République Algérienne Démocratique et populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Mémoire de magister

Présenté à l'Université de Guelma
Faculté des sciences et de l'ingénierie

Département d'Électronique
Spécialité : Informatique industrielle et imagerie

Présenté par : **NEMISSI Mohamed**

CLASSIFICATION AUTOMATIQUE NEURO-FLOUE AVEC APPRENTISSAGE ÉTIQUETÉ

Sous la direction du : Dr **SERIDI Hamid**

Juin 2004

Soutenu le 30 juin 2004 devant la Commission d'Examen

JURY

Président :	H. TEBBIKH	Pr , Université de Guelma
Rapporteur :	H. SERIDI	M/C, Université de Guelma
Examinateurs :	B. BENSAKER	M/C, Université de Annaba
	B/H. BOUKROUCHE	M/C, Université de Guelma
	B. BENZELTOUT	PHD, Université de Guelma

Remerciement

Je remercie ALLAH le tout puissant pour son aide et
qui par sa providence nous existons.

- *Je tiens à exprimer ma profonde reconnaissance pour mon encadreur M. Hamid SERIDI qui m'a consacré un temps et une disponibilité d'esprit considérables.*
- *Je remercie Messieurs les membres du jury d'avoir accepter la tache de lire mon mémoire et de me faire leurs commentaires.*
- *Je tiens également à remercier sincèrement M. Ahmed LATTI pour son inappréciable aide.*

Je remercie tous ceux qui m'ont aidés soit de prés ou de loin
à réaliser ce travail

Mohamed NEMISSI

الملخص

إن الشبكات العصبونية، و التي يعتبر المخ البشري كنموذج لإنشائها، تمثل عامل هام بالنسبة للباحثين في مجال الذكاء الاصطناعي نضرا لقدراتها الكبيرة على التدريب التلقائي، الحساب المتوازي و التعميم . لقد استعملت بنجاح في عدة مجالات لإعادة معـــرفة الأشكـــال، لكن من بين مساوئها طول مدة تدريبها.
بدلا من تغيير بنيتها، تعديل علاقات تدريبها أو حتى ضبط ثوابتها، و بهدف تسريع عملية تدريبها، نقترح طريقة جديدة "التصنيف الأوتوماتيكي مع التدريب بالبطاقات" و التي تعتمد على تغيير التمثيل الخاص بأمثلة التدريب، و ذلك بإضافة البطاقات. ان هذه الطريقة تتميز بفعاليتها و سهولة إدراجها، بالإضافة إلى إمكانية استعمالها مع المصنفات التي تعتمد على إدماج الشبكات العصبونية والمنطـــق الغامض

Résumé

Les réseaux de neurones artificiels, dont le cerveau humain a servi de modèle pour leur modélisation, présentent un intérêt majeur pour les chercheurs en l'intelligence artificielle vu leurs grandes capacités d'apprentissage automatique, de calcul parallèle et de généralisation. Ils sont utilisés avec succès dans plusieurs applications de la reconnaissance des formes, mais parmi leurs inconvénients est la lenteur de leur phase d'apprentissage. Plutôt que d'essayer de modifier leurs structures, de changer leurs relations d'apprentissages ou même de régler leurs paramètres, et afin d'accélérer leurs phases d'apprentissage, nous proposons une nouvelle approche "la classification avec apprentissage étiqueté" qui s'articule essentiellement sur la modification de la représentation des exemples d'apprentissage en leur ajoutant une caractéristique additionnelle (les étiquettes). L'implémentation de cette technique est simple, robuste et s'applique avec les classificateurs neuro-flous.

Abstract

The Artificial Neural Networks, which human brain was used as model for their modeling, are of major interest for the researchers in the field of Artificial Intelligence considering their great capacities of auto-learning, parallel computing and generalization. They are used successfully in several applications of the Pattern Recognition, but among their disadvantages is the slowness of their training. Rather than to try to modify their structures, to change their relations of training or to even regulate their parameters, and in order to accelerate their training, we propose a new approach, which is articulated primarily on the modification of the representation of the examples, by adding an additional feature (labels). The implementation of this technique is simple, robust and applies with the Neuro-Fuzzy Classifiers.

Sommaire

Introduction

Dieu a créé le monde avec une immense précision, les êtres biologiques sont dotés d'une régularité illimitée. De ce fait, la nature était toujours une source à partir de laquelle les inventeurs s'inspiraient pour mettre en évidence les découvertes les plus importantes, par exemple : en observant les oiseaux qui volent, le secret de la propulsion dans les airs étant découvert ; l'étude de l'œil humaine aidait à la construction des systèmes de vision telle que la caméra ; les bras manipulateurs des robots sont inspirés de ceux des êtres humains... D'une manière générale, les grandes découvertes révèlent des merveilles de la nature. Dans le cadre de l'intelligence artificielle, le cerveau humain a servi de modèle pour la modélisation des réseaux de neurones artificiels, ces derniers qui présentent un intérêt majeur pour les chercheurs du domaine vu leurs grandes capacités de calcul parallèle et de généralisation.

Depuis les temps les plus anciens, l'homme a cherché un appui matériel pour effectuer ces taches quotidiennes, en premier lieu, il eut recourt à des simples machines, actuellement, et vu l'évolution technologique moderne, il rêve de concevoir des machines qui pourrait reproduire son intelligence, qui ont la capacité d'apprendre, de s'adapter faces aux différentes situations, et l'aptitude de résoudre les problèmes rencontrés. En dotant ces machines de la perception, il a sûrement repoussé les limites de la création, ainsi le développement des théories de la reconnaissance des formes a pour but de permettre aux machines d'être capables de réagir selon les excitations externes, d'être sensibles à leurs environnements et capables de prendre connaissance du monde gouvernant en se servant de leurs propres capteurs.

De nos jours, on est arrivé presque a parlé la machine ; ils nous suffit de prononcer un nom, et notre portable obéira en composant le numéro correspondant, ils existent aussi des logiciels qui rédigent un texte qu'on leur dicte... Ces produits, utilisant la reconnaissance des formes, nécessitent une phase d'apprentissage pour pouvoir effectuer ces fonctionnalités : Notre portable a besoin d'enregistrer des noms par son composeur vocal, qui associer leurs numéros de téléphone et les logiciels précédents nécessitent une phase d'entraînement qui leur permet

de s'adapter avec les voix des utilisateurs. Cependant, l'apprentissage automatique est considéré comme l'une des importantes préoccupations des chercheurs de l'intelligence artificielle.

Dans ce travail, d'une part nous présentons quelques méthodes de la classification, on se base sur celles qui utilisent les réseaux de neurones artificiels et, avec un accent particulier, sur celles qui utilisent les systèmes Neuro-Flous, vu que ces derniers sont basés sur la fusion de deux importantes technologies et tentent de se bénéficier de leurs avantages et de compenser leurs inconvénients. D'autre part, et à fin d'améliorer les performances de ces classificateurs, on propose une nouvelle approche d'apprentissage qui repose essentiellement sur la modification de la représentation des exemples d'apprentissage. Ainsi, plutôt que d'essayer de modifier leurs structures, de changer ses relations d'apprentissages ou même de régler leurs paramètres, l'apprentissage étiqueté est basé sur l'ajout d'une caractéristique additionnelle (les étiquettes) aux exemples d'apprentissage.

Le présent mémoire se compose de cinq chapitres :

Le premier chapitre constitue une introduction aux méthodes de la classification, dans lequel on présente quelques approches statistiques de la classification.

Le deuxième chapitre se compose en deux parties : la première constitue une introduction aux concepts et structures des réseaux de neurones artificiels (ANN), dans laquelle ils seront présentés les différents types des ANN, leurs apprentissages et leurs applications. La deuxième partie, quant à elle, est consacrée à la classification par le Perceptron Multi Couche (MLP, Multi Layered Perceptron), le type le plus utilisé des ANN. Dans cette partie on présente les différentes étapes pour l'élaboration d'un algorithme d'apprentissage du MLP en introduisant de différentes techniques de stabilisation et d'accélération de la convergence.

Le troisième chapitre a pour objectif de présenter quelques systèmes flous et Neuro-Flous dédiés à la classification, il se divise en deux parties : La première constitue une introduction à la logique floue et aux systèmes flous, tandis que la deuxième est consacrée aux systèmes Neuro-Flous et leur architectures.

Le quatrième chapitre est consacré à la mise au point de l'approche proposée, dans ces paragraphes on présente les différentes étapes pour effectuer la classification avec apprentissage étiqueté, et les méthodologies de son application avec le Perceptron Multi Couche (MLP, Multi Layered Perceptron), le RVFLN (Random Vector Functional Link Neural Network) et le Classificateur neuro-flou (NFC, Neuro Fuzzy Classifier).

Le cinquième chapitre est réservé aux tests, son but est d'évaluer les performances de l'application de cette technique avec les classificateurs précédents sur de différents types de données, on utilise alors les bases de données suivantes : Iris; Cuisse humaine et Texture. Pour apprécier notre travail nous comparons les résultats obtenus avec celles d'autres travaux effectués sur les même bases de données à savoir : le système SUCRAGE (l'approche numérique et l'approche symbolique) ; les arbres flous et les systèmes d'inférences flous.

CHAPITRE I :

Reconnaissance des formes & Classification

I.1 INTRODUCTION

La reconnaissance des formes (RDF) est une partie intégrante de l'intelligence artificielle, son but réside essentiellement dans l'automatisation des processus d'analyse de données et de prise de décision, le développement de ces méthodes tente de concevoir des systèmes capables d'effectuer la perception d'une façon aussi similaire que possible que l'être humain. Ces techniques se trouvent dans plusieurs applications de divers domaines : l'industrie, la médecine, la robotique, le militaire, la lecture automatique et le traitement d'image. En permettant d'extraire un mot à partir d'un document manuscrit, une cible à partir d'une image satellite, un son de voix à partir d'un bruit, une cellule de cancer à partir d'une image microscopique et les caractéristiques biochimiques à partir d'un échantillon de sang, la RDF accorde la possibilité d'obtenir une information pertinente qui nous intéresse à partir des données volumineuses contenant une quantité d'informations importante.

I.2 REPRÉSENTATION DES OBJETS - ESPACE CARACTERISTIQUE

Dans une population *(P)* formée d'un ensemble d'objets, chacun d'eux peut être représenté par un ou plusieurs attributs, qui sont des variables descriptives de ces objets ayant des valeurs réelles mesurables ou sensées (comme la couleur, la taille, le poids ...). Une caractéristique est soit un attribut ou une fonction d'un ou plusieurs attributs comme le quotient largeur sur longueur, l'algorithme d'un produit ou un paramètre estimé des données... Les caractéristiques doivent être observables : Du lieu, ils sont [1] :

1. Mesurables directement.
2. Obtenues à partir d'une fonction des variables mesurables.
3. Estimées à partir des autres variables mesurables directement et qui sont corrélées avec eux.
4. Calculées ou estimées par une combinaison des trois méthodes précédentes.

Les caractéristiques utilisées dans la classification doivent être indépendantes statiquement, c'est à dire aucune caractéristique ne doit être déterminée à partir d'une autre, et doivent, avec un nombre le plus réduit que possible, permettre de différencier efficacement les classes. D'où la nécessité de la représentation de chaque objet par un vecteur caractéristique permettant d'éviter au maximum la redondance et le superflu.

Si chaque objet d'une population *(P)* est représenté par un vecteur caractéristique $X(x_1\ x_2 \quad x_N)$ ou $x_1, x_2, \ldots x_N$ prennent leur valeur dans $R_1, R_2, \ldots R_N$,on appelle espace caractéristique l'espace euclidien de dimension N formé par le produit cartésien : $E^N = R_1 \times R_2 \times \ldots \times R_N$. Ainsi le passage d'un espace de représentation, formé des attributs, à un espace caractéristique à tendance de :

1. Retenir les informations utiles [1].
2. Éliminer, le plus possible, les informations redondantes qui causent des bruits et dégradent les performances de la classification [2].
3. Rendre les mesures de données plus efficaces pour la décision [3].

I.3 LA CLASSIFICATION

I.3.1 Définition d'un classificateur

La classification est le processus de regroupement des objets à partir de leurs représentations. En reposant sur le niveau de ressemblance, soit entre les exemples à classifier ou entre l'exemple à classifier et les exemples prototypes représentant les différentes classes, un classificateur fourni une sortie $Z(q)$ correspondante à la classes C_q de l'exemple *(q)* représenté par son vecteur caractéristique $X^{(q)}$ (figure I.1)

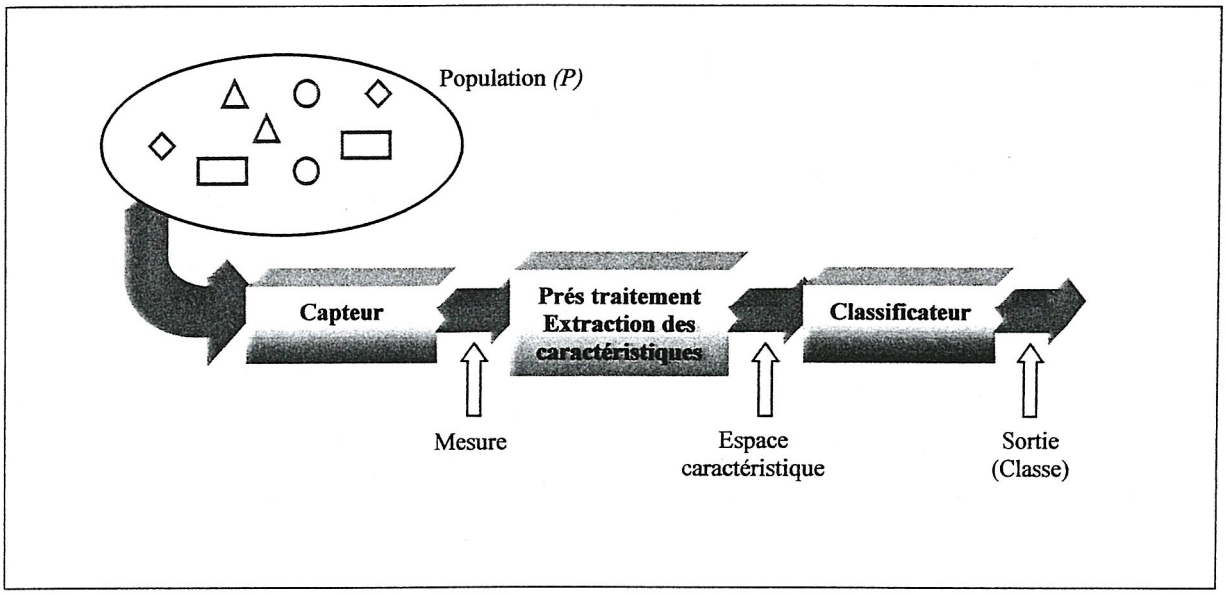


Figure (I.1) : Processus de la reconnaissance des formes

I.3.2 Fonction discriminante

Pour illustrer la séparation des classes dans l'espace caractéristique, considérons d'abord le cas bidimensionnel d'un ensemble d'exemples appartenant à deux classes C_1 et C_2, chacun d'eux étant représenté par deux caractéristiques x_1 et x_2, l'espace caractéristique est donc le plan. La classification de ces exemples revient à former une frontière de décision permettant de les séparer en deux classes. Ainsi, le problème de la classification peut être considéré comme étant un problème de réalisation des fonctions discriminantes $\Phi_1(X)$ et $\Phi_2(X)$ correspondantes aux classes C_1 et C_2. Ces fonctions permettent de classer chaque exemple (X) selon la règle :

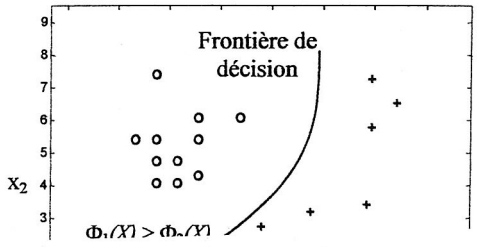
$$X \in C_1 \quad Si \quad \Phi_1(X) > \Phi_2(X) \tag{I.1.a}$$

$$X \in C_2 \quad Si \quad \Phi_2(X) > \Phi_1(X) \tag{I.1.b}$$

La frontière de décision est formée par l'ensemble des points de l'espace caractéristique dont les valeurs prises par les fonctions discriminantes sont égales :

$$\Phi_1(X) = \Phi_2(X) \tag{I.2}$$

Cette frontière de décision réalise une partition de l'espace caractéristique en deux domaines de décision D_1 et D_2 (figure I.2)



جوان
Juin
June

14

-2004-

الإثنين
Lundi
Monday

Note

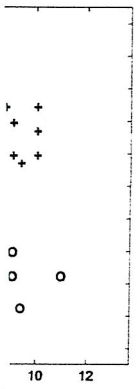
ié avec un fonction

(I.3)

e de décision sera la
ique dont les valeurs

16
17
18
19
20

Important



Juin

Sam	Dim	Lun	Mar	Mer	Jeu	Ven
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Classes linéairement séparables – classes non linéairement séparables: Considérons de nouveau le cas bidimensionnel et bi-classes, on dit que deux classes sont linéairement séparables s'ils le sont par une ligne droite (figure I.3.a), et inversement on dit alors qu'elle ne sont pas linéairement séparables (figure I.3.b).

D'une manière générale, pour une population dont les exemples appartiennent à deux classes C_1 et C_2, et représentés par un espace caractéristique de dimension (N) : S'il existe un hyperplan (H), de dimension $(N-1)$, de telle façon que tous les exemples de la classe C_1 sont d'un coté, et ceux de la classe C_2 sont de l'autre coté, on dit que ces deux classes sont linéairement séparables avec un hyperplan (H), dans l'exemple précédent, cas bidimensionnel, l'hyperplan est une ligne droite. Soit un ensemble de (Q) exemples appartenant à deux classes linéairement séparables $(C_1$ *et* $C_2)$, chacun d'eux étant représenté par son vecteur caractéristique X. La règle de décision est:

$$\begin{cases} X \in C_1 \quad Si \quad \sum_{i=1}^{N} x_i w_i + b \succ 0 & \text{(I.4.a)} \\ X \in C_2 \quad Si \quad \sum_{i=1}^{N} x_i w_i + b \prec 0 & \text{(I.4.b)} \end{cases}$$

La frontière de décision (l'hyperplan de dimension N-1) est constituée de l'ensemble des points $X(x_1 x_2 \quad x_N)$ vérifiant la relation:

$$\sum_{i=1}^{N} x_i w_i + b = 0 \qquad \text{(I.5)}$$

Sous forme matricielle :

$$XW + b = 0 \qquad \text{(I.6)}$$

I.3.4 Modes de Classification (Supervisée et Non-Supervisée)

Selon qu'on dispose ou non d'un ensemble de prototypes, il existe deux types principaux de la classification : La classification non supervisé et la classification supervisée, le premier type consiste à élaborer un système capable d'effectuer la classification en se basant sur l'unique information apportée par les exemples à classifier.

La classification supervisée, quant à elle, nécessite la disposition d'un ensemble d'exemples prototypes qu'on sache leur appartenance à priori, et qui représentent les classes en présence. Dans ce mode la séparation s'effectue soit en assignant à l'exemple a classé, la classe représentée par les prototypes dont il ressemble de plus, soit en réalisant, à partir d'un apprentissage supervisé, un modèle capable d'effectuer la classification.

I.4 DIFFÉRENTES APPROCHES DE LA CLASSIFICATION

La classification statistique est historiquement la première approche de la classification, elle rentre dans le cadre de la théorie de la décision. En utilisant des distributions empiriques des attributs, Fisher [4] est, le premier qui l'employa pour s'aider à la classification. La

classification par la production des règles et la classification structurelle sont des approches plus récentes que la précédente, elles sont basées sur la transformation des attributs en symboles. Les approches modernes sont celles qui utilisent les Réseaux de Neurones Artificiels et la Logique Floue. Les différentes méthodes sont représentées sur le graphe ci-dessous (figure I.4) :

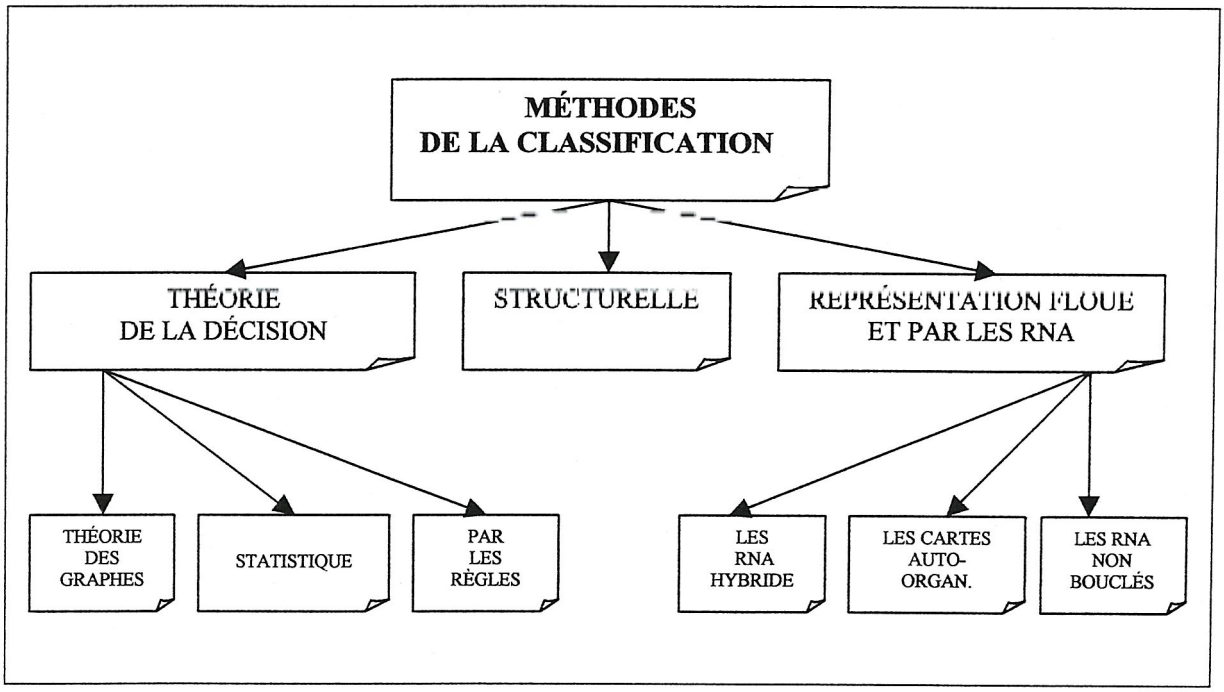


Figure (I.4) : Les différentes approches de la classification

I.5 LA CLASSIFICATION STATISTIQUE

I.5.1 Théorie de décision de Bayes

Développé par Thomas Bayes, la théorie de décision Bayesienne constitue une approche statistique fondamentale pour résoudre les problèmes de la classification [5][6][7][8], elle permet de déterminer les probabilités d'appartenance d'un objet à partir de son observation.

Soit à classifier un objet qui peut appartenir à l'une des K classes C_1, C_2, ... C_K, il s'agit de déterminer les probabilités que celui-la appartient à l'une d'elles. Si on connaît la probabilité à priori $P(C_k)$ pour qu'il appartienne à la classe C_k, ainsi et pour décider à quelle classe il revient, on a intérêt à opter systématiquement à la classe ayant la plus grande probabilité à priori, par exemple dans le cas d'un tir au hasard d'une bille à partir d'un récipient contenant 5 billes bleues et 4 billes rouges, on décide toujours que c'est une bille bleue. Heureusement en pratique on ne prend jamais de décision dans telle situation (avec peu d'information). En fait, on dispose de plus d'informations, chaque objet étant représenté par plusieurs caractéristiques qui aident à la décision en donnant des informations supplémentaires. Soit X le vecteur caractéristique représentant cet objet, et $P(X/ C_k)$ la fonction de densité de probabilité conditionnelle d'observer X, étant donner la classe C_i, selon Bayes, et en

connaissant $P(X/C_k)$, on peut déterminer la probabilité à posteriori $P(C_k/X)$ que sa classe soit C_k, comme suit :

$$P(C_k / X) = \frac{P(X / C_k) P(C_k)}{P(X)} \qquad , k=1, 2, \ldots K \tag{I.7}$$

Avec

$$P(X) = \sum_{k=1}^{K} P(X / C_k) P(C_k) \tag{I.8}$$

Cependant, la règle de Bayes permet de déterminer les probabilité à posteriori d'appartenance d'un objet à partir de l'observation de son vecteur caractéristique, on peut donc décider son appartenance selon la règle :

$$X \in C_i \quad \text{Si } P(C_i/X) = Max_{k=1,2,\ldots K} \{ P(C_k/X) \} \tag{I.9}$$

Pour illustrer les différentes formes des densités de probabilité, prenons l'exemple bi-classes des exemples de la bases de données Iris appartenant aux classes C_1 et C_2, et on considère qu'ils sont représentés par les deux caractéristiques : Quotient largeur pétale sur longueur pétale et largeur sur longueur sépale (figure I.5).

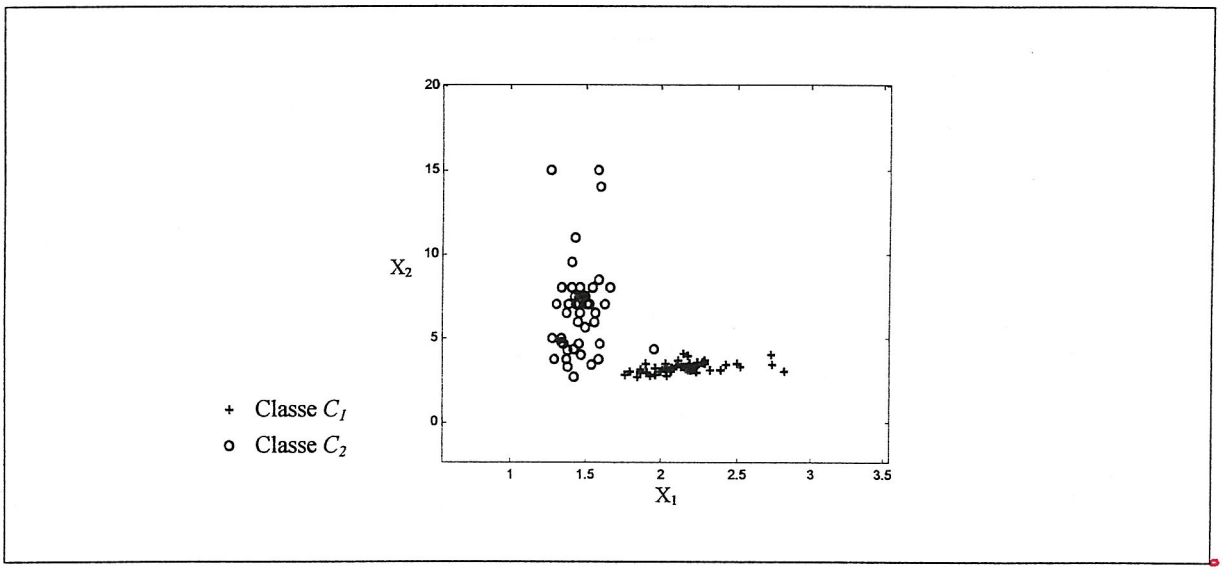


Figure (I.5) : Les exemples de la première et la deuxième classe de la base de données Iris

En supposant que la probabilité à priori d'appartenance aux classes C_1 et C_2 sont égales ($P(C_1) = P(C_2)=0.5$), les fonction de densité de probabilité $P(X/C_1)$ et $P(X/C_2)$, ainsi que les probabilités à posteriori d'appartenance aux classes C_1 et C_2 ($P(C_1/X)$ et $P(C_2/X)$), correspondantes à la première caractéristique, sont représentées sur la figure (I.6)

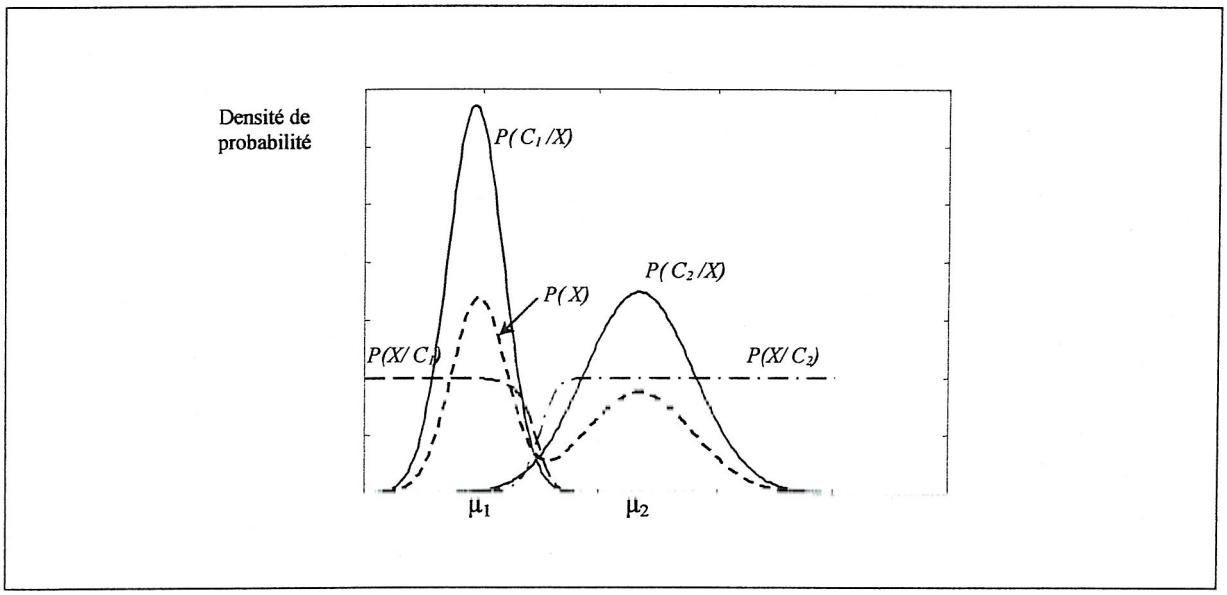


Figure (I.6) : Densité de probabilité de la première caractéristique (largeur sur longueur pétale)

Erreur de classification

Dans le cas bi-classe l'erreur associée à la règle de décision précédente est :

$$P(erreur) \begin{cases} P(C_1/X) & \text{Si on choisit } C_2 \quad \text{(I.10.a)} \\ P(C_2/X) & \text{Si on choisit } C_1 \quad \text{(I.10.b)} \end{cases}$$

Dans le cas unidimensionnel à deux classes, la probabilité d'erreur sera :

$$P(erreur) = \int_{R2} P(X/C_1)P(C_1) + \int_{R1} P(X/C_2)P(C_2) \qquad \text{(I.11)}$$

Où R1 et R2 sont les régions de décision correspondantes aux classes C_1 et C_2 (figure I.7). La probabilité d'erreur dépende du choix du seuil de décision t, et elle est minimale pour le choix de $t=t_0$ ce qui correspond à prendre la règle suivante, qui n'est que la règle de Bayes, comme règle de décision :

$$X \in C_1 \quad \text{Si } P(C_1/X)P(C_1) > P(C_2/X)P(C_2) \qquad \text{(I.12.a)}$$
$$X \in C_2 \quad \text{Si } P(C_2/X)P(C_2) > P(C_1/X)P(C_1) \qquad \text{(I.12.b)}$$

Dans le cas multi-classes :

$$X \in C_i \text{ Si } P(C_i/X)P(C_i) = Max_{k=1,2,\ldots K}\{P(C_k/X)P(C_k)\} \qquad \text{(I.13)}$$

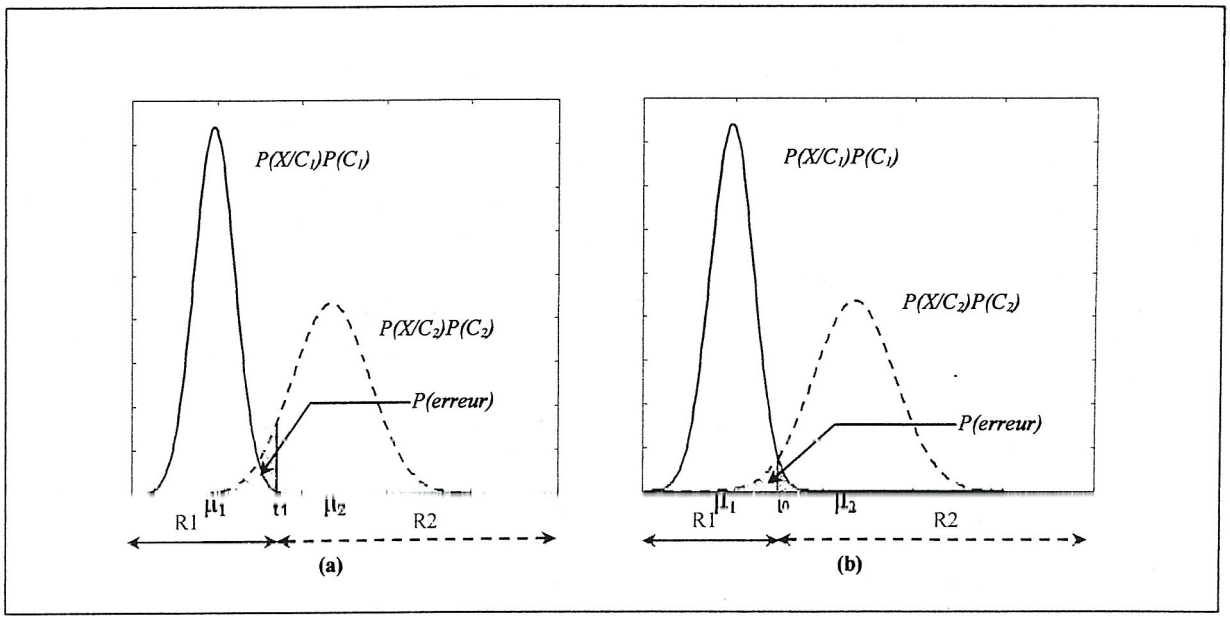


Figure (I.7) : Effet du choix de seuil sur la probabilité d'erreur

I.5.2 Méthode des k-plus proches voisins :

La méthode des plus proches voisins est une ancienne technique qui consiste à assigner l'exemple à classifier, à la classe dont appartient l'exemple le plus proche, après avoir déterminer les distances entre cet exemple et chacun des exemples disponibles. Soit un ensemble de *(Q)* exemples, appartenant aux *(K)* classes $\{C_1, C_2, \ldots C_K\}$, représentés par leur vecteurs caractéristiques $\{X^{(1)}, X^{(2)}, \ldots X^{(Q)}\}$, pour classifier un nouvel exemple représenté par le vecteur X, il s'agit de déterminer les distances d_q entre X et chaque vecteur $X^{(q)}$ (figure I.8), données par :

$$d_q = \left\| X - X^{(q)} \right\| \qquad \text{(I.14)}$$

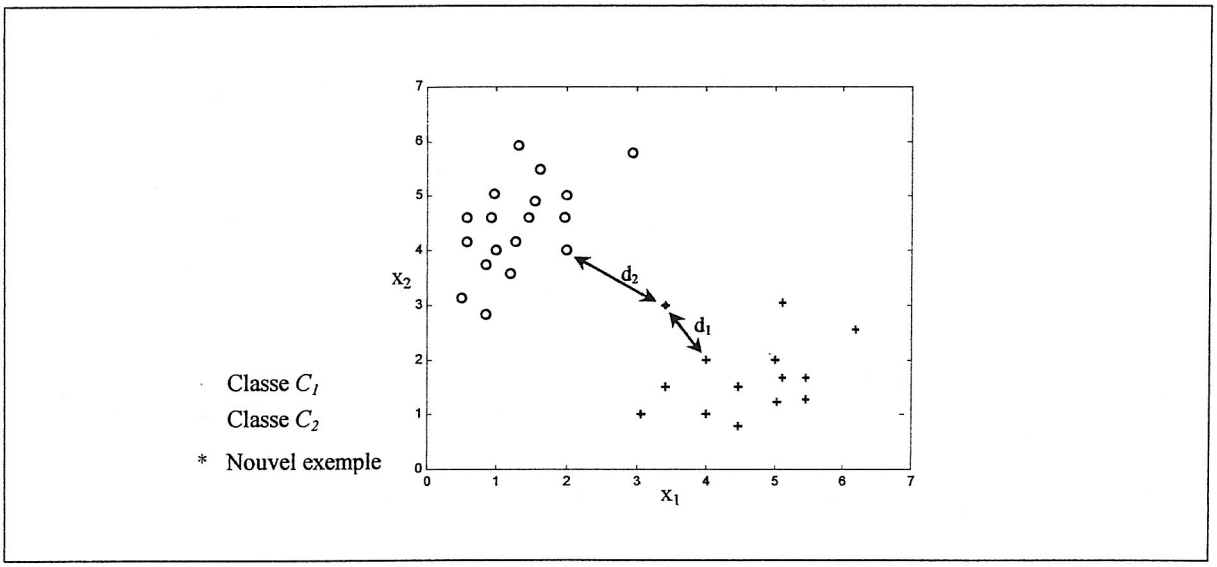


Figure (I.8) : Classification par la méthode des plus proches voisins

En suite d'assigner à cet exemple la classe C_m de l'exemple (m) correspondant à la distance la plus petite (d_m), les fonctions discriminantes sont alors :

$$\phi_m = \min(d_i)\, Pour\ tout\ X^{(i)} \in C_m \tag{I.15}$$

L'inconvénient de cette méthode, c'est que le plus proche exemple peut être de la classe incorrecte alors que la majorité de ces voisins ne le sont pas. Ainsi, pour éviter ce problème, on assigne à cet exemple la classe la plus représentée par les proches voisins d'ou le nom méthode des k-plus proches voisins, donc la classification sera par la détermination de la somme minimum des distances. Les fonctions discriminantes seront :

$$\phi_m = \min\left(\sum_{i=1}^{k} d_i\right), X_i \in C_m \tag{I.16}$$

1.5.3 Classificateur euclidien

C'est le classificateur statistique le plus simple, il consiste à assigner, à l'exemple à classifier, la classe dont son vecteur moyen est le plus proche du vecteur caractéristique de cet exemple au sens de la distance euclidienne. Ainsi les fonctions discriminantes sont données par :

$$\phi_i(X) = -\frac{1}{2}(X - M_i)'(X - M_i) \tag{I.17}$$

Ou M_i étant le vecteur moyen de la classe C_i.

L'application de ce classificateur pour la classification de la base de donnée Iris (les exemples de la première et la deuxième classe, représentés par deux caractéristiques), donne la frontière de décision représentée sur la figure (I.9) :

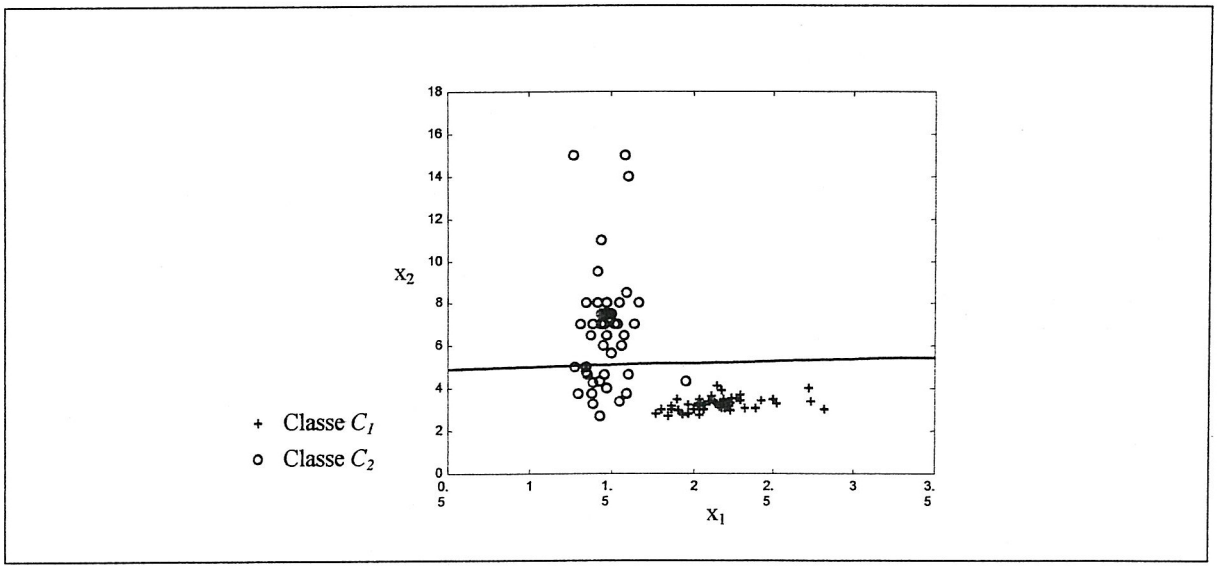


Figure (I.9) : Classification de la base de données Iris par le classificateur euclidien

1.5.4 Classificateur Gaussien :

Basé sur la supposition que les exemples de chaque classe ont une distribution Gaussienne multi variable, le classificateur gaussien consiste à établir des fonctions discriminantes estimées à partir des caractéristiques des exemples présents. Les fonctions discriminantes sont de la forme :

$$\phi_i(X) = -\frac{1}{2}(X - M_i)' \Sigma^{-1}(X - M_i) - \frac{1}{2}\|\Sigma\| + \ln(P(C_i)) \qquad \text{(I.18)}$$

Ou $P(C_i)$ est la probabilité à priori d'appartenance de la classe C_i et *ln* est le logarithme népérien. Le résultat de l'application de ce classificateur, à la base de données précédente, est représenté sur la figure (I.10), les probabilités à priori $P(C_1)$ et $P(C_2)$ sont prises égale à *« 0.5 »* :

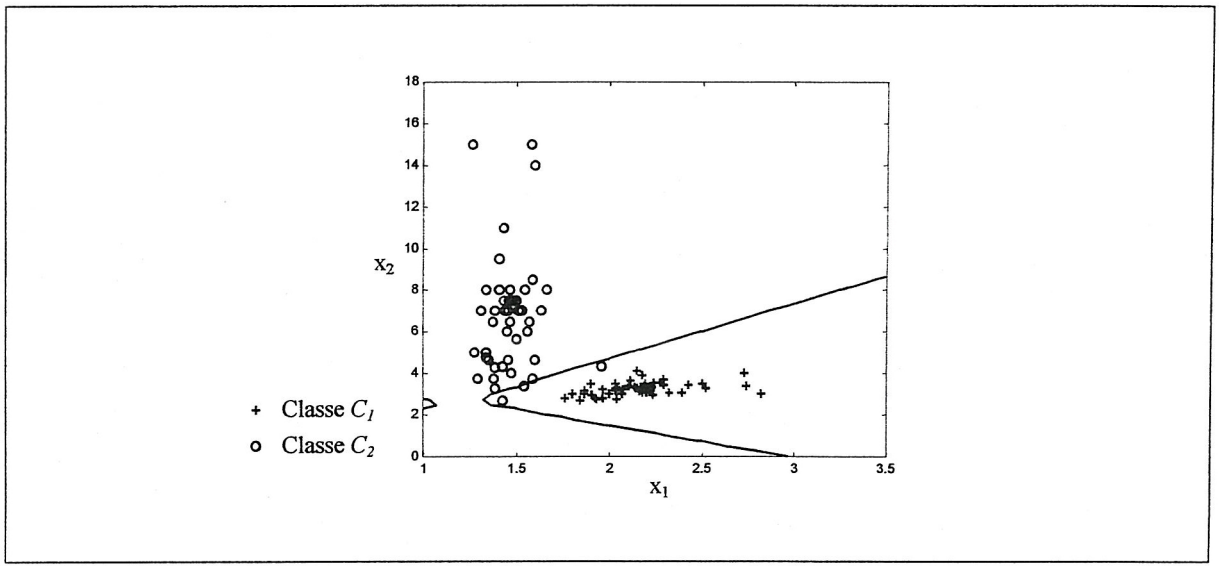


Figure (I.10) : Classification de la base de données Iris par le classificateur gaussien

Chapitre II :

Classification Par les Réseaux de Neurones

II.1 INTRODUCTION

La modélisation des réseaux de neurones, dont leurs structures et leurs fonctionnements tentent d'imiter les neurones du cerveau humain, est un exemple qui montre l'intérêt d'inspirations du fonctionnement des êtres biologiques. Les réseaux de neurones artificiels (ANN, Artificiel Neural Network) sont inspirés des principes selon lesquelles le système nerveux traite les informations d'où ils peuvent être considérés comme étant un modèle mathématique simplifié de celui là.

Contrairement aux calculateurs conventionnels qui ne peuvent être programmés que pour effectuer des taches précises, les ANNs peuvent apprendre de nouvelles fonctionnalités, ils présentent l'avantage d'auto-adaptation en se performant d'effectuer des taches de classification, de contrôle et de prédiction par l'ajustement de leurs interconnections, ainsi ils s'adaptent bien avec les systèmes dynamique.

Ce chapitre est devisé en deux parties : La première, constitue une introduction aux concepts et structures des réseaux de neurones artificiels. Dans laquelle, ils seront présentés les différents types des ANNs, leur apprentissage et leurs applications. La deuxième partie, quant à elle, est consacrée à la classification par le Perceptron Multi Couche (MLP, Multi Layered Perceptron), le type le plus utilisé des ANNs. Dans cette partie, en présente les différentes étapes pour l'élaboration d'un algorithme d'apprentissage du MLP. en introduisant des différentes techniques de stabilisation et d'accélération de la convergence.

Première Partie :

LES RÉSEAUX DE NEURONES ARTIFICIELS

II.2 APPLICATION DES RESEAUX DE NEURONES

Les ANNs sont destinés à être appliqués dans la reconnaissance des formes où ils sont utilisés dans plusieurs applications, parmi lesquelles : le réseau de Shea et Lin [9] qui détecte les explosives dans les bagages des aéroports et le réseau de Cun et Al [10] qui lit automatiquement les codes postaux de la poste américaine.

Il faut également signaler leurs applications dans de nombreux domaines : Dans la finance pour la prévision du marché, dans la télécommunication pour l'analyse des signaux et l'élimination du bruit, dans l'industrie pour le contrôle des systèmes et les diagnostiques des pannes, dans la médecine pour les diagnostiques médicaux...

II.3 LES NEURONES BIOLOGIQUES

II.3.1 Présentation

Les neurones biologiques sont des cellules spécialisées dans la transmission du message nerveux. Le cerveau humain contient plusieurs milliards de neurones interconnectés, chacun d'eux est connecté à environ dix mille d'autres neurones. Le neurone biologique (figure II.1) est composé de quatre parties :

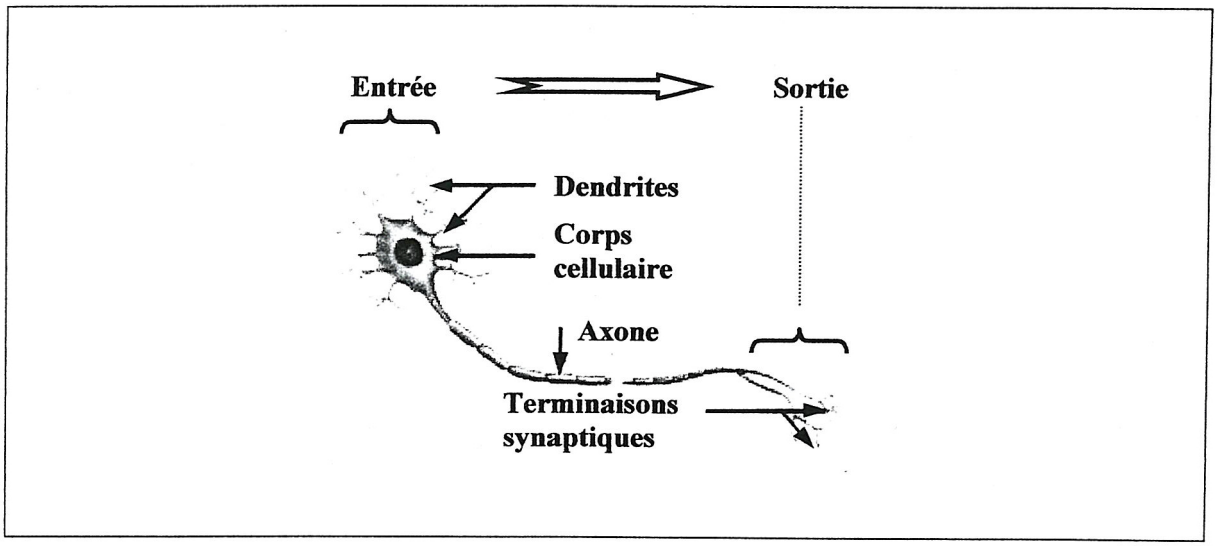


Figure II.1 : Le neurone biologique

Le corps cellulaire : il contient le noyau du neurone, sa taille est de quelques microns de diamètre, c'est le centre de l'influx nerveux qui représente l'état d'activité du neurone.

Les dendrites : ce sont les entrées principales du neurone, ils captent les signaux envoyés vers lui, leur taille est de quelques dizaines de microns de longueur

L'axone : Une cellule nerveuse ne comporte généralement qu'un seul axone, c'est la sortie du neurone, qui conduit l'influx nerveux se forme de potentiel d'action jusqu'aux neurones suivants. Il se termine par une synapse qui transmet chimiquement le message électrique aux dendrites du neurone prochain. L'axone est plus long que les dendrites, il se ramifie à son extrémité ou il se connecte aux autres neurones, sa taille peut varier de quelques millimètres à plusieurs mètres.

La synapse : c'est une jonction entre deux neurones, elle est essentielle dans le fonctionnement du système nerveux

II.3.2 Fonctionnement

Pour chaque neurone, la transmission de l'information est dans un seul sens : des dendrites vers les axones. Il reçoit les informations provenant d'un ou plusieurs d'autres neurones, sous forme d'un signal électrique, par ces dendrites. Si la somme de ces signaux est excitatrice le neurone émet à son tour un signal électrique qui se propage par les axones aux connecteurs terminaux, et ensuite aux dendrites des autres neurones. Selon Hebb [11] : L'apprentissage des neurones se fait par la modification des résistances électriques des connections dendrites.

II.4 LE NEURONE FORMEL

Les premiers travaux sur les ANN sont dus aux efforts de McCulloch et Pitts [12], bio-physiciens de l'université de Chicago, ils ont proposé un modèle simple des neurones formels (figure II.2).

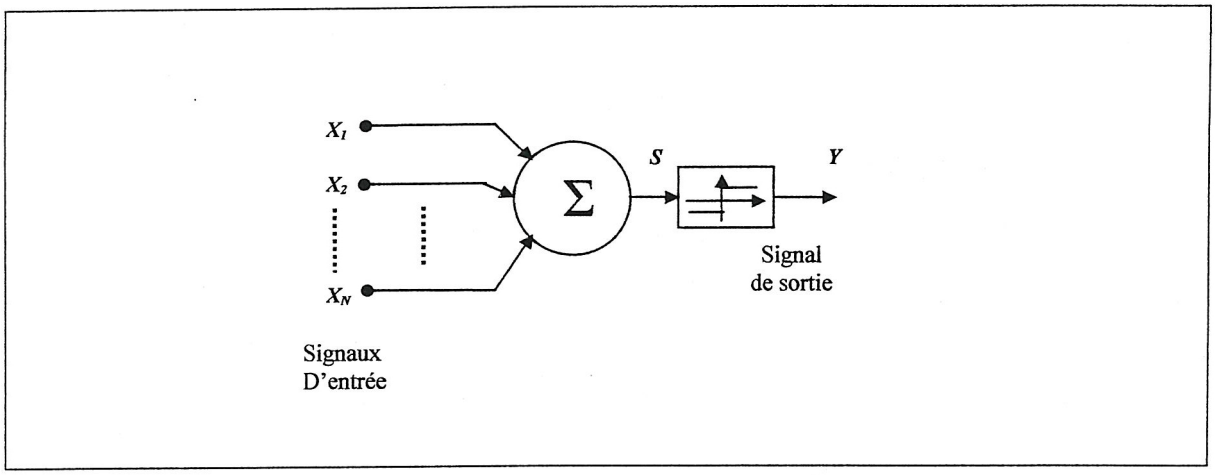


Figure II.2 : Le Neurone formel (modèle de McCulloch et Pitts)

Ce neurone formel est un modèle mathématique qui réalise une somme pondérée des signaux qui lui parviennent, il est activé si cette somme est positive. Ce modèle n'a pas possédé une règle d'apprentissage jusqu'à 1949 ou Hebb [11] a proposé un principe d'apprentissage sans avoir donner des équations (figure II.3).

Règle de Hebb [11] **:** Si deux neurones d'une part et d'une autre d'une synapse sont activés de façon synchrone et répétée, la connexion synaptique sera renforcée. La modification donc d'un poids ne dépend que des neurones connectés à ces extrémités.

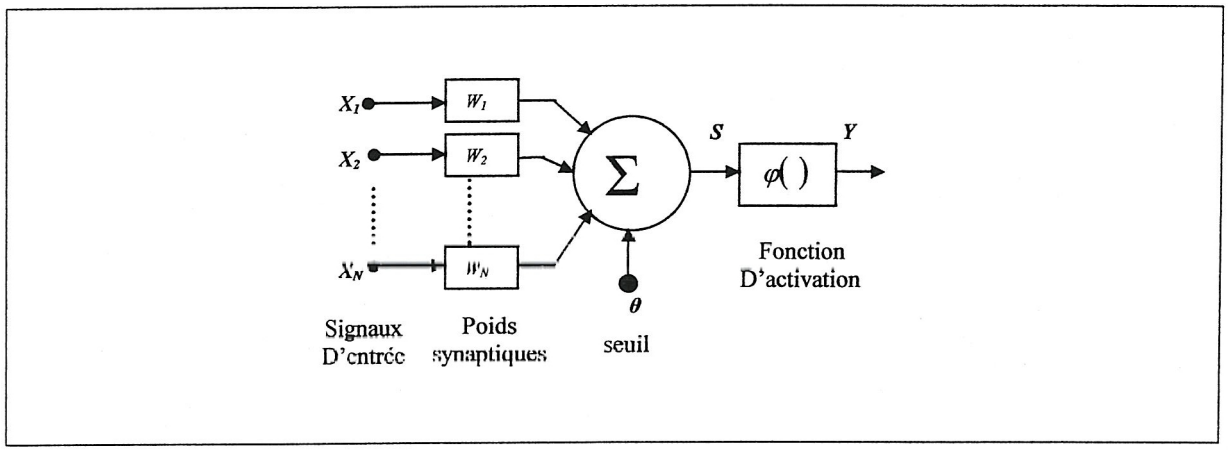


Figure (II.3) : Neurone formel avec apprentissage Hebbien

Dans le modèle original de McCulloch et Pitts, la fonction d'activation était le seuil de Heaviside, mais il existe plusieurs d'autres types de fonctions utilisées comme fonctions d'activation, quelques-unes sont représentées sur la figure (II.4), sa sortie sera donc donnée par :

$$Y=\varphi\left(\sum_{i=1}^{N} w_i\, x_i - \theta\right) \qquad \text{(II.1)}$$

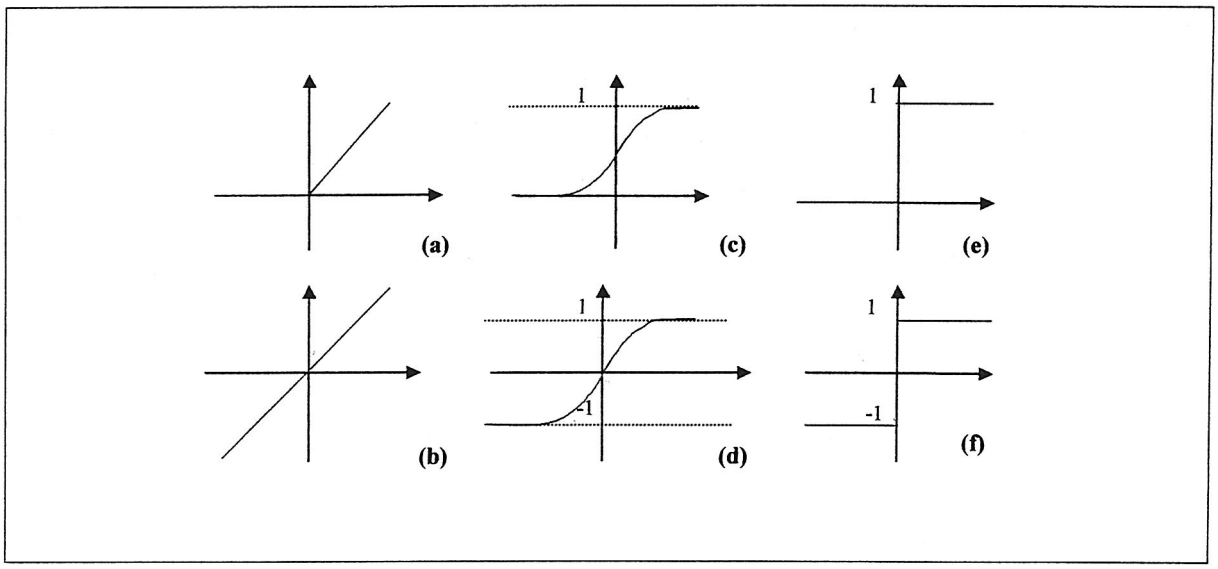


Figure II.4 : Quelques types de fonctions d'activation
(a) : Fonction linéaire positive (b) : Fonction linéaire
(c) : Sigmoïde unipolaire (d) : sigmoïde bipolaire
(e) : Seuil de Heaviside (f) : fonction signe

II.5 ARCHITECTURE DES ANNS

Un réseau de neurones artificiels est un ensemble de neurones formels associés en couches, et fonctionnant en parallèle. On distingue deux types d'architecture : les ANNs non bouclés FANN, Feed Forward Neural Network) et les ANNs bouclés (Back Forward Neural Network)

Les réseaux non bouclés (FANN) : Ce sont des réseaux dont l'information se propage de l'entrée vers la sortie sans retour, c'est le type le plus utilisé dans la classification, l'approximation des fonctions et la modélisation des procédés, parmi ces réseaux : i) le perceptron : Qui est le premier et le plus simple des ANNs. ii) le Perceptron Multi-Couches : C'est un réseau ayant une ou plusieurs couches cachées. iii) les réseaux RBF : Ils ont la même architecture que le PMC mais avec des fonctions du type Gaussien.

Les réseaux bouclés (back-forward) : Appelés aussi les réseaux récurrents, ces ANNs comprennent des boucles ramenant la valeur d'une ou de plusieurs sorties vers l'entrée avec un retard, ils peuvent alors être considérer comme des systèmes dynamiques, parmi ces réseaux : i) les cartes topographiques de Kohénen : Ils suivent un apprentissage non supervisé, établissant une carte directe topologique en fonction d'ensemble d'entrée. ii) le réseau de Hopfield : Il suit aussi un apprentissage non supervisé, c'est un réseau dont chaque neurone est connecté à tous les autres neurones. Ces réseaux seront présentés dans les paragraphes suivants

II.6 LES RÉSEAUX NON BOUCLES (FANN) :

II.6.1 Le perceptron :

II.6.1.1 Présentation :

Le Perceptron, créé par Rosenblatt en 1958 [13], est l'un des plus simples classificateurs neuronaux. Il comporte une seule couche de neurones qui reçoivent en entrée N valeurs $x_1, x_2, \ldots x_n$ et calcule une sortie Y (figure II.5). Un perceptron est défini par les poids synaptiques et un seuil (ou biais).

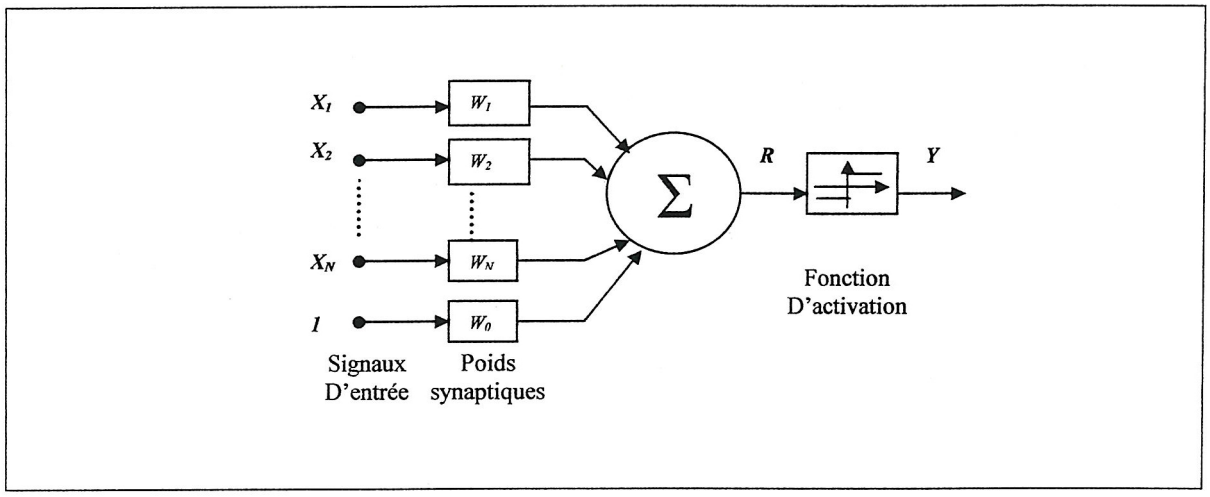


Figure II.5 : Le Perceptron

Les variables d'entrée peuvent être booliennes ou réelles, les poids peuvent être entiers ou réels. Sa fonction d'activation dans le modèle originale est la fonction Heaviside, alors sa sortie prend ces valeurs dans *{0,1}*. Le modèle le plus utilisé est celui dont la sortie prend ces valeurs dans *{-1,1}*, il suffit donc de remplacer la fonction Heaviside par une fonction bipolaire comme la fonction signe.

La sortie *Y* du perceptron est donnée par :

$$Y=h(R) \tag{II.2}$$

Ou h est la fonction d'activation, *R* est donnée par :

$$R = \sum_{n=1}^{N} x_n w_n + w_0 \tag{II.3}$$

II.6.1.2 Classification par le perceptron :

Le perceptron, et comme son nom l'indique, est désigné à être appliqué à la perception des formes, il divise l'espace caractéristique en deux sous-espaces délimités par un hyperplan. En effet la sortie du perceptron est $Y = 1$ si $R \geq 0$, sinon $Y = -1$, donc l'ensemble des vecteurs d'entrée dont $R = w_1x_1 + w_1x_1 + ... + w_Nx_N - w_0 = 0$ forme un hyperplan H dans l'espace caractéristique, H constitue une partition de ce dernier en demi-espace gauche H^+ et demi-espace droit H^- comme suit :

$$Si\ w_1x_1 + w_1x_1 + ... + w_Nx_N > w_0\ \ Alors\ \ X \in H^+$$
$$Si\ w_1x_1 + w_1x_1 + ... + w_Nx_N < w_0\ \ Alors\ \ X \in H^-$$

La figure (II.6) montre un exemple d'un espace caractéristique bidimensionnel divisé en deux sous-espace H^+ et H^- par une droite (constituant l'hyperplan dans le cas bidimensionnel).

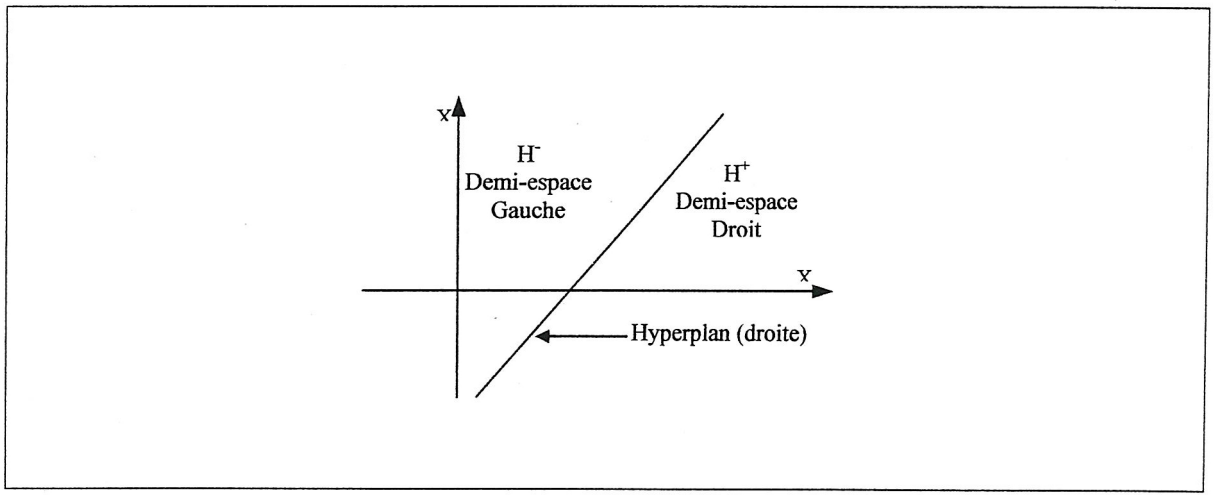


Figure (II.6) : Espace caractéristique divisé en deux sous-espaces

La règle d'apprentissage du perceptron [13] consiste à ajuster chacun de ces poids w_n à l'itération $(i+1)$ comme suit :

$$w^{(i+1)} = w^{(i)} + \eta\left(t^{(q)} - y^{(q)}\right)x^{(q)} \qquad \text{(II.4)}$$

Ou $x^{(q)}$ est l'entrée du réseau, $t^{(q)}$ est sa sortie désirée et η est le taux d'apprentissage (une constante positive choisie). Donc si le perceptron fourni la sortie désirée, il n'aura pas de changement de poids, sinon il aura selon l'équation précédente, et ainsi de suite jusqu'à ce que le perceptron arrive à classifier tous les exemples correctement. Le choix de η est critique, s'il est choisi avec une petite valeur l'apprentissage sera lent, et s'il est choisi avec une grande valeur il aura un risque de divergence de l'algorithme d'apprentissage [1].

II.6.1.3 Classification par une couche de perceptrons :

Dans un réseau constitué d'une couche de perceptrons (figure II.7), chaque perceptron sépare l'espace caractéristique en deux sous-espaces, alors si ce réseau est constitué de M perceptrons, il divise l'espace caractéristique en $2M$ sous-espaces.

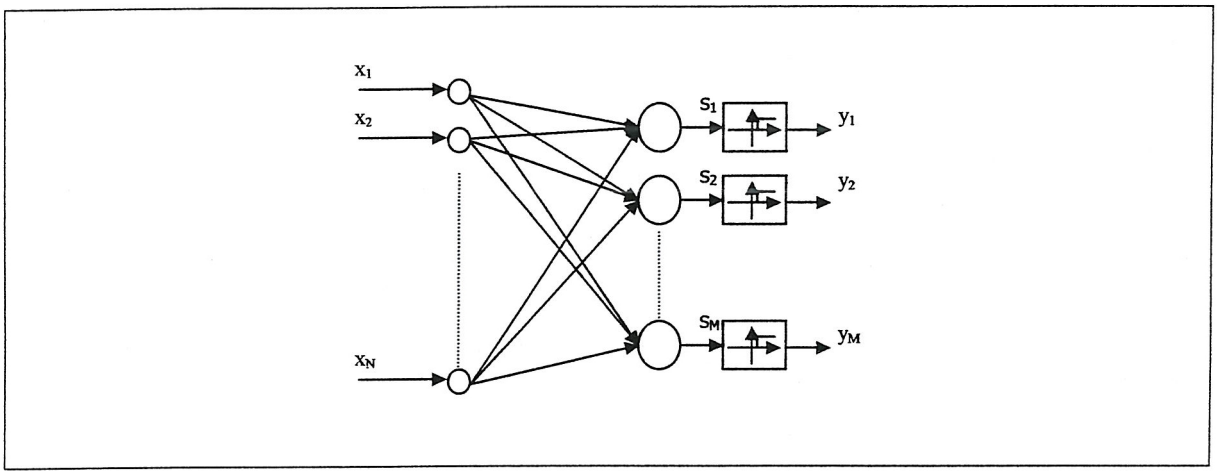
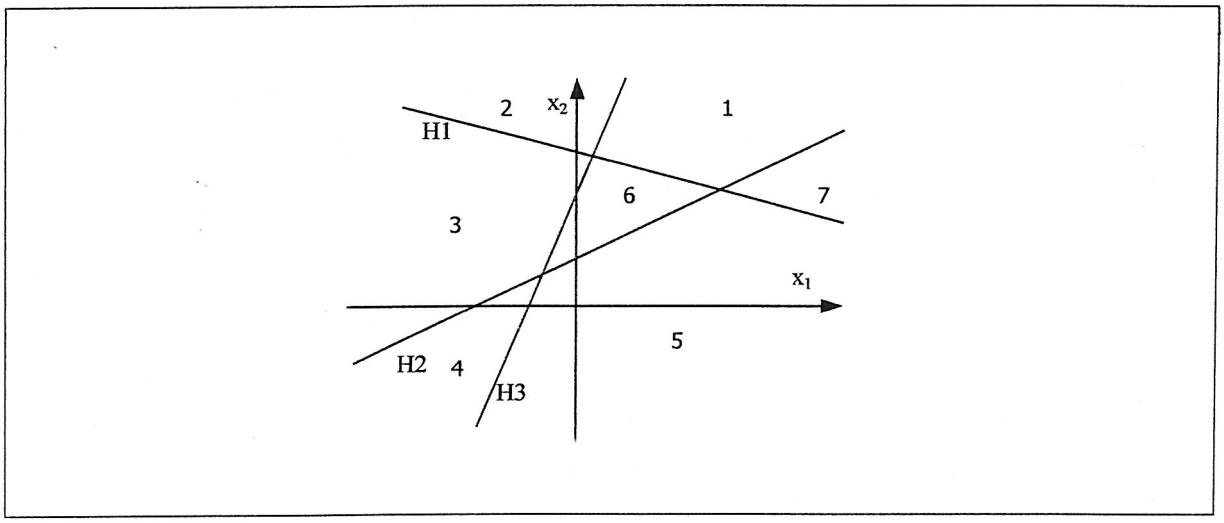


Figure (II.7) : Un réseau avec une couche de Perceptrons

Dans une telle partition chaque sous-espace est convexe et chaque intersection de 2M sous-espaces forme une région convexe [1], le nombre maximum possible de ces régions est $k_1 = 2^M$, et le nombre minimum possible est $k_2 = M+1$. Sur la figure (II.8), est illustrée une partition d'espace caractéristique bidimensionnel, par un réseau comportant trois neurones, en sept régions convexes.



Figure(II.8) : Séparation d'un espace caractéristique par un réseau de trois (3) perceptrons

II.6.2 L'ADALINE

II.6.2.1 Présentation

Proposée par Widrow [14], L'ADALINE (i.e. ADAptative LInear Element) est un modèle de neurones fondé sur le travail de Mcculloch et Pitts, il a une fonction d'activation linéaire (figure II.9), sa sortie est donnée par :

$$Y=R=\sum_{i=1}^{N} w_i\, x_i \qquad \text{(II.5)}$$

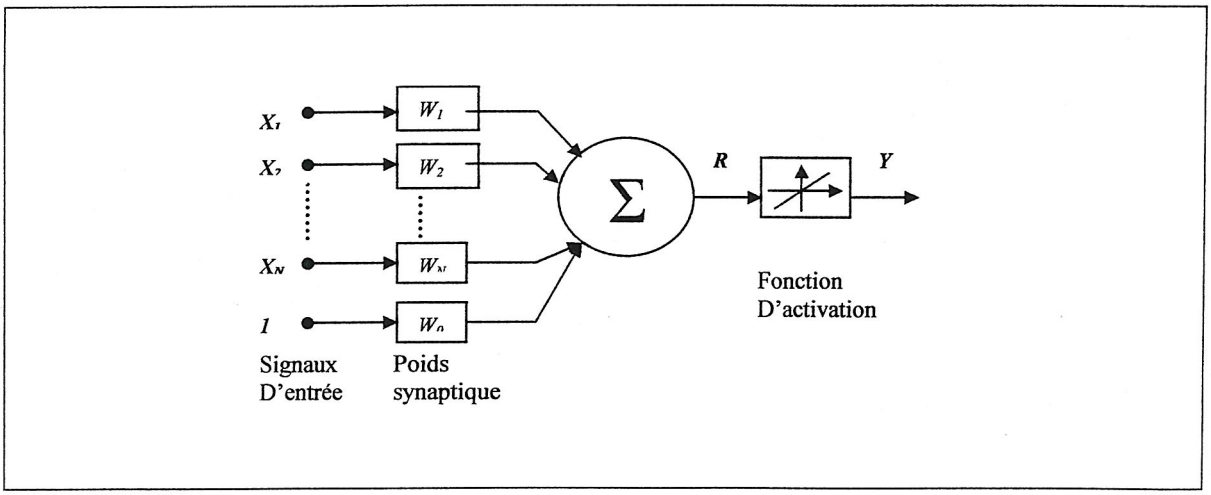


Figure (II.9) : L'ADALINE

L'apprentissage de l'ADALINE est effectué par la règle de Widrow[14] en minimisant l'erreur quadratique *(E)* donnée par :

$$E = \sum_{q=1}^{Q} \left(t^{(q)} - y^{(q)}\right)^2 \qquad \text{(II.6)}$$

la règle de Widrow est une technique de descente de gradient effectuant la mise à jour de chaque poids w à l'itération $(i+1)$ comme suit :

$$w^{(i+1)} = w^{(i)} - \eta \nabla E$$
$$= w^{(i)} + \eta \sum_{q=1}^{Q} \left(t^{(q)} - y^{(q)}\right) x^{(q)} \qquad \text{(II.7)}$$

Donc l'apprentissage de l'ADALINE consiste à ajuster ces poids jusqu'à obtenir une erreur assez faible entre la sortie désirée et la sortie calculée, c'est-à-dire contrairement au perceptron ou l'apprentissage s'acheva quand tous les exemples seront classés correctement. La classification avec une couche d'ADALINEs est semblable à celle d'une couche de perceptrons, chaque neurone divise l'espace caractéristique en deux demi-espaces.

II.6.3 Exemple de classification (comparaison entre le perceptron et L'ADALINE)

Pour comparer les performances de classification du perceptron et l'ADALINE, soit à classifier l'ensemble des exemples qui appartiennent à la classe 1 (sétosas) et la classe 2 (versicolores) de la base de données Iris. Chaque exemple étant représenté par deux caractéristiques : Largeur sépale / longueur sépale et largeur pétale / longueur pétale. Nous effectuons la classification avec un perceptron ensuite avec l'ADALINE, chacun d'eux ayant deux entrées et par conséquent deux poids à ajuster en plus d'un seuil.

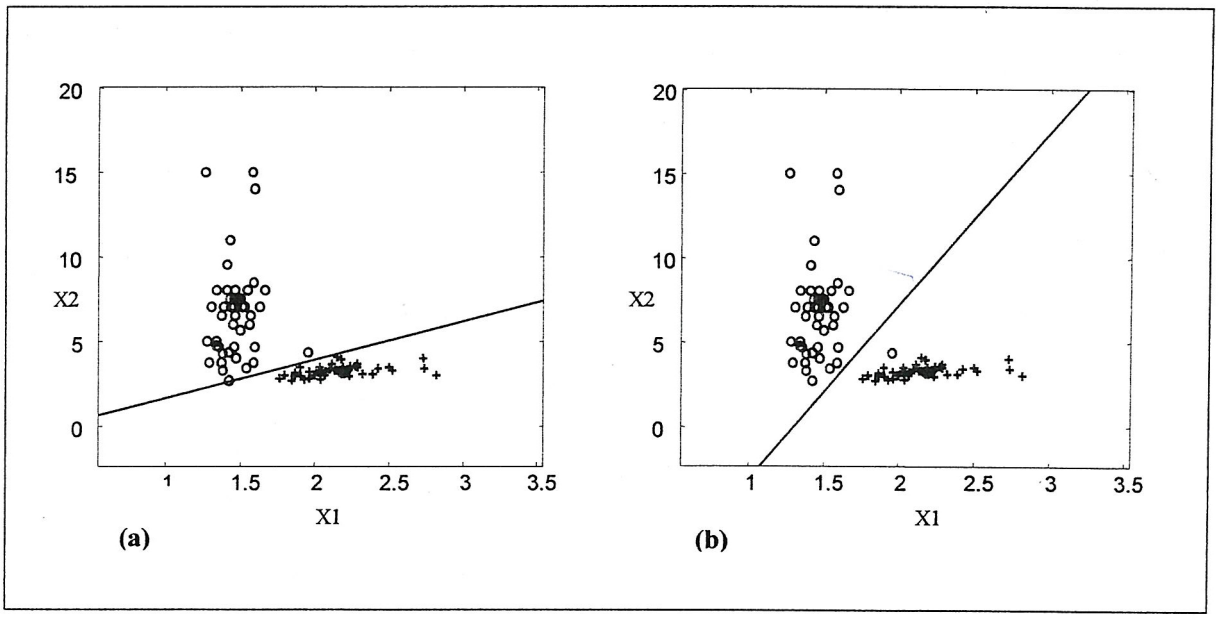


Figure (II.10) : Frontière de la classification de la base de données Iris
(a) : Classification par la Perceptron
(b) : Classification par l'ADALINE

D'après la figure (II.10), on constate que la frontière de décision de l'ADALINE sépare l'espace caractéristique d'une manière plus performante que celle du perceptron, elle est plus généralisable vu que la frontière de décision formée accorde la possibilité de mieux classifier de nouveaux exemples. En effet, cette différence est due de fait que l'ADALINE effectue la classification en minimisant une erreur quadratique de tous les exemples contrairement au perceptron qui opte à classifier tous les exemples correctement.

II.6.4 Limitation

Exemple de Minsiky-Paper : Le fameux exemple de Minsiky-Paper en 1960 [15] a diminué l'intérêt du perceptron et des autres ANN pendant deux décades [1], il montre l'échec du perceptron à être capable d'apprendre le XOR logique. Soit l'ensemble de quatre exemples caractérisant l'exemple du XOR logique, leurs vecteurs caractéristiques et leurs vecteurs de sortie correspondants sont représentés sur le tableau :

Exemple	Vecteur caractéristique	Sortie désirée
1	(0,0)	0
2	(0,1)	1
3	(1,0)	1
4	(1,1)	0

Tableau (II.1) : Données du XOR logique

Pour l'apprentissage nous utilisons un perceptron ayant deux entrées et une sortie (donc $N = 2$; $M = 1$; $Q = 4$; $K = 2$), nous pouvons constater et comme le montre la figure (II.11) qu'il est impossible de séparer les deux classes par le perceptron. D'ailleurs l'espace caractéristique du XOR logique ne peut pas être séparé par une seule droite quelle que soit son orientation.

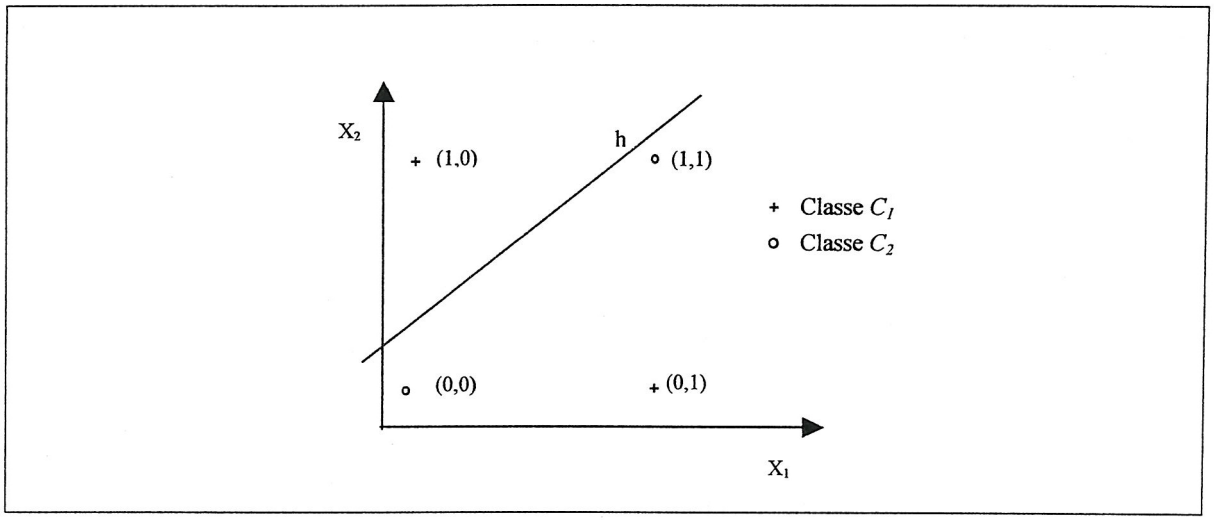


Figure (II.11) : Classification du XOR logique

II.6.5 Réseaux de fonctions à base radiale :

II.6.5.1 Présentation :

Les réseaux de fonctions à base radiale (réseaux RBF) sont utilisés pour la première fois par Broomhead & Lowe [16], leur modélisation tente d'imiter le fonctionnement d'un certain neurone biologique à réponse de syntonisation locale, c'est à dire qui ne répond qu'à une partie de l'espace du signal d'entrée.

Fonction à base radiale : c'est une fonction qui fournie une sortie différente de zéro seulement pour des entrées qui se situent dans une région locale de l'espace des entrées (champ récepteur), la plus courante c'est la fonction gaussienne donnée par :

$$f(X)=\exp\left(\frac{\|X-v_i\|}{2\sigma_i^2}\right) \qquad \text{(II.8)}$$

Avec v_i : le centre du champ récepteur (ayant une sortie maximale) et σ_i : facteur d'adoucissement qui défini la taille du champ récepteur (figure II.12)

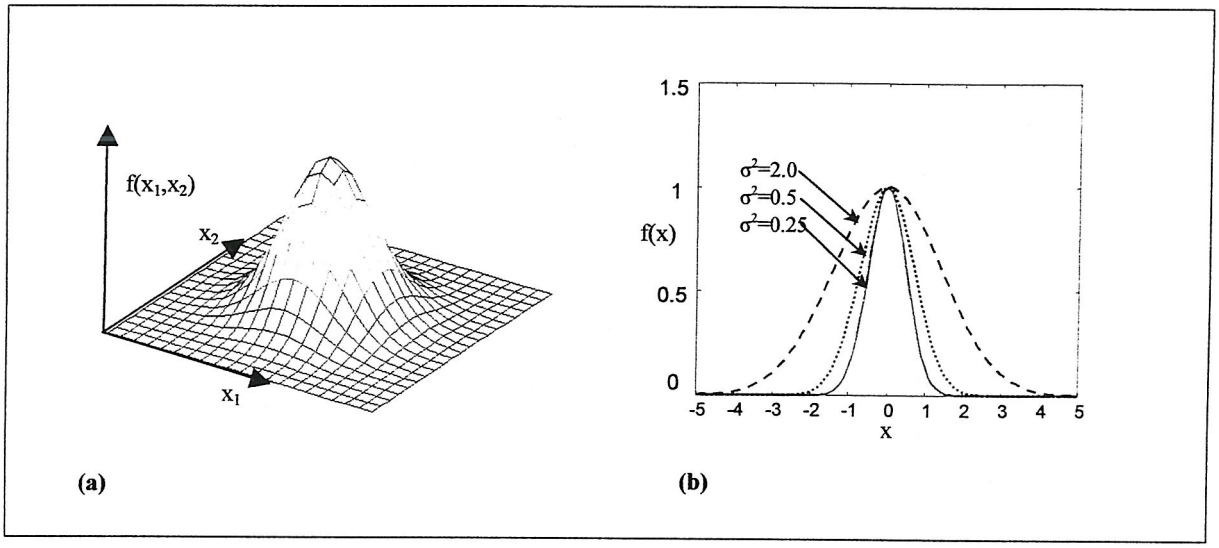


Figure (II.12) : (a) : Fonction RBF bidimensionnelle
(b) : Fonction RBF unidimensionnelle pour différentes valeurs de σ

Le neurone RBF : c'est un neurone qui réalise sa sortie en calculant la distance, entre chaque entrée x_n et son vecteur prototype v_i (code book vector), donnée par $\|X - v_i\|$. Sa sortie est d'autant plus forte que cette distance est plus petite, c'est à dire d'autant que le vecteur d'entrée est très semblable à son vecteur prototype.

Un réseau RBF, tel qu'il a été proposé par Broomhead & Lowe [16], se compose de trois couches (figure II.13) dont la couche cachée comporte « M » neurones RBF connectés

linéairement à la couche de sortie, seule ces connections sont ajustables. Les neurones de la couche de sortie ont des fonctions d'activations linéaires.

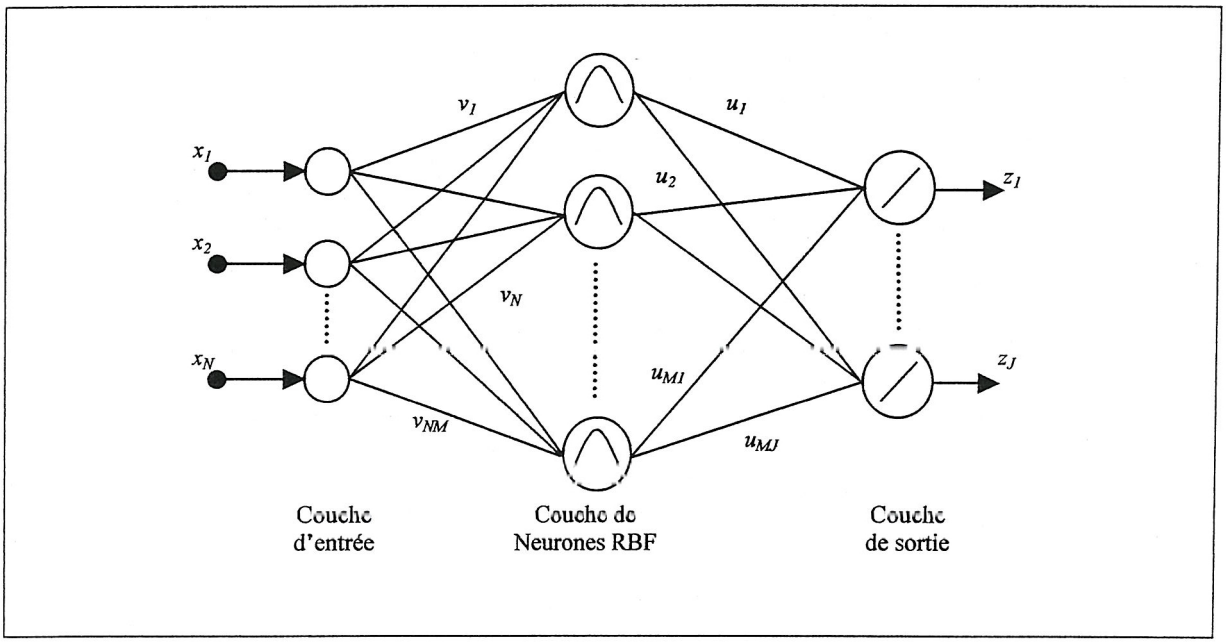


Figure (II.13) : Un réseau RBF

L'utilisation des réseaux RBF à la classification consiste, et comme tout classificateur, à déterminer la classe de l'exemple représenté à l'entrée du réseau par son vecteur caractéristique. Leur principe est de déterminer les distances entre chaque vecteur d'entrée présenté avec les vecteurs prototypes de chaque neurone caché, chaque classe étant représentée par un ou plusieurs vecteurs prototypes (figure II14), il s'agit ensuite de recoder la classe à laquelle appartient cet exemple par la couche de sortie, en trouvant le vecteur prototype le plus proche et en lui attribuant sa classe.

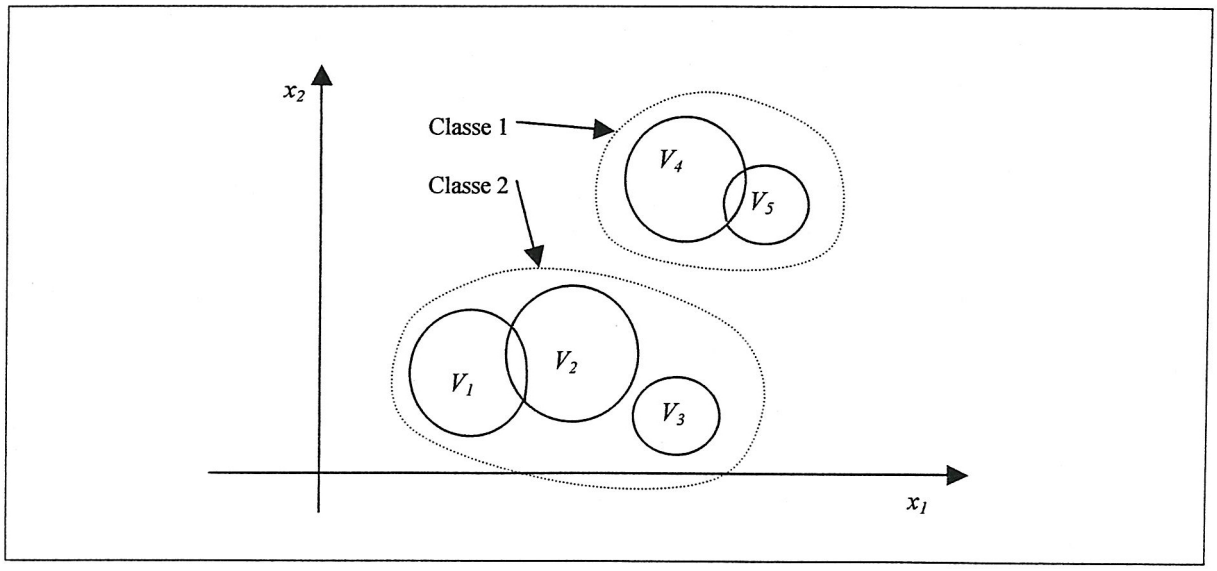


Figure (II.14) : Classification avec un réseau RBF en deux classes dont la première est représentée par 2 prototypes et la deuxième par 3 prototypes

II.6.6 Functional Link Neural Network (FNN)

Crée par Pao et al. [17], les FLNNs (figure II.15) sont réseaux avec une seule couche de neurones, et qui génèrent automatiquement des caractéristiques additionnelles aux vecteurs caractéristiques. En utilisant le tenseur, si les caractéristiques initiales sont x_1, x_2, …, x_N, les caractéristiques additionnelles seront donc :

$$x_{N+1} = x_1 x_2,\ x_{N+1} = x_1 x_3, \ldots$$

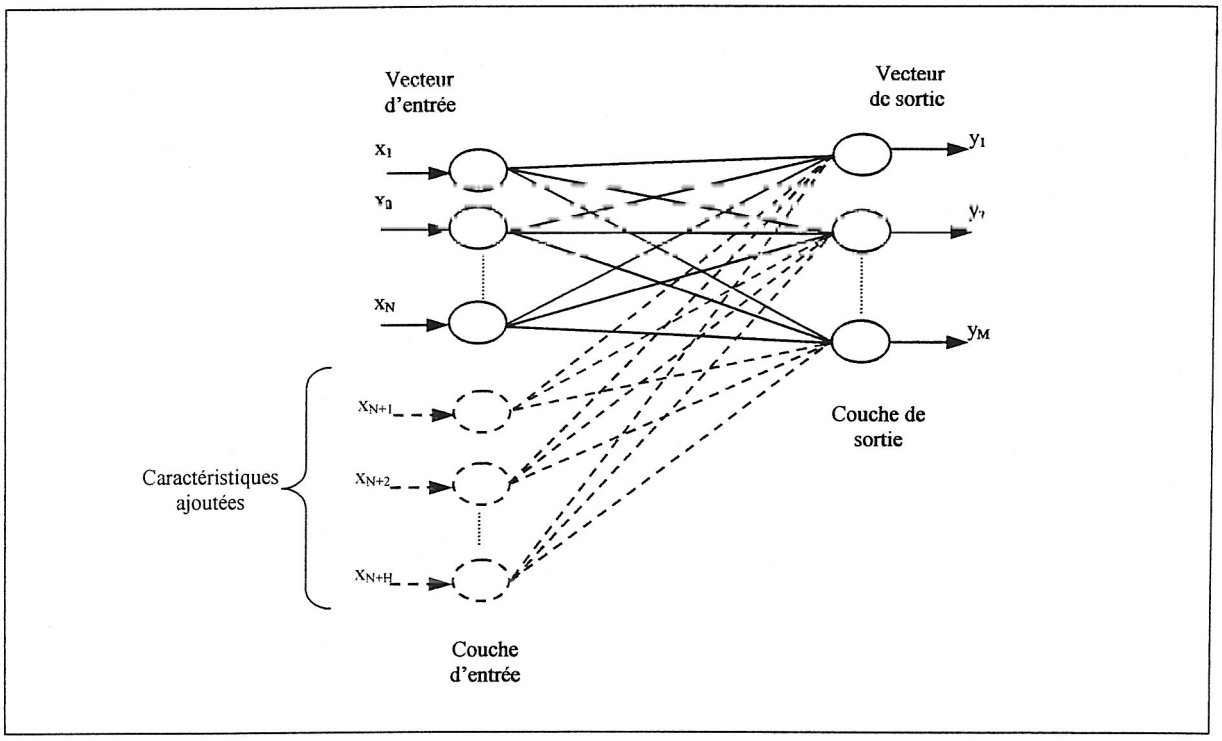


Figure (II.15) : Functional Link Neural Network

la sortie du m^{ème} neurone est donnée par :

$$y_m = f\left(\sum_{n=1}^{N+H} x_n w_{nm} \right) \qquad \text{(II.9)}$$

la mise à jour du poids w_{nm} à l'itération (i+1) est donnée comme suite :

$$w_{nm}^{(i+1)} = w_{nm}^{(i)} - \eta \frac{\partial E^{(i+1)}}{\partial w_{nm}} \qquad \text{(II.10)}$$

Où E est l'erreur quadratique totale donnée par :

$$E = \sum_{q=1}^{Q} \sum_{m=1}^{M} \left(t_m^{(q)} - y_m^{(q)} \right)^2 \qquad \text{(II.11)}$$

Par exemple, pour le cas du XOR logique (figure II.16.a) dont les deux classes sont non linéairement séparables, après l'ajout de la caractéristique additionnelle $x_3=x_1x_2$ (figure II.17.b) ces classes deviennent linéairement séparables par un plan, et par conséquent peuvent être classer par un réseau à couche.

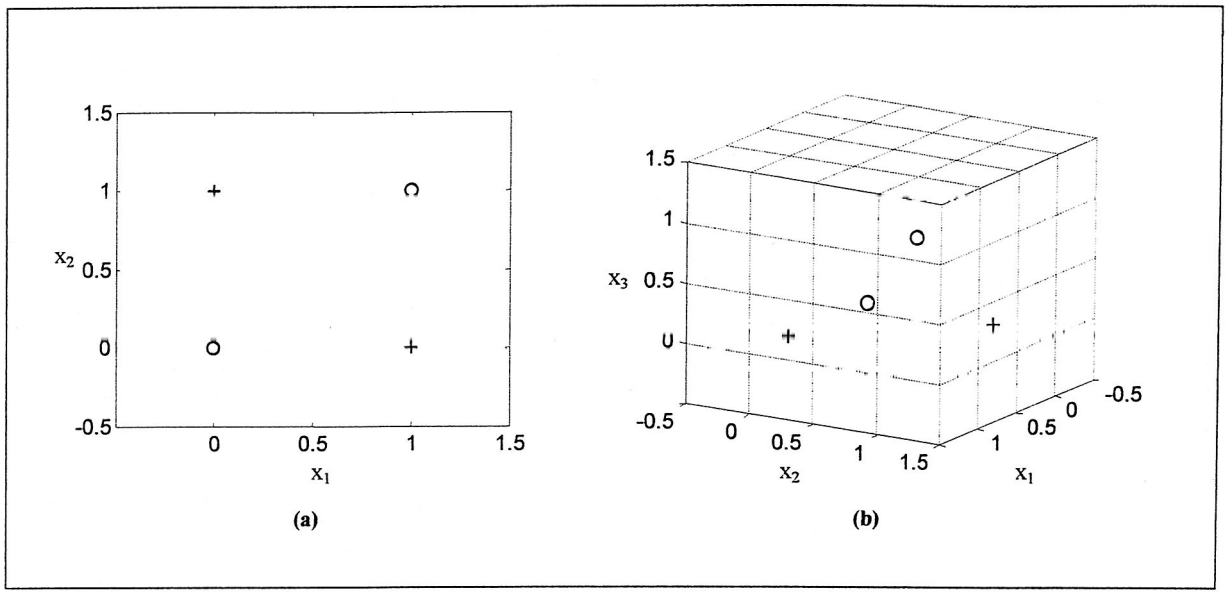


Figure (II.16) : Exemple du XOR logique
(a) : Données initiales
(b) : Données avec caractéristique additionnelle

II.6.7 Le RVFLNN

Le RVFLNN (Random Vector Functional Link Neural Network) est un modèle du perceptron développé par Pao et al. [18], c'est un agrandissement du PMC généralisé par Giles et Maxwell [19]. Le RVFLN est un réseau ayant les sorties de la couche cachée du PMC comme des entrées supplémentaires (figure II.17). Le réseau possède donc $N+M$ entrées, selon Pao les (M) entrées fournies par la couche cachée forme un ensemble d'entrées optimisées. Son algorithme d'apprentissage consiste à adapter les poids de sortie seulement d'où il présente l'avantage de simplicité et rapidité d'apprentissage par rapport au MLP.

La j^{eme} sortie du réseau est donnée par :

$$z_j = g(s_j) = g\left(\sum_{n=1}^{N} x_n v_{nj} + \sum_{m=1}^{M} y_m u_{mj} \right) \qquad \text{(II.12)}$$

La m^{eme} sortie de la couche cachée est donnée par :

$$y_m = h(r_m) = h\left(\sum_{n=1}^{N} x_n w_{nm} \right) \qquad \text{(II.13)}$$

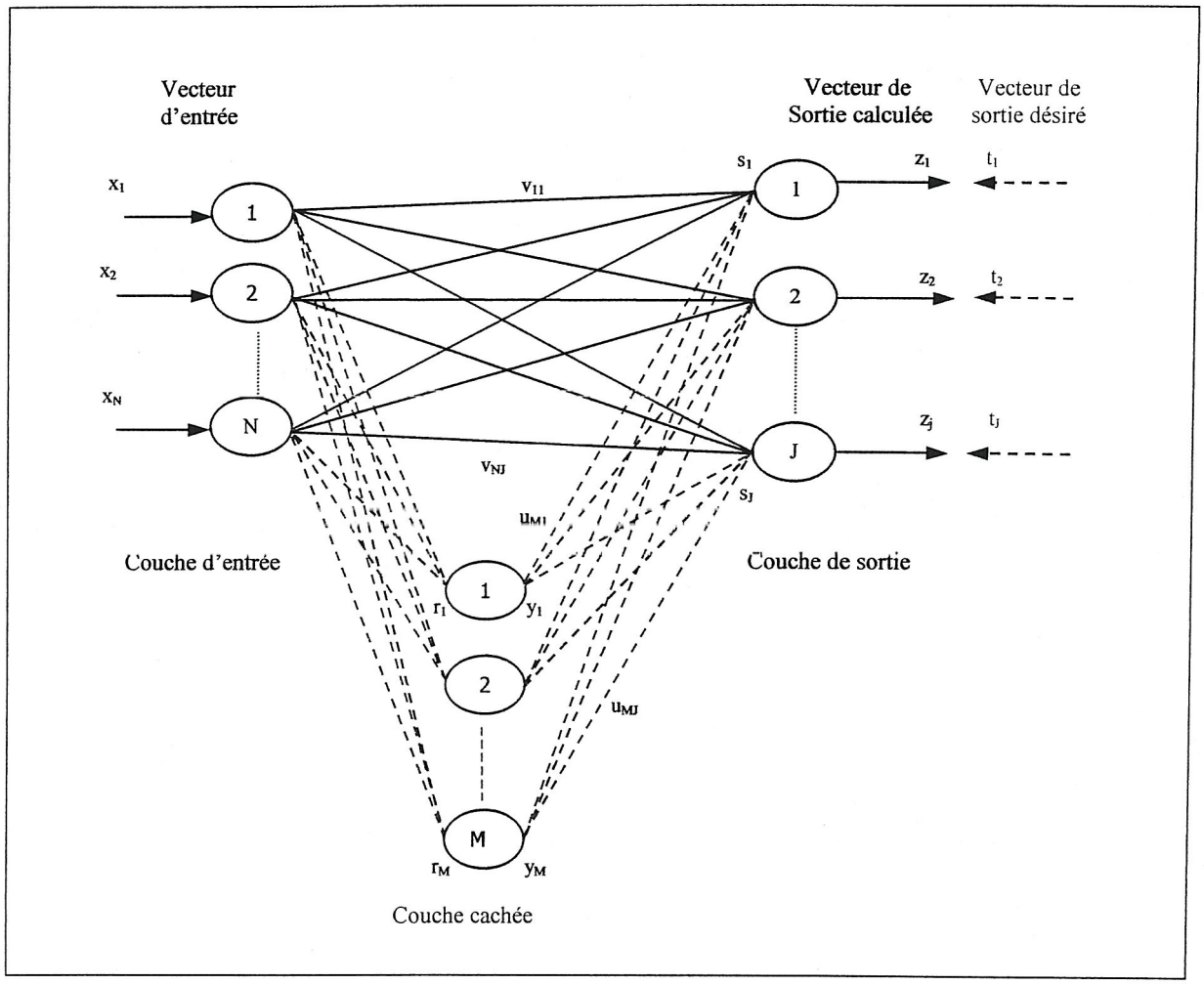


Figure (II.17) : Un réseau RVFLNN

L'apprentissage du RVFLNN est très simple et plus rapide que celui du PMC. La méthode d'apprentissage de Pao et al. est différente de la rétro-propagation, elle consiste à ajuster seulement les poids $\{v_{nj}\}$ et $\{u_{mj}\}$, les poids $\{w_{nm}\}$ sont non ajustables, mais ils doivent être sélectionner selon la contrainte suivante [1] : La sortie y_m de chaque neurone caché, n'est pas saturée. Pour assurer cette contrainte une solution est de sélectionner les poids $\{w_{nm}\}$ avec des petites valeurs positives et négatives. L'apprentissage du RVFLNN consiste à minimiser l'erreur quadratique totale (E) donnée par :

$$E = \sum_{q=1}^{Q} \sum_{j=1}^{J} \left(t_j^{(q)} - z_j^{(q)} \right)^2 \qquad \text{(II.14)}$$

L'ajustement des poids s'effectue de la façon suivante :

$$\Delta u_{mj}^{(i+1)} = -\eta \frac{\partial E^{(i)}}{\partial u_{mj}} \qquad \text{(II.15.a)}$$

$$\Delta v_{nj}^{(i+1)} = -\eta \frac{\partial E^{(i)}}{\partial v_{nj}} \qquad \text{(II.15.b)}$$

L'adaptation des poids à l'itération (i+1) sera donc comme suit :

$$u_{nm}^{(i+1)} = u_{nm}^{(i)} + \eta \cdot \sum_{q=1}^{Q} \left(t_j^{(q)} - z_j^{(q)}\right) g'\left(s_j\right) y_m \qquad \text{(II.16.a)}$$

$$v_{mj}^{(i+1)} = u_{mj}^{(i)} + \eta \cdot \sum_{q=1}^{Q} \left(t_j^{(q)} - z_j^{(q)}\right) g'\left(s_j\right) x_n \qquad \text{(II.16.b)}$$

II.7 LES RÉSEAUX BOUCLÉS (FEED-BACK NEURAL NETWORK)

II.7.1 La carte auto-organisatrice de Kohonen

II.7.1.1 Présentation

Proposée par Kohonen [20] au début des années 80s, les cartes auto-organisatrice (SOM, Self Organization Map) sont des réseaux établissant des cartes discrètes ordonnées topologiquement. En fait, les structures nerveuses topologiques fussent l'objet de plusieurs travaux [21][22] depuis les années 70s. Une carte auto-organisatrice se compose de deux couches : une couche d'entrée qui à pour but de refléter les vecteurs d'entrée, et une couche compétitive de sortie pouvant être en 1D, 2D ou 3D (figure II.18)

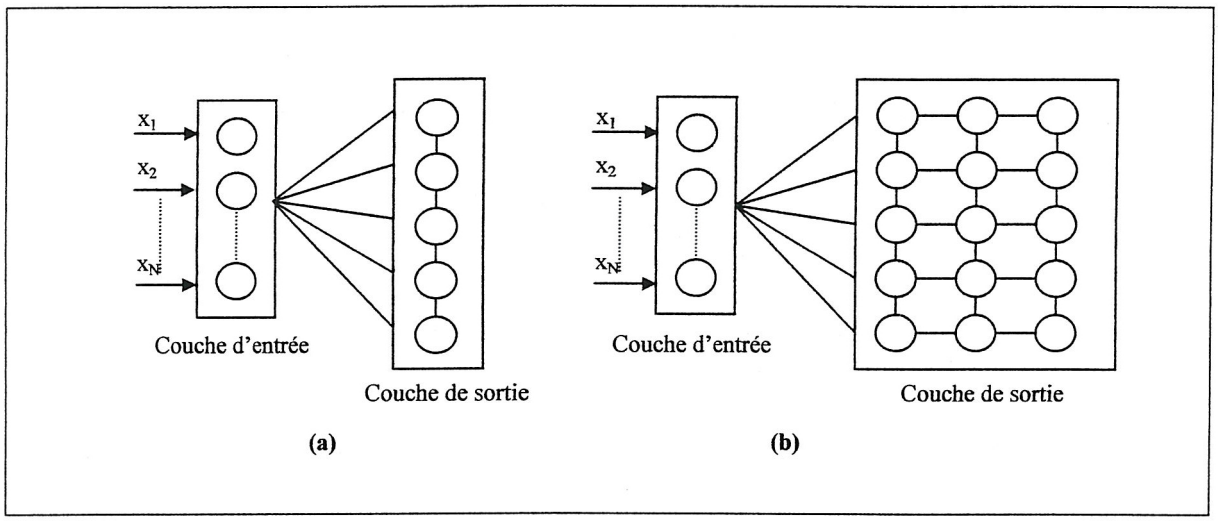


Figure (II.18) : Des cartes auto-organisatrice de Kohonen
(a) : Carte 1D
(b) : Carte 2D

II.7.1.2 Apprentissage :

L'apprentissage de la carte auto-organisatrice s'effectue de la façon suivante : A chaque présentation d'un exemple $X^{(q)}$, elles seront calculées les distances D_{qm} entre ce vecteur et les vecteurs $V^{(m)}$ correspondants à chaque neurone *(m)* de la couche de sortie. Ces distances sont données par :

$$D_{qm}=\left\|X^{(q)}-V^{(m)}\right\|$$

Où $\|\ \|$ désigne la distance euclidienne. Le neurone gagnant, qui gagne la compétition, est celui ayant la plus petite distance, les poids de son vecteur seront modifiés, ainsi que celle des neurones voisins, selon la règle compétitive :

$$V^{(m)}=V^{(m)}+\eta\left(X^{(q)}-V^{(m)}\right) \qquad \text{(II.17)}$$

Les vecteurs des autres neurones, soit qu'ils gardent leurs anciennes valeurs ou ces dernières seront diminuer, le gain d'apprentissage est alors une fonction de la distance. Ainsi chaque neurone améliore les poids de son vecteur, lorsqu'il gagne lui-même ou, l'un de ces voisins.

Après l'apprentissage, les neurones de la couche compétitive se spécialisent sur un type bien déterminer de données et au même type de réponse. Vu que l'obtention d'une structure globale nécessite des grandes distances de voisinage, et d'autre part que la précision du comportement de chaque neurone nécessite une petite distance, Kohonen [23], recommande l'utilisation d'une grande distance initiale ensuite de la réduire linéairement au cours de l'apprentissage jusqu'où le voisinage final d'un neurone contiendra celui-ci seulement.

II.7.2 Learning Vector Quantization

Crée à l'origine par lind et al. [24] et gray [25] dans un contexte de compression des données, la technique LQV (Learning Vector Quantization) est adaptée ensuite par Kohonen [23] pour qu'elle soit appliquée dans le cadre de la reconnaissance des formes. LVQ est une simple technique, qui a pour but de déterminer les valeurs optimales d'un ensemble de vecteurs prototypes représentant les données à traiter utilisées dans la classification. Ces réseaux sont similaires à la carte auto-organisatrice, mais ils suivent un apprentissage supervisé d'où la nécessité d'avoir un ensemble d'exemples qu'on connaît leur appartenance à priori. Leur apprentissage consiste à déterminer, à chaque présentation d'un exemple, le neurone gagnant puis de l'améliorer si le résultat est correct, sinon le réduire. La règle d'adaptation est comme suit :

$$V^{(m)}=\delta(t)\,\eta\left(V^{(m)}-X^{(q)}\right) \qquad \text{(II.18)}$$

Avec $\delta(t) = +1$ si le résultat est correct et $\delta(t) = -1$ dans le cas contraire.

Deuxième Partie :

CLASSIFICATION PAR LE PERCEPTRON MULTI-COUCHES

II.8 INTRODUCTION

Contrairement aux réseaux à deux couches qui sont limités par leur incapacité à produire des frontières de décision complexes, les réseaux multicouches ont de grandes capacités de calcul et peuvent formé des régions de décision arbitrairement complexes [26]. Malgré que cette conception a déjà été envisagée par Resemblatt [13], il a fallu attendre la découverte de la rétro-propagation, dans le début des années 80, par Remelhart & al. [27] pour l'apprentissage du MLP. Cette découverte donna donc renaissance aux réseaux de neurones après la diminution de leur intérêt par la conclusion pessimiste de Minsky et Paper en 1969[15]. Le but de cette partie est l'élaboration d'un algorithme d'apprentissage du MLP basé sur la rétro-propagation et comportant des différentes techniques de stabilisation et d'accélération de sa convergence, les différentes étapes, seront donc présentées pour la construction de cet algorithme ainsi que les améliorations apportées.

II.9 LE PERCEPTRON MULTI-COUCHES

Le Perceptron Multi-Couches (MLP, Multi Layered Perceptron) est une réalisation en série du perceptron. Il comprend en plus de la couche d'entrée et la couche de sortie, une ou plusieurs couches intermédiaires (dites cachées) dont chaque neurone est connecté à tous les neurones de la couche précédente et la couche suivante, les neurones de la même couche ne sont pas connectés entre eux. Comme le montre la figure (II.19): Chaque vecteur caractéristique $X(x_1\ x_2\ \dots\ x_N)$, représentant un exemple (objet), lorsqu'il est présenté au réseau tous ces composants seront transmis aux neurones de la première couche. Les sorties de ces neurones $Y\ (y_1\ y_2\ \dots\ y_M)$ seront à leur tour transmis aux neurones de la couche suivante et ainsi de suite :

$$y_m = h(r_m) \tag{II.19}$$

$$r_m = \sum_{n=1}^{N} x_n w_{nm} \tag{II.20}$$

Où h est la fonction d'activation des neurones de la couche cachée.

Dans le cas indiqué sur la figure (perceptron avec une seule couche cachée), la couche suivante c'est la couche de sortie. Le vecteur de sortie $Z\ (z_1\ z_2\ \dots\ z_J)$ est composé des sorties des neurones de cette dernière :

$$z_j = g(s_j) \tag{II.21}$$

$$s_j = \sum_{m=1}^{M} y_m u_{mj} \qquad \text{(II.22)}$$

Où g est la fonction d'activation des neurones de la couche de sortie.

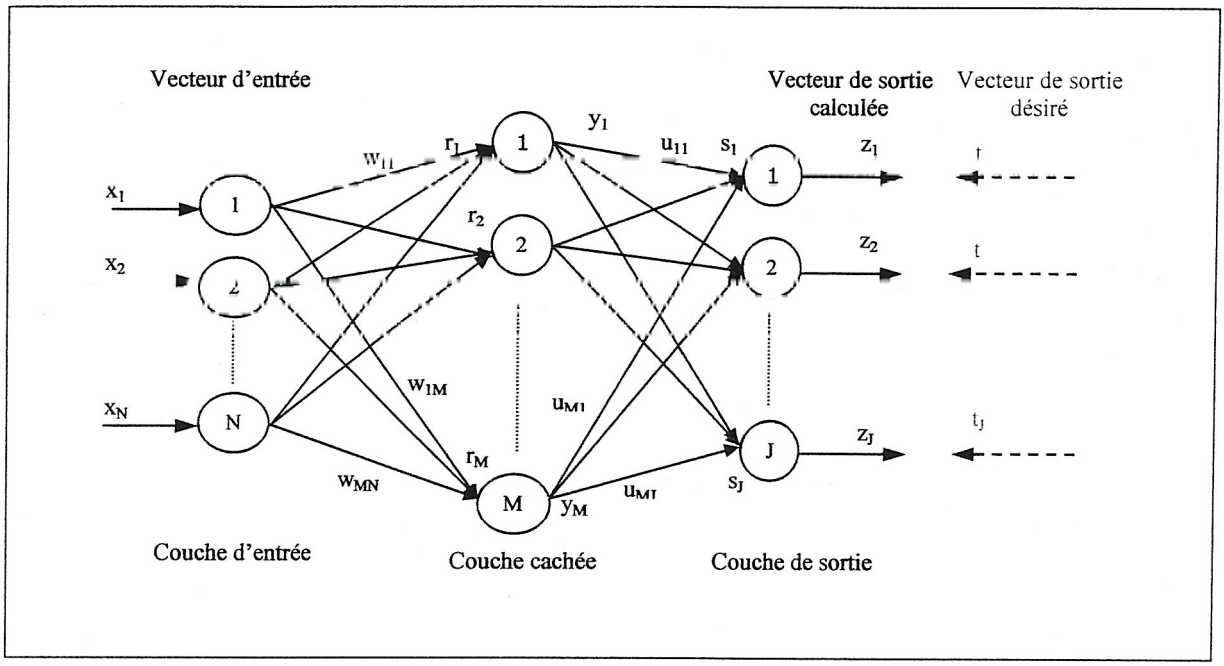


Figure (II.19) : MLP avec une seule couche cachée

II.10 ARCHITECTURE DU RÉSEAU

II.10.1 Nombre de couches cachées

Dans de nombreux cas l'utilisation d'une seule couche cachée est suffisante pour avoir des résultats satisfaisants, en effet Cybenko [28] à démontrer qu'un MLP avec une seule couche cachée ayant une fonction d'activation continue et non linéaire, est suffisant pour approximer, au sens des moindres carrées, avec une erreur arbitrairement faible pour un ensemble donné d'apprentissage, n'importe laquelle transformation continue représentée par cet exemple. D'autre part Bourlard et Wellekens [29] ont démontré que les sorties d'un MLP avec une seule couche cachée peuvent être interprétées comme des données approximations, au sens des moindres carrées des probabilités a posteriori des classes si : i) la couche de sortie comporte un neurone par classe. ii) le critère d'optimisation est celui des moindres carrées de l'erreur. iii) le nombre de neurones cachée est suffisamment grand. iv) l'apprentissage ne converge pas vers un minimum local. On se limite donc par l'utilisation d'un PMC avec une seule couche cachée

II.10.2 Nombre de neurones du réseau :

Dans le contexte de la classification, le nombre de neurones de la couche d'entrée du MLP est égal à la dimension de l'espace caractéristique vue que le rôle de cette couche est de présenter chaque composante du vecteur caractéristique d'entrée aux neurones de la couche suivante. D'autre part le nombre de neurones de la couche de sortie est égale au nombre de classes, soit un neurone par classe [29] (la sortie de ce neurone est 1 est les autres sont à 0 ou –1).

Nombre de neurones de la couche cachée : Plus que cette couche possède de neurones plus le temps d'apprentissage est grand. En effet, Hinton [30] estime que le temps réel d'apprentissage d'un MLP est proportionnel au cube de nombre W de poids de réseau $(W=NM+MJ)$, selon lui $O(W)$ époques est nécessaire pour l'apprentissage du MLP. Il est donc interessant de construire un réseau ayant le minimum possible de neurones pour que l'apprentissage soit assez rapide, mais cela diminue ces capacités de généralisation. Ils existent deux approches pour déterminer le nombre de neurones cachés nécessaires : La première consiste à commencer avec un nombre petit, ensuite à l'augmenter, et l'autre à commencer avec un nombre grand, et de diminuer ensuite. Dans les deux cas on teste à chaque fois l'erreur, et on s'arrête lorsque l'erreur de validation cesse de croître.

II.10.3 Fonction d'activation

Pour que le MLP soit capable de générer des fonctions de discrimination non linéaire, on devait utiliser des fonctions d'activations non linéaires, d'autre part l'algorithme d'apprentissage du MLP, la rétro-propagation du gradient, nécessite des fonctions d'activation continues et dérivables. Ainsi la fonction la plus utilisée c'est la fonction sigmoïde donnée par :

$$f(x)=\frac{1}{1+e^{-\alpha x}} \qquad \text{(II.23)}$$

C'est une fonction paramétrique dont le paramètre α influe sur sa réponse (figure II.20)

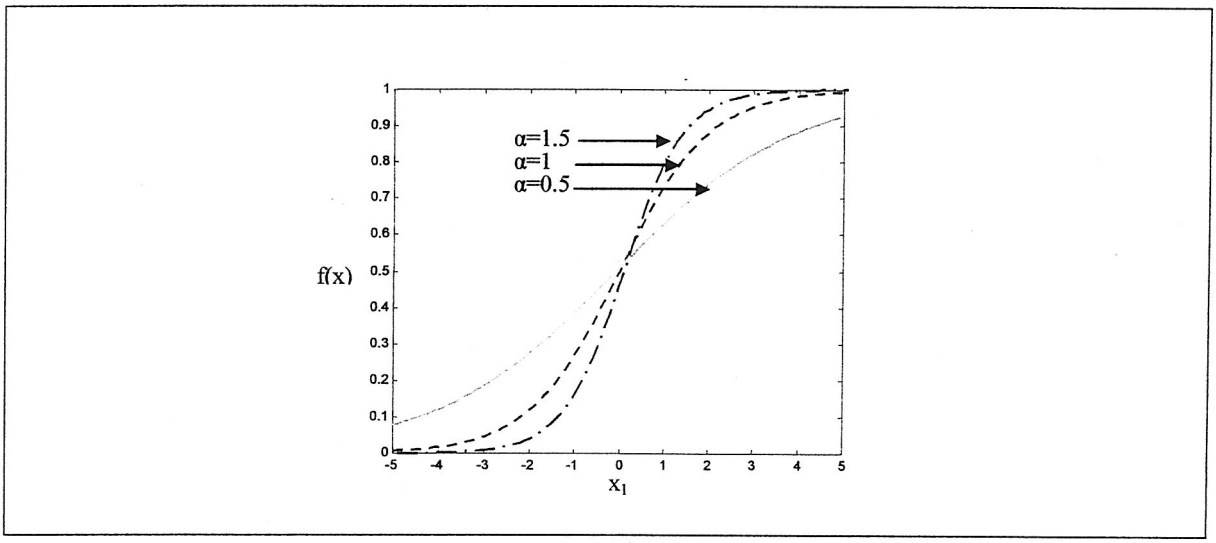


Figure (II.20) : La fonction sigmoïde avec différentes valeurs de α

II.11 APPRENTISSAGE DU MLP

La classification par le MLP consiste à fournir une sortie $Z^{(q)}(z_1\ z_2\ ...z_N)$ correspondante à la classe $c(q)$ de l'exemple (q) à classifier, représenté par son vecteur caractéristique $X^{(q)}\ (x_1\ x_2\ ...x_N)$, présenté à l'entrée du réseau (figure II.21). Comme tous les ANN, les connaissances du PMC sont représentées par ces poids. Ainsi son apprentissage consiste à adapter ces derniers d'une manière à ce que le réseaux soit capable de réaliser la tache désiré (classification, modélisation, ...etc.) en minimisant l'erreur quadratique total
données par :

$$E = \sum_{q=1}^{Q} \sum_{j=1}^{J} \left(t_j^{(q)} - z_j^{(q)}\right)^2 \qquad \text{(II.24)}$$

L'apprentissage du MLP est supervisé, on doit donc disposer d'une base de données contenant des échantillons dont on connaît leur appartenance à une classe, c'est à dire une suite de Q vecteurs caractéristiques $X^{(q)}$ associés avec une suite de même nombre de vecteurs de sorties désirées $T^{(q)}$ dont chacun représente la classe du vecteur d'entrée associé. L'algorithme d'apprentissage du PMC est celui de la rétro-propagation du gradient.

La rétro-propagation du gradient (Back Propagation) : est une technique de descente du gradient, c'est une généralisation de la règle de Widrow-Hoff. La rétro-propagation du gradient, tel que son nom l'indique, permet d'ajuster les poids en propageant l'erreur de la couche de sortie vers la couche d'entrée (figure II.21)

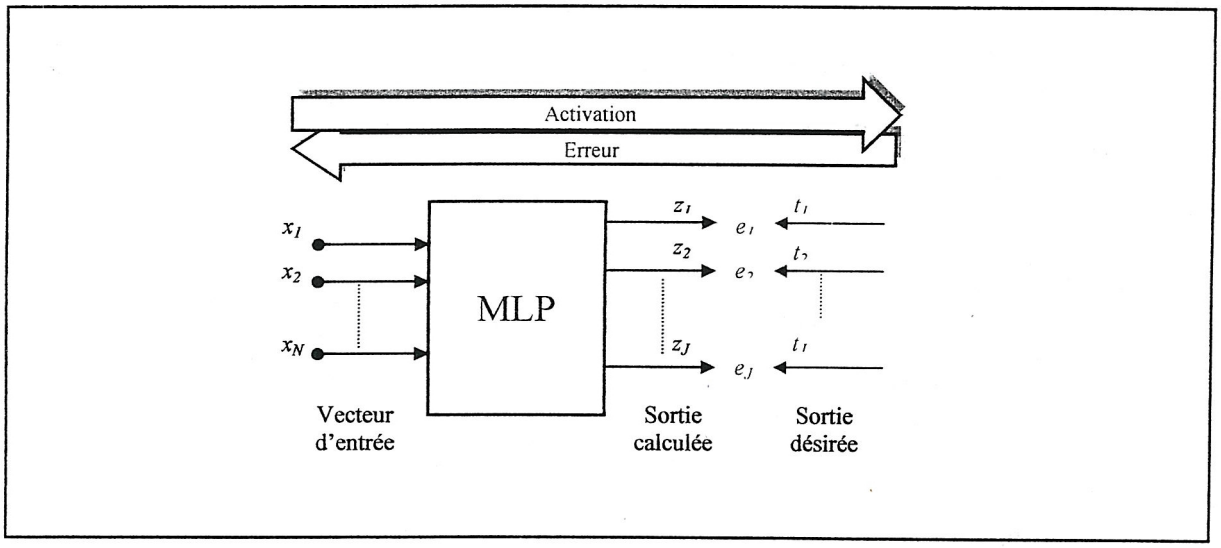


Figure (II.21) : Apprentissage du perceptron par la rétro-propagation du gradient

On devait donc posséder d'un ensemble de Q pairs {entrée / sortie désirée} appelé ensemble d'apprentissage. L'ajustement des poids w_{nm} et u_{mj} à l'itération $(i+1)$ est donné par les équations :

$$\Delta W_{nm}^{(i+1)} = -\eta_1 \frac{\partial E^{(i)}}{\partial W_{nm}} \qquad \text{(II.25.a)}$$

$$\Delta u_{mj}^{(i+1)} = -\eta_2 \frac{\partial E^{(i)}}{\partial u_{mj}} \qquad \text{(III.25.b)}$$

Où η_1 et η_2 sont des constantes positives, inférieurs à 1, appelées pas d'apprentissage

la mise à jour des poids serait comme suit (Annexe)

$$u_{mi}^{(i+1)} = u_{mj}^{(i)} + \eta_1 \cdot (t_j - z_j) \cdot g' \cdot (s_j) y_m \qquad \text{(II.26.a)}$$

$$w_{nm}^{(i+1)} = w_{nm}^{(i)} + \eta_2 \cdot \left(\sum_{j=1 i=1}^{J} (t_j - z_j) \cdot g'(s_j) \cdot u_{mj} \right) \cdot (h'(r_m) \cdot (x_n) \qquad \text{(II.26.b)}$$

Dans le cas de la sigmoïde unipolaire :

$$\frac{\partial E}{\partial u_{mj}} = -2.(t_j - z_j) z_j (1 - z_j) y_m \qquad \text{(II.27.a)}$$

$$\frac{\partial E}{\partial w_{nm}} = -2 \cdot \left\{ \sum_{j=1 i=1}^{J} (t_j - z_j).z_j \cdot (1 - z_j).u_{mj} \right\} * y_m \cdot (1 - y_m).(x_n) \qquad \text{(II.27.b)}$$

Et dans le cas de la sigmoïde bipolaire :

$$\frac{\partial E}{\partial u_{mj}} = -(t_j - z_j)(1 - z_j)(1 + z_j) y_m \qquad \text{(II.28.a)}$$

$$\frac{\partial E}{\partial w_{nm}} = \frac{-1}{2} \cdot \left\{ \sum_{j=1 i=1}^{J} (t_j - z_j) \cdot (1 + z_j) \cdot (1 - z_j) \cdot u_{mj} \right\} (1 + y_m)(1 - y_m)(x_n) \qquad \text{(II.28.b)}$$

II.11.1 Algorithme d'apprentissage de la Back-Propagation :

Pour un ensemble d'apprentissage de Q exemples, l'apprentissage du MLP se fait comme il l'indique l'algorithme (II.1)

Algorithme (II.1)

Etape 1 : introduction des données

$X^{(q)}$ (vecteurs caractéristiques)

$T^{(q)}$ (vecteurs de sorties désirées)

Etape 2 : détermination des dimensions de données

N (nombre de caractéristique qui sera le nombre de neurones de la couche d'entrée)

Q (nombre d'exemples d'apprentissage)

J (nombre de classes qui sera le nombre de neurones de la couche de sortie)

Initialisation des paramètres :

α (paramètre des sigmoïde)

I (nombre d'itération)

η_1 et η_2 (pas d'apprentissage).

M (nombre de neurones de la couche cachée)

U_{mi} et W_{nm} (poids initiaux avec des valeurs aléatoires)

Etape 3 : Apprentissage

Pour i=1 à I (pour chaque itération)

Pour q =1 à Q (pour chaque exemple)

$X = X^{(q)}$ (introduction du vecteur caractéristique de l'exemple q)

$T = T^{(q)}$ (introduction du vecteur de sortie de l'exemple q)

Sous-programme de la mise à jour

Pour m=1 à M

Pour j=1 à J

Ajustement de chaque poids u_{mj} selon (III.26.a)

Pour n=1 à N

Ajustement de chaque poids w_{nm} selon (III.26.b)

$$E(i)=\sum_{q=1}^{Q} E_p(q)$$

Etape 4 : Fin

Sous-programme de la mise à jour

Pour m =1 à M

Pour n =1 à N

$$r_m = \sum_{n=1}^{N} w_{nm}\, x_n$$

$$y_m = \frac{1}{1+e^{-\alpha r_m}}$$

Pour j =1 à J

$$s_j = \sum_{n=1}^{J} u_{nm}\, y_n$$

$$z_j = \frac{1}{1+e^{-\alpha s_j}}$$

$$E_p(q) = \sum_{j=1}^{J} (t_j - z_j)^2$$

II.12 ACCÉLÉRATION DE L'APPRENTISSAGE

La rétro-propagation du gradient est historiquement l'algorithme le plus ancien et le plus utilisé pour l'apprentissage du MLP [1]. Mais c'est un processus lent et peu fiable [31], il est donc très important d'accélérer sa convergence particulièrement lorsqu'on saurait que la base de données la plus représentative est souvent de très grande taille et requiert un temps d'apprentissage important, ainsi beaucoup de techniques ont été développées pour l'accélération de sa convergence.

La convergence de la back-propagation dépend de [1] : i) la base de données utilisée (la taille et la qualité d'exemples d'apprentissage). ii) l'architecture du réseau. iii) le gain d'apprentissage et sa stratégie d'adaptation. iv) la valeur du moment. v)les poids initiaux. vi) les fonctions d'activation. Donc pour améliorer l'apprentissage nous présentons dans les paragraphes suivants quelques techniques qui seront implantées dans l'algorithme d'apprentissage.

II.12.1 Modes d'apprentissage

L'apprentissage du MLP par la rétro-propagation consiste en deux modes, ils se différent selon la méthode d'ajustement des poids. Pour le premier mode la modification des poids s'effectue à chaque présentation d'un exemple d'apprentissage, donc il à tendance à minimiser les erreurs quadratiques partielles de chaque exemple. Le second mode, appelé le full-propagation, consiste à minimiser l'erreur quadratique totale en ajustant les poids après la présentation de tous les exemples, les poids sont donc modifiés par l'accumulation des modifications apportées par la présentation de chaque exemple. L'ajustement des poids se fait de la manière suivante :

$$u_{mj}^{(i+1)} = u_{mj}^{(i)} + \eta_1 \cdot \sum_{q=1}^{Q} \left(t_j^{(q)} - z_j^{(q)}\right) \cdot g'\left(s_j\right) \cdot y_m \qquad \text{(III.29.a)}$$

$$w_{nm}^{(i+1)} = w_{nm}^{(i)} + \eta_2 \sum_{q=1}^{Q} \left\{ \sum_{j=1 i=1}^{J} (t_j^{(q)} - z_j^{(q)}) \cdot g'(s_j) \cdot u_{mj} \right\} \cdot \left(h'(r_m)\right) \cdot (x_n^{(q)}) \qquad \text{(III.29.b)}$$

Cependant le full propagation présente l'avantage d'effectuer l'adaptation des poids une seule fois pour tous les exemples et par conséquent une accélération d'apprentissage, et permet aussi d'éviter le problème du mode de présentation des exemples d'apprentissage.

II.12.2 formalisation matricielle

Le full-propagation offre la possibilité d'une meilleure formalisation matricielle des différentes équations et d'éviter le plus possible les boucles dans l'algorithme d'apprentissage, il suffit donc de représenter les vecteurs caractéristiques de tous les exemples d'apprentissage $\{X^{(1)} ; X^{(2)} ; X^{(Q)}\}$ par une matrice X, ainsi que les poids synaptiques $\{W_{nm}\}$ et $\{U_{mj}\}$ qui seront remplacer par les matrices W et U. la mise à jour du réseau serait :

$$Y=WX ; R=h(X)$$
$$S=RU ; Z=g(S)$$

En posant G et H les matrices ayant comme éléments les dérivées dg/ds_j et dh/dr_m respectivement, on propose d'adapter les poids à l'itération $(i+1)$ comme suit :

$$U^{(i+1)} = U^{(i)} + \eta_1 Y'*((T-Z).*G) \quad \text{(III.30.a)}$$

$$W^{(i+1)} = W^{(i)} + \eta_2 X'(((T-Z).*G*U').*H \quad \text{(III.30.b)}$$

Où (*) désigne le produit matricielle, (.*) : Le produit élément par élément et (') : L'opérateur transposé.

Vérification : Pour vérifier les équations précédentes, on utilise un petit programme sous MATLAB-Symbolic Maths. On se limite à un réseau ayant deux neurones d'entrées, deux neurones de sortie et trois neurones cachés. Et on considère qu'on dispose de quatre exemples, ainsi on aura : $Q=4$; $N=2$; $M=3$; $J=2$. Ce programme consiste à calculer ΔW et ΔU en utilisant une fois les boucles et une fois les équations matricielles décrites précédemment et à déterminer aussi la différence entre ces deux méthodes. En posant $E = (T-Z)$, l'exécution de ce dernier donne les résultats présentés à l'Annexe.

l'exécution des programmes utilisant les boucles une fois, et utilisant les matrices une autre fois pour le back propagation et le Full propagation (les équations précédentes sans valables aussi pour le Back propagation, il suffit de remplacer les matrices X et T par les vecteurs $X^{(q)}$ et $T^{(q)}$ de chaque exemple) sur la base de données 10-5-10 (qui sera représentée dans le paragraphe suivant), permet de conclure que l'utilisation de la formalisation matricielle réduit le temps d'apprentissage à 12.7 % pour le Back propagation et à 11 % pour le full propagation.

Comparaison entre la back-propagation et le full-propagation : Pour comparer ces deux modes nous effectuons des tests semblables aux ceux de Looney [1], Fahlman [32], Riedmiller et al. [33] sur la base de données standards 10-5-10 (ten5ten), dont les vecteurs d'entrée et de sortie réalisent une rotation de dix, donc dix vecteurs ayant les composants nuls sauf un seul qui vaut « *1* » forment les entrées et les sorties du réseau (tableau II.1)

Exemple	Vecteurs d'entrée	Vecteurs de sortie
1	(1 0 0 0 0 0 0 0 0 0)	(1 0 0 0 0 0 0 0 0 0)
2	(0 1 0 0 0 0 0 0 0 0)	(0 1 0 0 0 0 0 0 0 0)
3	(0 0 1 0 0 0 0 0 0 0)	(0 0 1 0 0 0 0 0 0 0)
4	(0 0 0 1 0 0 0 0 0 0)	(0 0 0 1 0 0 0 0 0 0)
5	(0 0 0 0 1 0 0 0 0 0)	(0 0 0 0 1 0 0 0 0 0)
6	(0 0 0 0 0 1 0 0 0 0)	(0 0 0 0 0 1 0 0 0 0)
7	(0 0 0 0 0 0 1 0 0 0)	(0 0 0 0 0 0 1 0 0 0)
8	(0 0 0 0 0 0 0 1 0 0)	(0 0 0 0 0 0 0 1 0 0)
9	(0 0 0 0 0 0 0 0 1 0)	(0 0 0 0 0 0 0 0 1 0)
10	(0 0 0 0 0 0 0 0 0 1)	(0 0 0 0 0 0 0 0 0 1)

Tableau (II.1) : Base de données 10-5-10

On utilise un réseau d'architecture 10-5-10, c'est à dire la couche d'entrée comporte dix neurones *(N=10)*, la couche cachée comporte cinq neurones *(M=10)* et 10 neurones pour la couche de sortie (J=10). D'autre part on utilise 0.1 à la place de 0 et 0.9 à la place de 1, les fonctions d'activation seront des sigmoïdes unipolaires ayant $\alpha = 2.4$, les gains d'apprentissage $\eta_1 = \eta_2 = 0.3$, les résultats obtenus (pour 1000 itérations) sont indiqués sur le tableau (III.2) : l'erreur quadratique moyenne ; le taux de classification des exemples

d'apprentissage et le temps d'apprentissage pour dix exécutions, et à chaque fois en garde les mêmes poids initiaux.

exécution	Back propagation			Full propagation		
	EQM	Taux	temps	EQM	Taux	temps
1	0.0347	100	5.660	0.0347	100	0.770
2	0.0307	100	5.760	0.0362	100	0.880
3	0.0363	100	5.830	0.0364	100	0.770
4	0.0355	100	5.830	0.0355	100	0.770
5	0.0383	100	5.440	0.0378	100	0.770
6	0.0373	100	5.770	0.0373	100	0.770
7	0.0345	100	5.710	0.0364	100	0.820
8	0.0339	100	5.820	0.0340	100	0.820
9	0.0391	100	5.880	0.0397	100	0.830
10	0.0364	100	5.770	0.0371	100	0.770

Tableau (II.2) : Résultats d'apprentissage de la base de données 10-5-10 par la Back-propagation et le Full-propagation

Les résultats précédents nous permettent de conclure que l'algorithme utilisant la formalisation matricielle du Full propagation nécessite un temps moins que celui nécessaire pour le Back propagation (algorithme utilisant la formalisation matricielle aussi) de 13% pour données des erreurs quadratiques moyenne presque égales.

II.12.3 La saturation des neurones

Dans le cas la saturation d'un neurone, et vu que la dérivée de la sigmoïde serait très faible, la correction ainsi apportée est presque nulle, même si la sortie de ce neurone est assez différente de la sortie désirée.

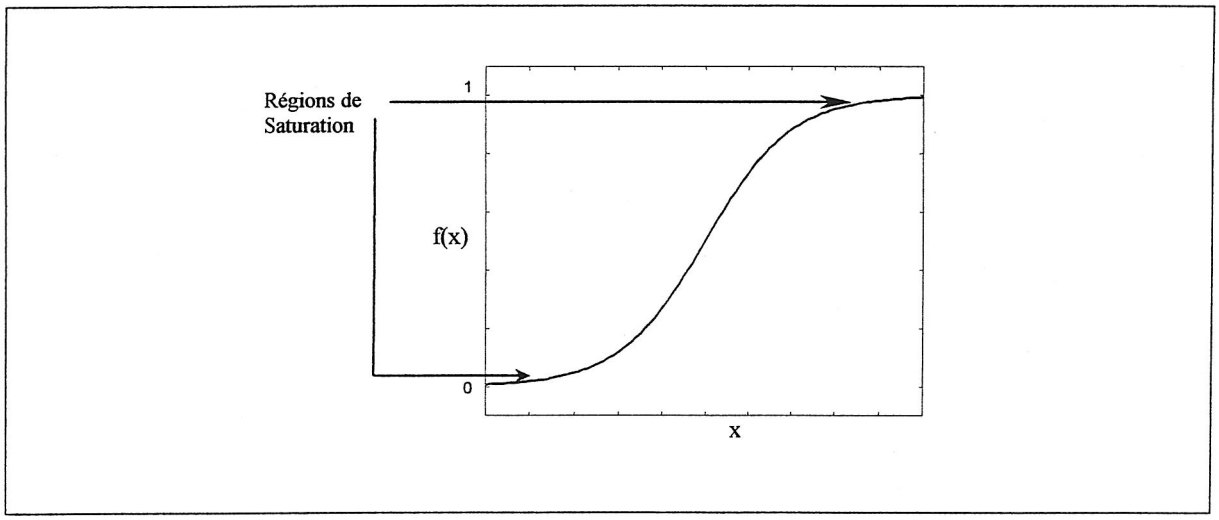


Figure (II.22) : Régions de saturation de la sigmoïde

En effet un poids peut se trouver dans la région de saturation si la sigmoïde converge vers zéro [35], et il faut un nombre important d'itérations pour sortir de telle région, et peut être ne

jamais sortir. Pour éviter ce problème Fahlman [32] a ajouté une petite valeur ($\Delta = 0.1$) pour chaque dérivée, Looney [1], et après avoir effectuer des tests sur plusieurs données, a modifié cette technique en ajoutant cette petite valeur seulement pour les dérivées inférieures à 0.1. La figure (II.22) montre les régions de saturation d'une sigmoïde unipolaire.

De plus, pour éviter ce problème on normalise les données pour qu'elles s'échelonnent entre « 0.1 » et « 0.9 » par l'équation [1] :

$$X = 0.8\frac{(X - X\min)}{(X\max - X\min)} + 0.1 \qquad \text{(II.31)}$$

II.12.4 Les poids initiaux

Les poids initiaux représentent l'un des facteurs dont dépend la convergence de l'algorithme d'apprentissage, donc une initialisation convenable assure un bon début d'apprentissage. Il sont choisis généralement avec des petites valeurs aléatoires réparties uniformément avec une moyenne de zéro [32], en plus ils doivent être d'une magnitude inférieur de *0.5* pour éviter la saturation des neurones au début de l'apprentissage [1], pour la même raison Russo [35] suggère que leurs magnitude soit inférieure de *2.4/M* (*M* étant le nombre totale des lignes arrivant au neurone dont réside le poids). D'autre part Nguyen & Widrow [36] ont mis au poins une méthode d'initialisation qui consiste à distribuer les pois initiaux dans la couche cachée de telle façon que chaque exemple présenté permet d'accorder un apprentissage efficace aux neurones cachés. Ils ont posé : $\beta = 0.7(M^{1/N})$ ou M est nombre des neurones cachés et N est la dimension de l'espace caractéristique. Ensuite, et après avoir initialiser les poids aléatoirement (soit Wa la matrice des poids ainsi obtenue), leur technique consiste à calculer chaque poids initial w_{nm} selon la relation :

$$w_{nm} = \frac{\beta}{\sum_{n=1}^{N}\sum_{m=1}^{M} wa_{nm}} wa_{nm} \qquad \text{(II.32)}$$

Nous utilisons donc cette technique pour l'initialisation des poids de la couche cachée, et la méthode de Russo pour les poids de la couche de sortie [1].

II.12.5 Ajustement du pas d'apprentissage :

Dans la phase d'apprentissage du MLP par la rétro-propagation du gradient, le pas d'apprentissage (η_1 ou η_2) définit la taille de la descente sur la surface de l'erreur, donc s'il est petit l'adaptation des poids sera avec des petits pas, et par conséquent l'apprentissage sera lent, et s'il est grand, il risque de dépasser le minimum global et risque aussi d'apparaître les oscillations. Sa valeur est donc critique, et la détermination d'un gain optimale n'est pas évidente. Cependant il est intéressant de modifié ce paramètre itérativement au cours de l'apprentissage, comme celle qui consiste à l'augmenter si l'erreur décroît, le réduire si l'erreur augmente et le laisser inchangé si cette dernière change légèrement, ainsi l'utilisation de cette méthode rend le choix de ce paramètre peu critique. D'autres techniques sont aussi proposées comme l'ajout du moment [27], quick-prop [32] et la règle delta bar delta [37].

La règle delta bar delta : élaborée par Jacobs [37], cette technique est basée sur l'utilisation d'un pas d'apprentissage différent pour chaque poids et l'ajuster itérativement, l'adaptation du poids w_{nm} (et u_{mj}) à l'itération *(i+1)* sera donc de la forme :

$$w_{nm}^{(i+1)} = w_{nm}^{(i)} - \eta_{nm} \frac{\partial E^{(i)}}{\partial w_{nm}} \qquad \text{(II.32)}$$

Cette technique consiste à ajuster chaque poids en se basant sur la forme locale de l'erreur définie par les signes de ces gradients, donc l'ajustement de chaque pas d'apprentissage dépend de la variation de l'erreur en fonction du poids associé, ainsi il devrait être augmenter dans le cas ou les dérivées de plusieurs modifications ont le même signe et devrait être réduit s'ils changent de signes. La modification des pas se fait de la façon suivante :
On détermine pour chaque pas :

$$\Delta_{nm}^{(i)} = \beta\left(\Delta_{nm}^{(i-1)}\right) + (\beta - 1)\left(\partial E^{(i)} / \partial w_{nm}\right) \qquad \text{(II.33)}$$

ensuite on l'ajuste comme suit :

$$\eta_{nm}^{(i+1)} = \eta_{nm}^{(i)} + \delta \quad si \quad \left(\Delta_{nm}^{(i-1)}\right)\left(\partial E^{(i)} / \partial w_{nm}\right) > 0 \qquad \text{(II.34.a)}$$

$$\eta_{nm}^{(i+1)} = \theta\,\eta_{nm}^{(i)} \quad si \quad \left(\Delta_{nm}^{(i-1)}\right)\left(\partial E^{(i)} / \partial w_{nm}\right) < 0 \qquad \text{(II.34.b)}$$

$$\eta_{nm}^{(i+1)} = \eta_{nm}^{(i)} \quad si \quad \left(\Delta_{nm}^{(i-1)}\right)\left(\partial E^{(i)} / \partial w_{nm}\right) = 0 \qquad \text{(II.34.c)}$$

Où δ, β et θ sont des petites constantes positives pré spécifiées par l'utilisateur, Jacobs a posé $\delta=0.05$, $\theta = 0.3$ *et* $\beta = 0.7$. la valeur initiale des $\eta_{nm}^{(0)}$ est généralement mise à « *0.1* ». Pour tester la règle Delta Bar Delta on effectue l'apprentissage de la base de donnée précédente *(10-5-10)* par le Back propagation et le full propagation pour 300 itérations et avec les même paramètres (η= *0.1*, δ=*0.04*, θ = *0.3 et* β = *0.7*), les même poids initiaux serons utilisés à chaque exécution, les résultats ainsi obtenus sont présentés sur le tableau (II.3) :

Exécution	Back propagation			Full propagation		
	EQM	Taux	temps	EQM	Taux	temps
1	0.0468	100	8.19	0.0293	100	0.82
2	0.0659	100	8.30	0.0273	100	0.77
3	0.0658	100	8.46	0.0268	100	0.83
4	0.0609	100	8.41	0.0262	100	0.77
5	0.0555	100	8.41	0.0261	100	0.77
6	0.0444	100	8.40	0.0263	100	0.77
7	0.0910	100	8.34	0.0260	100	0.77
8	0.0527	100	8.35	0.0263	100	0.83
9	0.0472	100	8.40	0.0253	100	0.82
10	0.0685	100	8.41	0.0274	100	0.77

Tableau (III.3) : Résultat d'apprentissage de la base de données (10-5-10) en utilisant la règle delta Bar Delta et l'algorithme Nguyen-Widrow-Russo d'initialisation des poids

D'après les résultats précédents on conclue qu'on utilisant la règle Delta bar Delta avec le Full propagation accorde une convergence plus rapide (300 itérations suffisantes), d'autre part le temps d'exécution est de 10% du temps nécessaire pour le Back propagation.

II.12.6 Le moment (momentum)

Cette technique est proposée à l'origine par Rumelhart et al. [27], elle consiste à ajouter un terme d'inertie au corrections apportées à chaque poids. Ainsi l'adaptation du poids w_{nm} à l'itération (i+1) est donnée par :

$$w_{nm}^{(i+1)} = w_{nm}^{(i)} - \eta \frac{\partial E^{(i)}}{\partial W_{nm}^{i}} + \mu\left(w_{nm}^{(i)} - w_{nm}^{(i-1)}\right) \qquad \text{(II.35)}$$

Ou μ (le moment) est une constante positive, avec $0<\mu<1$. Donc la modification de chaque poids, en plus quelle est proportionnelle à la dérivée de l'erreur par rapport à ce poids, elle est en fonction de la correction appliquée à l'itération précédente. Cette technique, en diminuant la correction apportée dans le cas où elle serait opposée à la correction précédente, à tendance d'éliminer les oscillations.

Le choix de la valeur de μ semble relativement simple [32], en pratique les valeurs du moment choisie s'échelonnent généralement entre 0.6 et 0.9 [26].

II.12.7 Technique de Zurada & Yamada

Le paramètre *(α)* agit sur la forme des sigmoïdes (figure II.20), et par conséquent sur la réponse des neurones, donc une modification de ce paramètre améliore la convergence de l'algorithme de l'apprentissage [1]. La technique de Zurada et Yamada [38] consiste à traiter ce paramètre comme un poids additionnel et à l'ajuster itérativement, chaque α_r étant ajusté à l'itération *(i+1)* comme suit :

$$\alpha_r^{(i+1)} = \alpha_r^{(i)} - \eta \frac{\partial E^{(i)}}{\partial \alpha_r} \qquad \text{(II.36)}$$

Pour les paramètres de la couche de sortie :

$$\frac{\partial E^{(i)}}{\partial \alpha_j} = \left(\frac{\partial E^{(i)}}{\partial z_j}\right)\left(\frac{\partial z_j}{\partial \alpha_j}\right)$$

$$= -2\left(t_j - z_j\right) g'\left(s_j\right) s_j \qquad \text{(II.37.a)}$$

Et pour les paramètres de la couche cachée :

$$\frac{\partial E^{(i)}}{\partial \alpha_m} = \sum_{j=1}^{J} \left(\frac{\partial E^{(i)}}{\partial z_j}\right)\left(\frac{\partial z_j}{\partial s_j}\right)\left(\frac{\partial s_j}{\partial y_m}\right)\left(\frac{\partial y_m}{\partial \alpha_m}\right)$$

$$= \left[-2\sum_{j=1}^{J}\left(t_j - z_j\right) g'\left(s_j\right) u_{mj}\right]\left[h'\left(r_m\right) r_m\right] \qquad \text{(II.37.b)}$$

Algorithme II.2 :

Etape 1 : introduction des données

$X^{(q)}$ (vecteurs caractéristiques)

$T^{(q)}$ (vecteurs de sorties désirées)

Etape 2 : Détermination des dimensions de données

Initialisation des paramètres

Appel à sous programme d'initialisation

Etape 3 : APPRENTISSAGE

Pour i=1 à I (pour chaque itération)

Appel à Sous-programme de simulation du réseau

Détermination de $\partial E/\partial w_{nm}$

Détermination de $\partial E/\partial u_{mj}$

$\partial E/\partial u_{mj} = \partial E/\partial u_{mj} + 0.1$

$\partial E/\partial w_{nm} = \partial E/\partial w_{nm} + 0.1$

Ajustement de poids

Pour m=1 ***à*** M

Pour n=1 ***à*** N

Ajustement de chaque poids u_{mj} selon (III.30.a)

Pour j=1 ***à*** J

Ajustement de chaque poids w_{nm} selon (III.30.b)

Ajustement des pas selon delta bar delta

Pour m=1 ***à*** M

Pour n=1 ***à*** N

Si $\left(\Delta_{nm}^{(i-1)}\right)(\partial E^{(i)} / \partial W_{nm}) > 0$

Alors $\eta_{nm}^{(i+1)} = \eta_{nm}^{(i)} + \delta$

SiNon $\left(\Delta_{nm}^{(i-1)}\right)(\partial E^{(i)} / \partial W_{nm}) < 0$

Alors $\eta_{nm}^{(i+1)} = \theta\, \eta_{nm}^{(i)}$

SiNon $\eta_{nm}^{(i+1)} = \eta_{nm}^{(i)}$

$\Delta_{nm}^{(i)} = \beta\left(\Delta_{nm}^{(i-1)}\right) + (\beta - 1)(\partial E^{(i)} / \partial W_{nm})$

Pour j=1 ***à*** J

Si $\left(\Delta_{mj}^{(i-1)}\right)(\partial E^{(i)} / \partial u_{mj}) > 0$

Alors $\eta_{mj}^{(i+1)} = \eta_{mj}^{(i)} + \delta$

SiNon $\left(\Delta_{mj}^{(i-1)}\right)(\partial E^{(i)} / \partial u_{mj}) < 0$

Alors $\eta_{mj}^{(i+1)} = \theta\ \eta_{nm}^{(i)}$

SiNon $\eta_{mj}^{(i+1)} = \eta_{mj}^{(i)}$

$\Delta_{mj}^{(i)} = \beta\left(\Delta_{mj}^{(i-1)}\right) + (\beta - 1)(\partial E^{(i)} / \partial u_{mj})$

Ajustement de (α) selon Zurada & Yamada

Pour m=1 ***à*** M

Ajustement de α_m selon (II.37.a)

Pour j=1 ***à*** J

Ajustement de α_j selon (II.37.b)

$$E(i)=\sum_{q=1}^{Q} E_p(q)/Q$$

Etape 4 : fin

II.13 APPLICATION

Pour apprécier les performances l'algorithme d'apprentissage du MLP, basé sur le Full propagation et comportant l'algorithme de Nguyen-Widrow-Russo de l'initialisation des poids initiaux, la méthode delta bar delta et la technique de Fahlmande. Nous avons effectuer l'apprentissage du MLP, sur la base de données Iris, selon cet l'algorithme (algorithme II.2) sans et avec moment, puis avec l'ajustement des fonctions d'activation [39]. Le réseau utilisé est d'architecture 4-12-3, les paramètres de la règle Delta Bar Delta sont : $\eta = 0.1$, $\delta = 0.005$, $\theta = 0.3$ et $\beta = 0.7$. L'évolution de l'erreur quadratique moyenne au cours de l'apprentissage est illustrée sur la figure (II.23)

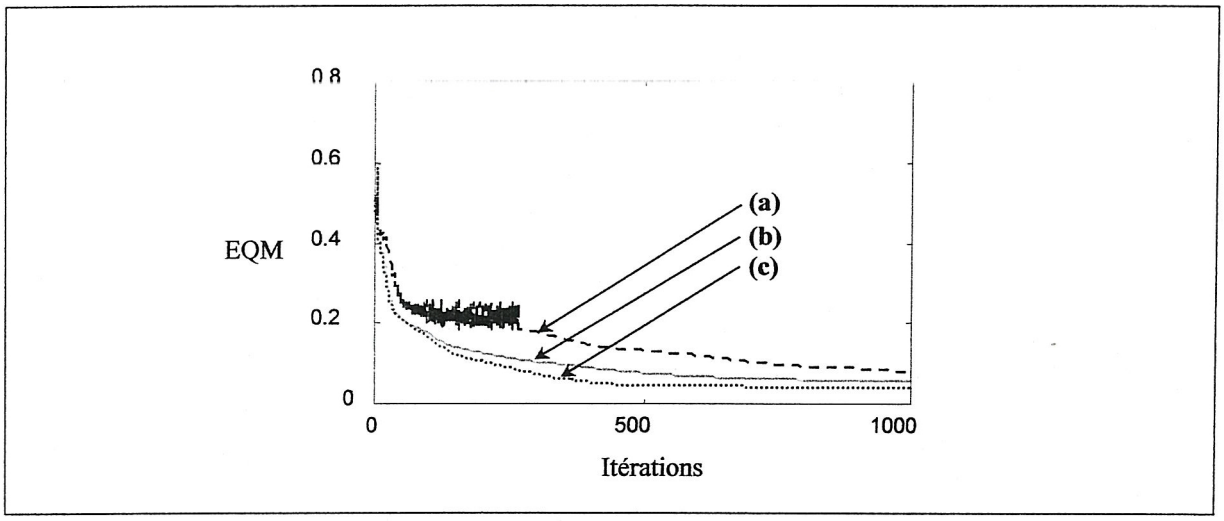


Figure (II.23) : Evolution de l'EQM au cours de l'apprentissage de Iris
(a) : Algorithme sans moment
(b) : Algorithme avec moment
(c) : Algorithme avec moment et avec l'ajustement des fonctions d'activation

II.14 CONCLUSION

L'intérêt majeurr des réseaux de neurones artificiels réside principalement dans leurs capacités d'apprentissage et de généralisation, ils peuvent s'adapter sur les exemples d'apprentissage pour réaliser un modèle capable d'effectuer la classification d'une manière rapide, vu leurs grandes capacités de calcul parallèle. Parmi leurs inconvénients : c'est l'impossibilité de décrire le modèle réalisé ; la possibilité qu'ils peuvent être prisonniers dans minimum local et le manque de techniques qui permettent de déterminer automatiquement leurs paramètres et leurs architectures. De ce fait, la deuxième partie de ce chapitre été consacré à compenser quelques inconvénients du MLP, vu qu'il est le type le plus utilisé des ANNs et qu'ils existent plusieurs techniques destinés à accélérer et à stabiliser son apprentissage. Ainsi, en se basant sur ces techniques, on a élaboré un algorithme d'apprentissage du MLP qui permet d'accélérer sa phase d'entraînement, de le stabiliser, de lui éviter, le plus possible, d'être prisonnier dans les minimums locaux et de rendre le choix de ces paramètres peu critique.

CHAPITRE III :

Classification Neuro-Floue

III.1 INTRODUCTION

De nos jours, ils existent plusieurs systèmes complexes comportant des problèmes différents dont chacun nécessite un type de calcul distinct, ce qui donna lieu à la naissance des nouveaux systèmes hybrides combinant les réseaux de neurones, la logique floue, les systèmes experts et les algorithmes génétiques. Les systèmes Neuro-Flous font partie de cet ensemble d'approches qui visent à générer des systèmes plus robustes, plus efficaces et facilement interprétables, ils présentent un intérêt majeur pour les chercheurs de l'intelligence artificielle vu que leur utilisation permet, d'une part de bénéficier des avantages des réseaux de neurones artificiels et de la logique floue, et d'autre part de compenser leurs inconvénients.

Ce chapitre a pour objectif de présenter quelques systèmes flous et Neuro-Flous dédiés à la classification, il se divise en deux parties : La première constitue une introduction à la logique floue et aux systèmes flous, tandis que la deuxième est consacrée aux systèmes Neuro-Flous et leurs architectures.

Première partie :

LES SYSTÈMES FLOUS

III.2 INTRODUCTION À LA LOGIQUE FLOUE

Le concept de la logique floue était introduit en 1965 par Lotfi Zadeh [40], professeur de l'université de Californie de Barkley (USA) en publiant son article intitulé : Ensembles flous (fuzzy sets). A cette époque elle n'a pas eu un grand succès, malgré quelques travaux effectués, il a fallu attendre la fin des années quatre-vingt pour apparaître, au Japon puis en Europe, de nombreux produits industriels utilisant la logique floue. Ces produits ont été portés à large public, d'où l'essor de cette théorie.

Plutôt d'utilisé des variables numériques précises, la logique floue fait appel à des variables floues comme : faible grand, petit...Elle tente de formaliser les phénomènes traités comme le fait le cerveau humain, cette théorie est basée sur des concepts floues similaires à notre raisonnement. En effet, nos connaissances face à la plupart des situations sont imparfaites, les informations qu'on manipule sont souvent imprécises et incertaines, par exemple : Pour choisir un appartement, il nous suffit de savoir qu'elle est proche de notre travail sans une approche de la distance exacte, prenons aussi le cas d'un conducteur qui s'approche d'une intersection, il se dit par exemple: « Si ma vitesse n'est pas très élevée, et si le feu de réglementation est rouge et pas assez loin, je freine doucement », il n'y a personne qui dit : « Si ma vitesse est supérieure à 80 Km et le feu est à 65 m je freine avec une force de 103 Newtons ».

III.3 LES ENSEMBLES FLOUS (FUZZY SETS)

La logique floue repose sur la constatation qu'ils existent des données qui ne peuvent pas être représentés par des variables Booliennes ayant deux valeurs « 0 » ou « 1 » (cas de la logique classique), par exemple une chambre de température 18° n'est vraiment chaude n'est vraiment froide, une personne de 30 ans n'est vraiment jeune n'est vraiment vielle.

Dans le cadre de la logique classique (Boolienne) un sous-ensemble A d'un ensemble X est défini par une fonction caractéristique μ_A qui prend la valeur « 0 » pour les éléments de X qui n'appartiennent pas à A, et la valeur « 1 » pour ceux appartenant à A (figure III.1.a), ainsi chaque élément x de l'ensemble X soit qu'il appartient à A où qu'il ne le soit pas.

$$\mu_A(x) : A \longrightarrow \{0,1\}$$

La logique floue est basée sur la notion d'appartenance partielle, ainsi un élément peut appartenir partiellement à un ensemble. Un sous-ensemble floue B est définie par une fonction d'appartenance μ_B qui associé à chaque élément x son degré $\mu_B(x)$, compris entre « 0 » (non-appartenance) et « 1 » (appartenance totale), avec le quel x appartient à B (figure III.1.b)

$$\mu_B(x) : B \longrightarrow [0,1]$$

Cependant, plus que $\mu_B(x)$ tend vers « 1 », plus le degré d'appartenance de x à A est élevé et inversement.

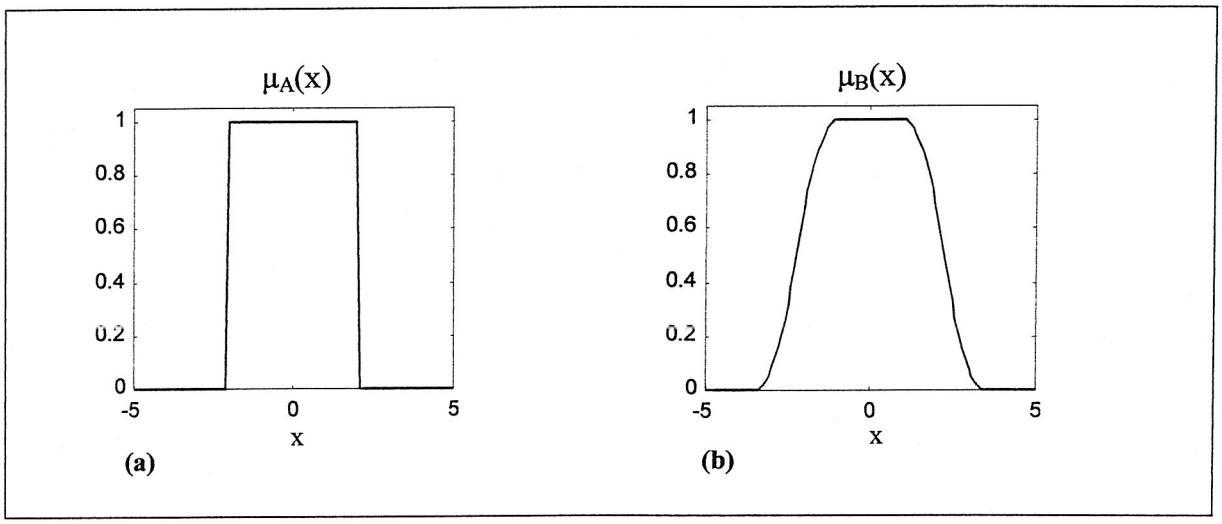


Figure (III.1) : Fonctions d'appartenances
(a) : Cas de la logique classique.
(b) : Cas de la logique floue

Sur la figure (III.2) est présenté un exemple de partitionnement des ages en trois ensembles flous (Jeune, Moyen-age, Vieux), selon la logique floue cette classification peut être faite comme le montre cette figure, cependant une personne de 27 ans appartient à l'ensemble des jeunes avec un degré égale à 0.8, avec un degré de 0.2 à l'ensemble des moyen-ages et avec un degré nul aux vieux.

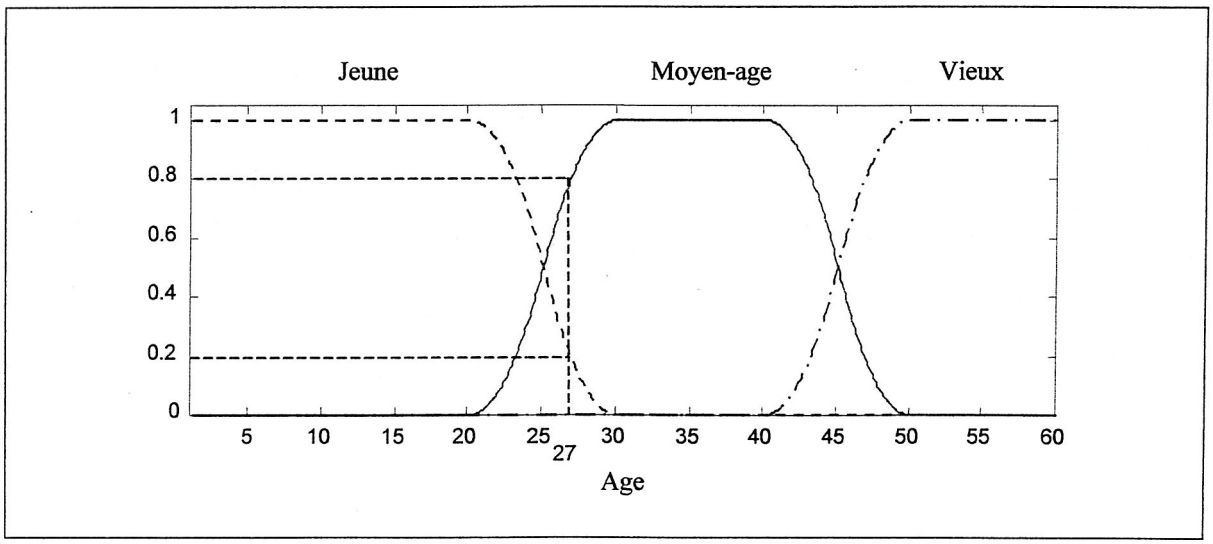


Figure (III.2) : Fonctions d'appartenances caractérisant les sous-ensembles flous : Jeune ; Moyen-age et vieux

III.4 LES OPÉRATEURS DE LA LOGIQUE FLOUE

III.4.1 L'opérateur Non (Complément)

Soit A un sous ensemble de l'ensemble X : L'ensemble complémentaire de A est défini par les élément de X qui n'appartiennent pas à A ($C = Non\ (A) = \overline{A}$). Dans le cadre de la logique floue la fonction d'appartenance de l'ensemble complémentaire est donnée comme suit :

$$\mu_C(x) = Non(A) = 1 - \mu_A(x) \qquad \text{(III.1)}$$

Sur la figure (III.3) sont données les fonctions d'appartenances des compléments d'une fonction gaussienne et une autre triangulaire.

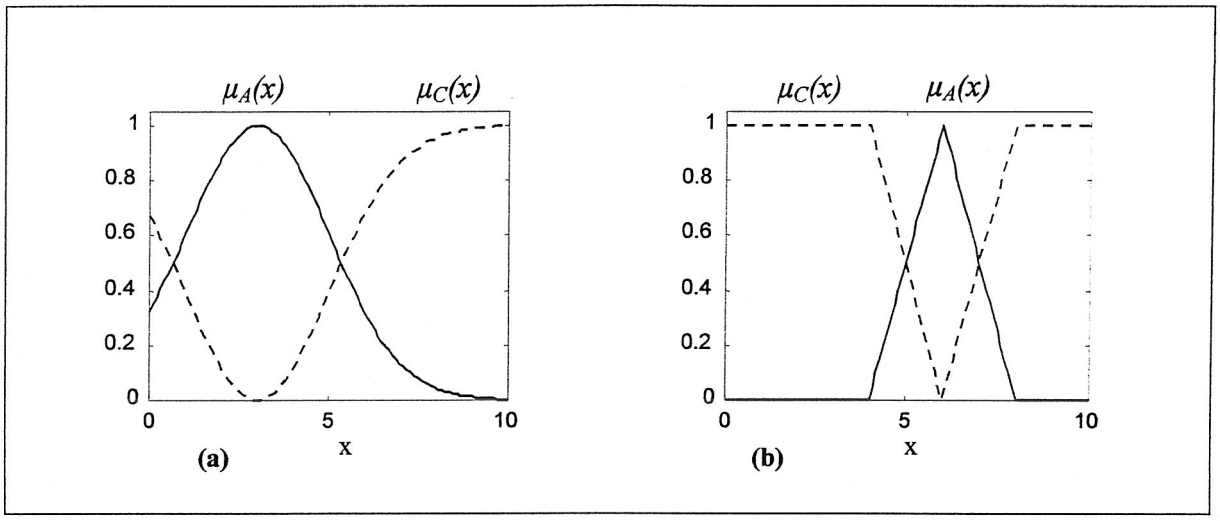


Figure (III.3) : Exemples de l'opérateur Non

III.4.2 L'opérateur Et (Intersection)

Soit A et B deux sous-ensembles, la fonction d'appartenance la plus utilisée (de Zadeh) de l'ensemble C formé par l'intersection de A et B ($C = A\ Et\ B = A \cap B$) est réalisée par le minimum des fonctions d'appartenances μ_A et μ_B, on aura donc :

$$\mu_C(x) = Min\ \{\mu_A(x), \mu_B(x)\} \qquad \text{(III.2.a)}$$

L'opérateur Et peut également être réalisé par le produit de μ_A et μ_B :

$$\mu_C(x) = \mu_A(x)\ \mu_B(x) \qquad \text{(III.2.b)}$$

La figure (III.4) illustre un exemple de l'intersection de deux fonctions d'appartenances.

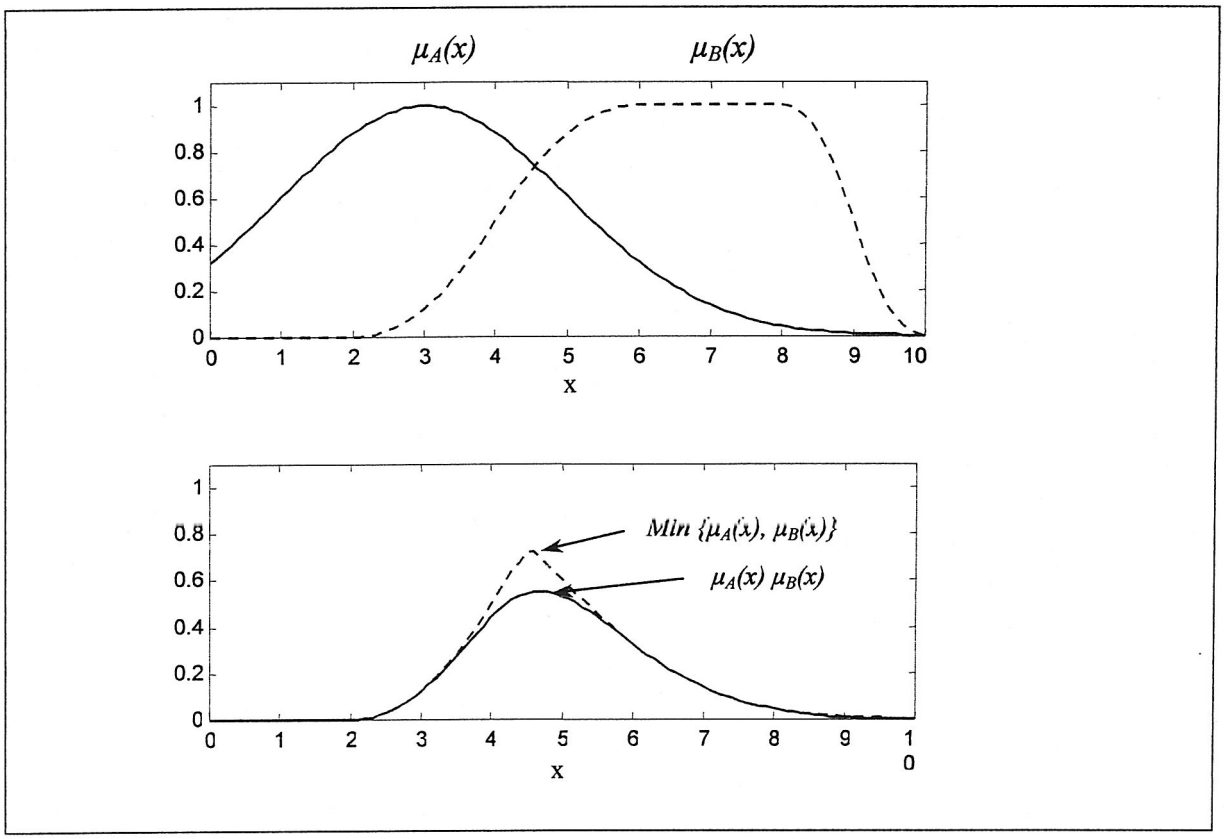


Figure (III.4) : Exemple de l'opérateur Et

La différence entre l'application de l'opérateur Et réalisé par le produit et celui réalisé par le minimum sur deux ensemble floue *A* et *B*, caractérisés par des fonctions d'appartenances gaussiennes, pour deux variables floues *x* et *y*, est illustrée sur la figure (III.5)

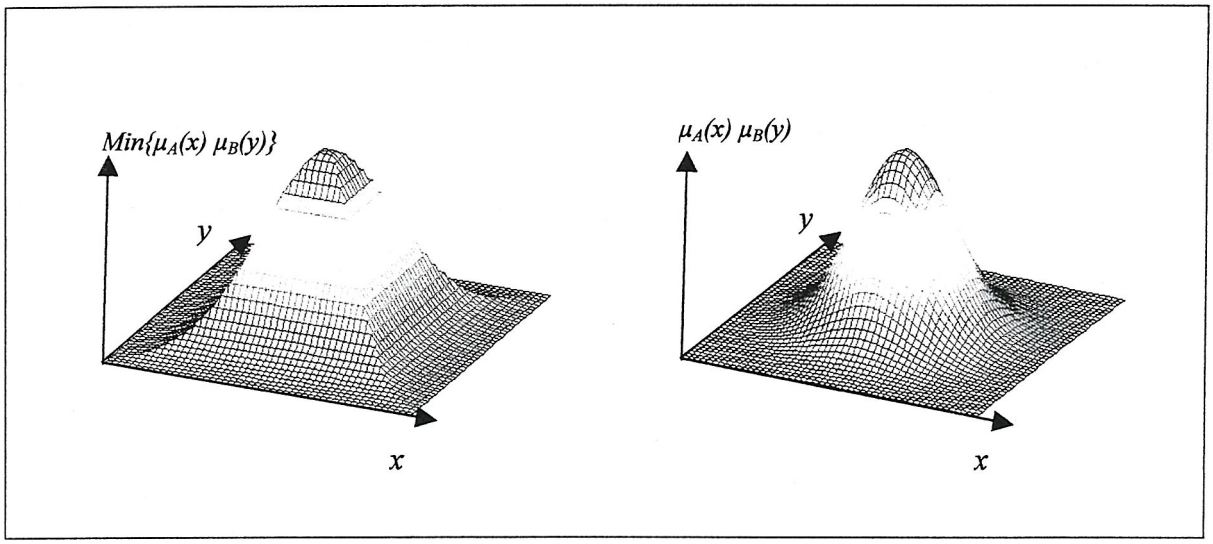


Figure (III.5) : Exemple de l'opérateur Et de deux fonctions d'appartenaces gaussiennes des variables x et y

L'opérateur Et flou est donné par :

$$\lambda \, Min \{\mu_A(x), \mu_B(x)\} + \frac{(1-\lambda)}{2} [\mu_A(x) + \mu_B(x)] \qquad \text{(III.2.c)}$$

Avec λ est une constante appartient à *[0 1]*

III.4.3 L'opérateur Ou (Union)

La fonction d'appartenance la plus utilisée de l'ensemble C formé de l'union de *A* et *B* *(C = A Ou B = A ∪ B)* est réalisée par le maximum des fonctions d'appartenances μ_A et μ_B, on aura donc :

$$\mu_C(x) = Max \{\mu_A(x), \mu_B(x)\} \qquad \text{(III.3.a)}$$

L'opérateur Ou peut également être réalisé par la moyenne de μ_A et μ_B :

$$\mu_C(x) = \mu_A(x) + \mu_B(x) / 2 \qquad \text{(III.3.b)}$$

La figure (III.6) illustre un exemple d'union de deux fonctions caractéristiques.

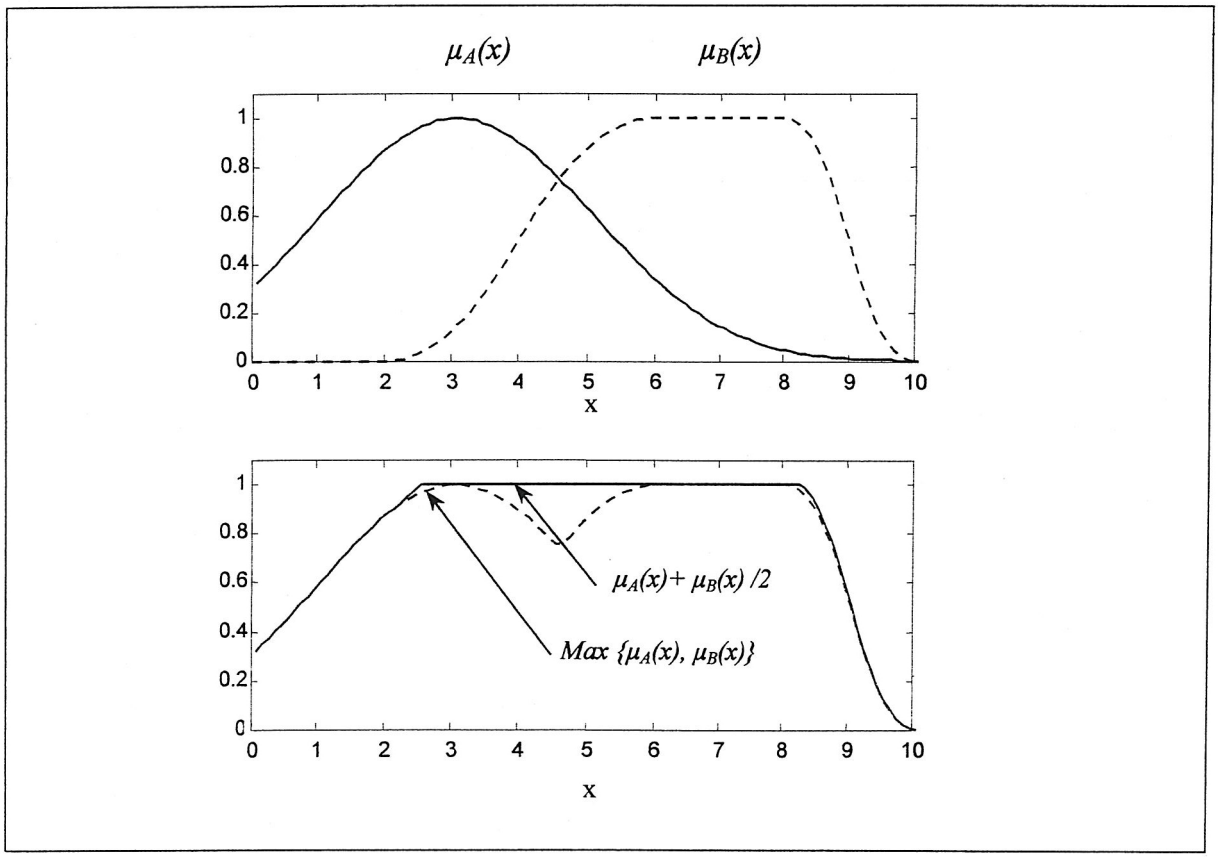


Figure (III.6) : Exemple de l'opérateur Ou

La différence entre l'application de l'opérateur Ou réalisé par la moyenne et celui réalisé par le maximum sur deux ensemble floue A et B, caractérisés par des fonctions d'appartenances gaussiennes sur les mêmes ensembles flous est illustrée sur la figure (III.7)

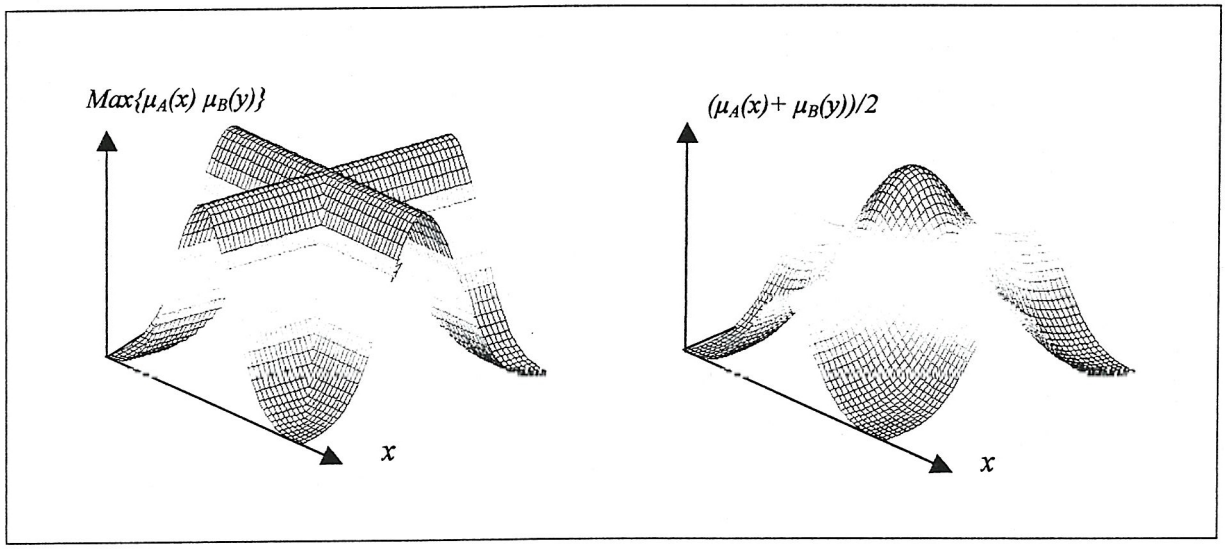


Figure (III.7) : Exemple de l'opérateur Ou de deux fonctions d'appartenaces gaussiennes des variables x et y

L'opérateur Ou flou est donné par :

$$\lambda \, Max \, \{\mu_A(x), \mu_B(x)\} + \frac{(1-\lambda)}{2} [\, \mu_A(x) + \mu_B(x)] \qquad \text{(III.3.c)}$$

Avec λ est une constante appartient à *[0 1]*

III.5 LES RÈGLES SI-ALORS FLOUES

La façon de notre raisonnement est formalisée selon la logique floue par l'utilisation des règles Si-Alors basées sur des expressions floues similaires à notre langage. Par exemple :

***Si** la Résistance est Grande **Alors** le Courant est Faible*

Où le *Courant* et la *Résistance* sont des variables floues, et *Faible* et *Grande* sont des termes linguistiques désignant des sous-ensembles flous qui doivent être caractérisées par des fonctions d'appartenances. Une règle floue se compose de deux parties : i) La prémisse : Formée par une combinaison de prépositions liées entre elles par les opérateurs Et, Ou et Non, dans l'exemple précédent la proposition '*la résistance est grande*' constitue la prémisse de cette règle et ii) La conclusion : C'est la partie conséquence, dans la règle précédente la conclusion est la proposition '*le courant est faible*'.

Une autre forme des règles Si-Alors floues est proposée par Sugeno et Takage [41] : les ensembles flous sont introduits seulement dans la premiére partie (la prémisse), comme :

***Si** la Fréquence est Grande **Alors** la Tension est 220 V*

La conclusion (*Tension*) est numérique.

III.6 SYSTÈME D'INFÉRENCE FLOU (FIS)

Un Système d'Inférence Flou (FIS, Fuzzy Inference System) comporte une base de règles contenant un ensemble des règles Si-Alors Floues et une base de connaissances qui définit les fonctions d'appartenances des ensembles flous. Ce processus est composé de trois étapes (figure III.8): i) Fuzzification : Cette étape consiste à déterminer le degré d'appartenance de chaque variable d'entrée aux états utilisés dans les prémisses des règles, et qui sont décrites par leurs fonctions d'appartenances. C'est une étape délicate et requiert de l'expérience. ii) Inférence : Il s'agit de la détermination des degrés d'activations des règles par la combinaison des propositions de leurs prémisses, ces règles sont prédéfinies dans le système. iii) Défuzzification : Le rôle de cette étape est de fournir de valeurs précises des variables de sorties du système, pour qu'elles puissent être exploitées.

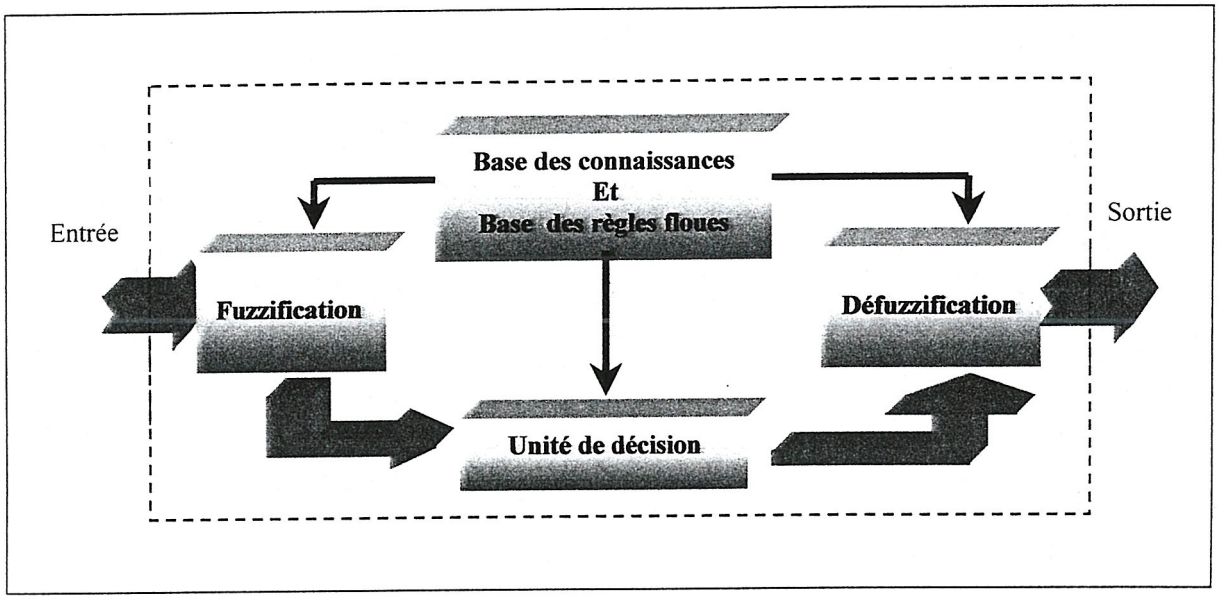


Figure (III.8) : Système d'Inférence Flou (FIS)

III.6.1 Système d'Inférence Flou de Mamdani

Ce système [42] consiste en six étapes :

1. Détermination d'un ensemble des règles floues.
2. Fuzzification des entrées en utilisant les fonctions d'appartenances aux ensembles flous d'entrées.
3. Combinaison des entrées fuzzifiées correspondantes aux règles floues.
4. Détermination des conséquences des règles.
5. Combinaison des conséquences pour avoir la distribution de la sortie.
6. Défuzzification de la distribution de la sortie.

La figure (III.9) met en évidence un exemple de ce système avec deux variables d'entrée x et y. La fuzzification de la variable d'entrée x s'effectue en utilisant deux sous ensembles flous

A_1 et A_2, alors que celle de y s'effectue en utilisant B_1 et B_2. La sortie est déterminée en calculant le centre de gravité de la distribution de sortie déterminée par l'addition des conséquences de règles.

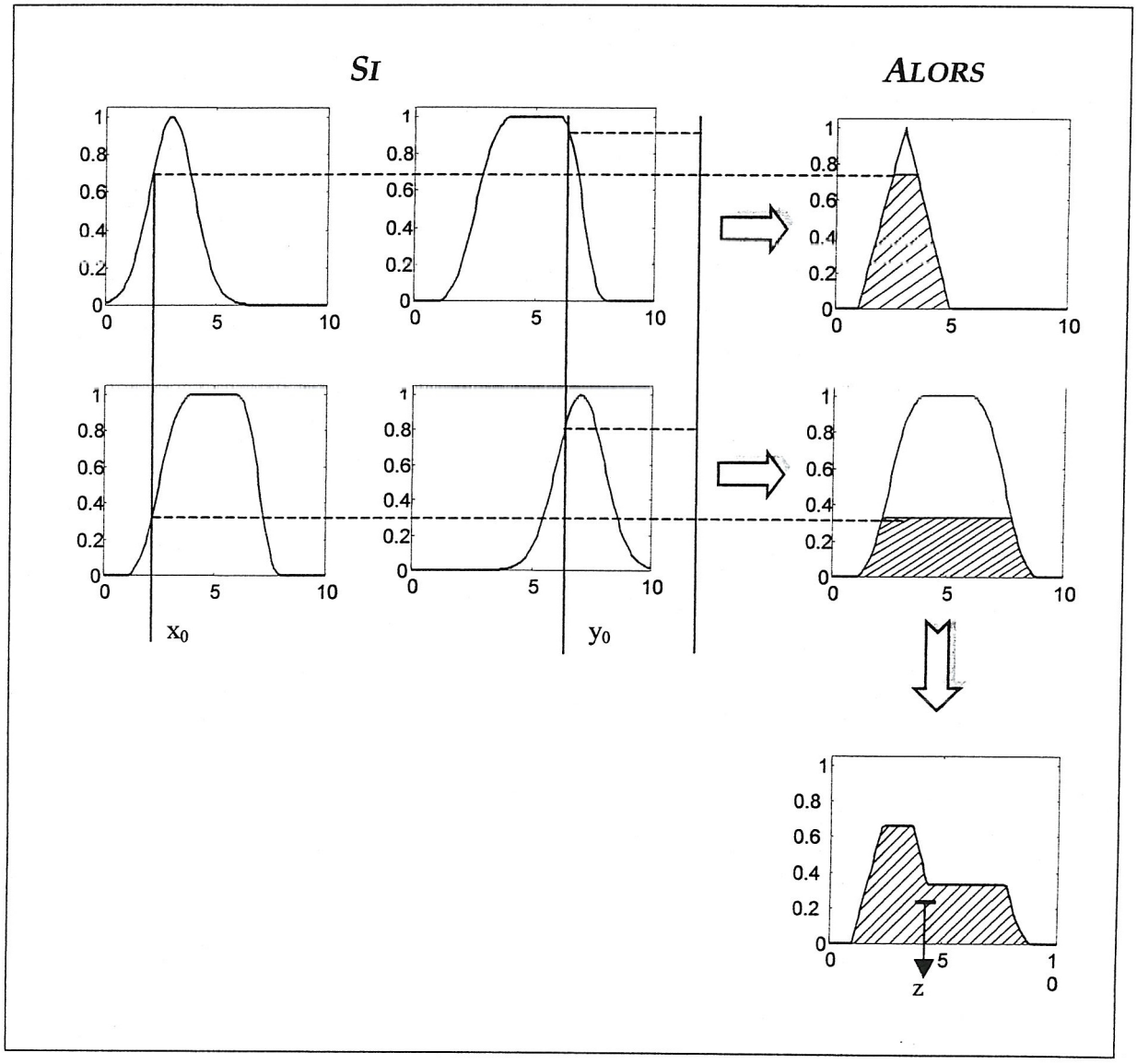


Figure (III.9) : Système d'inférence de Mamdani

III.6.2 Système d'Inférence Flou de Sugeno et Takage

Le système de Sugeno et Takage [41] se diffère à celui de Mamdani sur la façon de détermination de la sortie. Ce système permet d'avoir une sortie numérique déterminée par l'addition des résultats de la multiplication des entrées avec des constantes. Un exemple à deux variables d'entrée et deux règles est, comme suit :

$$\begin{cases} \textbf{R1 :} \ \textbf{Si}\ x \text{ est } A_1 \text{ et } y \text{ est } B_1 \ \textbf{Alors}\ z_1 = p_1x + q_1y + c_1 \\ \textbf{R2 :} \ \textbf{Si}\ x \text{ est } A_2 \text{ et } y \text{ est } B_2 \ \textbf{Alors}\ z_2 = p_2x + q_2y + c_2 \end{cases}$$

Et :

$$\begin{cases} \alpha_1 = \mu_{A1}(x_0) \textbf{ Et } \mu_{B1}(y_0) & \text{(III.4.a)} \\ \alpha_2 = \mu_{A2}(x_0) \textbf{ Et } \mu_{B2}(y_0) & \text{(III.4.b)} \end{cases}$$

Les sorties sont données par :

$$\begin{cases} z_1 = p_1 x + q_1 y + c_1 & \text{(III.5.a)} \\ z_2 = p_2 x + q_2 y + c_2 & \text{(III.5.b)} \end{cases}$$

La sortie du système sera donnée par :

$$z = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} \qquad \text{(III.6)}$$

Cet exemple est illustré sur la figure (III.10)

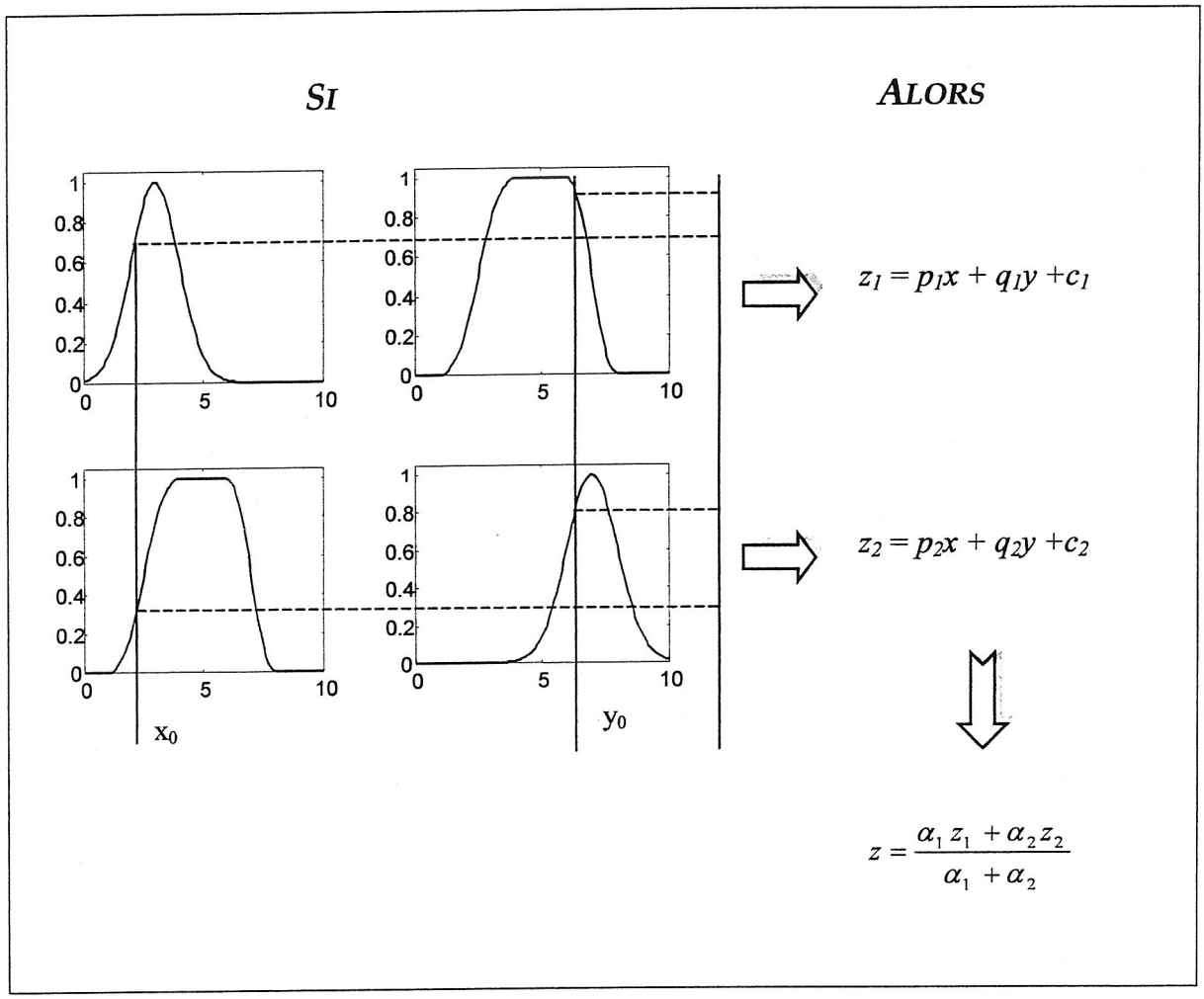


Figure (III.10) : Système d'inférence Flou de Sugeno et Takage

Deuxième Partie :

CLASSIFICATION NEURO-FLOUE

III.7 LES SYSTÈMES NEURO-FLOUS (SNF)

III.7.1 Objectifs des SNF

La création des Systèmes Neuro-Flous (SNF), qui sont formés par l'intégration des reseaux de neurones et des systèmes flous, a pour but de combiner leurs avantages, à savoir: i) L'auto adaptation des ANN en ajustant leurs poids; ii) Leurs puissances de généralisation; iii) Et la possibilité d'utiliser les connaissances à priori pour l'initialisation des paramètres des FIS.

D'autre part la fusion de ces deux technologies complémentaires tente de compenser leurs désavantages, comme: i) L'absence d'une méthode systématique de transformation des connaissances d'un expert humain en base de connaissance d'un FIS ; ii) La nécessité d'une méthode qui permet d'ajuster les paramètres des fonctions d'appartenances d'un FIS ; iii) Il est très difficile de définir une structure à priori des ANN, par exemple il n'existe pas une technique automatique qui permet de déterminer le nombre des neurones cachés d'un MLP ; iv) L'impossibilité d'utiliser les connaissances à priori pour l'initialisation des ANN ; v) La difficulté d'interprétation du modèle réalisé par les ANN après leurs apprentissages.

III.7.2 Architecture des SNF

Les systèmes Neuro-Flous sont le résultat d'incorporation des systèmes flous dans les réseaux de neurones et les ont entraînés avec leurs algorithmes d'apprentissage, ainsi chaque étape des FIS sera effectuée par une couche de neurones (figure III.11)

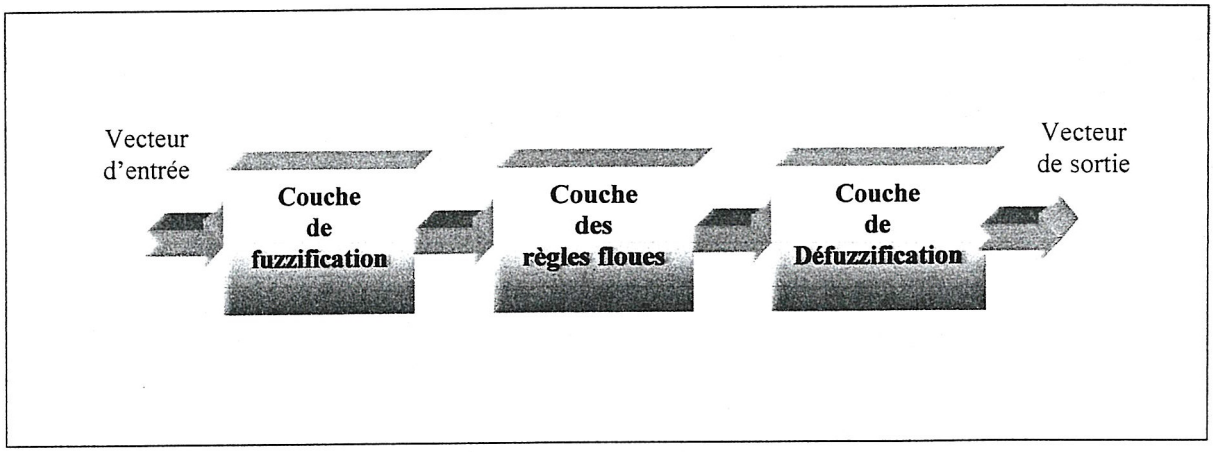


Figure (III.11) : Architecture d'un système Neuro-Flou

III.8 SNF BASÉ SUR LE MODÈLE DE MAMDANI

Ce système se compose de cinq couches (figure III.12) [42] :

La 1ère couche : Le rôle de cette couche est de transmettre les entrées vers la deuxième couche.

La 2ème couche : Chaque neurone de cette couche correspond à un terme linguistique cratérisé par une fonction d'appartenance, son rôle est la détermination des degrés d'appartenances des variables d'entrée aux ensembles flous (la fuzzification).

La 3ème couche : Chaque neurone de cette couche constitue une prémisse d'une règle, il effectue le Et des variables qui lui arrivent.

La 4ème couche : Son rôle est de combiner ces entrées (venus de la troisième couche) et de déterminer le degré d'appartenance aux termes linguistiques de sortie.

La 5ème couche : le rôle de cette couche est de combiner toutes les conséquences des règles et de fournir les sorties déffuzzifieés.

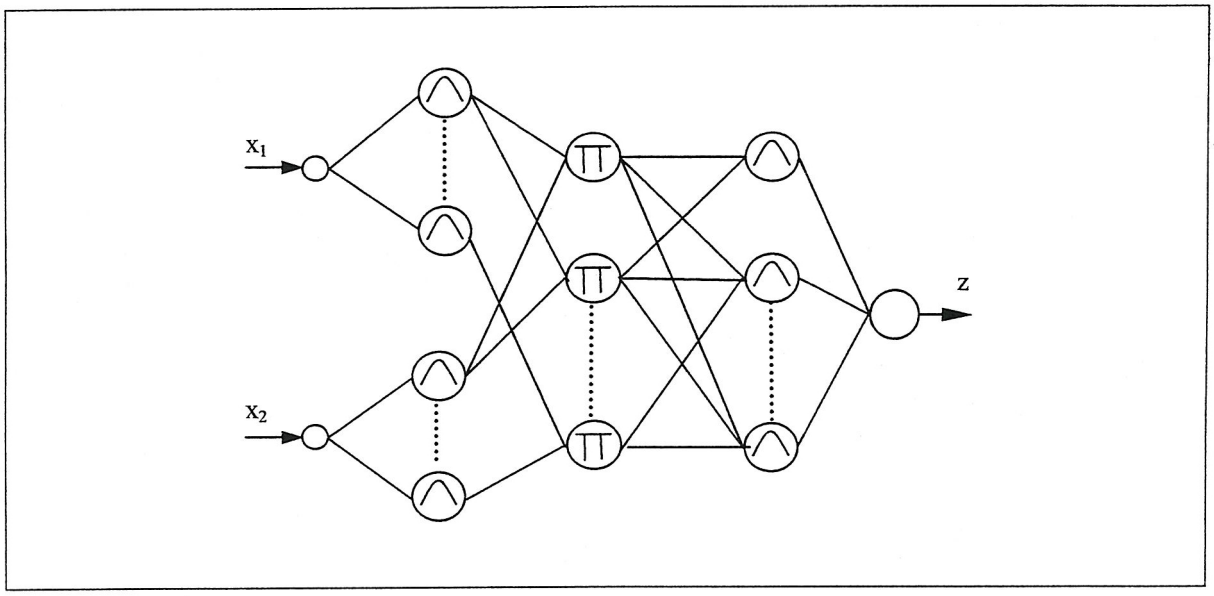


Figure (III.12) : SNF basé sur le modèle de Mamdani

III.9 SNF BASÉ SUR LE MODÈLE DE TAKAGÉ ET SUGENO

Ce modèle se compose de six couches (figure III.13), dont les trois premières sont similaires à celle du modèle précédent, le rôle des trois couches restantes est comme suit [41] :

La 4ème couche : Chaque neurone *i* de cette couche permet de déterminer α_i.

La 5ème couche : Chaque neurone *i* de cette couche permet de déterminer s_i selon la relation :

$$s_i = p_i x + q_i y + c_i \tag{III.7}$$

La 6ème couche : l'unique neurone de cette couche permet de déterminer la sortie *z* du système donnée par :

$$z = \frac{\sum \alpha_i s_i}{\sum \alpha_i} \tag{III.8}$$

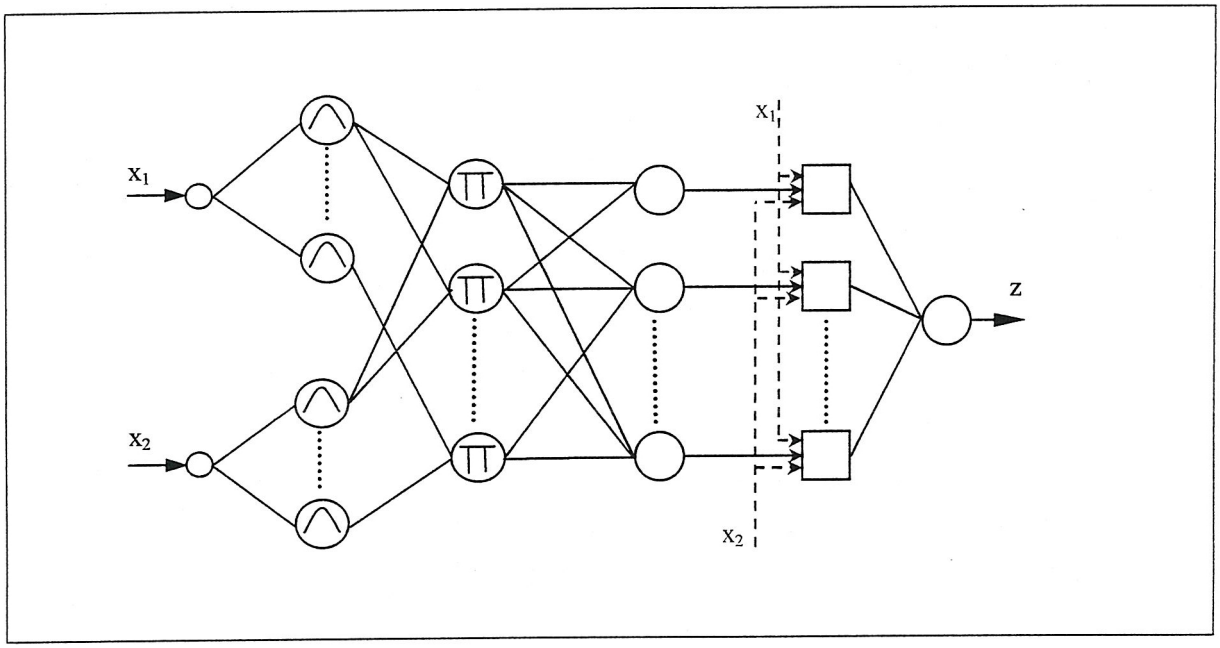


Figure (III.13) : SNF basé sur le modèle de Sugéno et Takagé

III.10 CLASSIFICATEUR NEURO-FLOU (NFC)

III.10.1 Architecture

Le classificateur neuro-flou de Jang [43] [44] (NFC, Neuro Fuzzy Classifier) est une structure désignée à être utilisée dans les applications de la reconnaissance des formes. Le NFC présente l'avantage d'utiliser les connaissances à priori (acquises par les experts humains où les techniques de visualisation) pour initialiser ces paramètres, ainsi il permet de commencer l'apprentissage d'un point d'initialisation non-loin du point optimal [45]. D'autre part, les paramètres obtenus après l'apprentissage peuvent être transformés en une structure basée sur les règles Si-Alors floues.

Le NFC à une architecture basée sur les réseaux de neurones et pouvant comprendre dans sa structure les règle Si-Alors floues utilisées dans la classification, ces règles peuvent être exprimées comme suit :

Si x_1 *est A1 et* x_2 *est B2* ***Alors*** X *est* C

Où x_1 et x_2 sont les variables d'entrée ; $A1$ et $A2$ sont des termes linguistiques caractérisés par leur fonctions d'appartenance ; X est l'objet représenté par x_1 et x_2 ; et C est le nom de la classe.

Sur la figure (III.14) est représenté un NFC avec deux variables d'entrée x_1 et x_2 et deux sorties z_1 et z_2 correspondantes à deux classe C_1 et C_2, il est composé de trois couches permettant d'établir un système de classification basé sur les règle précédentes.

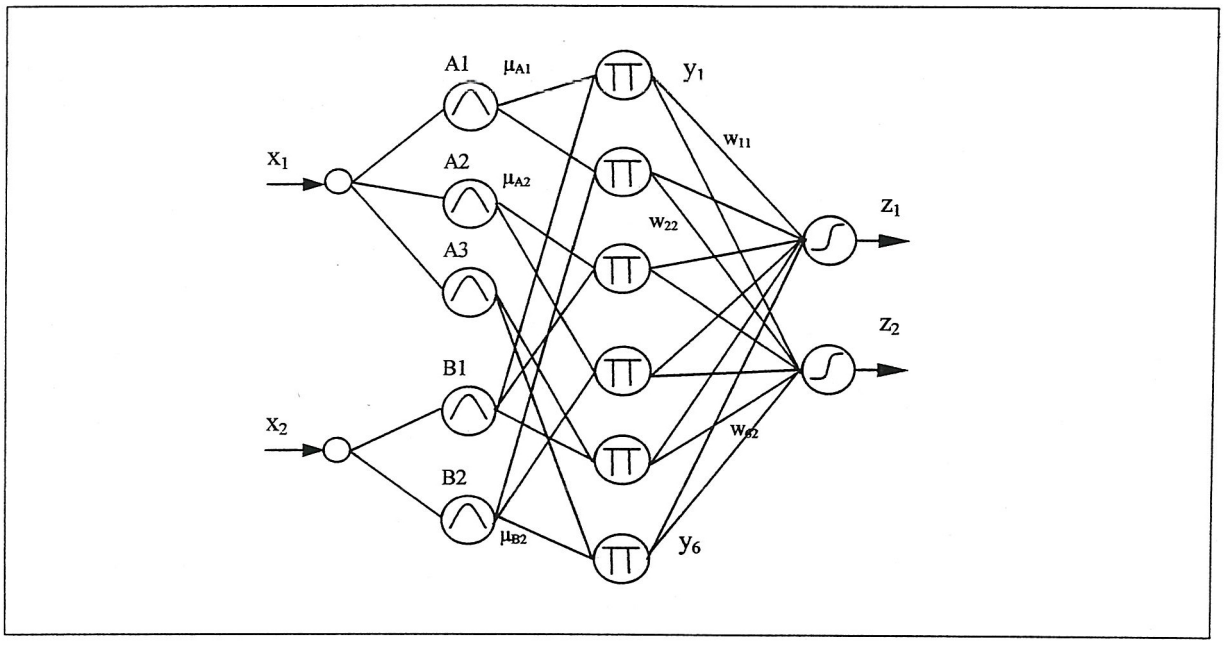


Figure (III.14) : Classificateur neuro-floue avec deux variables d'entrée et deux sorties

1ère couche : Chaque neurone de cette couche correspond à une variable linguistique représentée par une fonction d'appartenance. Dans le cas des fonctions gaussiennes, sa sortie, est de la forme :

$$\mu_{A1}(x_i) = \exp\left(-\left(\frac{x_i - m_{A1}}{a_{A1}}\right)^2\right) \qquad \text{(III.9)}$$

Avec : x_i est la caractéristique arrivant à ce neurone, m_{ij} et a_{ij} les paramètres de la fonction d'appartenance μ_{ij}. Les poids u_{ij} reliant cette couche peuvent être interprétés comme étant a_{ij}.

2ème couche : Les neurones de cette couche fournissent le produit des signaux arrivant (sorties des neurones de la première couche). Par exemple la sortie du 2ème neurone est :

$$y_2 = \mu_{A1}(x_1) \cdot \mu_{B2}(x_2) \qquad \text{(III.10)}$$

Les sorties de ces neurones constituent les prémisses des règles, elles décrivent le degré d'appartenance des caractéristiques d'entrée (x_1, x_2) à la région floue formée par les ensembles floues caractérisée par les termes linguistiques $A1$et $B2$.

3ème couche : chaque classe est représentée par un neurones de cette couche, ainsi le nombre de ces neurones est égale à celui des classes présentes. La sortie de la k^{ème} neurone est :

$$z_k = h\left(\sum_{m=1}^{M} w_{mk} y_m\right) \tag{III.11}$$

Où $h(.)$ est la fonction d'activation (fonction sigmoïde) , donnée par :

$$h(s) = \frac{1}{1+e^{-as}} \tag{III.12}$$

III.10.2 Apprentissage

Ce classificateur peut suivre un apprentissage selon la méthode de descente du gradient [46], en minimisant l'erreur quadratique E donnée par :

$$E = \frac{1}{2}(Z-T)^2 \tag{III.13}$$

Où Z est la sortie calculée et T est la sortie désirée. L'adaptation du poids w_{nj} à l'itération $(i+1)$ est comme suit :

$$w_{mj}^{(i+1)} = w_{mj}^{(i)} - \eta_w \frac{\partial E^{(i)}}{\partial w_{mj}}$$

$$w_{mj}^{(i+1)} = w_{mj}^{(i)} - \eta_w \frac{\partial E^{(i)}}{\partial z_j}\frac{\partial z_j}{\partial w_{mj}}$$

$$= w_{mj}^{(i)} - \eta_w \frac{\partial}{\partial z_j}\left(\sum_{j=1}^{J} z_j - t_j\right)\frac{\partial}{\partial w_{mj}}\left(\sum_{m=1}^{M} w_{mj} y_m\right)$$

$$= w_{mj}^{(i)} - \eta_w (z_j - t_j) h'(s_j) y_m \tag{III.14}$$

La mise à jour des paramètres des fonctions d'appartenances s'effectue par :

$$m_{Ai}^{(i+1)} = m_{Ai}^{(i)} - \eta_m \frac{\partial E^{(i)}}{\partial m_{Ai}} \tag{III.15.a}$$

$$a_{Ai}^{(i+1)} = a_{Ai}^{(i)} - \eta_m \frac{\partial E^{(i)}}{\partial a_{Ai}} \tag{III.15.b}$$

III.10.3 Exemple d'application

Soit à classifier les exemples illustrés sur la figure (III.15), ils appartiennent à deux classes C_1 et C_2, et ils sont représentés par deux caractéristiques x_1 et x_2. On effectue leur classification avec un NFC ayant l'architecture suivante : i) Une couche d'entrée composée de deux neurones vu que les exemples à classifier sont représentés par deux caractéristiques. ii) Suite à l'emplacement de ces exemples on peut définir les sous-ensembles flous représentés sur la figure, ces derniers sont caractérisés par les termes linguistiques TP, P, G et TG (très petit, Petit, grand et Très Grand) pour la première caractéristique et TP, P, G_1, G_2, TG_1, TG_2 pour la deuxième, ainsi la deuxième couche sera composée de six neurones. iii) Chaque neurone de la troisièmes couche définira une règle, ainsi le nombre sera égale à 24, iv) Finalement la couche de sortie sera composée de deux neurones chacun d'eux correspond à l'une des deux classes ($Z(C_1) = [1\ 0]$ et $Z(C_2 = [0\ 1]$)

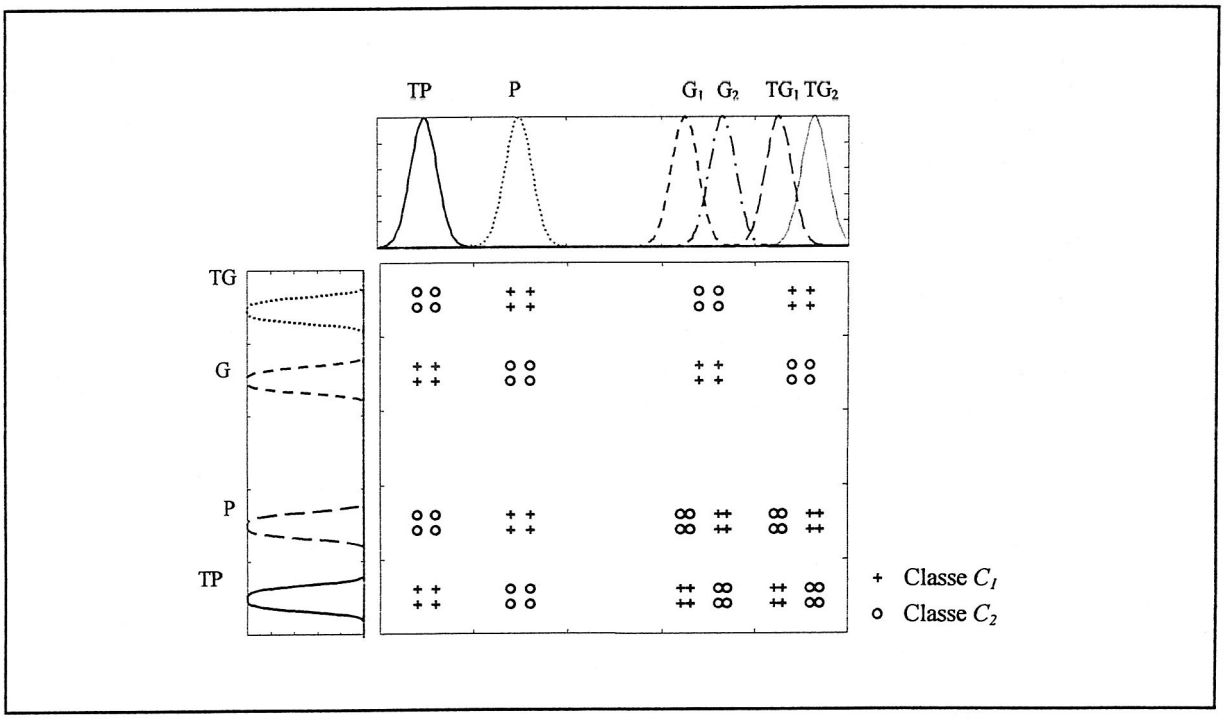


Figure (III.15) : Exemple de classification

Ce classificateur permet d'établir, par ça structure, les règles Si-Alors floues de la forme :

Si x_1 *est TP et* x_2 *est G* ***Alors*** *X(x1 x2) appartient à la classe* C_1

Ou plus précisément :

Si x_1 *est TP et* x_2 *est G* ***Alors*** $Z(X) = [1\ 0]'$

Dans ce cas (avec ces ensembles flous) ce classificateur arrive à séparer l'ensemble des exemples après une itération d'apprentissage, sans avoir ajuster les paramètres des fonctions d'appartenances.

CHAPITRE IV:

Classification avec apprentissage Étiqueté

IV.1 INTRODUCTION

Les objets d'une population peuvent être représentés par des différents attributs, la distinction de quelque sous-populations est plutôt simple qu'elle nécessite un seul attribut, mais la séparation d'autres sous-populations est plus difficile et exige plus d'attributs, par exemple pour la classification d'un ensemble de fruits la couleur permit de distinguer les oranges des citrons, mais ne permit pas de différencier les citrons des pamplemousses. En effet les performances de la classification dépendent, d'une manière intense, sur la représentation des objets à reconnaître, par exemple le perceptron serait rapidement entraîné lorsque les caractéristiques sont choisies de telle sorte que les classes soient linéairement séparables dans l'espace caractéristique. Ainsi, l'approche qu'on propose révèle de cette idée, de rendre les classes linéairement séparables pour faciliter l'apprentissage, et ce par l'ajout d'une caractéristique additionnelle (les étiquettes).

Ce chapitre est consacré à la mise au point de cette approche. Ainsi, dans ces paragraphes on présente les différentes étapes pour effectuer la classification avec apprentissage étiqueté, et les performances de son application avec le Perceptron Multi Couche (MLP, Multi Layered Perceptron), le RVFLNN (Random Vector Functional Link Neural Network) et le Classificateur neuro-flou (NFC, Neuro Fuzzy Classifier).

IV.2. CLASSIFICATION AVEC APPRENTISSAGE ÉTIQUETÉ

IV.2.1 Méthodologie de l'approche

La classification avec apprentissage étiqueté est une technique qui rentre dans le cadre la reconnaissance des formes par les réseaux de neurones, elle est basée sur la modification de la représentation des objets à classer en leur ajoutant une caractéristique additionnelle (les étiquettes) afin d'accélérer la phase d'apprentissage et d'améliorer les performances de classification.

La classification avec apprentissage étiqueté consiste en trois étapes : à modifier la représentation des exemples d'apprentissage, en suite d'entraîner le réseau en se basant sur la nouvelle représentation, et en fin d'effectuer des tests pour classer chaque nouvel exemple.

Soit un ensemble de $N+M$ exemples, dont N exemples appartiennent à la classe C_1 et M à la classe C_2, et chaque exemple étant représenté par deux caractéristiques. En ajoutant les étiquettes, la représentation de ces exemples est donnée sur le tableau (IV.1) :

Exemple	Caractéristiques sans étiquettes	Caractéristiques avec étiquettes	Classe
1	$(x_{1,1}\ \ x_{1,2})$	$(x_{1,1}\ \ x_{1,2}\ \ L_1)$	C_1
2	$(x_{2,1}\ \ x_{2,2})$	$(x_{2,1}\ \ x_{2,2}\ \ L_1)$	C_1
...	...	...	...
N	$(x_{N,1}\ \ x_{N,2})$	$(x_{N,1}\ \ x_{N,2}\ \ L_1)$	C_1
N+1	$(x_{N+1,1}\ x_{N+1,2})$	$(x_{N+1,1}\ x_{N+1,2}\ L_2)$	C_2
N+2	$(x_{N+2,1}\ x_{N+2,2})$	$(x_{N+2,1}\ x_{N+2,2}\ L_2)$	C_2
...	...	...	...
N+M	$(x_{N+M,1}\ x_{N+M,2})$	$(x_{N+M,1}\ x_{N+M,2}\ L_2)$	C_2

Tableau (IV.1) : Données sans et avec étiquettes

Après avoir entraîné le réseau en se basant sur des exemples d'apprentissage normalisés (avec les étiquettes), le modèle neuronal construit est alors basé sur la nouvelle représentation, il est adapté à classifier des exemples ayant cette caractéristique additionnelle. Pour que ce classificateur puisse effectuer sa tache, chaque exemple à classifier doit avoir une étiquette. On propose donc, pour classifier un nouvel exemple, de le présenter au réseau à chaque fois avec une étiquette (L_1 puis L_2), ensuite de déterminer pour chaque classe la différence entre la sortie calculée et la sortie désirée. On aura :

$$E_1(X) = \textit{somme quadratique } \{Z(C_1) - Z_1(X)\} \quad \text{(IV.1.a)}$$
$$E_2(X) = \textit{somme quadratique } \{Z(C_2) - Z_2(X)\} \quad \text{(IV.1.b)}$$

Avec $Z(C_i)$ et $Z_i(X)$ sont respectivement la sortie désirée et la sortie calculée correspondantes à la classe C_i. En fin, cet exemple sera attribuer à la classe ayant la plus petite erreur. La règle de décision est :

$$X \in C_1 \ Si \ E_1(X) < E_2(X) \quad \text{(IV.2.a)}$$
$$X \in C_2 \ Si \ E_2(X) < E_1(X) \quad \text{(IV.2.b)}$$

Les sorties des réseaux de neurones peuvent être considérées comme les probabilités à posteriori d'appartenance aux classes présentes [29]. En appliquant l'apprentissage étiqueté on a exploité cette intéressante propriété des réseaux de neurones.

Dans le cas multi classes (K classes), on fait correspondre chaque classe C_k avec une étiquette L_k (soit K étiquettes pour l'ensemble des classes). On ajoute à chaque exemple d'apprentissage l'étiquette correspondante à sa classe, on effectue l'apprentissage et pour classer un nouvel exemple X on détermine pour chaque classe l'erreur $E_i(X)$ donnée par :

$$E_i(X) = somme\ quadratique\ \{Z\ (C_i) - Z_i(X)\} \qquad \text{(IV.3)}$$

Avec $Z(C_i)$ et $Z_i(X)$ sont respectivement la sortie désirée et la sortie calculée correspondantes à la classe C_i.

La règle de décision est :

$$X \in C_i \ \ Si \ \ E_i(X) = min\ \{E_1(X), E_2(X), ..., E_K(X)\} \qquad \text{(IV.4)}$$

IV.2.2 Algorithme de la classification avec l'Apprentissage Étiqueté

Pour un ensemble de (Q) exemple d'apprentissage appartenant au (K) classes et avec un ensemble de (Q_t) exemples de test, le processus de la classification avec apprentissage étiqueté sera selon l'algorithme suivant :

Algorithme (IV.1)

Etape 1 : Ajout des étiquettes

Pour $q = 1$ jusqu'à Q
  Pour $k = 1$ jusqu'à K
    Si la classe de l'exemple (q) **est** k
    Alors Lui ajouter l'étiquette L_k

Etape 2 : Apprentissage du réseau

Etape 3 : Test des nouveaux exemples :

Pour $q = 1$ jusqu'à Q_t
  Pour $k = 1$ jusqu'à K
    - Ajouter l'étiquettes L_k
    - Simulation du réseau
    - $Erreur\ (k) = somme\text{-}quadratique(Z(C_k) - Z_K)$
    - $E_{min} = minimum\ (Erreur)$

  Pour $k = 1$ jusqu'à K
    Si $Erreur\ (k)$ **est** E_{min}
    Alors Résultat $(q) = k$

Fin

La classification avec apprentissage étiqueté par les réseaux de neurones peut être représentée par le schéma présenté sur la figure (IV.1) :

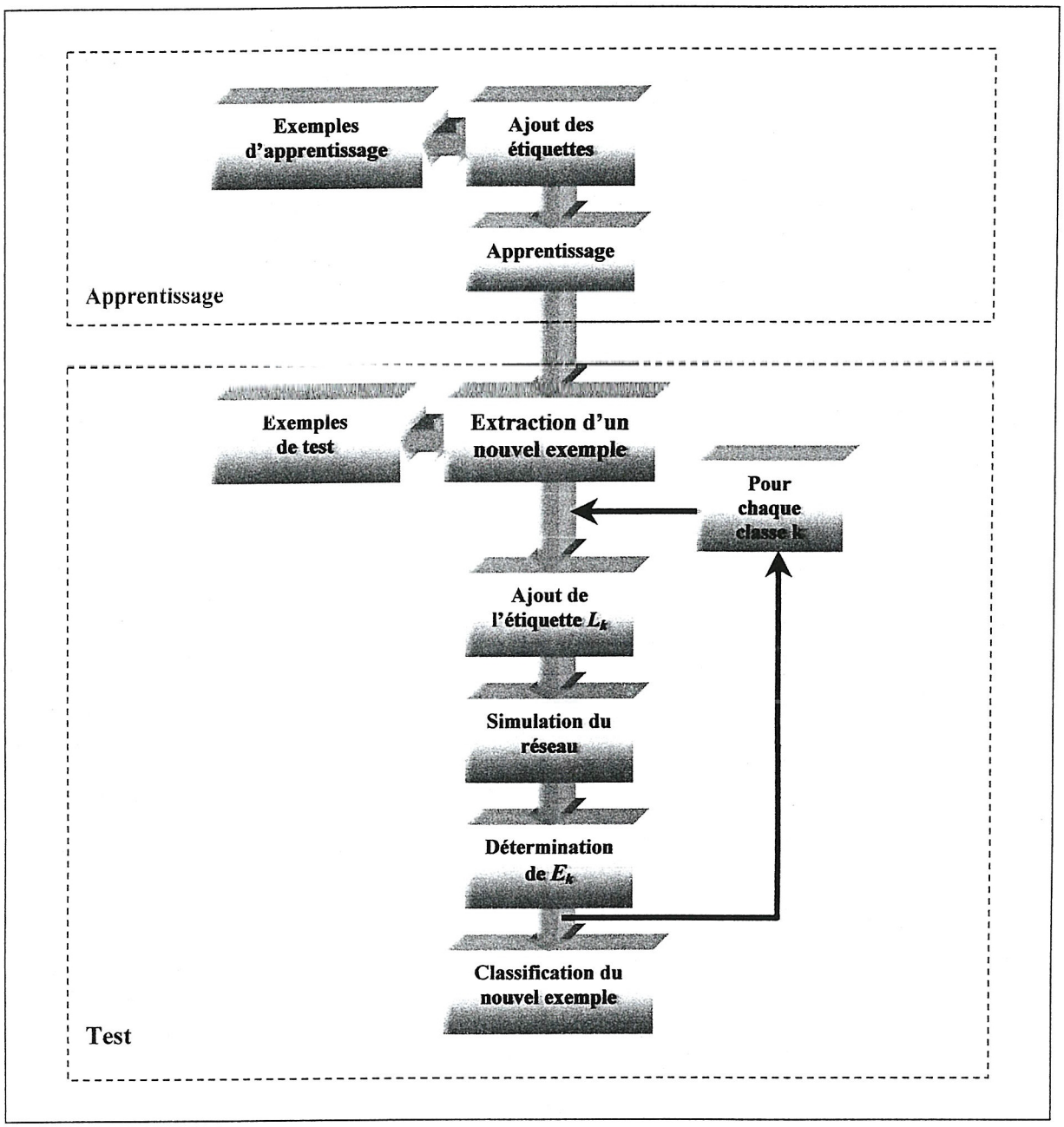


Figure (IV.1) : Classification avec apprentissage étiqueté

IV.2.3 Choix des étiquettes

Le choix d'un ensemble d'étiquettes ayant des valeurs écartées permet d'avoir une accélération importante de l'apprentissage, mais risque de dégrader les capacités de généralisation du réseau, et au contraire un choix de valeurs proches permet de le garder mais n'accélère pas l'apprentissage, ainsi l'ensemble de leurs valeurs est un compromis entre l'accélération et la généralisation. En effet, les valeurs des étiquettes ne doivent pas être loin des valeurs des autres caractéristiques, et vu que ces dernières s'échelonnent

généralement entre 0 et 1 (ou 0.1 et 0.9), on propose de choisir un ensemble d'étiquettes autour de 0.5 avec un écart *(σ)* entre eux, par exemple pour deux classes ces valeurs seront $L_1=0.5-\sigma/2$ *et* $L_2=0.5+\sigma/2$, pour trois classes $L_1=0.5-\sigma$, $L_2=0.5$ *et* $L_3=0.5+\sigma$, et ainsi de suite.

IV.3 APPLICATION AVEC LE MLP

La classification avec apprentissage étiqueté par le MLP consiste a lui entraîné après avoir ajouter les étiquettes à touts les exemples d'apprentissage, en suite, pour classer un nouvel exemple, à effectuer des tests avec chacune des étiquettes présentes [47]. L'application de cette approche avec le MLP modifie son architecture en ajoutant un neurone (correspondant à la caractéristique additionnelle) à la couche d'entrée (figure IV.2).

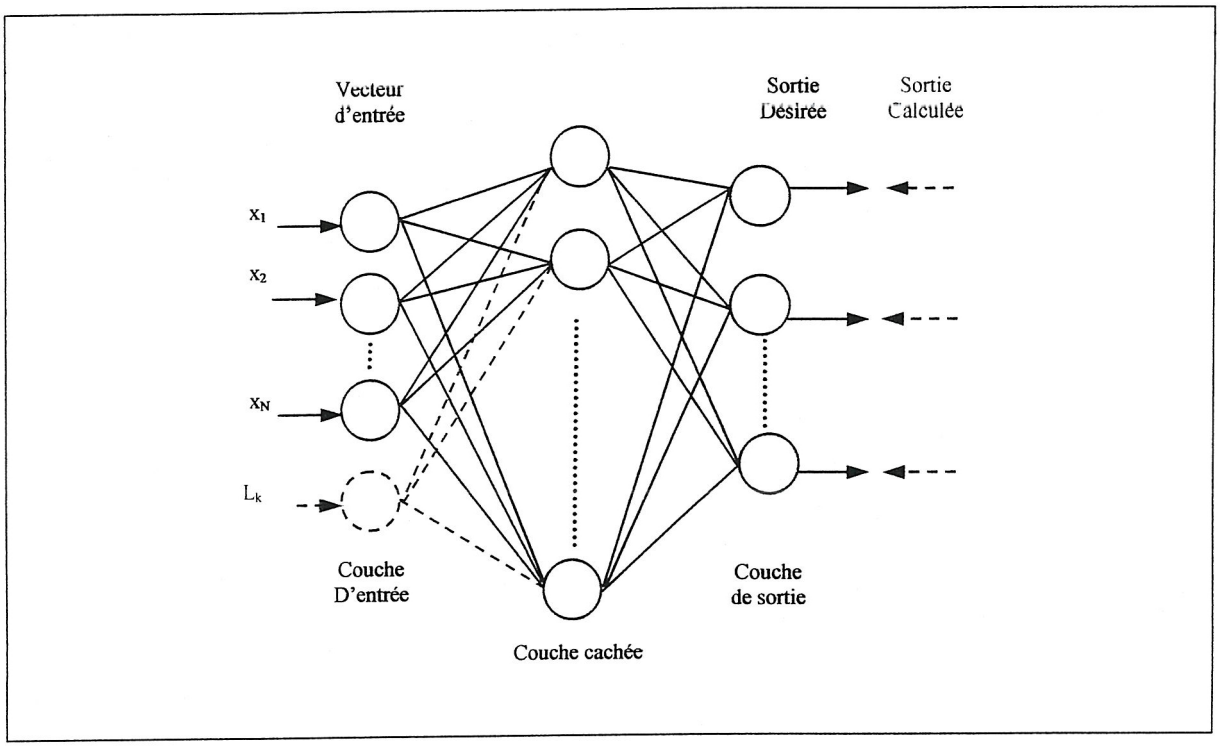


Figure (IV.2) : MLP avec apprentissage étiqueté

La caractéristique additionnelle L_k sera traitée comme les autres caractéristiques, le vecteur X sera :

$$X(x_1 \ x_2 \ldots x_N \ L_k)$$

La $m^{ème}$ sortie de la couche cachée sera :

$$y_m = h\left(\sum_{n=1}^{N} x_n w_{nm} + L_k w_{N+1,m} \right) \qquad \text{(IV.5)}$$

Les sorties de la couche de sortie restent inchangées, la sortie de la $j^{ème}$ neurone est :

$$z_j = g\left(\sum_{m=1}^{M} y_m . u_m \right) \qquad \text{(IV.6)}$$

IV.3.1 Exemple de classification

Pour apprécier l'application de cette approche avec le MLP, soit à classer la base de données représentée sur la figure (IV.3), Cette dernière est constituée de 16 exemples appartenant aux deux classes (C_1 *et* C_2) non linéairement séparables, 12 d'entre eux seront utilisés pour l'apprentissage et les 4 restants seront réservés pour le test. On réalise la classification de cette base de données avec un MLP d'architecture 2-4-2, sans et avec étiquettes. L'algorithme d'apprentissage du MLP est basé sur le full propagation [1], l'ajustement des gains est effectué avec la règle delta bar delta [37], les paramètres des sigmoïdes sont constants et les poids initiaux sont générés par l'algorithme de Nguyen-Widrow-Russo [35][36]. On grade les mêmes poids initiaux pour l'apprentissage sans et avec étiquettes, les poids additionnels correspondant aux étiquettes sont mis à *« 0 »*.

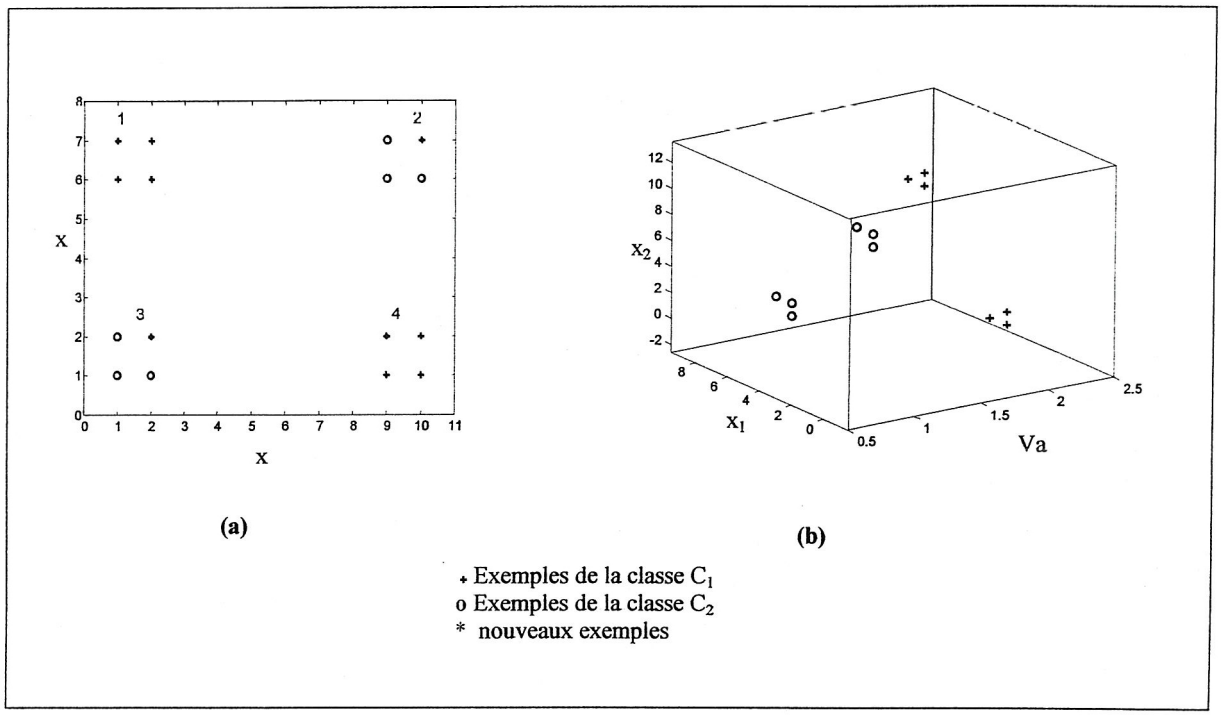


Figure (IV.3) : Représentation des exemples d'apprentissage et des exemples de test
(a) : Caractéristiques sans étiquettes
(b) : Caractéristiques avec étiquettes

Les résultats de classification obtenus (les sorties du réseau, les erreurs correspondantes à chaque étiquette et la classe attribuée à chaque exemple) sont donnés sur le tableau (IV.2). Les sorties désirées étant $Z(C_1) = [0.9\ 0.1]$ pour la classe C_1 et $Z(C_2) = [0.1\ 0.9]$ pour la classe C_2. Les étiquettes utilisées sont : $L_1 = 0.475$ *et* $L_2 = 0.525$ $(\delta=0.05)$.

Exemple	Z_1	Z_2	E_1	E_2	Classe
1	(0.932 0.073)	(0.931 0.074)	0.002	1.373	C_1
2	(0.842 0.163)	(0.838 0.166	0.007	1.083	C_1
3	(0.250 0.749)	(0.154 0.845)	0.844	0.006	C_2
4	(0.158 0.841)	(0.090 0.903)	1.100	0.000	C_2

Tableau (IV.2) : Résultat de la classification avec un MLP

D'après les résultats précédents, on constate que la sortie fournie par le réseau indique la classe correcte pour chaque exemple soit avec l'étiquette correspondante ou l'autre, ainsi l'utilisation de ces deux étiquettes accorde une simplicité de décision. Par exemple pour le premier cas :

$$Z_1(X^{(1)}) = (0.932\ 0.073) \qquad (pour\ L_1)$$
$$Z_2(X^{(1)}) = (0.931\ 0.074) \qquad (pour\ L_2)$$

En terme de probabilité, le réseau fourni pour cet exemple les probabilité suivantes :

$$P(C_1/X^{(1)}) = 0.932 \text{ et } P(C_2/X^{(1)}) = 0.073 \qquad (pour\ L_1)$$
$$P(C_1/X^{(1)}) = 0.931 \text{ et } P(C_2/X^{(1)}) = 0.074 \qquad (pour\ L_2)$$

IV.3.2. Effet du choix des étiquettes :

Pour apercevoir l'effet de changement des valeurs des étiquettes, en effectue la classification de cette base de données avec les étiquettes $L_1=0.5-\sigma/2$ et $L_2=0.5+\sigma/2$, pour différentes valeurs de σ, les résultats obtenues sont donnés sur le tableau (IV.3). Pour chaque valeur de σ on réalise quatre exécutions (en gardant à chaque fois les même poids initiaux pour les deux modes d'apprentissage).

L_1	L_2	σ	Itérations nécessaires pour avoir un taux de 100%			
			Exécution			
			1	2	3	4
0.450	0.550	0.10	150	78	85	78
0.550	0.450	0.10	134	77	214	102
0.475	0.525	0.05	130	86	95	95
0.525	0.475	0.05	125	82	98	113
0.400	0.600	0.20	100	85	75	91
0.600	0.400	0.20	157	81	164	108
Sans étiquettes			337	173	125	112

Tableau (IV.3) : Résultat de classification avec un MLP pour différentes étiquettes

D'après le tableau précédent : Dans les trois premières exécutions, il est nettement apparent que l'application de l'apprentissage étiqueté accélère l'apprentissage pour toutes les valeurs choisies des étiquettes. Tandis que pour la quatrième, le cas ou le MLP est rapidement entraîné, l'application de l'apprentissage étiqueté garde cette propriété. D'autre part, il est également apparent que le choix des étiquettes avec $\sigma = 0.05$ (soient $L_1=0.475$ et $L_2=0.525$) donne les meilleurs résultats. Sur les figures (IV.4), sont illustrés l'évolution au cours de l'apprentissage de l'erreur quadratique moyenne et le taux de classification correspondant à l'apprentissage sans étiquettes et à l'apprentissage étiqueté pour les quatre exécutions précédentes avec les étiquettes ($L_1=0.475$ et $L_2=0.525$).

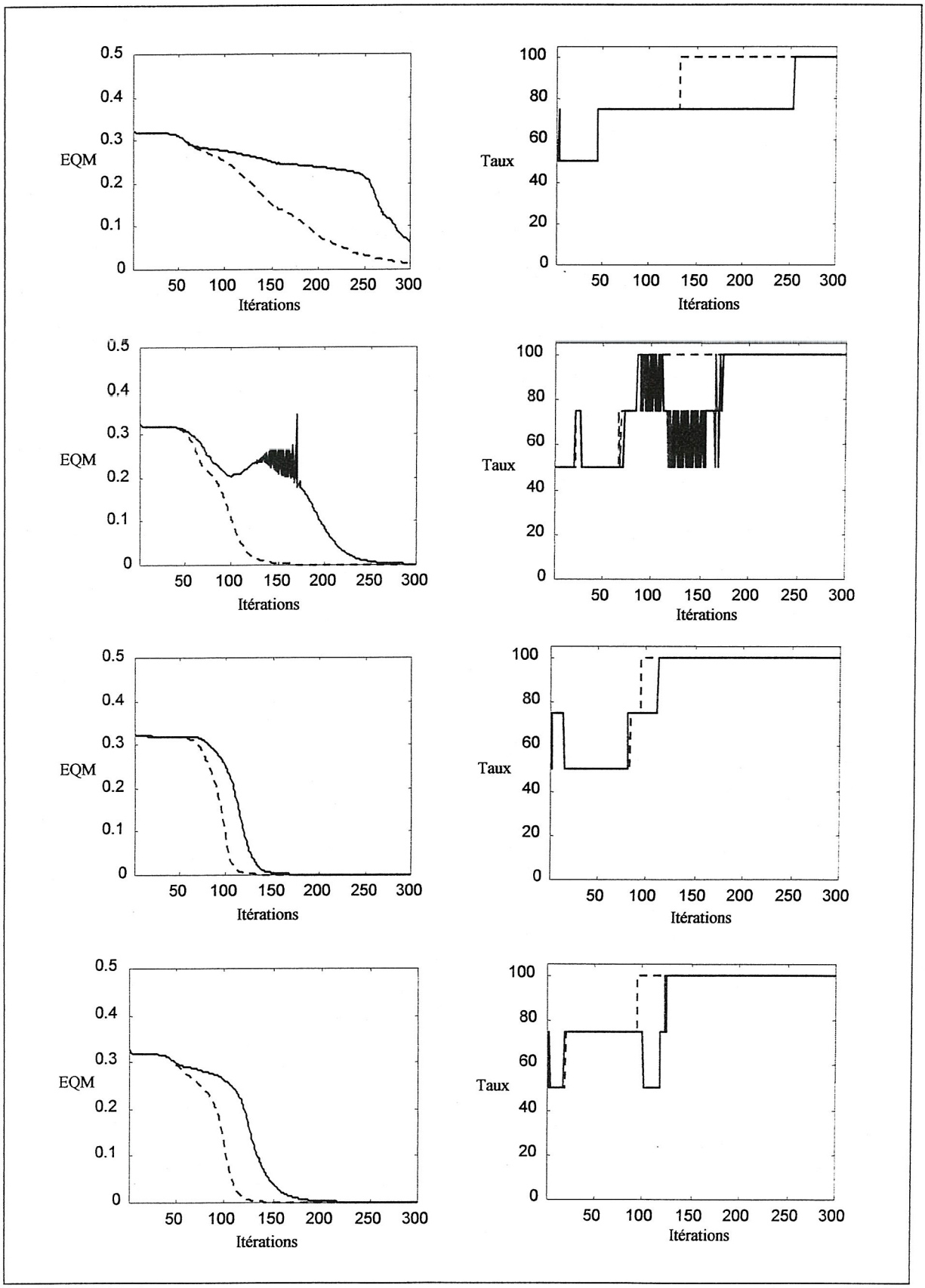


Figure (IV.4) : Evolution de l'erreur Quadratique Moyenne Et du Taux de classification des exemples de test pour les quatre exécutions

D'une manière générale, l'application de l'apprentissage étiqueté accélère l'apprentissage du réseau en lui permettant de sortir des minima locaux rapidement comme le montre la figure (IV.5), qui représente l'évolution de l'Erreur Quadratique Moyenne, au cours de l'apprentissage pour trois exécutions, sans étiquettes et avec apprentissage étiqueté.

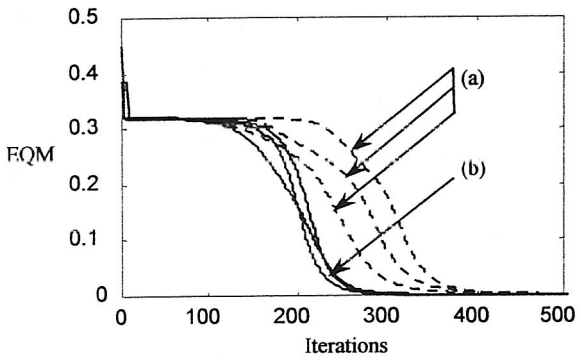


Figure (IV.5) : Evolution de l'erreur d'apprentissage pour trois exécutions
(a) : Apprentissage sans étiquettes
(b) : Apprentissage étiqueté

IV.4 APPLICATION AVEC LE RVFLNN

L'application de l'apprentissage étiqueté avec le RVFLNN (Random Functional Link Network) est similaire à celle avec le MLP [48]. Elle consiste à ajouter les étiquettes aux exemples d'apprentissage et à effectuer des tests avec chacune des étiquettes pour classer un nouvel exemple.

Le vecteur caractéristique devient de la forme :

$$X(x_1 \; x_2 \ldots x_N \; L_k)$$

La $m^{ème}$ sortie de la couche cachée sera :

$$y_m = h(r_m) = h\left(\sum_{n=1}^{N} x_n w_{nm} + L_k w_{N+1,m} \right) \qquad \text{(IV.7)}$$

La $j^{ème}$ sortie du réseau sera :

$$z_j = g(s_j) = g\left(\sum_{n=1}^{N} x_n v_{nj} + \sum_{m=1}^{M} y_m u_{mj} \right) \qquad \text{(IV.8)}$$

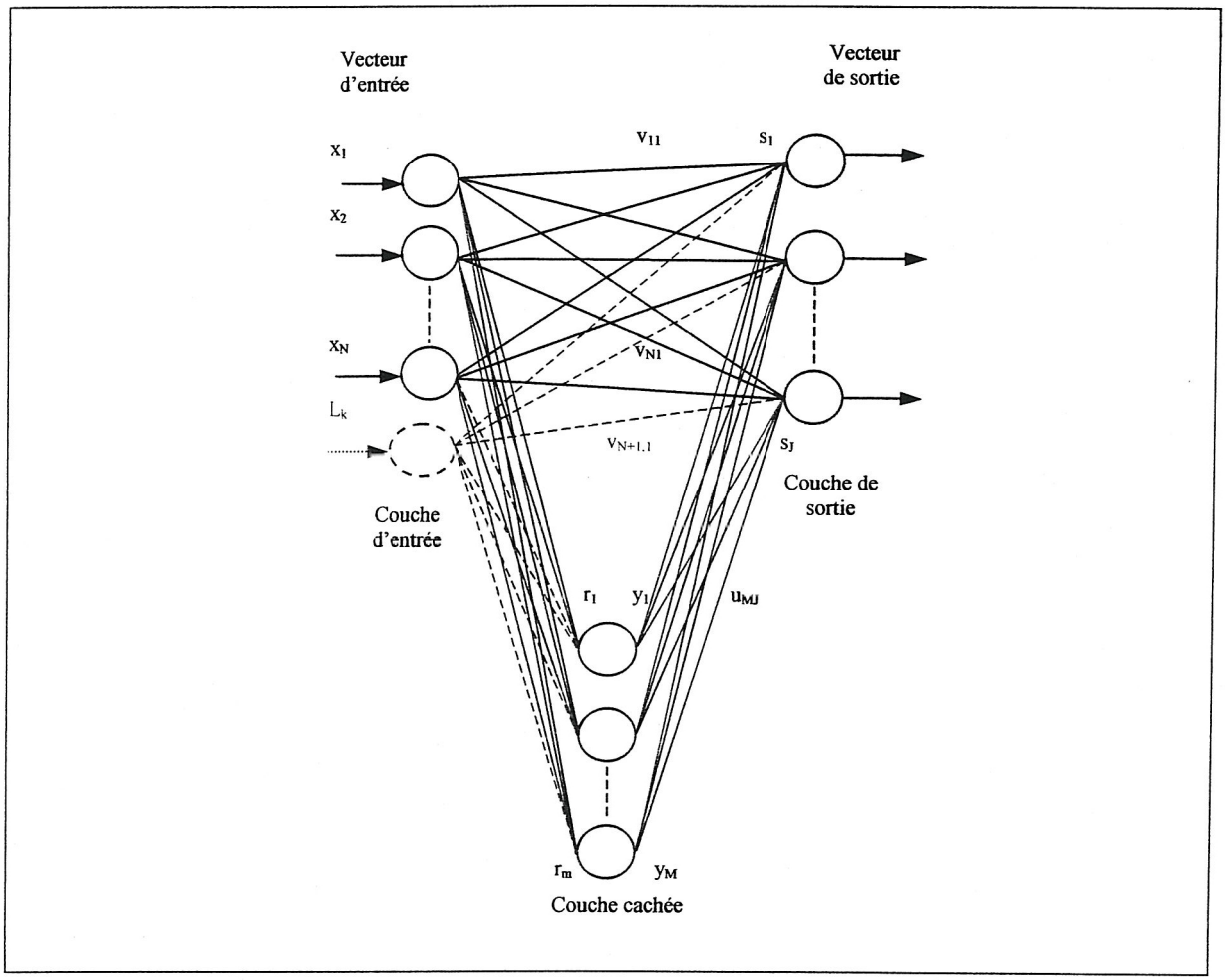


Figure (IV.6) : RVFLNN avec apprentissage étiqueté

IV.5 APPLICATION AVEC LE CLASSIFICATEUR NEURO-FLOU

La classification avec apprentissage étiqueté par le classificateur Neuro-Flou (NFC, Neuro-Fuzzy Classifier) est identique avec celle par les réseaux de neurones. L'application de cette méthode mène à remplacer les règles floues de la forme :

Si x_1 *est* Ai ***et*** x_2 *est* Bi ***Alors*** $X \in C_k$

Par les règles :

Si x_1 *est* Ai ***et*** x_2 *est* Bi ***et*** x_3 *est* Li ***Alors*** $X \in C_k$

Ou sous forme de langage naturel :

Si x_1 *est grand* ***et*** x_2 *est Petit* ***et*** *son étiquette est* Li ***Alors*** X *appartient à* C_k

La première étape de cette méthode consiste à ajouter les étiquettes à tous les exemples d'apprentissage, et par conséquent à ajouter, au NFC, un neurone à la couche d'entrée et K

neurones à la deuxième (K est le nombre de classes). Chaque neurone ajouté à la deuxième couche correspond à une fonction d'appartenance d'une étiquette. Par exemple pour le cas bidimensionnel à deux classes (C_1 et C_2) illustré sur la figure (IV.7), $A1$ et $A2$ sont les termes linguistiques caractérisés par fonctions d'appartenances μ_{A1} et μ_{A2} ; $B1$ et $B2$ sont caractérisés par μ_{B1} et μ_{B2} ; $L1$ et $L2$ sont les étiquettes correspondantes respectivement aux classes C_1 et C_2 et qui sont caractérisées par μ_{L1} et μ_{L2}. Ces dernières peuvent être gaussiennes ou triangulaires, ils ont la valeur d'étiquettes comme centre, en prenant le premier cas (de la fonction gaussienne) elle est de la forme :

$$\mu_{L1}(x)=\exp\left(-\frac{(x-a_{L1})^2}{(b_{L1})^2}\right)$$

$$=\exp\left(-\frac{(x-L1)^2}{(b_{L1})^2}\right) \qquad \text{(IV.9)}$$

Donc x_1 sera fuzzifié par μ_{A1} et μ_{A2} ; x_2 avec μ_{B1} et μ_{B2} ; et x_3 (la caractéristique additionnelle) le sera par μ_{L1} et μ_{L2}.

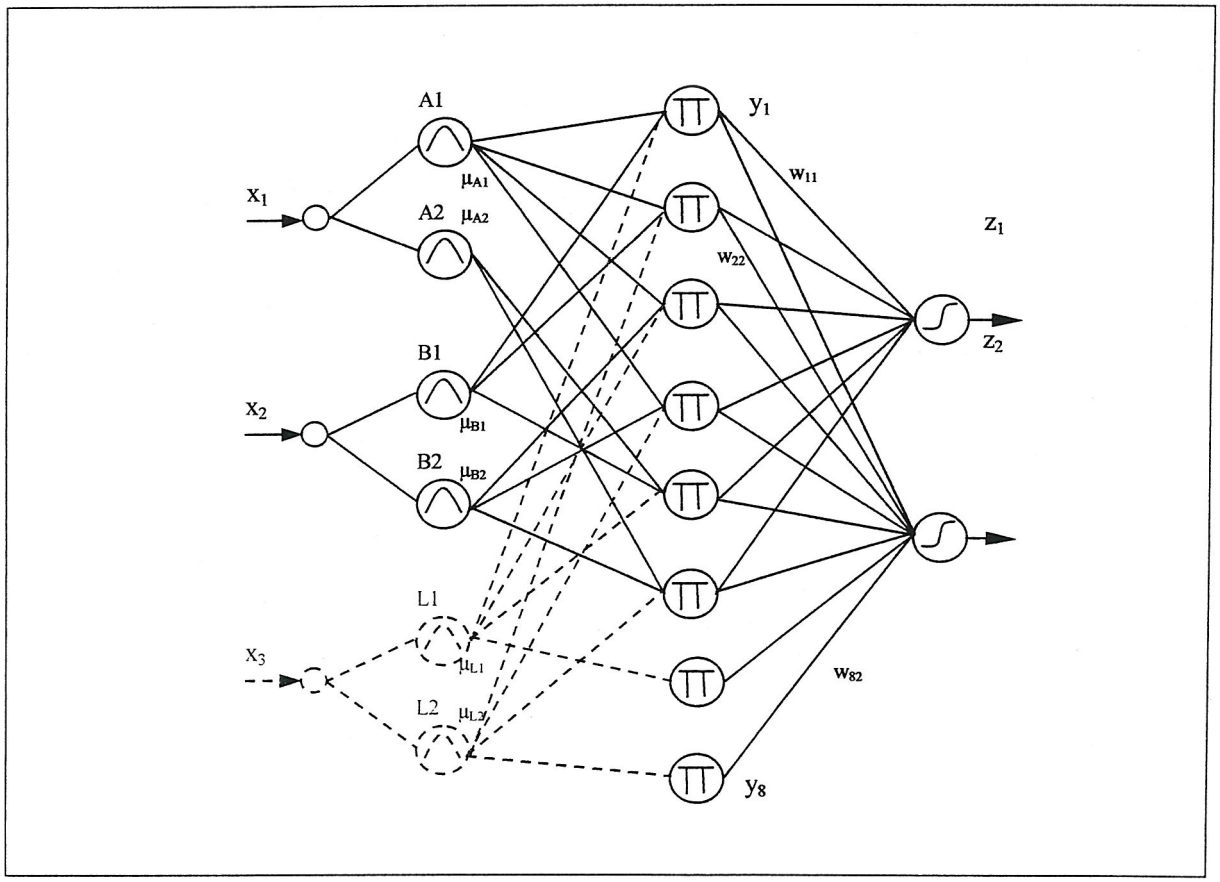


Figure (IV.7) : Classificateur Neuro-Flou avec apprentissage étiqueté

Les prémisses des règles floues du classificateur qui sont fournies par la troisième couche seront :

$$y_1 = \mu_{A1}(x_1)\ \mu_{B1}(x_2)\ \mu_{L1}(x_3)$$

$$y_2 = \mu_{A1}(x_1)\ \mu_{B1}(x_2)\ \mu_{L2}(x_3)$$

$$y_3 = \mu_{A1}(x_1)\ \mu_{B2}(x_2)\ \mu_{L1}(x_3)$$

$$y_4 = \mu_{A1}(x_1)\ \mu_{B2}(x_2)\ \mu_{L2}(x_3)$$

$$y_5 = \mu_{A2}(x_1)\ \mu_{B1}(x_2)\ \mu_{L1}(x_3)$$

$$y_6 = \mu_{A2}(x_1)\ \mu_{B1}(x_2)\ \mu_{L2}(x_3)$$

$$y_7 = \mu_{A2}(x_1)\ \mu_{B2}(x_2)\ \mu_{L1}(x_3)$$

$$y_8 = \mu_{A2}(x_1)\ \mu_{B2}(x_2)\ \mu_{L2}(x_3)$$

Les sorties du classificateur seront :

$$z_1 = h\left(\sum_{m=1}^{8} w_{m1} y_m\right) \qquad \text{(IV.10.a)}$$

$$z_2 = h\left(\sum_{m=1}^{8} w_{m2} y_m\right) \qquad \text{(IV.10.b)}$$

Les prémisses des règles floues établies par la troisième couche sont affectées par les fonctions d'appartenances des étqiuettes, plutôt que par les étiquettes elles-mêmes (figure IV.8). C'est à dire, contrairement au cas du MLP où le choix des valeurs des étiquettes influe sur les performances de la classification, dans ce cas ces performances dépendent des fonctions d'appartenances des étiquettes, et plus précisément par leurs paramètres *b*.

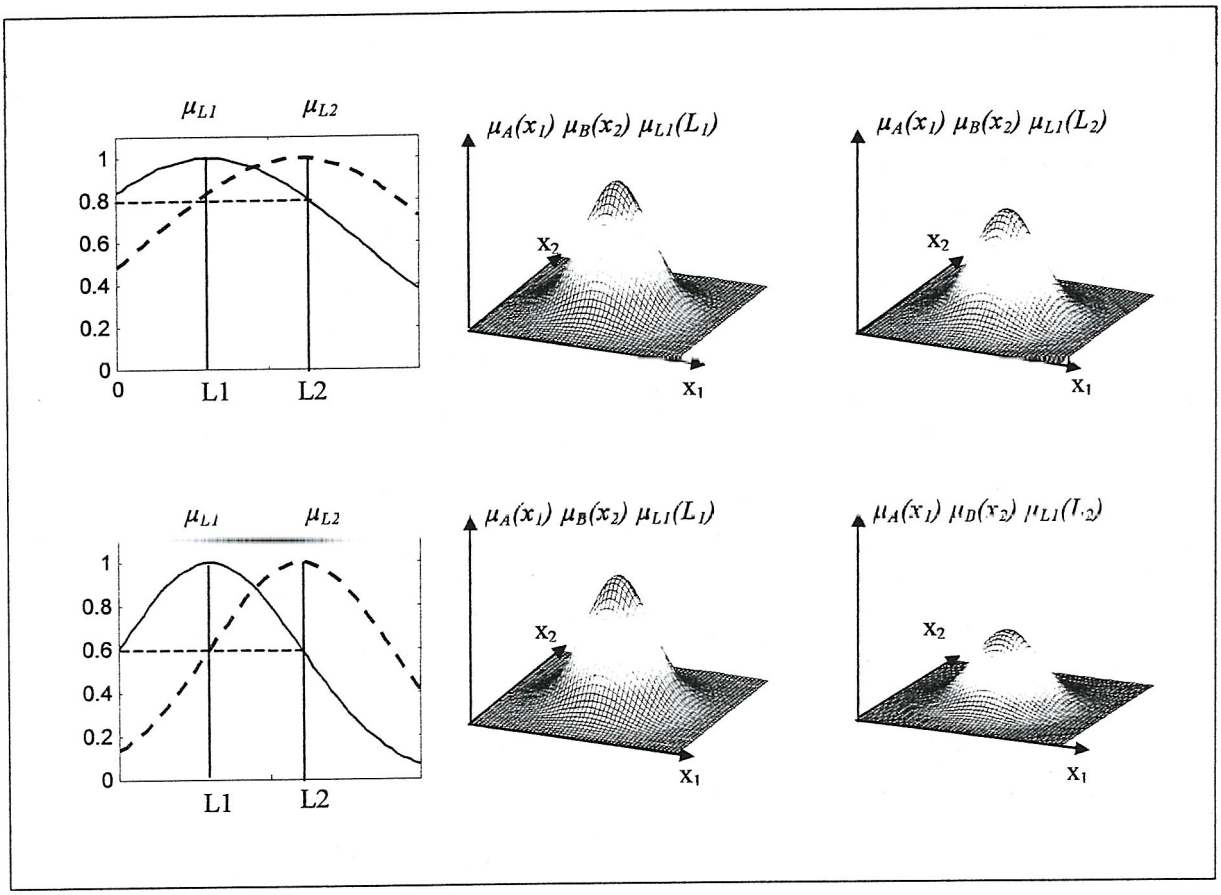


Figure (IV.8) : Effet des étiquettes sur la sortie de la troisième couche

IV.6 CONCLUSION

Ainsi présentée, la classification avec apprentissage étiqueté est une méthode générale, elle est applicable avec plusieurs types des réseaux de neurones. Cette technique s'articule essentiellement sur la modification de la représentation des exemples d'apprentissage en leur ajoutant une caractéristique additionnelle. Par conséquent, elle présente l'avantage de la simplicité de son implémentation dans la procédure de la classification vu que l'algorithme d'apprentissage demeure inchangé. D'autre part, elle est adaptable avec les autres techniques d'accélération et de stabilisation de l'apprentissage, par exemple dans le cas du MLP, l'apprentissage étiqueté est applicable avec un algorithme d'apprentissage comportant une technique d'initialisation des poids, une règle d'ajustement du gain d'apprentissage, l'ajout du moment et l'ajustement des paramètres des fonctions d'activations. Donc les performances du classificateur sont améliorées sont avoir modifié son algorithme d'apprentissage.

CHAPITRE V:

Tests & Résultats

V.1 INTRODUCTION

Dans le chapitre précédent, on a proposé une méthode de classification automatique basée sur une nouvelle approche d'apprentissage supervisé: La classification avec apprentissage étiqueté. On a également présenté les méthodologies de son application avec le Perceptron Multi Couche (MLP, Multi Layered Perceptron), le RVFLNN (Random Vector Functional Link Neural Network) et avec le Classificateur Neuro-Flou (NFC, Neuro Fuzzy Classifier) (figureV.1).

Dans ce chapitre, nous allons réaliser des tests avec les bases de données Iris, cuisse humaine et Texture afin d'apprécier les performances de cette nouvelle approche. Pour estimer sa robustesse nous utilisant la validation croisée, ensuite nous comparons les résultats obtenus avec ceux du SUCRAGE (l'approche numérique [49] et l'approche symbolique [50]), des systèmes d'inférences flous [51] et des arbres flous [52].

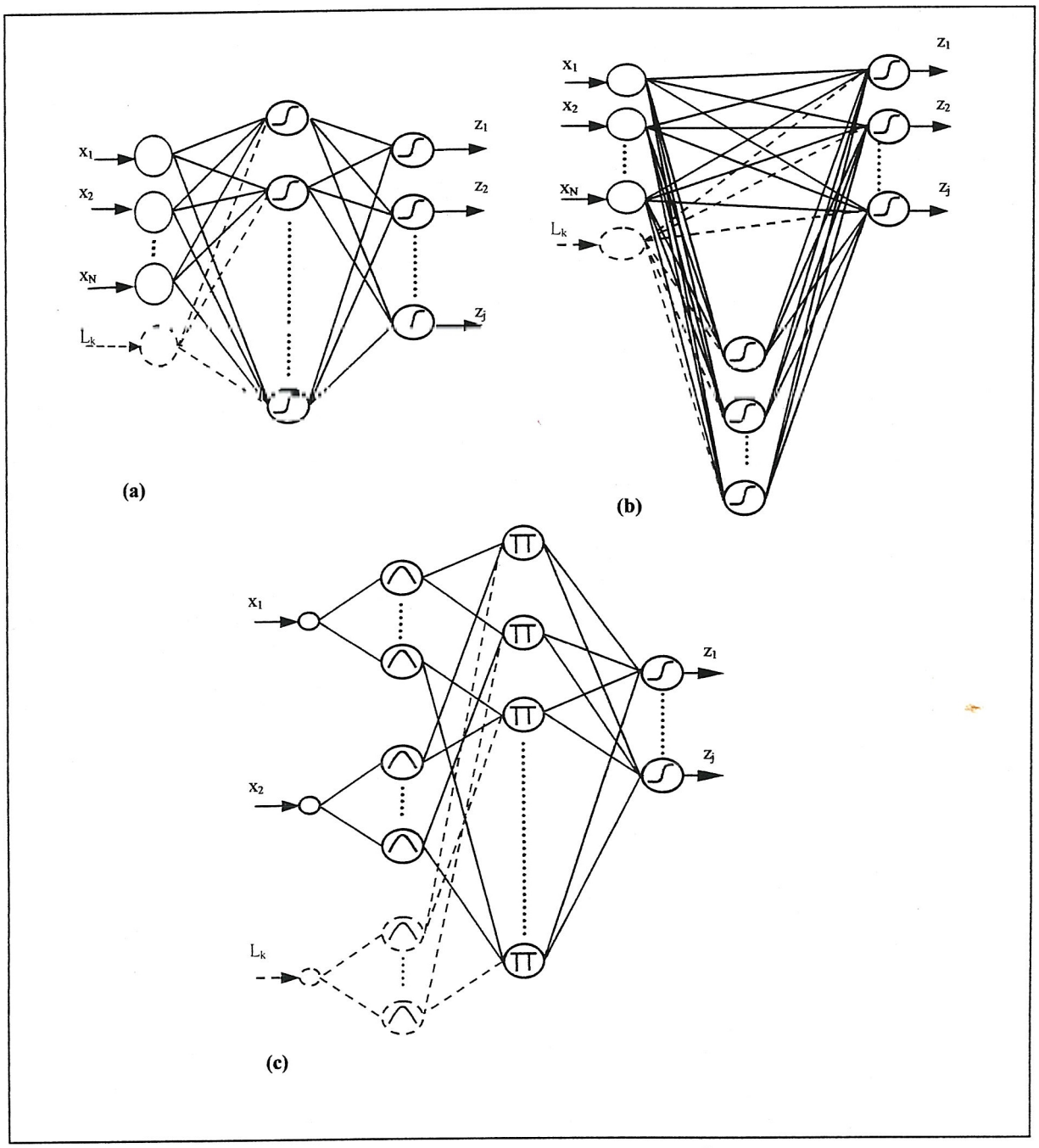


Figure (V.1) : Différentes architectures utilisées avec l'apprentissage étiqueté
(a) : Le Perceptron Multi Couche (MLP)
(b) : Le Random Vector Functional Link Neural Network (RVFLNN)
(c) : Le Classificateur Neuro-Floue (NFC)

V.2 ARCHITECTURES ET PARAMÈTRES DES CLASSIFICATEURS UTILISÉS

V.2.1 Architecture et paramètres du MLP

L'architecture du MLP utilisé pour classifier chacune de ces bases de données est comme suite: i) La couche d'entrée comporte un nombre de neurones égale à celui de caractéristiques qui représentent chaque exemple ; ii) Le nombre de neurones cachés est choisi empiriquement

V.3 CLASSIFICATION DE LA BASE DE DONNÉES IRIS

La base de données Iris de Fisher [4] est constituée de 150 échantillons de fleurs, répartie en trois classes : 50 sétosas, 50 versicolors et 50 virginias. Chaque fleur est décrite par 4 caractéristiques numériques : la longueur et la largeur du sépale, et la longueur et la largeur du pétale. Cette base est souvent utilisée comme référence pour évaluer les performances des classificateurs [53]. Pour évaluer leurs capacités de généralisation, on utilise une validation croisée d'ordre 10. C'est à dire dix bases contenant chacune 135 exemples sont utilisés pour l'apprentissage, et dix autres contenants chacune 15 exemples sont réservés pour le test.

V.3.1 Résultats de classification par le MLP

Pour classer la base de données Iris on utilise un MLP d'architecture 4-12-3, les paramètres de la règle Delta Bar Delta sont : $\eta = 0.1$, $\delta = 0.005$, $\theta = 0.3$ et $\beta = 0.7$. L'évolution du taux de classification en utilisant tous les exemples pour l'apprentissage, pour deux exécutions, sont illustrés sur la figure (V.2.). Les étiquettes sont $L1 = 0.475$, $L2 = 0.5$ et $L3 = 0.525$ ($\sigma = 0.025$).

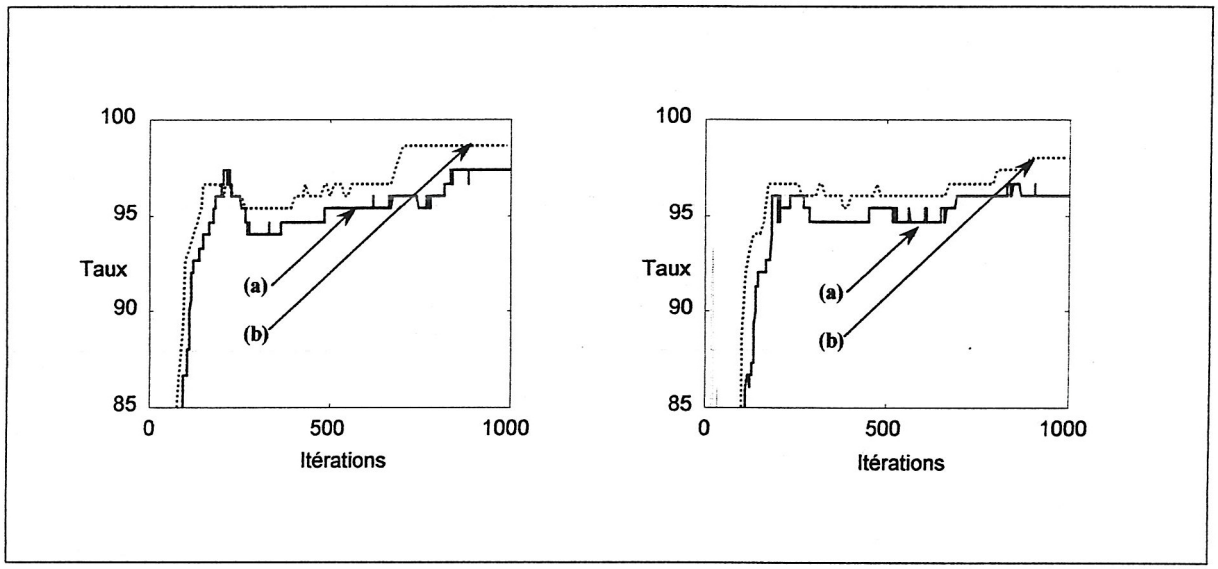


Figure (V.2) : Évolution du taux de classification au cours de l'apprentissage de Iris par le MLP pour deux exécutions
(a) : Sans étiquettes
(b) : Avec étiquettes

En utilisant la validation croisée, les résultats obtenus, avec le même ensemble d'étiquettes, sont donnés sur le tableau (V.1)

	Apprentissage sans étiquettes		Apprentissage étiqueté	
	Taux de classification (%)	Nombre d'itération	Taux de classification (%)	Nombre d'itération
Moy.	**97.33**	**323**	**98.00**	**255**

Tableau (V.1) : Résultats de la classification de Iris par le MLP

V.3.2 Résultats de classification par le RVFLNN

Les résultats de la classification obtenus, sur l'ensemble de tous les exemples de la base, par un RVFLNN d'architecture *(4-8-3)* et avec deux ensembles d'étiquettes, sont donnés sur le tableau (V.2).

Apprentissage étiqueté	σ	L1	L2	Taux (%)	Nombre d'itérations
	1.00	0.450	0.550	91.33	1900
	0.50	0.475	0.525	92.00	1950
Apprentissage sans Étiquettes				92.00	2080

Tableau (V.2) : Résultats de la classification de Iris par le RVFLNN

V.3.3 Résultat de classification par le NFC

Les fonctions d'appartenances, des variables d'entrée que nous utilisons, sont illustrées sur la figure (V.3) :

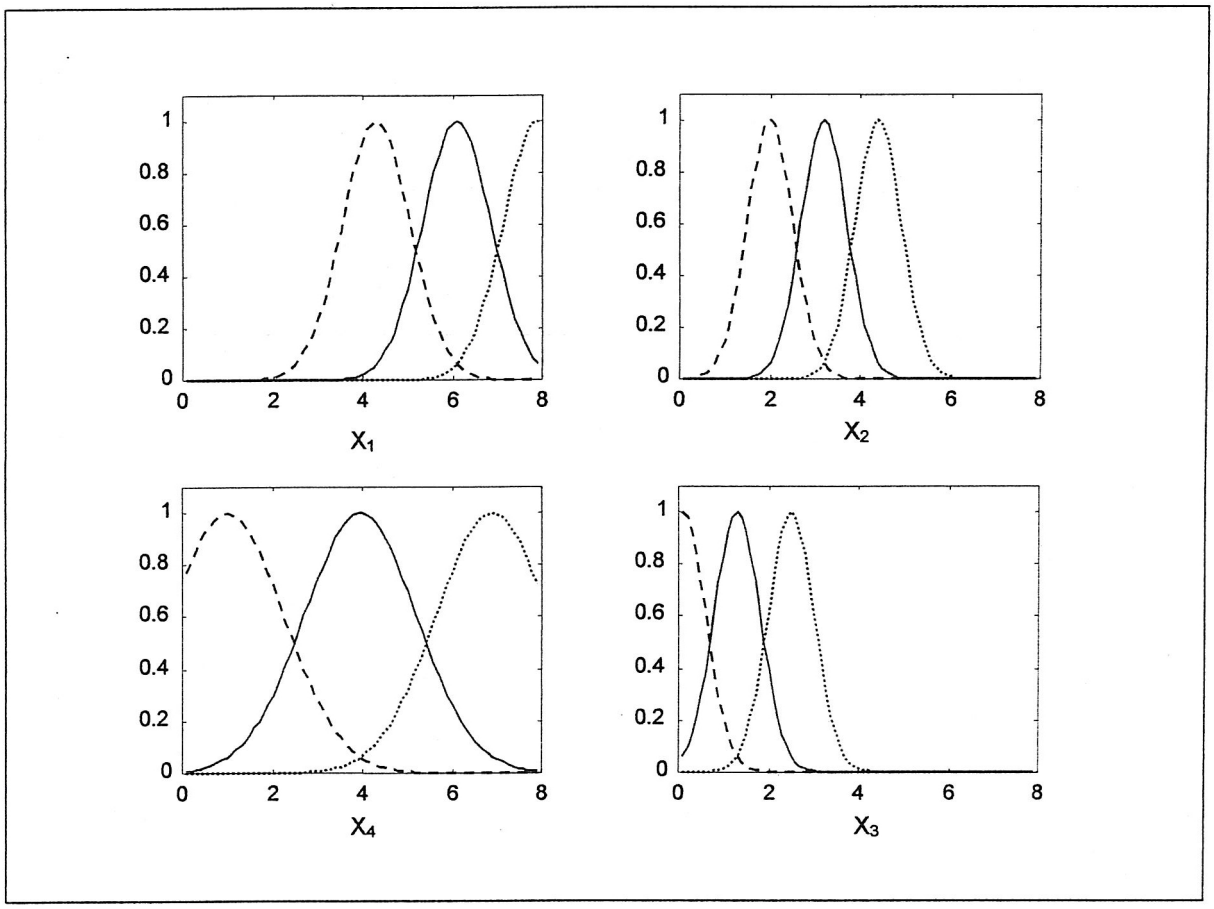


Figure (V.3) : Les fonctions d'appartenances caractérisant les ensembles flous : Petit, Moyen et Grand.

L'évolution du taux de classification au cours de l'apprentissage, en utilisant tous les exemples pour l'apprentissage, est illustrée sur la figure (V.4) :

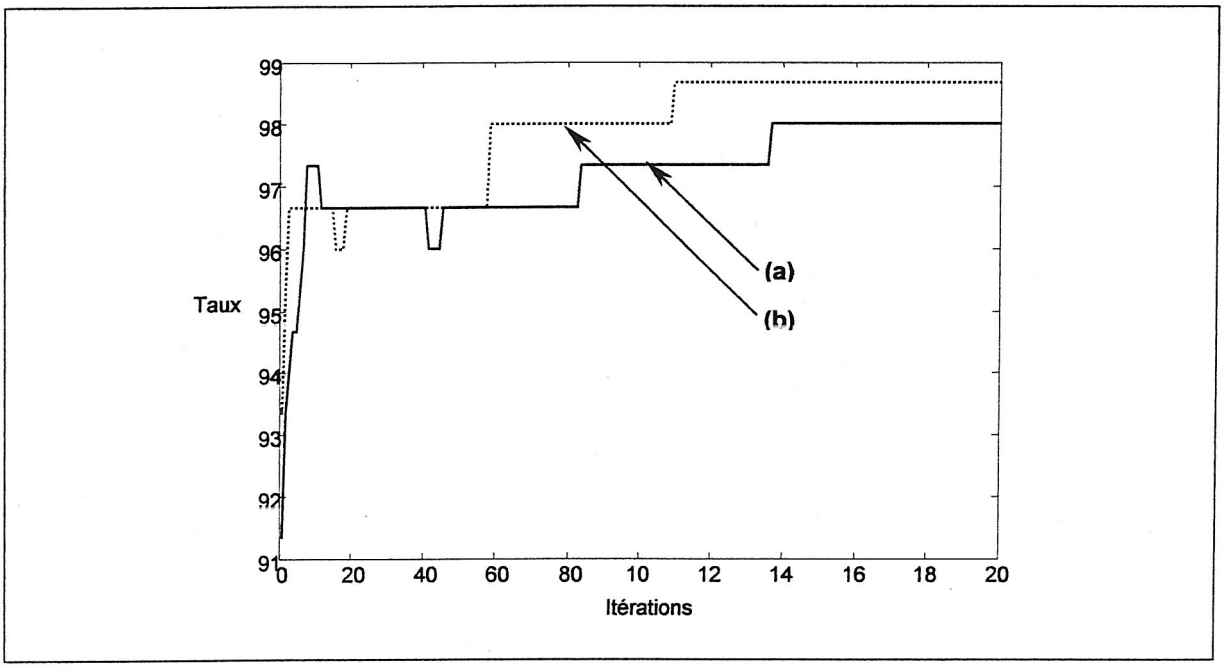


Figure (V.4) : Evolution du taux de classification au cours de l'apprentissage de Iris par le NFC
(a) : Sans étiquettes
(b) : Avec étiquettes

En utilisant la validation croisée, la moyenne du taux de classification et du nombre d'itérations, sont donnés sur le tableau (V.3) :

	Apprentissage sans étiquettes		Apprentissage étiqueté	
	Taux de classification (%)	Nombre d'itération	Taux de classification (%)	Nombre d'itération
Moy.	**96.67**	**12.7**	**97.33**	**8.6**

Tableau (V.3) : Résultats de la classification de Iris par le RVFLNN

V.3.4 Comparaison des résultats

Nos résultats ainsi que ceux du SIF, des Arbres Flous et du SUCRAGE (approche numérique et approche symbolique) sont donnés sur le tableau (V.4) :

Base de données Iris	MLP		RVFLNN		NFC		SUCRAGE		SIF	Arbres Flous
	S. É	A. É	S. É	A. É	S. É	A. É	N	S		
Taux (%)	97.33	**98.00**	92.00	92.00	96.67	97.33	**98.00**	**98.00**	95.57	95.30

Tableau (V.4) : Différents résultats de la classification de Iris

D'après ces résultats, on constate que de l'apprentissage étiqueté avec le MLP a permit d'avoir un taux de classification égal à celui du SUCRAGE et a amélioré le taux du NFC.

V.4 CLASSIFICATION DE LA BASE DE DONNÉES CUISSE HUMAINE

L'image de la figure (V.5) est acquise par la photographie couleur de cryosection (ce mode consiste à immerger le corps dans un gel, le congeler puis le découpé en couches de 1 mm pour l'homme et de 0.33 pour la femme qui sont arasées avant d'être photographiées), elle est prise du site "National Library of Medicine". Cette image est mise sous le format TIFF (Tag Image File Format). Elle a une taille de 670*415 pixels. Les attributs qui caractérisent un pixel sont en RVB (couleur rouge, vert et bleu). Le niveau de variation des couleurs est entre 0 et 255.

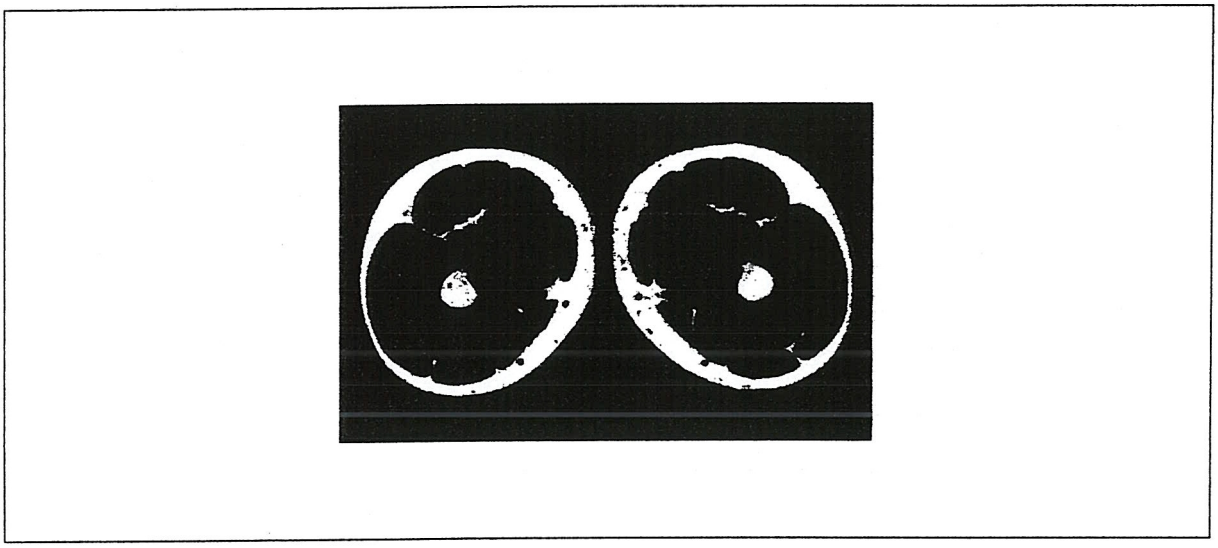


Figure (V.5) : Image d'une cryosection de cuisses humaines

Une classification manuelle a été faite par un expert (médecin, anatomiste) et quatre tissus ont été identifiés (graisse, os, moelle et muscle). Chacun de ses tissus correspond à une classe et chaque classe est représentée par un fichier de 300 pixels. L'échantillon obtenu est constitué de 1200 pixels, 300 pixels de chaque classe. Avec la méthode de validation croisée d'ordre 4, quatre ensembles d'apprentissage sont obtenus et chacun est constitué de 900 pixels; les ensembles de test correspondant contiennent donc 300 pixels. Les tests effectués par (Borgi [49]) ont montré que l'ajout des composants X et Y (pour repérer la position géométrique d'un pixel et de tenir compte de son voisinage) améliore les résultats.

V.4.1 Résultats de classification par le MLP

Le MLP utilisé est d'architecture *(5-12-4)*, les paramètres de la règle Delta Bar Delta sont les suivants : $\eta = 0.01$, $\delta = 0.0001$, $\theta = 0.3$ et $\beta = 0.7$.

L'évolution de l'Erreur Quadratique Moyenne et du taux de classification en utilisant tous les exemples pour l'apprentissage, sont illustrés sur la figure (V.6). Les étiquettes sont *L1 = 0.425, L2 = 0.475, L3 = 0.525 et L4 = 0.575 (σ = 0.05).*

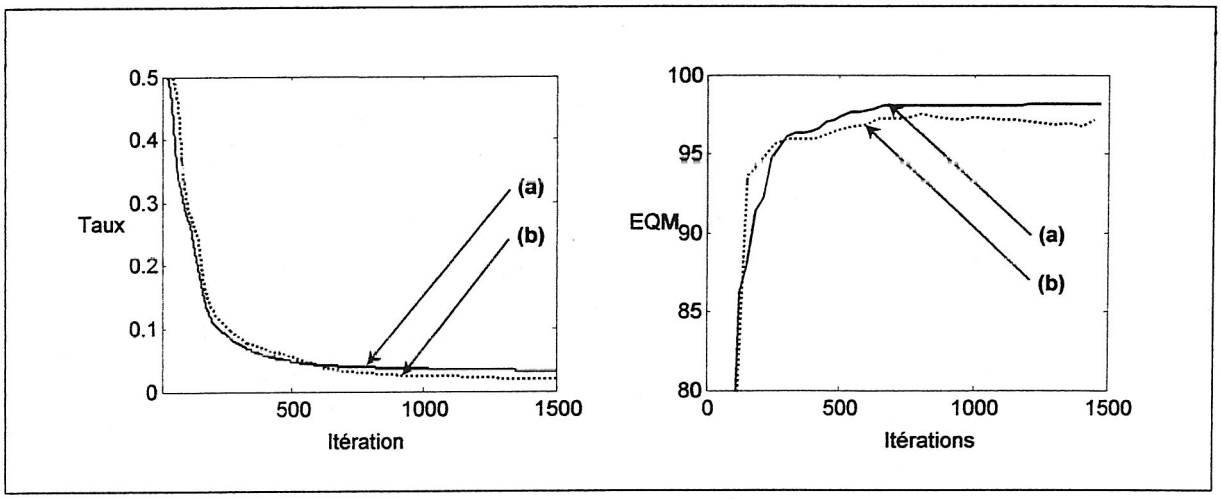


Figure (V.6) : Evolution du taux et de l'EQM aux de l'apprentissage de la Cuisse par le MLP
(a) : Apprentissage sans étiquettes
(b) : Apprentissage étiqueté

En utilisant la validation croisée, les résultats obtenus sont données sur le tableau (V.5) :

	Apprentissage sans étiquettes		Apprentissage étiqueté	
	Taux de classification (%)	Nombre d'itération	Taux de classification (%)	Nombre d'itération
Cuisse 0	99.67	445	99.67	360
Cuisse 1	100	470	99.67	410
Cuisse 2	99.33	455	99.33	410
Cuisse 3	93.67	1218	93.00	990
Moy.	**98.17**	**647**	**97.92**	**542.5**

Tableau (V.5) : Résultats de la classification de la cuisse par le MLP

V.4.2 Résultat de la classification par le RVFLNN

On a effectué la classification de cette base de données avec un RVFLNN d'architecture *(5-8-4)* et avec les mêmes étiquettes.

La figure (V.7) illustre l'évolution de l'EQM et du taux de classification au cours de l'apprentissage en utilisant tous les exemples. Les étiquettes sont les précédentes.

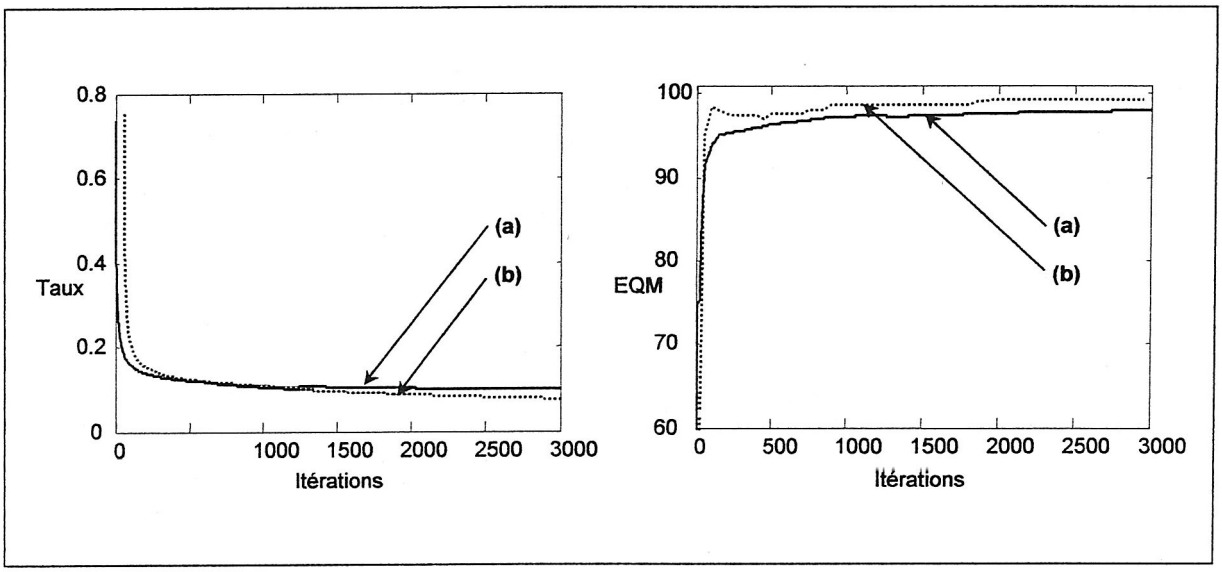


Figure (V.7) : Evolution du taux et de l'EQM au cours de l'apprentissage de la cuisse par le RVFLNN
(a) : Sans étiquettes
(b) : Avec étiquettes

les résultats obtenus par la validation croisée, sont donnés sur le tableau (V.6) :

	Apprentissage sans étiquettes		Apprentissage étiqueté	
	Taux de classification (%)	Nombre d'itération	Taux de classification (%)	Nombre d'itération
Cuisse 0	98.33	2300	99.33	1950
Cuisse 1	99.67	2270	100	1000
Cuisse 2	99.33	2590	99.33	1350
Cuisse 3	91.67	2273	92.67	2700
Moy.	**97.25**	**2385.3**	**97.83**	**1750**

Tableau (V.6) : Résultats de classification de la Cuisse par le RVFLNN

V.4.3 Résultat de la classification par le NFC

Les formes des fonctions d'appartenances utilisées pour la fuzzyfication des variables d'entrées sont illustrées sur la figure (V.8)

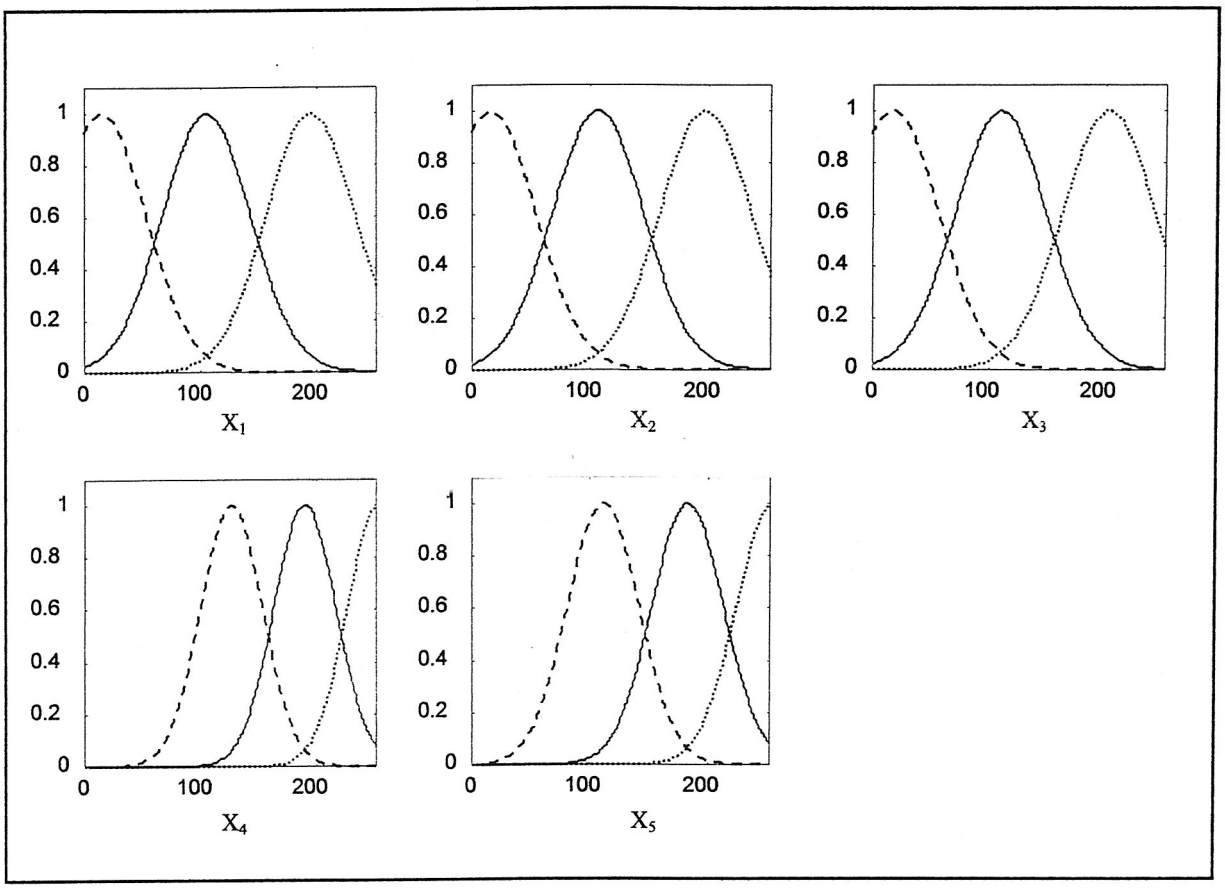


Figure (V.8) : Les fonctions d'appartenances caractérisant les ensembles flous : Petit, Moyen et Grand correspondant à la base de données cuisse humaine

L'évolution du taux et de l'EQM en utilisant tous les échantillons de la base de données sont représentés sur la figure (V.9).

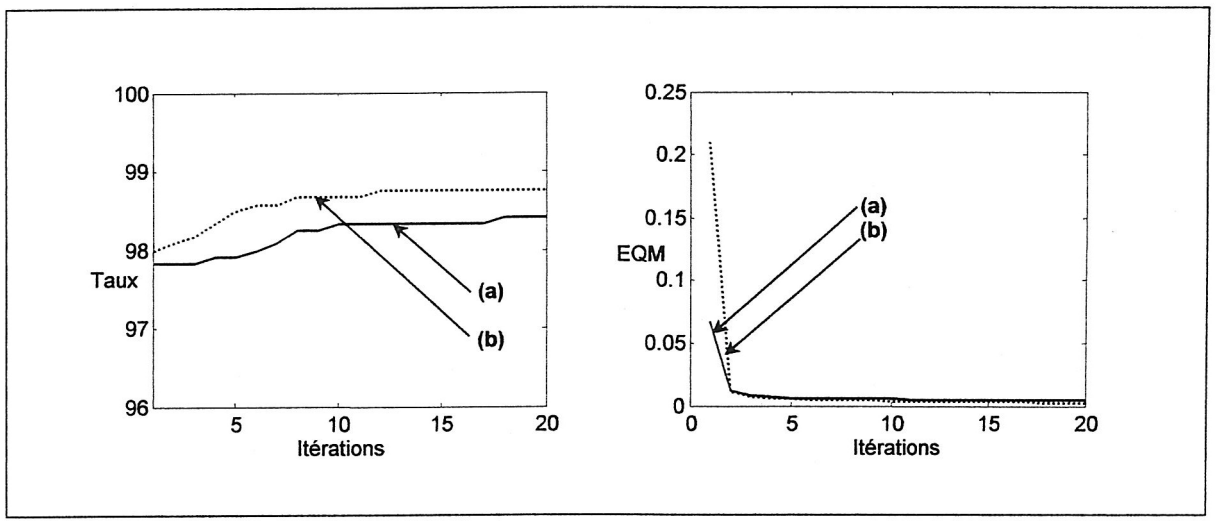


Figure (V.9) : Evolution du taux et de l'EQM au cours de l'apprentissage de la cuisse par le NFC
(a) : Sans étiquettes
(b) : Avec étiquettes

Les taux de classification et les nombres d'itérations obtenus en utilisant la validation croisée sont représentés sur le tableau (V.7) :

	Apprentissage sans étiquettes		Apprentissage étiqueté					
			$\mu_{Li}(Li) = 1$ $\mu_{Li}(Lj) = 0.80$		$\mu_{Li}(Li) = 1$ $\mu_{Li}(Lj) = 0.85$		$\mu_{Li}(Li) = 1$ $\mu_{Li}(Lj) = 0.90$	
	Taux (%)	Itérations	Taux (%)	Itérations	Taux (%)	Itérations	Taux (%)	Itérations
Cuisse 0	99.67	7	100	8	100	5	100	6
Cuisse 1	100	4	100	3	100	3	100	2
Cuisse 2	99.67	2	99.67	2	99.67	1	99.67	1
Cuisse 3	92.33	48	93.00	44	93.00	36	93.00	37
Moy.	**97.92**	**15.25**	**98.17**	**14.25**	**98.17**	**11.25**	**98.17**	**11.50**

Tableau (V.7) : Résultats de classification de la Cuisse par le NFC

V.4.4 Comparaison des résultats

Nos résultats ainsi que ceux du SUCRAGE sont donnés sur le tableau (V.8):

Base de données Cuisse	MLP		RVFLNN		NFC		SUCRAGE	
	S. É	A. É	S. É	A. É	S. É	A. É	N	S
Taux (%)	98.17	97.92	97.25	97.83	97.92	98.17	**99.08**	**99.08**

Tableau (V.8) : Différents résultats de la classification de la Cuisse

Malgré que nos résultats soient moins satisfaisants, cela n'empêche pas de voir les importantes améliorations apportées par l'apprentissage étiqueté avec le RVFLNN et le NFC.

V.5 CLASSIFICATION DE LA BASE DE DONNÉES TEXTURE

La figure (V.10) est une image constituée de deux microtextures différentes. Un prétraitement (le calcul des différentes corrélation locales) de l'image initiale a donné naissance à une série de 8 images dont chacune est le résultat d'une détection d'un attribut particulier. Un pixel est alors décrit par un vecteur à 8 attributs [54]. Ainsi, l'image des deux textures de la figure 8 est représentée par deux classes dont chacune est décrite par 8 fichiers contenant chacun les valeurs des pixels pour l'une des 8 composantes. L'échantillon obtenu est constitué de 400 pixels de chaque classe. La méthode de la validation croisée stratifiée d'ordre 4 conduit à 4 ensembles d'apprentissage contenant chacun 600 pixels et 4 ensembles de tests de 200 pixels chacun [49].

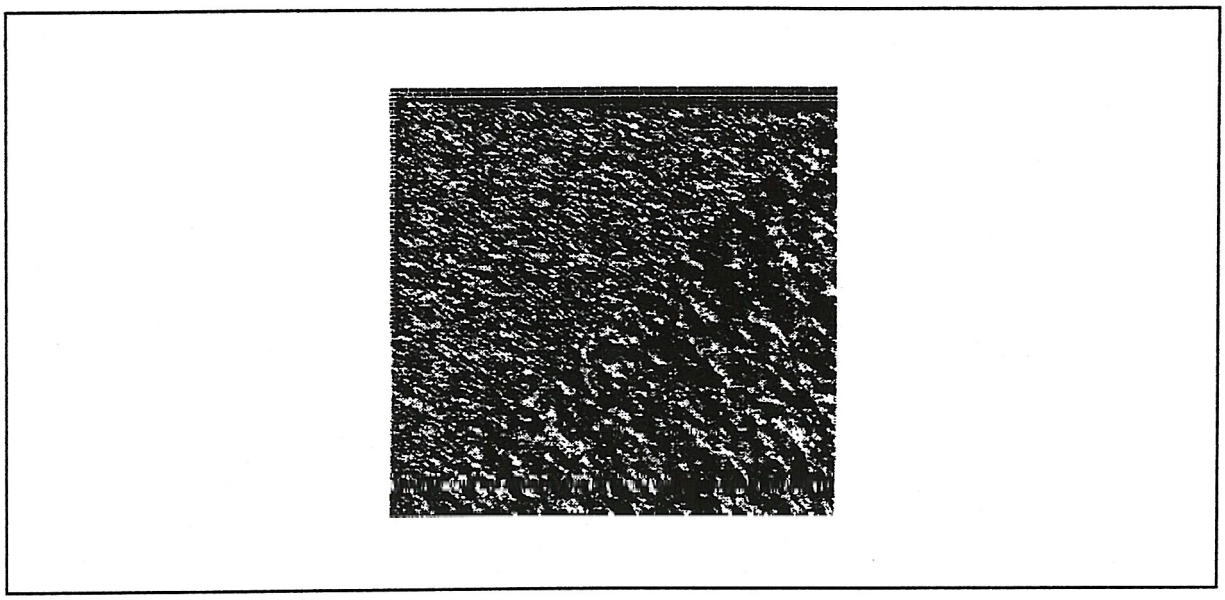


Figure (V.10) : Image des textures.

V.5.1 Résultats de classification par le MLP

Le MLP utilisé est d'architecture *(8-4-2),* les paramètres de la règle Delta Bar Delta sont les suivant : $\eta = 0.01$, $\delta = 0.0001$, $\theta = 0.3$ et $\beta = 0.7$.

L'évolution du taux et d'EQM au cours de l'apprentissage, en utilisant tous les exemples pour apprentissage, est illustrée sur la figure (V.11). Les étiquettes sont $L1 = 0.475$, $L2 = 0.525$ ($\sigma = 0.05$) :

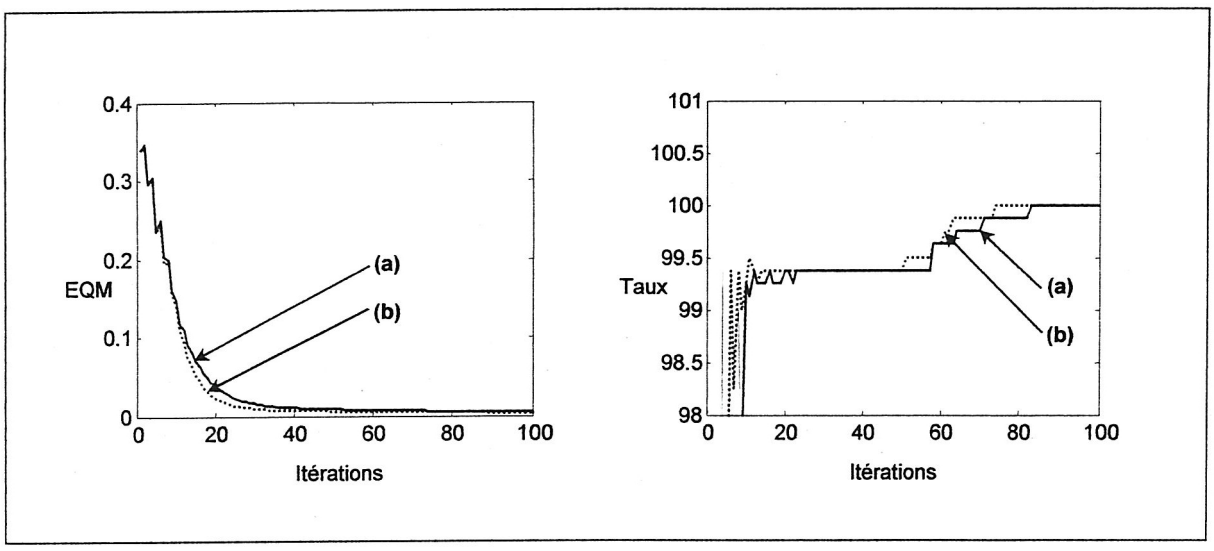


Figure (V.11) : Evolution du taux et de l'EQM au cours de l'apprentissage de la Texture par le MLP
(a) : Sans étiquettes
(b) : Avec étiquettes

Les résultats obtenus en utilisant la validation croisée, sont données sur le tableau (V.9) :

	Apprentissage sans étiquettes		Apprentissage étiqueté	
	Taux de classification (%)	Nombre d'itération	Taux de classification (%)	Nombre d'itération
Texture 0	96.5	16	97	13
Texture 1	100	7	100	7
Texture 2	100	5	100	5
Texture 3	100	5	100	5
Moy.	**99.13**	**8.75**	**99.25**	**7.50**

Tableau (V.9) : Résultats de classification de la Texture par le MLP

V.5.2 Résultat de classification par le RVFLNN

Nous avons utilisé un réseau d'architecture *(8-4-2)* entraîné avec un gain d'apprentissage qui vaut « 0.5 »,

L'évolution du taux et de l'EQM en utilisant tous les échantillons la base de données sont représentés sur la figure (V.12). Les étiquettes sont celles du cas précédent.

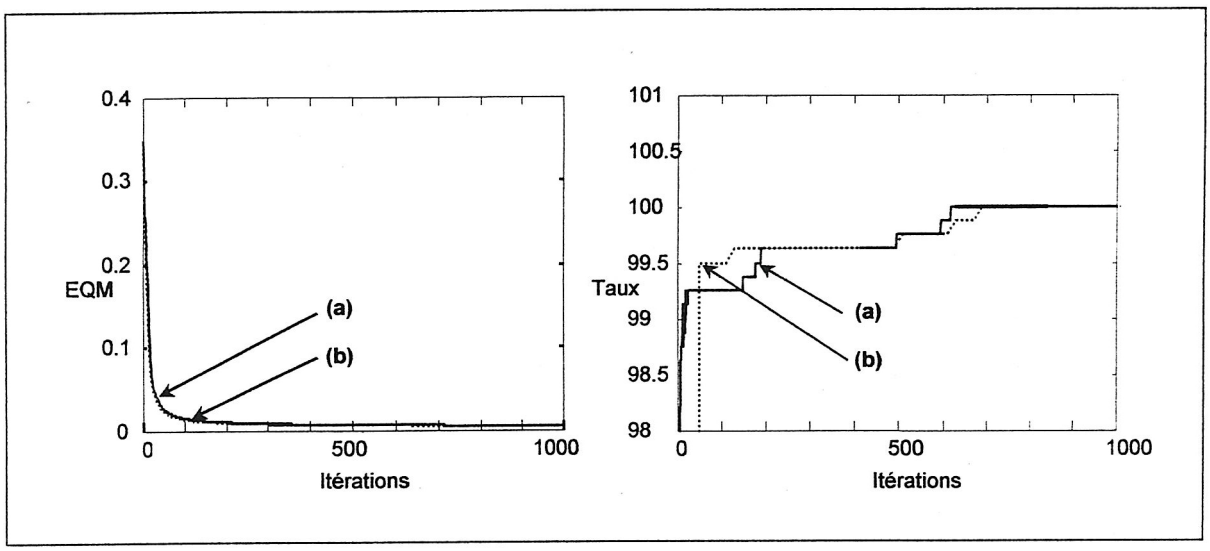


Figure (V.12) : Evolution du taux et de l'EQM au cours de l'apprentissage de la Texture par le RVFLNN
(a) : Sans étiquettes
(b) : Avec étiquettes

Les résultats obtenus en utilisant la validation croisée, sont donnés sur le tableau (V.10):

	Apprentissage sans étiquettes		Apprentissage étiqueté	
	Taux de classification (%)	Nombre d'itération	Taux de classification (%)	Nombre d'itération
Texture 0	95	550	95.5	600
Texture 1	100	7	100	8
Texture 2	100	2	100	2
Texture 3	100	4	100	4
Moy.	**98.75**	**140.75**	**98.88**	**153.50**

Tableau (V.10) : Résultats de classification de la Texture par le RVFLNN

V.5.3 Résultat de classification par le NFC

Les fonctions d'appartenance caractérisant les sous-ensembles flous (Petit, Moyen et Grand) sont illustrées sur la figure (V.13)

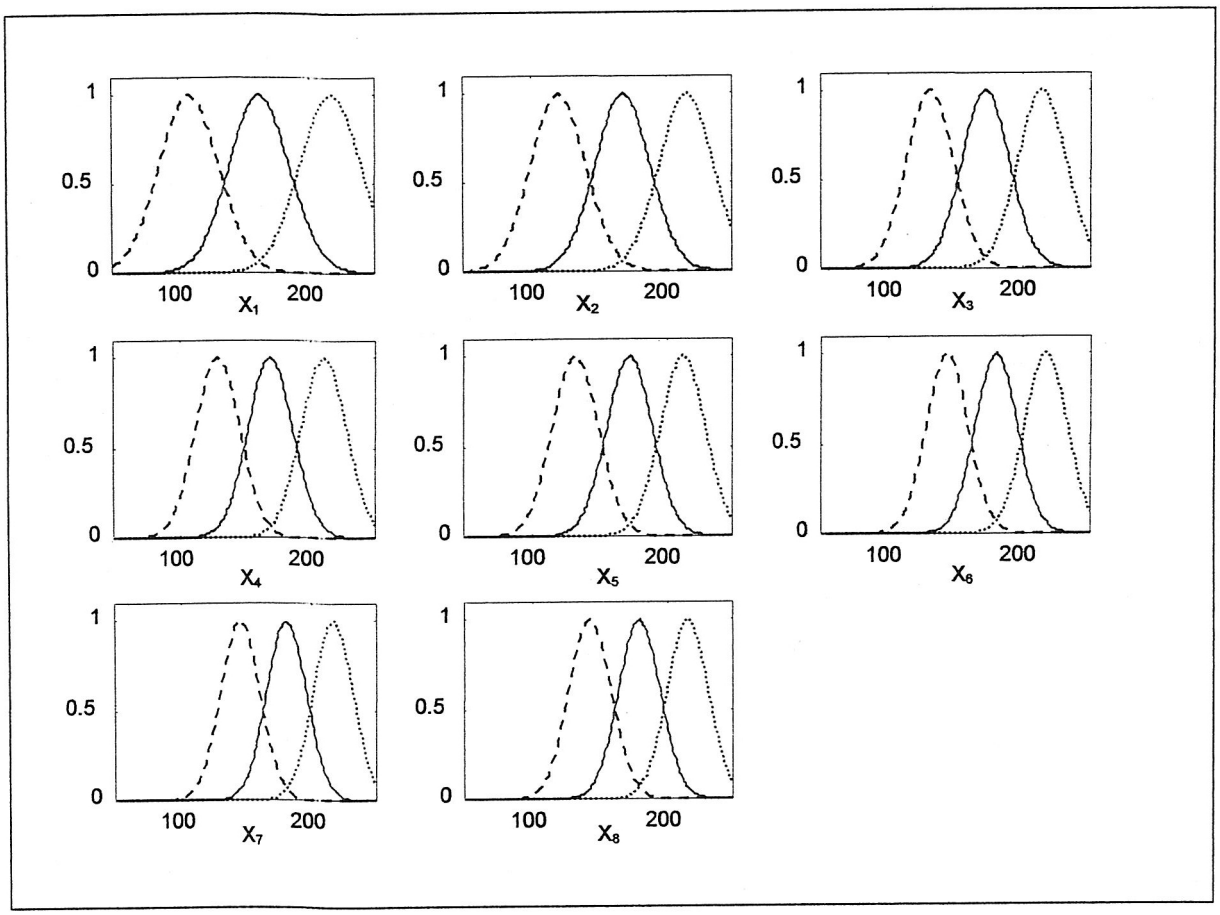


Figure (V.13) : Les fonctions d'appartenances caractérisant les sous-ensembles flous : Petit, Moyen et Grand correspondant à la base de données Texture.

Les résultats obtenus en utilisant la validation croisée sont donnés sur le tableau (V.11) :

	Apprentissage sans étiquettes		Apprentissage étiqueté					
			$\mu_{Li}(Li) = 1$ $\mu_{Li}(Lj) = 0.80$		$\mu_{Li}(Li) = 1$ $\mu_{Li}(Lj) = 0.85$		$\mu_{Li}(Li) = 1$ $\mu_{Li}(Lj) = 0.90$	
	Taux (%)	Itérations	Taux (%)	Itérations	Taux (%)	Itérations	Taux (%)	Itérations
Texture0	96	5	96	4	96	3	96	3
Texture1	100	1	100	1	100	1	100	1
Texture2	100	1	100	1	100	1	100	1
Texture3	100	1	100	1	100	1	100	1
Moy.	**99**	**2**	**99**	**1.75**	**99**	**1.5**	**99**	**1.5**

Tableau (V.11) : Résultats de classification de la Texture par le NFC

V.5.4 Comparaison des résultats

Nos résultats ainsi que ceux du SUCRAGE sont donnés sur le tableau (V.12):

Base de données Texture	MLP		RVFLNN		NFC		SUCRAGE	
	S. É	A. É	S. É	A. É	S. É	A. É	N	S
Taux (%)	99.13	**99.25**	98.75	98.88	99.00	99.00	97.50	97.25

Tableau (V.12) : Différents résultats de la classification de la base de données Texture

Les résultats obtenus par Le MLP, le RVFLNN et le NFC sont très satisfaisants, Avec l'apprentissage étiqueté nous constatons une amélioration de ces résultats. En effet, il accélère l'apprentissage du NFC, augmente le taux du RVFLNN et les deux pour le MLP.

V.6 CONCLUSION

Les tests expérimentaux réalisés avec ces bases de données nous permettent de dégager les poins suivants :

- L'apprentissage étiqueté permet d'augmenter le taux de classification et/ou diminuer le nombre d'itérations nécessaires pour l'apprentissage ; à l'exception du cas de classification de la base de données Cuisse humaine par le MLP.
- Bien que les résultats obtenus par ces classificateurs sur la base de données de la Cuisse humaine soient moins satisfaisants que ceux du SUCRAGE ; nous constatons une amélioration apportée par notre approche avec le RVFLNN et le NFC.
- La classification de la base de données Texture par le RVFLNN, le MLP et le NFC a permit d'avoir des résultats plus satisfaisants que ceux obtenus par le SUCRAGE. De plus, l'utilisation de l'apprentissage étiqueté les améliore.

- Notre approche s'adapte bien avec le classificateur Neuro-Flou, vu la souplesse du choix des étiquettes.
- Les taux de classification fournis par le MLP sont souvent les meilleurs, mais ces performances dépende de l'initialisation des poids ; par contre, le NFC est indépendant de l'initialisation (ces poids sont fixés à « 0.5 »). De plus, sa phase d'apprentissage est plus rapide.

Conclusion

Pour permettre aux systèmes d'être capables de prendre connaissance de leurs environnements, de s'adapter aux différentes situations et d'être plus autonome dans leurs raisonnements, ceci n'est possible qu'après les avoirs doter avec des stratégies d'apprentissage convenables. De ce fait, l'apprentissage automatique est la voie la plus intéressante qui permet de concevoir des machines plus intelligentes.

L'apprentissage automatique est l'une des propriétés des réseaux de neurones les plus importantes, il consiste à mémoriser leurs connaissances au niveau de leurs connections, or la lenteur de cette phase est l'un des inconvénients de leur utilisation. Ainsi, le but principal de l'apprentissage étiqueté est d'accéléré cette étape. Cette approche s'appuie sur le fait que l'apprentissage soit plus rapide si les classes sont linéairement séparables, d'où la modification de la représentation des exemples d'apprentissage permet, par l'ajout des étiquettes (i.e. l'annexion d'une dimension additionnelle à l'espace caractéristique), d'assurer la séparation des classes présentes. L'application de cette technique avec les réseaux de neurones permet dans tous les cas de figure d'éviter les minima-locaux et d'accélérer la phase d'apprentissage.

La conception des systèmes Neuro-Flous par l'intégration des systèmes d'inférences flous dans les réseaux de neurones a pour but de concevoir des systèmes interprétables, robuste et efficaces, la fusion de ces deux techniques permet d'utiliser les avantages de l'un et de contourner les inconvénients de l'autre. L'un des objectifs de cette fusion est d'ajusté les paramètres des fonctions d'appartenances des systèmes flous en utilisant les techniques d'apprentissage des réseaux de neurones. Dans notre approche, l'utilisation de l'apprentissage étiqueté avec le classificateur Neuro-Flou, nous permet d'avoir des résultats satisfaisants, sans avoir recours à l'ajustement des paramètres des fonctions d'appartenances. De ce fait, cette technique présente l'avantage de minimiser le temps d'exécution de chaque itération, vu que les poids de la couche de sortie seront les seules à être ajusté.

Les tests réalisés avec les bases de données Iris, Cuisse humaine et Texture, nous ont permis de conclure que notre approche améliore les performances en augmentant le taux de classification et/ou en accélérant l'apprentissage. Nous constatons également qu'elle s'adapte mieux avec le classificateur Neur-Flou, qui offre une souplesse dans le choix des étiquettes. De plus, contrairement aux réseaux de neurones, l'utilisation de ce classificateur est indépendante du choix des poids initiaux.

En perspectives, nous envisageons généraliser cette approche avec d'autres systèmes Neuro-Flous.

Bibliographie

[1] Looney C., Pattern Recognition Using Neural Networks, Oxford University Press, New York, 1997

[2] Devijver P., Statistical Pattern Recognition, appeared in Application of Pattern Recognition, edited by K. S. Fu, CRC Press, Boca Raton, FL, 15-36, 1982.

[3] Nadler M., and Smith E., Pattern Recognition Engineering, Wiley-Interscience, New York, 1993.

[4] Fisher R. A., The use of multiple measurements in taxonomic problems, Ann. Eugenics, Vol.7, part II, 179-188, 1936.

[5] Nilsson N. J., Learning machines: foundations of trainable pattern classifying systems, Mc Graw-Hill, N.Y., 1965.

[6] Meizel W. S., Computer-oriented approaches to pattern recognition, Academic Press, New York, 1972.

[7] Duda R.O., and Hart P.E., Pattern Recognition and Scene Analysis, J. Wiley, New York, 1973.

[8] Tout J. T., and Gonzalez R. C., Recognition Pattern Principals, Addison-Wesley, Readings, MA, 1974.

[9] Shea, P. M., Lin, V., Detection of explosives in checked airlines baggage using an artificial neural system, Int. Join Conf. On Neural Networks, vol.2, 31-34, 1989.

[10] Le Cun Y. H., Broser, B., Denker, J. S., Henderson D., Howard, R. E., Hubbard, W., Jackel, L. D., Backpropagation applied to handwritten ZIP code recognition, Neural Computation, vol. 1(4), 541-551.

[11] Hebb D.,The organization of the Behavior, Wiley, New York, 1949.

[12] Mcculloch W. S., and Pitts W., A logical calculus of the ideas immanent in nervous activity, Bull. Of Math. Biophys., vol. 5, 115-133, 1945.

[13] Rosenblatt F., The perceptron: a probabilistic model fir information storage and organization in the brain, Psycho. Review, vol. 654, 386-408, 1958.

[14] Widrow B., and Stearms S. D., Adaptive Signal Processing, Signal Processing Series, Prentice-Hall, Englewood Cliffs, NJ, 1985.
[15] Minsky M. L., and Papert S.A. Perceptrons, MIT Press, Cambridge, MA, 1969.
[16] Broomhead D.S.,and Lowe D., Multivariable Functional Interpolation and adaptive Networks, Complex Systems, Vol.2 ,321-355, 1988.
[17] Pao Y. H., Adaptive Pattern Recognition and Neural Network, Addison-Wesley, Reading, MA, 1989.
[18] Pao Y. H., Park G. H., and Sobjic D. J. , Learning and generalization characteristics of the Random vector functional link net, Neurocomputing, vol. 6, 163-180, 1994.
[19] Giles C. L., and Maxwell T., Learning invariance and generalization in high order neural networks, Applied Optics vol. 26, 4972-4978, 1987.
[20] Kohnen T., The Self-Organizing Map, proceeding of IEEE, Vol.78, 1464-1480, 1990.
[21] Linsker R., From basic Network Principles to Neural Architectures, Emergence of Orientation Columns Cells, Proc. Nat. Acad. Of science USA, Vol.83, 8779-8783, 1986.
[22] Linsker R., From basic Network Principles to Neural Architectures, Emergence of Orientation-Selective Cells, Proc. Nat. Acad. Of science USA, Vol.83, 8390-8394, 1986.
[23] Kohnen T., Things You Haven't Heard about the Self-Organizing Map, IEEE Neural Networks Conference, San Francisco, IEEE, 1147-1156, 1993.
[24] Lind Y., Buzo A., and Gray, R. M., An algorithm for vector quantizer design, IEEE trans. Commun., vol. 28, 115-133, 1980.
[25] Gray R.M., Vector Quantization, IEEE ASSP, vol.1, 4-29, 1984
[26] Jodouin J.F., Les réseaux neuromimétiques, Edition Hermès, Paris, 1994.
[27] Rumelhart D.E, Hinton G.E, et Williams R.J., learning internal representation by error propagation, parallel distributed processing: exploration in the micro structure of cognition, D.E. Rumelhart et J.L. McClelland (edition), MIT press Cambridge, 318-362, 1986.
[28] Cybenco G., Approximation by Superposition of a sigmoidal Function, Math, Control, Signals and systems, vol. 2, 303-314, 1989.
[29] Bourlard H. and Wellekens C. J., Links between markov models and multilayered perceptron, D. S. Touretzki, editor, Advances in Neural Information Processing systems, San Mateo, CA, IEEE, Morgan Kaufmann, vol. 1, 502-501, 1989.
[30] Hinton G. E., connectionist learning procedure, Artif. Intell., vol. 40, 185-134, 1989.
[31] Hoehfeld M. A., and Fahlman S. E., Learning with limited numerical precision using cascade correlation algorithm, IEEE Trans. Neural Networks, vol. 3, no. 4, 902-6111, 1992.
[32] Fahlman S. E., An Empirical Study of Learning Speed in Backpropagation, Technical Report CMU-CS-88-162, Carnegie Mellon University, 1988.
[33] Reidmiller M., and Braun H., A Direct adaptive method for faster backpropagation learning : the RPROP algorithm, Proc. 1993 IEEE Int. Conf. Neural Networks, San Francisco, Vol. 1, 586-591, 1993.
[34] Lee Y., Oh S. H., and Kim M., the effect of initial weights on premature saturation in back-propagation learning, Proc. 1991 Int. Joint Conf. Neural Networks, Seattle, vol. 1, 765-770.
[35] Russo A. P., Neural Networks for sonar signal processing, tutorial No. 8, IEEE Conf. on Neural Networks for Ocean engineering, Wachington, D.C., 1991.
[36] Nguyen D., and Widrow B., Improving the learning speed two-layer neural networks by choosing initial values of the adaptive weights, Proc. 1990 IEEE Int. Joint Conf. Neural Networks, San Diego, vol. 3, 21-26, 1990.

[37] Jacobs R.A., Increased rates of convergence through learning rate adaptation, Neural Networks, vol.1, No.4, 295-307, 1988.

[38] Zurada M., Lambda learning rule for feedforward neural networks, proc, IEEE int. Conf. Neural Networks, vol. 3, 1808-1811, 1993.

[39] Nemissi M., Bodouda H., Seridi H., H. Akdag, "Classification supervisée par les réseaux de neurones : Accélération de l'apprentissage", Recueil de des résumés, 1 ère Rencontre sur les Journées des sciences et Technologies Avancées (JSTA'2003), Guelma, 24-25 mai 2003, p53.

[40] Zadeh L. A., Fuzzy sets, in Information and Control, 8, 1965.

[41] Takagi T. and Sugeno M., Derivation of fuzzy control rules from human operators control actions, proc. Of the IFAC symp. On fuzzy Information, Knowledge Representation and decision Analysis, pp 55-60, 1983.

[42] Abraham A., and Nath B., Evolutionary Design of Neuro-Fuzzy Systems-A Generic framework, In Proceding of 4th Japan-Australia Joint workshop on intelligent and Evolutionary Systems, Japan, 2000.

[43] Sun C. T., and Roger Jang J. S., Adaptive network based fuzzy classification. In proc. of the Japan-U.S.A., Symposium on Flexible Automation, 1992.

[44] Sun C. T., and Roger Jang J. S., A neuro-fuzzy classifier and its applications, In proc. of IEEE international conference on fuzzy systems, San Francisco, 1993.

[45] Roger Jang J. S., Neuro-Fuzzy Modeling: Architecture, Analyses and Application, PHD thesis, University of California, Berkeley, 1992.

[46] Shi Y., and Mizumoto, M., Some consideration on conventional neuro-fuzzy learning algorithms by gradient descent method, *Fuzzy Sets and Systems*, Vol. 112, 51-63, 2000.

[47] Nemissi M. et Seridi H., Performances de l'application de l'apprentissage étiqueté avec le Perptron Multicouches, dans Proc. Conférence Internationale : Sciences Electronique, Technologies de l'Information et des Télécommunications (IEEE-SETIT'2004), Sousse-Tunisie, 15-20 mars 2004, p 159.

[48] Nemissi M., Seridi H., Akdag H., Performances de l'apprentissage étiqueté avec Random Vector Functional Link Neural Network, dans Proc. 4ème Séminaire National en Informatique (SNIB'2004), Biskra, 4-6 mai 2004, p320.

[49] Borgi, A., Apprentissage supervisé par génération de règles: le système SUCRAGE, Thèse de l'Université de Paris 6, 1999.

[50] Serid, H., Nouvelle Approche Qualitative du Traitement des Connaissances Incertaines et Contribution au Développement de Systèmes Experts Symboliques, Thèse de l'Université de Reims, 1998.

[51] Ishibuchi, H., Nozaki, K., Tanaka, H., Distributed representation of fuzzy rules and its application to pattern classification, Fuzzy Sets and Systems, Vol. 52, p 21-32, 1992.

[52] Marsala, C., Apprentissage inductif en présence des données imprécises: construction et utilisation d'arbres de décision flous, Thèse de Doctorat de l'Université Paris 6, janvier 1998.

[53] Mehrotra, K., Mohan C.K., Ranka, A., Elements of Artificial Neural Networks, Cambridge: MIT Press.

[54] Ould Ahmedou M. L., Amélioration de méthodes de classification automatique non supervisée pour la segmentation d'images multi-composants, Thèse de l'Université de Reims, 1998.

Annexe

1. Détermination de la dérivée $\partial E / \partial u_{mj}$:

$$\frac{\partial E}{u_{mj}} = \left(\frac{\partial E}{\partial z_j}\right)\cdot\left(\frac{\partial z_j}{\partial s_j}\right)\cdot\left(\frac{\partial s_j}{\partial u_{mj}}\right)$$
$$= \left[\frac{\partial}{\partial z_j}\left(\sum_{l=1}^{J}\left(t_l - z_l\right)^2\right)\right]\cdot\left[\frac{\partial}{\partial s_j} g\left(s_j\right)\right]\cdot\left[\frac{\partial}{\partial u_{mj}}\sum_{k=1}^{M} y_k u_k\right]$$
$$= -2\cdot\left(t_j - z_j\right) g'\left(s_j\right) y_m$$

2. Détermination de la dérivée $\partial E / \partial w_{nm}$:

$$\frac{\partial E}{\partial w_{nm}} = \left(\frac{\partial E}{\partial r_m}\right)\cdot\left(\frac{\partial r_j}{\partial w_{nm}}\right)$$
$$= \left[\left(\frac{\partial E}{\partial y_m}\right)\cdot\left(\frac{\partial y_m}{\partial r_m}\right)\right]\cdot\left(\frac{\partial r_m}{\partial w_{nm}}\right)$$
$$= \left[\frac{\partial}{\partial y_m}\left(\sum_{l=1}^{J}\left(t_l - z_l\right)^2\right)\right]\cdot\left[\frac{\partial}{\partial r_m} y_m\right]\cdot\left[\frac{\partial r_m}{\partial w_{nm}}\right]$$
$$= \left[\frac{\partial}{\partial y_m}\left(\sum_{l=1}^{J}\left(t_l - z_l\right)^2\right)\right]\cdot\left[\frac{\partial}{\partial r_m} h\left(r_m\right)\right]\cdot\left[\frac{\partial}{\partial w_{nm}} * \left(\sum_{k=1}^{N} x_k \cdot w_{km}\right)\right]$$
$$= \left[\sum_{j=1}^{J}\frac{\partial}{\partial s_j}\sum_{l=1}^{J}\left(t_l - z_l\right)^2 \cdot\left(\frac{\partial s_j}{\partial y_m}\right)\right]\cdot\left(h'\left(r_m\right)\right)\left(x_n\right)$$

$$= \left[\sum_{j=1}^{J} (-2)(t_l - z_l)\, g'(s_j) \left(\frac{\partial}{\partial y_m} \sum_{k=1}^{M} y_i u_{ij} \right) \right] \cdot (h'(r_m))(x_n)$$

$$= \left(\sum_{j=1}^{J} \sum_{i=1} (-2)(t_j - z_j) g'(s_j) u_{mj} \right) \cdot (h'(r_m))(x_n)$$

3. Détermination de la dérivée de la sigmoïde unipolaire :

$$g(s) = \frac{1}{1+e^{-\alpha s}}$$

$$g'(s_j) \quad \frac{d}{ds_j} g(s_j)$$

$$= \frac{d}{ds_j} \left(\frac{1}{1+e^{-\alpha s_j}} \right)$$

$$= \alpha \cdot e^{-\alpha s_j} \left(\frac{1}{1+e^{-\alpha s_j}} \right)^2$$

$$= \alpha (1+e^{-\alpha s_j} - 1)\, z_j^2$$

$$= \alpha\, z_j^2 (1 - z_j^{-1})$$

4. Détermination de la dérivée de la sigmoïde bipolaire :

$$g(s) = 2\left(\frac{1}{1+e^{-\alpha s}} \right) - 1$$

Ou aussi :

$$g(s) = \left(\frac{1+e^{-\alpha s}}{1-e^{-\alpha s}} \right)$$

$$g'(s) = 2\alpha . e^{-\alpha s} \left(\frac{1}{1+e^{-\alpha s}} \right)^2 \qquad (*)$$

D'autre part on a :

$$g(s) + 1 = 2\left(\frac{1}{1+e^{-\alpha s}} \right)$$

d'où :

$$e^{-\alpha s} = \frac{1 - g(s)}{1 + g(s)}$$

$$1 + e^{-\alpha s} = \frac{2}{1 + g(s)}$$

remplaçant $e^{-\alpha s}+1$ et $e^{-\alpha s}$ par leurs valeurs dans (*) :

$$g'(s)=2\alpha\frac{1-g(s)}{1+g(s)}\left(\frac{1+g(s)}{2}\right)^2$$

$$=\alpha\frac{(1+g(s))(1-g(s))}{2}$$

5. Exécution du programme en MATLAB-SYMBOLIC

```
» programme  demonstration

deltaW_matrice =

[ y11*e11*g11+y21*e21*g21+y31*e31*g31+y41*e41*g41,
y11*e12*g12+y21*e22*g22+y31*e33*g33+y41*e42*g42]
[ y12*e11*g11+y22*e21*g21+y32*e31*g31+y42*e41*g41,
y12*e12*g12+y22*e22*g22+y32*e33*g33+y42*e42*g42]
[ y13*e11*g11+y23*e21*g21+y33*e31*g31+y43*e41*g41,
y13*e12*g12+y23*e22*g22+y33*e33*g33+y43*e42*g42]

deltaW_boucles =

[ y11*e11*g11+y21*e21*g21+y31*e31*g31+y41*e41*g41,
y11*e12*g12+y21*e22*g22+y31*e33*g33+y41*e42*g42]
[ y12*e11*g11+y22*e21*g21+y32*e31*g31+y42*e41*g41,
y12*e12*g12+y22*e22*g22+y32*e33*g33+y42*e42*g42]
[ y13*e11*g11+y23*e21*g21+y33*e31*g31+y43*e41*g41,
y13*e12*g12+y23*e22*g22+y33*e33*g33+y43*e42*g42]

deltaW_boucles - deltaW_matrice =

[ 0, 0]
[ 0, 0]
[ 0, 0]

deltaU_matrices =
[
x11*(e11*g11*u11+e12*g12*u12)*h11+x21*(e21*g21*u11+e22*g22*u12)*h21+x31*(e31*g31*u11+e33*g33
*u12)*h31+x41*(e41*g41*u11+e42*g42*u12)*h41,
x11*(e11*g11*u13+e12*g12*u21)*h12+x21*(e21*g21*u13+e22*g22*u21)*h22+x31*(e31*g31*u13+e33*g33
*u21)*h32+x41*(e41*g41*u13+e42*g42*u21)*h42,
x11*(e11*g11*u22+e12*g12*u23)*h13+x21*(e21*g21*u22+e22*g22*u23)*h23+x31*(e31*g31*u22+e33*g33
*u23)*h33+x41*(e41*g41*u22+e42*g42*u23)*h43]
[
x12*(e11*g11*u11+e12*g12*u12)*h11+x22*(e21*g21*u11+e22*g22*u12)*h21+x33*(e31*g31*u11+e33*g33
*u12)*h31+x42*(e41*g41*u11+e42*g42*u12)*h41,
x12*(e11*g11*u13+e12*g12*u21)*h12+x22*(e21*g21*u13+e22*g22*u21)*h22+x33*(e31*g31*u13+e33*g33
*u21)*h32+x42*(e41*g41*u13+e42*g42*u21)*h42,
x12*(e11*g11*u22+e12*g12*u23)*h13+x22*(e21*g21*u22+e22*g22*u23)*h23+x33*(e31*g31*u22+e33*g33
*u23)*h33+x42*(e41*g41*u22+e42*g42*u23)*h43]
```

Résumé

Les réseaux de neurones artificiels, dont le cerveau humain a servi de modèle pour leur modélisation, présentent un intérêt majeur pour les chercheurs en l'intelligence artificielle vu leurs grandes capacités d'apprentissage automatique, de calcul parallèle et de généralisation. Ils sont utilisés avec succès dans plusieurs applications de la reconnaissance des formes, mais parmi leurs inconvénients est la lenteur de leur phase d'apprentissage. Plutôt que d'essayer de modifier leurs structures, de changer leurs relations d'apprentissages ou même de régler leurs paramètres, et afin d'accélérer leurs phases d'apprentissage, nous proposons une nouvelle approche "la classification avec apprentissage étiqueté" qui s'articule essentiellement sur la modification de la représentation des exemples d'apprentissage en leur ajoutant une caractéristique additionnelle (les étiquettes). L'implémentation de cette technique est simple, robuste et s'applique avec les classificateurs neuro-flous.

Mots clés :

Apprentissage supervisé, classification, reconnaissance des formes, réseaux de neurones, systèmes neuro-flous.