donc poser :

$$f(w) = \begin{cases} 1 & si\ wx > 0 \quad (conduction) \\ 0 & si\ x = 0 \quad (non\ conduction) \end{cases} \tag{5.11}$$

Le poids synaptique d'un canal donné sera égal à 1 en présence d'au moins une impulsion et de 0 dans le cas contraire.

Une façon d'établir la conduction en fonction du seuil est de diviser la période de traitement en intervalle de temps en fonction du nombre de quantrons parallèles ou filtres nécessaires au type d'application envisagée. De cette façon, la non linéarité de la valeur de sortie du quantron par rapport à son vecteur d'entrée pour une période de traitement prédéfinie (5.11) est conservée. Cette période est considérablement supérieure à la période d'émission du vecteur d'entrée. Dans le cas de l'application de reconnaissance de caractères, les intervalles de temps nécessaires à la validation de chaque caractère sont déterminés. On a :

$$y_m = \begin{cases} t \quad avec \quad t_{\inf\_m} < t < t_{\max\_m} \quad et \quad t = \min\{t\ si\ R_m(t) > \Gamma_m \\ 0 \end{cases} \tag{5.12}$$

avec :

$$\Gamma_m = \sum canal\ d'entrée\ devant\ émettre\ un\ signal = \sum i \tag{5.13}$$

où

y_m : sortie du $m^{ième}$ quantron en parallèle ;

t_{inf_m} : borne inférieure de l'intervalle de temps de validation de la fonction résultante du quantron m durant un cycle de traitement donné ;

t_{max_m} : borne supérieure de l'intervalle de temps nécessaire pour calculer la résultante de sortie du quantron m durant la période de traitement du stimulus d'entrée;

Γ_m : seuil critique du quantron m ou encore amplitude minimum requise à la fonction résultante pour permettre une sortie t, si cet instant est compris dans l'intervalle de temps prédéfini ;

m : nombre de filtres de sortie nécessaire pour l'application envisagée ;

L'obtention d'un meilleur écart des seuils critiques de chaque quantron est proportionnelle à la dimension du vecteur d'entrée. Ceci augmentera alors le rendement de classification du réseau de quantrons parallèles à une seule couche.

A partir de cet ensemble d'hypothèses, la conception d'un algorithme d'apprentissage du quantron simplifié est envisageable. Le quantron aura un seuil de décision égal à la somme des canaux d'entrée du quantron ayant émis au moins une impulsion dans la période d'apprentissage donnée.

5.2.1 **Algorithme du réseau de quantrons parallèles**

Le modèle d'algorithme du réseau de quantrons parallèles est inspiré de l'algorithme p-delta. Il peut être appliqué à un filtre de sortie constitué par un quantron d'une LSM. Dans ce cas, la première étape consiste à identifier les neurones de sortie du liquide qui seront connectés aux entrées du filtre (quantron). Il existe certaines réponses (neurones de sortie) du module liquide de la LSM qui ont un comportement similaire aux entrées du système. De plus, il serait important de construire un algorithme d'apprentissage indépendant des modèles de sortie du module liquide. Ceci amène à définir un algorithme qui pourra être appliqué à un réseau de quantrons parallèles à une seule couche dont les données d'entrée peuvent ou non être soumises à un système de traitement préalable (ex : machine liquide). Contrairement au perceptron qui ne peut résoudre que les problèmes linéaires, le quantron a la possibilité de résoudre des problèmes non linéaires. Ainsi, un système de prétraitement aura pour but d'augmenter le rendement de son taux de classification.

La fonction de l'erreur quadratique est donnée par l'équation suivante :

$$E = \frac{1}{2}\sum (d_j - y_j)^2 \tag{5.14}$$

avec d_j et y_j les paramètres représentant respectivement la valeur désirée et la valeur obtenue pour chaque sortie du quantron.

La sortie du quantron correspond au moment où la résultante de celle-ci atteint la valeur du seuil critique ; ce dernier est prédéfini comme la somme des impulsions de signaux de chaque canal

d'entrée du quantron. La normalisation de l'amplitude du signal de chaque canal permet de déduire que la valeur du seuil est égale au nombre de canaux d'entrée activés (signal).

Le paramètre S du quantron est défini comme étant la constante dont la valeur est égale au pas itératif de la fréquence d'émission d'une donnée d'entrée. Cette normalisation du paramètre S est valable et justifiable car l'influence de sa variation n'aura plus d'importance sur le mécanisme d'apprentissage. L'influence du paramètre de déphasage θ est aussi prise en compte. Ce paramètre servira à calibrer les impulsions d'entrée des canaux d'émission données dans l'intervalle de temps de validation alloué au quantron (filtre).

La dérivée de la fonction de coût peut donc être évaluée en fonction du paramètre du poids synaptique selon la formule suivante :

$$\frac{\delta E}{\delta W} = \frac{\delta E}{\delta y}\frac{\delta y}{\delta W} = -(d_j - y_j)\frac{\delta y}{\delta W} \tag{5.15}$$

L'intervalle de conduction de la sortie de chaque quantron a été choisi au départ en fonction du seuil critique. L'amplitude d'un élément de la résultante de sortie du quantron simplifié est égale à 1 s'il y a conduction, et 0 si le dit canal ne conduit pas. Donc, la dérivée de la fonction de sortie du quantron est considérée proportionnelle à la dérivée de sa fonction résultante dans l'intervalle prédéfini de traitement du stimulus d'entrée. L'équation (5.14) devient donc :

$$\frac{\delta E}{\delta W} = \frac{\delta E}{\delta y}\frac{\delta y}{\delta W} = -(d_j - y_j)\frac{\delta R(t)}{\delta W} = -(d_j - y_j)x\alpha(u(t-\theta) - u(t-\theta-S)) \tag{5.16}$$

où α est une constante de proportionnalité

L'actualisation des paramètres du quantron se fera dans l'objectif que la valeur de la fonction de coût E soit plus petite que le facteur de tolérance λ.

La mise à jour des paramètres des poids synaptiques du quantron j s'effectuera en tenant compte de la simplification de celui-ci par le vecteur unitaire dans le cas d'une conduction. D'où l'expression suivante :

$$W_j(i+1) = \begin{cases} 1 \; si \; x_j \neq 0 \\ 0 \; si \; x_j = 0 \end{cases} \tag{5.17}$$

avec :

x_j : valeur d'entrée du canal j différente de 0 s'il y a présence d'une impulsion.

$W_j(i+1)$: poids synaptique de l'entrée j à l'itération $i+1$.

La mise à jour du paramètre de déphasage est réalisée en fonction de la conduction du canal de données d'entrée et de l'intervalle de temps de validation alloué durant le cycle d'apprentissage du quantron en question. Ainsi, il importe de s'assurer que chaque canal d'entrée du quantron émet au moins une impulsion. Cette dernière devra être active dans l'intervalle de temps de décision de la validation de la résultante de quantron. Ceci se traduit par :

$$\theta_j(i+1) = \begin{cases} \theta_j(i) & si \quad x_j \neq 0 \quad et \quad t_{\min\_j} \leq \theta_j(i) < t_{\max\_j} \\ \theta_j(i) + \mu \times \Delta\theta & si \quad x_j \neq 0 \quad et \quad \theta_j(i) < t_{\min\_j} \\ \theta_j(i) - \mu \times \Delta\theta & si \quad x_j \neq 0 \quad et \quad \theta_j(i) > t_{\max\_j} \\ 0 & autrement \end{cases} \tag{5.18}$$

avec :

μ : facteur d'ajustement du déphasage. Il est positif et inférieur à 1 ;

$\Delta\theta$: variation de déphasage calculée entre l'intervalle ciblé de validation de la sortie et le déphasage courant.

Ainsi, la résultante sera valable au moins dans l'intervalle de temps alloué à la prise de décision de la valeur de sortie du quantron. Donc, la valeur de sortie du quantron sera une valeur comprise entre $t_{\min\_j} \leq y_j < t_{\max\_j}$ lorsque ce dernier conduit.

L'ajustement de la valeur de sortie en fonction du facteur de tolérance γ de l'erreur en sortie est fait par l'ajustement de la valeur constante S du quantron. Il s'effectue lors de la détermination, à la sortie, des valeurs désirées en fonction des échantillons d'entrée. Ainsi, durant l'apprentissage, si le facteur de tolérance de l'erreur est diminué, l'ajustement de la valeur constante de S deviendra :

$$S_j(i+1) = \begin{cases} S_j(i) & si \quad \gamma \geq E \\ S_j(i) + \beta \times \Delta\theta & si \quad \gamma < E \end{cases} \tag{5.19}$$

où β est le facteur d'ajustement du paramètre S représentant le noyau du quantron

L'ensemble des éléments de l'algorithme du quantron parallèle est défini. Il s'agit du modèle de quantron utilisé et des fonctions de mise à jour. Donc, une implémentation de cet algorithme sera présentée pour un exemple d'application de reconnaissance de caractères.

5.2.2 **Implémentation de l'algorithme du quantron parallèle**

Un modèle de quantron parallèle a été implémenté à l'aide de l'algorithme du quantron parallèle présentement en développement. L'application implémentée est la reconnaissance de caractère telle que définie dans le chapitre 4. Elle est limitée à une matrice de 3x2 (6 pixels) pour cette implémentation. Une série de caractères a été prise en compte et leur apprentissage supervisé s'est effectué selon les considérations sur les valeurs désirées.

La figure suivante présente un exemple de simulation d'apprentissage d'un stimulus donné.

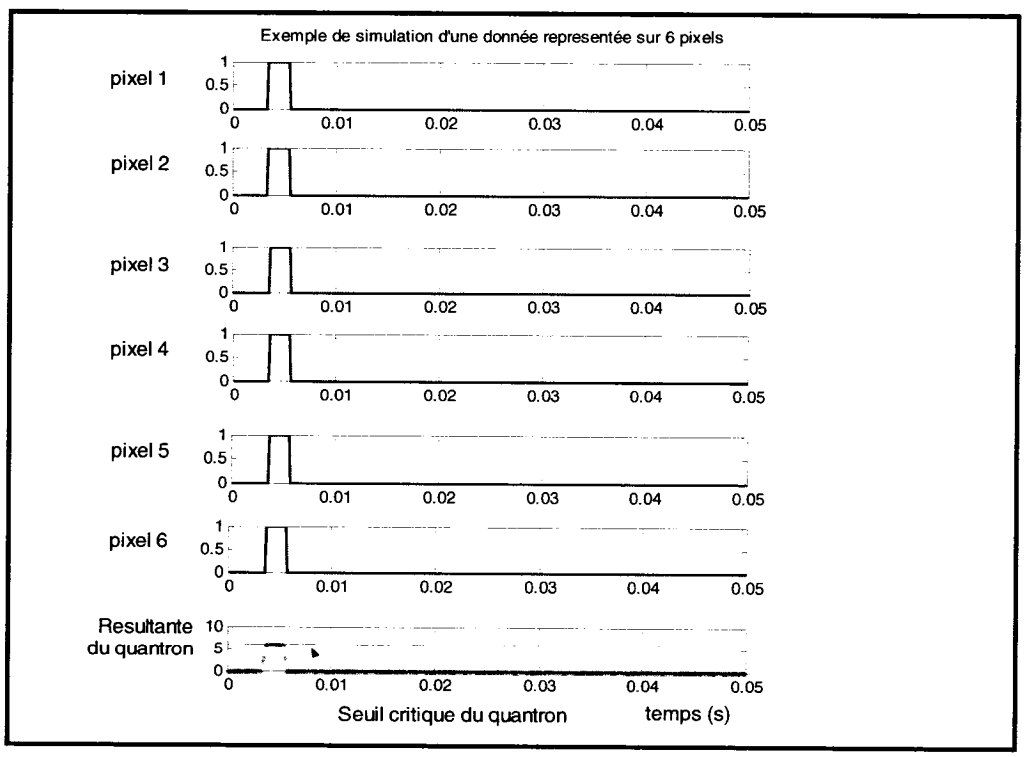


Figure 5-4 : Simulations de l'apprentissage du quantron parallèle à une seule couche

Dans la figure 5-4, la partie supérieure montre les signaux de chaque pixel de la matrice de données d'entrée. Elle est constituée de 6 pixels dans cet exemple. Les pixels 1à 6 émettent une impulsion durant la période d'apprentissage. La dernière courbe de cette figure montre l'évolution de la fonction résultante après chaque itération de la période d'apprentissage. La somme des pixels actifs correspond à l'amplitude de la fonction résultante durant la période de conduction. Ainsi, la sortie du quantron sera égale au premier instant de la période d'activation de la fonction résultante.

5.3 Conclusion

Comme dans le cas général des structures dynamiques, les machines à état liquide sont constituées de deux blocs interconnectés. Ils représentent respectivement la partie de convolution et un filtre linéaire. La partie de convolution, dans le cas de la machine à état liquide est une structure du microcircuit qui matérialise les états liquides du système.

La machine à état liquide effectue un traitement temporel des données d'entrée de façon explicite dans son module liquide. Cette caractéristique facilite l'interprétation de ces données. Le même état liquide permet d'interpréter plusieurs sorties. La machine à état liquide est résistante à la présence du bruit.

La construction du module liquide est réalisée par un microcircuit ayant pour fonction de transformer les données d'entrée de petite dimension en données de très grande dimension. Le résultat de cette transformation servira d'entrée à sa seconde composante appelée filtre de sortie.

Le filtre de sortie, habituellement conçu à base du perceptron, pourra être obtenu à base du quantron.

Cette modification de conception est exécutable grâce à la création d'un algorithme d'apprentissage du quantron parallèle qui permet d'entrainer un réseau de quantrons à une seule couche. L'implémentation de cet algorithme a été effectuée et testée pour un ensemble de caractères définit sur 6 pixels. Les travaux futurs permettront de valider le fonctionnement de cet algorithme pour une dimension de données d'entrée plus grande. Les résultats de cette implémentation devront montrer qu'en augmentant la dimension des données d'entrée, les performances de classification du réseau parallèle augmentent considérablement. Ceci permettra de conclure qu'en utilisant le module liquide comme prétraitement des données, il est non seulement plausible d'obtenir un système capable de résoudre n'importe quel type de fonction mais de l'agrémenter également d'un meilleur rendement.

CHAPITRE 6

DISCUSSION ET CONCLUSION

La première partie de ce travail a été consacrée à la présentation d'un prototype de modélisation de réseaux de neurones multicouches. Ces réseaux utilisent l'algorithme de rétropropagation de l'erreur optimisé comme paradigme d'apprentissage. L'implémentation sert d'analyse du modèle classique, le perceptron multicouche, pour une application de classification de caractères. Cette analyse a pour but de faciliter l'adaptation de la rétropropagation de l'erreur pour le nouveau modèle de réseaux multicouches. Ce dernier possède le quantron comme élément fondamental.

Le quantron est un nouveau type de neurone artificiel dont le fonctionnement a été étudié. Il est considéré comme neurone dynamique et statique. En entrée, il analyse des impulsions fréquentielles de potentiels post-synaptiques. Il génère, à sa sortie, un instant de conduction si le stimulus présenté en entrée atteint un seuil critique prédéterminé. L'étude du quantron fait constater que contrairement au perceptron, qui se limite à la résolution de problèmes linéaires, il est capable de résoudre des problèmes non linéaires. Cette caractéristique se doit de motiver la recherche d'un algorithme d'apprentissage pour étudier des familles d'applications de classification.

La deuxième partie de cette recherche a été axée sur la conception d'un algorithme de rétropropagation de l'erreur pour un réseau de quantrons multicouches. Il est destiné à la résolution de la même application de reconnaissance de caractères. Son implémentation a été réalisée en plusieurs phases. La première consiste à redéfinir un modèle approximatif du

quantron pour la phase d'apprentissage. Le modèle d'approximation du quantron utilisé est un modèle de fonction quadratique. Ce choix se justifie par le fait que sa fonction de transfert, dans une phase de conduction, se comporte comme une parabole dont l'amplitude maximum de la courbe se trouve au voisinage de l'instant définissant la sortie du système. L'algorithme de reconnaissance choisi est la méthode de descente du gradient adaptée aux différents paramètres du quantron. De cette implémentation découle la preuve que le quantron peut être employé en réseau multicouche avec un algorithme d'apprentissage par rétropropagation de l'erreur adapté à ses paramètres. Ses performances en termes d'apprentissage (convergence) sont considérablement élevées contrairement au perceptron multicouche. Cependant, dans les travaux futurs, il faudrait procéder à des tests de validation plus poussés afin de déterminer les limites fonctionnelles de ce nouvel algorithme.

L'implémentation matérielle des réseaux multicouches nécessite plus de ressources matérielles, réduisant ainsi la performance du système. La technique de rétropropagation de l'erreur par la descente du gradient est biologiquement improbable par rapport au fonctionnement du cerveau humain. C'est dans cette optique qu'une nouvelle approche de conception de réseaux, utilisant l'idée de la machine à état liquide, a été étudiée. Elle est présentée en dernière partie avec une analyse de l'utilisation du quantron dans ce système.

Il a donc été aussi démontré que le quantron est utilisable comme filtre de sortie de la machine à état liquide. Ce filtre, habituellement réservé aux perceptrons, exige un nouvel algorithme d'apprentissage pour un réseau de quantrons parallèles à une seule couche. Le modèle du quantron simplifié a été utilisé afin d'accélérer la convergence de l'algorithme d'apprentissage lors de cette implémentation.

L'utilisation du quantron dans une machine à état liquide comme filtre de sortie offre en termes de conception plusieurs avantages. Sa fonction résultante permet la génération des impulsions de sortie du microcircuit de la partie liquide. Elle produit aussi une sortie correspondant à l'instant de conduction du quantron contrairement au perceptron qui compte le nombre d'impulsions de sortie du même microcircuit de l'état liquide. Elle génère une sortie positive, dans le cas de conduction, et négative dans le cas contraire. Ceci, contrairement à la machine à état liquide conventionnelle qui permet au système muni du quantron comme filtre d'avoir une grande flexibilité d'analyse. Cependant, il faut toujours identifier les sorties du liquide qui doivent être interprétées par le filtre de sortie.

L'originalité de ce travail de recherche réside principalement dans la conception de nouveaux algorithmes d'apprentissage de rétropropagation de l'erreur d'un réseau de quantrons multicouches d'une part et dans l'implémentation d'un second modèle d'algorithme pour les quantrons parallèles à une seule couche. Ce modèle joue le rôle de filtre de sortie des machines à état liquide. Les applications utilisées sont relativement simples et limitées au besoin de prouver le concept de fonctionnement des différents modèles d'algorithmes développés.

La prochaine étape de la recherche devra particulièrement tenir compte de la nouvelle approche de conception de réseau de neurones, la machine à état liquide, comme une solution plausible pour résoudre des problèmes de régression. Dans cette nouvelle machine, il faudrait implémenter le quantron comme filtre de sortie. Il importe aussi d'analyser la possibilité d'utiliser le quantron dans le microcircuit afin de transformer la machine à état liquide en un seul module de microcircuit qui est conçu totalement de quantrons représentant le liquide. Ainsi, les vecteurs de sortie de la nouvelle machine de quantrons seront des instants précis. Ceci augmente la possibilité d'exploiter les caractéristiques du quantron comme un neurone versatile.

RÉFÉRENCES

[1] ALDWORTH P. (1999), System-on-chip Bus Architecture for embedded Applications, *International Conference on Computer Design*, p. 297-298.

[2] AYALA J. L. and LÓPEZ-VALLEJO M. L. (2002), Data-quantization policies for power and area minimization of hardware neural networks, *XVII Design of Circuits and Integrated Systems Conference*, Santander, Spain.

[3] AMIN H. & AL (1997), Piecewise linear approximation applied to non linear function of neural network, *IEE Proc- Circuits Devices Syst*, Vol. 144(6), p. 313-317.

[4] AUER P., BURGSTEINER H. M. and MAASS W. (2002), The p-Delta Learning Rule for Parallel perceptrons, *submitted for publication. Online available at http://www.igi.tugraz.at/maass/p_delta_learning.pdf.*

[5] AUER P., BURGSTEINER H. M. and MAASS W. (2002), Reducing Communication for Distributed Learning in Neural Networks, Springer Berlin / Heidelberg, Vol. 2415.

[6] BELATRECHE A., MAGUIRE L. P., McGINNITY M. and XIANG WU Q., An Evolutionary Strategy for Supervised Training of Biologically Plausible Neural Networks.

[7] BOHTE S. M., KOK J. N. and LA POUTR H. (2002), Error-backpropagation in temporally encoded networks of spiking neurons, *Neurocomputing*, Vol. 48, p. 17–37.

[8] CANADIAN MICROELECTRONIC CORPORATION (2002), Block Authoring Flow: Soft IP Block, ICI-104 Version 1.1.

[9] CANADIAN MICROELECTRONIC CORPORATION (2002), Tutorial on CMC's Digital IC Design Flow, Document ICI-096, Part of tutorial Release, Vo1.4.

[10] CARIANI P. (1995), As if time really mattered: Temporal strategies for neural coding of sensory information, *Brain and self-organization*, ed. K. Pribram Erlbaum, Hillsdale NJ, p. 161-229.

[11] CHANG H., COKE L., HUNT M., MARTIN G., McNELLY A. and TOLDD L. (1999), Surviving the SOC revolution, Kluwer Academic Plublishers

[12] COHEN B. (1999), VHDL Coding Styles and Methodologie, 2nd Edition, *Kluwer Academic Plublishers*.

[13] COHEN B. (2001), Component Design by Example ... a Step-by-Step Process Using VHDL with UART as Vehicle, ISBN 0-9705394-0-1.

[14] EGMONT-PETERSEN M., DE RIDDER D. AND HANDELS H. (2002), Image processing with neural networks – a review, *the journal of the Pattern recognition*, Vol. 35, No. 10.

[15] ELISSEEFF A. and PONTIL M. (2003), Leave-one-out error and stability of learning algorithms with applications, *Advances in Learning Theory: Methods, Models and Applications, NATO Science Series III*: Computer and Systems Sciences, Vol. 190, J. Suykens et al. Eds.

[16] GERSTNER W. and M. KISTLER W. (2002), Spiking Neuron Models, *Cambridge University Press*, www.epfl.ch/~gerstner/SPNM/SPNM.html.

[17] HAYKIN S. (1999), Neural networks: A comprehensive foundation, 2nd Edition, *Hardcover*.

[18] HENNESSY J. L. and PATTERSON D. A. (1996), .Architecture des ordinateurs, une approche quantitative 2e édition, *Thomson Publishing et Morgan Kaufman Publishers*, France.

[19] HODGKIN A. L. and HUXLEY A. F. (1952), A quantitative description of membrane current and its application to conduction and excitation in nerve, *Journal of physiology (London) 117*, p. 500–544.

[20] HOPFIELD J. (2002), Artificial Neural Networks, *International Conference Proceedings*, Madrid, Spain, August.

[21] IGEL C. and HÜSKEN M. (2000), Improving the Rprop Learning Algorithm, *Proceedings of the Second International Symposium on Neural Computation*, NC'2000, ICSC Academic Press, p. 115-121.

[22] KOLEN J. F. and POLLACK J. B. (1994), Back-Propagation without Weight Transport, *The Proceedings of the IEEE World Congress on Computational Intelligence.*

[23] LABIB R. (1999), New single neuron structure for solving nonlinear problems, IJCNN '99. *International Joint Conference on Neural Networks*, Vol. 1, p. 617-620.

[24] LABIB R. (1999), Processus de diffusion : outils de modélisation, de prévision et de contrôle, *thèse de doctorat*, École Polytechnique de Montréal.

[25] LAMIREL C., DUCLOY J. and KAMMOUN H. (2000), A self organizing map (SOM) extended model for information discovery in digital library context, *Processings of the International workshop multimedia Data Mining*, ACM SIGKDD conference.

[26] LeCun Y, Bottou L., Bengio Y., and Haffner P. (1998), Gradient-Based Learning Applied to Document Recognition, *Proceedings of the IEEE,* Vol. 86, no. 11, p. 2278-2324.

[27] LEE J.D (1997), Object recognition Using a neural network with optimal feature extraction, *Math. Comput. Modlling*, Vol. 25, No. 12, p. 105-117.

[28] MAASS W., NATSCLAGER T. and MARKRAM H. (2001), Real-time computing without stable states: a new framework for neural computation based on pertubations, preprint, Technische Universitt Gratz.

[29] MAASS W. and BISHOP C. M. (1999) Pulsed Neural Networks, Cambridge: *MIT Press*, p. 408.

[30] MACKAY D. J. C. (2003), Information theory, inference, and learning algorithms, Cambridge *university press.*

[31] MATWORKS (2004), MATLAB User's and Reference Manual. Software Version 7.0, *The language of technical computing.*

[32] MCCULLOCH W. and Pitts W. (1943), A logical calculus of the idea immanent in nervous activity, *Bulletin of Mathematical Biophysics*, Vol. 5, p 115-133.

[33] MEYER Y., JAFFARD S. and RIOUL O. (1987), L'analyse par Ondelettes, *Pour la Science*, p. 28-37.

[34] Natschläger T., Maass W., and Markram H.(2002), The "liquid computer": A novel strategy for real time computing on time series, *Special Issue on Foundations of Information Processing of TELEMATIK*, Vol. 8(1), p. 39- 43.

[35] *Neural Micro Circuit*, http://www.lsm.tugraz.at/download/index.html

[36] PIERRE S. (1998), Inferring New Design Rules by Machine Learning : A Case Study of Topological Optimization , *IEEE Trans. on Man, Systems, and Cybernetics*, Vol. 28A, no 5, p. 575-585.

[37] PRESMAN R. S. (2001), Software Engineering: A Partitionner's Approach, 5th Edition, *McGraw-Hill Higher Education.*

[38] RIEDMILLER M. and BRAUM H. (1993), A direct adaptive method for faster back propagation learning: The rprop algorithm, *In Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA.

[39] SAITOH T. and KANEKO T. (2000), Automatic recognition of Wild Flowers, *International conference on pattern recognition*, Vol. 2, p. 2507.

[40] SALEH R. (2001), System-on-Chip Trends Conference, *System-on-Chip Workshop Proceedings,* Ottawa Canada.

[41] SANGIOVANNI-VINCENTILLI A., and MARTIN G. (2001), Platform-based design and software design methodology for embedded systems, *IEEE design and test of computers*, p. 23-33.

[42] SHIRANITA K., HAYASHI K., OTSUBO A., MIYAJIMA T. and TAKIYAMA R., Determination of meat quality by Image Processing and Neural Network Techniques, Vol. 2, p. 989-992.

[43] SIN-CHUN Ng.; CHI-CHUNG C. and SHU-HUNG L. (2004), Magnified gradient function with deterministic weight modification in adaptive learning, Neural *Networks IEEE Transaction*,Vol. 15, Issue 6, p. 1411-1423.

[44] STAUNSTRUP J. and WOLF W. (1997), Hardware/Software Co-Design, Principles and Practice, *Kluwer Academic Publishers.*

[45] SYNOPSYS (2001), DesignWare AMBA-based Microprocessor Sub-system, Speeding SoC Design with the Next Generation of Reuse, *Synopsys DW Star IP for MIPS seminar.*

[46] UNIVERSITY OF CALIFORNIA, Irvine, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[47] WARRIS C. J., LEE J.-D. and McCORMIK P. G. (1997), Object recognition Using a neural network with optimal feature extraction, *Math. Comput. Modlling*, Vol. 25, No. 12, p. 105-117.

[48] Wolf D., Romero R. and Marques E. (2001), Using Embedded Processors in Hardware Models of Artificial Neural Networks, *Proc. of the Brazilian Simposium of Intelligent Automation (SBAI'2002)*, Canela-Brazil.

[49] XU J., and WOLF W. (2002), Platform-based design and the first generation Dilemma, *Proc. 9th IEEE/DATC Electronic Design Processes Workshop.*