الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التـعليم العالي والبحث العلمي

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

Thèse de Doctorat

Présentée à l'Université de Guelma
Faculté des Sciences et Sciences de l'Ingénierie

Département de Génie Electrique
Spécialité : Electronique

Présenté par : M. NEMISSI Mohamed

Thème : Classification et reconnaissance des formes par algorithmes hybrides

---

Sous la direction de : Prof. SERIDI Hamid

JURY

Prof : SELLAMI Mokhtar	**Université d'Annaba**	**Président**
Prof : SERIDI Hamid	**Université de Guelma**	**Rapporteur**
Prof : BOUKROUCHE A/Hani	**Université de Guelma**	**Examinateur**
Dr : MOUSSAOUI A/Krim	**Université de Guelma**	**Examinateur**
Dr : KHADIR Med Tarek	**Université d'Annaba**	**Examinateur**
Dr : FARAH Nadir	**Université d'Annaba**	**Examinateur**

2009

Classification et reconnaissance des formes par algorithmes hybrides

Présenté par : M. NEMISSI Mohamed

Sous la direction de : Prof. SERIDI Hamid

JURY

Prof : SELLAMI Mokhtar	Université d'Annaba	Président
Prof : SERIDI Hamid	Université de Guelma	Rapporteur
Prof : BOUKROUCHE A/Hani	Université de Guelma	Examinateur
Dr : MOUSSAOUI A/Krim	Université de Guelma	Examinateur
Dr : KHADIR Med Tarek	Université d'Annaba	Examinateur
Dr : FARAH Nadir	Université d'Annaba	Examinateur

Au nom de Dieu, le Tout miséricorde, le Miséricordieux.

Je ne trouverais jamais les mots pour lui témoigner ma reconnaissance.

Louanges à Allah, Seigneur des univers, qui m'a permis d'exister et ainsi d'accomplir ce travail ardu que j'ai commencé il y a cinq ans.

Remerciement

Je tiens en premier lieu à remercier Mr Hamid SERIDI d'avoir proposé et encadré ce sujet. Je lui exprime ma profonde gratitude pour m'avoir fait profiter de ses connaissances, de ses méthodes de travail, et surtout de sa rigueur scientifique.

J'adresse mes plus vifs remerciements à Mr Mokhtar SELAMI d'avoir accepté la présidence du jury.

Je tiens à exprimer ma profonde gratitude à Mr Abelhani BOUKROUCHE, Mr Mohamed Tarek KHADIR, Mr Nadir FAREH et Mr Abdelkarim MOUSSAOUI qui ont accepté de rapporter cette thèse. Je les remercie pour le temps qu'ils ont consacré et pour l'intérêt qu'ils ont porté à mon travail.

Je voudrais aussi remercier Mr Herman AKDAG de m'avoir accueilli chaleureusement au sein du CRestic et de ses précieux conseils.

Je tiens à exprimer ma sincère reconnaissance à Mr Fethi ACHOURI et Mr Ahmed LAATI qui m'ont prodigué de précieux conseils concernant la rédaction de cette thèse.

Je voudrais remercier tous mes amis qui m'ont encouragé et m'ont soutenu durant la réalisation de ce travail.

Un grand merci a ma famille et en particulier a mes parents pour leurs sacrifices et leurs encouragements.

Que tous ceux qui m'ont aidé de près ou de loin dans l'élaboration de ce travail trouvent ici l'expression de ma sincère gratitude.

Résumé

Les travaux de recherche menés dans cette thèse s'attachent à la classification par les systèmes neuronaux, les systèmes mutli-réseaux de neurones et les systèmes neuro-flous. Plus précisément, nous nous intéressons à l'amélioration des performances de l'apprentissage par la Retro-propagation. Nous proposons à cet égard une nouvelle méthode de classification : *la classification étiquetée*, et introduisons trois modèles qui en résultent : *le perceptron multicouche étiqueté*, *le classificateur neuro-flou étiqueté* et *les systèmes étiquetés multi-réseaux de neurones*.

La méthode proposée s'articule essentiellement sur l'ajout d'une caractéristique additionnelle, *les étiquettes*, à tous les exemples d'apprentissage, et à la réalisation de plusieurs tests pour classer les nouveaux exemples. L'idée de base consiste à simplifier l'entrainement en rendant les exemples d'apprentissage linéairement séparables, et à exploiter les propriétés des réseaux de neurones, notamment l'estimation des probabilités a posteriori, afin d'établir la décision finale.

Abstract

This work deals with the classification using neural systems, neuro-fuzzy systems and the systems of multiple neural networks. In particular, our research is focused on improving the Back-propagation performances. In this purpose, we present a new classification method, referred to as *the labeled classification*, and introduce three classification models based on its application: the *labeled multi layered Perceptron, the labeled neuro-fuzzy classifier* and *the labeled system of multiple neural networks*.

The proposed approach seeks to improve the training process through the addition of an extra feature, *labels*, to all training examples, and carrying out tests with these labels to classify new examples. The basic idea relies on simplifying the training process by making the training examples linearly separable, and exploiting the proprieties of neural networks, especially the posterior probabilities estimation, to perform the final decision.

Sommaire

Partie III

Introduction

Contexte

La reconnaissance des formes est un domaine de recherche qui, depuis son apparition, n'a pas cessé d'évoluer et de couvrir plus d'applications dans notre vie quotidienne. La complexité croissante des systèmes traités dans ce domaine a donné lieu à des approches très diverses et a amené à envisager des outils plus innovants. Parmi les méthodes introduites dans cette discipline, les réseaux de neurones et la logique floue ont prouvé leurs efficacités dans plusieurs applications. Ceci est dû aux grandes capacités d'apprentissage et d'approximation des réseaux de neurones et aux capacités de la logique floue de traitement des données imprécises et incertaines.

Par ailleurs, les systèmes actuels de la reconnaissance des formes nécessitent le traitement de différents problèmes dont chacun exige un type de calcul distinct. Ceci a donné lieu à la naissance de nouveaux systèmes hybrides combinant de différents outils afin de concevoir des systèmes plus robustes. L'idée principale ayant motivé cette nouvelle orientation réside dans le fait qu'une coopération judicieuse de plusieurs outils complémentaires, devrait résulter une amélioration des performances. Les systèmes neuro-flous constituent un important exemple de cette tendance actuelle d'intégration de différents outils dans des architectures hybrides. Ces systèmes mettent à profit les propriétés de deux puissantes méthodes de l'intelligence artificielle ; les réseaux de neurones et la logique floue.

Ce travail se situe dans ce cadre de recherche, et s'articule autour de la classification par les systèmes neuronaux, les systèmes mutli-réseaux de neurones, et les systèmes neuro-flous. Plus précisément, nous nous intéressons aux performances de l'apprentissage par la Retro-propagation qui constitue l'algorithme principal d'apprentissage des systèmes précédents. Dans ce contexte, nous proposons une nouvelle méthode de classification : *la classification étiquetée*.

Motivations et Contributions

Dans ce travail, nous introduisons trois modèles de classification qui résultent de l'application de la méthode proposée : *le perceptron multicouche étiqueté*, *le classificateur neuro-flou étiqueté* et *les systèmes étiquetés multi-réseaux de neurones*. Les principes et les

objectifs de la classification étiquetée se révèlent essentiellement dans le premier modèle dont sa mise en œuvre se base sur le réseau de neurone le plus employé et le plus étudié : le perceptron multicouche.

L'apprentissage du perceptron multicouche en utilisant la Retro-propagation a été abondamment évoqué dans la littérature des réseaux de neurones. Ces travaux se sont intéressés non seulement à évaluer ses performances et à montrer son importance, mais aussi à corriger ses problèmes. Cela constitue un sujet d'un grand intérêt pour les chercheurs qui désirent exploiter les capacités de ce réseau et améliorer ses performances pour étendre son champ d'applications. Les approches proposées se sont étendues sur différents aspects : la génération des poids initiaux, l'introduction d'un ajustement des paramètres d'apprentissage, la modification des équations de mise à jour par l'ajout d'autres termes, utilisation des fonctions coût modifiées…etc.

A la différence des méthodes précédentes, notre approche ne requiert aucune modification de l'algorithme d'apprentissage. La classification étiqueté s'articule essentiellement sur la l'ajout d'une caractéristique additionnelle, *les étiquettes*, à tous les exemples d'apprentissage, et à la réalisation de plusieurs tests pour classer les nouveaux exemples. Nous tirons profit des propriétés du perceptron multicouche, notamment l'estimation des probabilités a posteriori, afin d'améliorer son apprentissage.

Le deuxième modèle proposé se base sur les systèmes neuro-flous dont leur mise en oeuvre vise l'exploitation des propriétés des réseaux de neurones et des systèmes flous. L'idée ayant motivé à introduire ce modèle réside dans le fait que, même avec cette hybridation, ces systèmes souffrent toujours de la lenteur d'apprentissage des réseaux de neurones.

La troisième application de la classification étiquetée concerne les systèmes multi-réseaux de neurones. L'implémentation des problèmes de classification multi-classes en utilisant ces systèmes consiste à attribuer à chaque réseau la tâche de résoudre une partie du problème entier. Le fait que les sous-problèmes attribués aux différents réseaux ne sont pas équivalents dans leur complexité nous a incités à proposer un système comportant des réseaux renforcés qui s'en chargent des parties difficiles. Nous tentons donc d'introduire un système hybride comportant dans une même structure deux catégories de réseaux : des réseaux simples et des réseaux étiquetés.

Organisation

Le présent manuscrit s'organise en trois parties :

La première partie constitue une introduction aux fondements théoriques des outils mis en œuvre dans ce travail. Cette partie comporte deux chapitres :

- Le premier chapitre est consacré aux concepts généraux de la reconnaissance des formes et la classification. Ce chapitre présente tout d'abord le processus général de

la reconnaissance des formes et quelques aspects fondamentaux de la classification. Il décrit ensuite la théorie de décision de Bayes qui constitue l'approche fondamentale des méthodes statistiques de classification. Finalement, il présente quelques classificateurs de base.

- Le deuxième chapitre constitue une introduction aux réseaux de neurones. Il évoque leurs principes de modélisation, leur apprentissage et leurs architectures. Nous présentons également à la fin de ce chapitre quelques modèles de base de réseaux de neurones.

La deuxième partie porte sur la classification par le perceptron multicouche et la classification neuro-floue. Cette partie comporte également deux chapitres :

- Le troisième chapitre est consacré à la classification par le perceptron multi couches. Nous tentons d'y étudier les performances de classification par ce réseau et d'analyser quelques travaux qui concernent son apprentissage avec la retro-propagation.
- Le quatrième chapitre s'intéresse à la classification neuro-floue. Il donne en premier lieu un aperçu général sur les objectifs et l'architecture des systèmes neuro-flous, et décrit ensuite le classificateur neuro-flou qui constituera la deuxième application de la méthode proposée.

Dans la troisième partie, nous présentons la méthode proposée, ses applications et son évaluation.

- Le cinquième chapitre détaille le processus de la classification étiquetée et présente les deux premiers modèles proposés : le perceptron multicouche étiqueté et le classificateur neuro-flou étiqueté.
- Le sixième chapitre concerne l'application de la classification étiquetée avec les systèmes multi-réseaux de neurones. Nous présentons dans un premier temps les principes et les objectifs de ces systèmes en évoquant les deux stratégies les plus utilisées pour la décomposition des problèmes mutli-classes. Nous décrivons ensuite le modèle proposé et nous indiquons comment le concept de la classification étiquetée peut être introduit selon ces deux stratégies de décomposition.
- L'objectif du septième chapitre est l'évaluation des performances de la méthode proposée. Nous employons quatre bases de données : Iris, vin, la cuisse humaine et la texture, afin de tester les modèles proposés sur différents types de données. Pour mettre en valeurs les améliorations apportées, nous comparons nos modèles, d'une part, avec leurs versions originales et, d'autre part, avec d'autres travaux.

Finalement, une conclusion et quelques perspectives concluent ce travail.

Partie I

Chapitre 1 : Concepts de base

Chapitre 2 : Les réseaux de neurones

Chapitre 1

Concepts de base

Ce chapitre est consacré aux concepts généraux de la reconnaissance des formes et la classification. Dans cette partie du manuscrit, nous nous inspirons essentiellement des ouvrages de Duda et al. [1], de Bishop [2] et celui de Kuncheva [3]. Nous décrierons tout d'abord le processus général de la reconnaissance des formes et quelques aspects fondamentaux de la classification. Nous présenterons ensuite la théorie de décision de Bayes qui constitue l'approche fondamentale des méthodes statistiques de classification. Finalement, nous présenterons quelques classificateurs de base, et évoquerons les méthodes de combinaison des classificateurs.

1.1 Introduction

La facilité avec laquelle nous pouvons reconnaitre un visage ou une voix, de comprendre des paroles ou lire des caractères manuscrits, de décider si une pomme est mûre par son odeur et d'identifier nos clés de voiture dans notre poche par toucher, donne une fausse idée au sujet du surprenant processus qui est à la base de ces actes de reconnaissance des formes [1]. Ces actions, permettant la prise de décisions à partir de données brutes, sont cruciales dans la survie de tous les êtres biologiques et Dieu nous a dotés de systèmes cognitifs extrêmement sophistiqués pour effectuer de telles taches.

Comme discipline scientifique, la reconnaissance des formes (RDF) s'attache à l'extraction de l'information utile à partir de données brutes. Elle s'intéresse donc au développement des systèmes intelligents qui remplissent des tâches de perception et de prise de décision. Selon bishop [2], le terme reconnaissance des formes englobe une grande gamme de problèmes de traitement d'information ayant de grandes importances pratiques. Les formes à identifier dépendent de l'application : ils peuvent être des images, des lettres, des paroles, des cibles militaires, des formes d'onde d'un signal… etc. D'une façon générale, les formes se définissent comme un ensemble de caractéristiques ou de comportements qui peut être différencié et classé à partir d'une collecte d'informations.

1.2 Processus de la reconnaissance de formes

Comme les humains utilisent leurs sensations pour acquérir les connaissances relatives à leur environnement afin de réagir avec lui, de même un système de RDF commence par

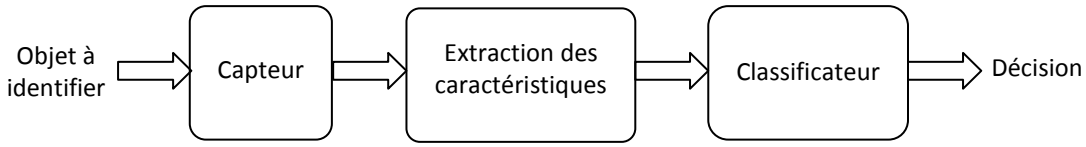


Fig. 1.1. Schéma de base d'un processus de RDF

l'acquisition des données par des capteurs. Une représentation des données brutes est ensuite produite et présentée au classificateur qui produit une décision. Un système de RDF de base se compose principalement d'un capteur, d'un dispositif d'extraction des caractéristiques et d'un classificateur (Fig. 1.1). Un système plus complet inclura également des étapes de prétraitement, d'évaluation et de post traitement. Un processus de RDF comporte généralement les étapes suivantes :

1.2.1 Acquisition des données

Il s'agit de mesurer des grandeurs qui caractérisent les objets à classer par des capteurs afin de fournir une première représentation. Par exemple dans le cas d'un système de RDF basé sur le traitement d'image, cette étape consiste à capter l'image au moyen des scanners ou des caméras et à la convertir en grandeurs numériques adaptées au système de traitement utilisé.

1.2.2 Prétraitement des données

Cette étape consiste à préparer les données issues des capteurs afin d'être exploitées. Il s'agit généralement de réduire les bruits et de chercher à ne garder que l'information significative de l'objet à classer. Par exemple dans le cas de la reconnaissance de caractères, les opérations de prétraitement les plus utilisées sont: le redressement de l'écriture, le lissage, la normalisation et la squelettisation.

1.2.3 Extraction des caractéristiques

Le but de l'étape d'extraction des caractéristiques est d'élaborer un ensemble de caractéristiques contenant celles les plus pertinents pour la tâche de classification. Les informations superflues ou inutiles dégradent les performances du classificateur et devront être enlevées, alors que le contenu valable devra être préservé.

Comme le montre le diagramme de la figure (1.2), les caractéristiques sont quantitatives ou qualitatives [3]. Les caractéristiques discrètes avec un grand nombre de valeurs possibles sont traitées comme quantitatifs tandis que les caractéristiques qualitatives (catégoriques) sont celles avec un petit nombre de valeurs possibles, avec ou sans graduations. La classification statistique opère avec des caractéristiques numériques, les valeurs de ces caractéristiques sont rangées comme des vecteurs de dimension $N : X = [x_1 \quad x_2 \dots \quad x_N] \in R^N$. L'espace réel constitue donc l'espace de représentation. La RDF syntaxique (syntactic

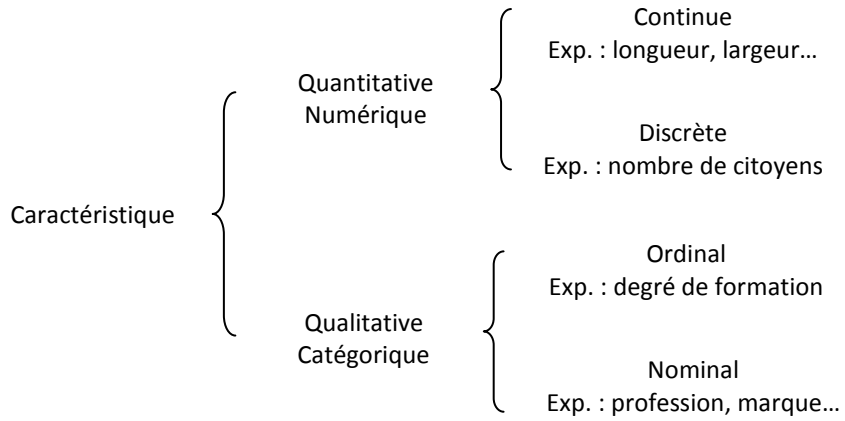


Fig. 1.2. Catégorisation des caractéristiques

pattern recognition), et contrairement à la RDF statistique, traite seulement des caractéristiques qualitatives.

Parfois un objet peut être représenté par de multiples sous-ensembles de caractéristiques. Par exemple, dans la vérification d'identité, trois différentes modalités peuvent être employées : front de visage, profil de visage et voix.

Il est à noter qu'il y a également des caractéristiques non mesurables que les humains peuvent évaluer intuitivement, mais à peine expliquer. Ceux-ci incluent le sens de l'humour, l'intelligence, la beauté… etc.

1.2.4 Phase de classification

La classification s'insère dans le processus de la RDF comme étant l'étape finale de décision. C'est la phase la plus importante qui permet de classifier des objets d'après leurs représentations. La tâche du classificateur est d'autant plus facile que la représentation des objets est pertinente.

1.3 La classification

1.3.1 Classificateurs, fonctions discriminantes et régions de décision

La classification consiste à attribuer de façon automatique un objet à une classe parmi d'autres possibles. Un classificateur doit donc attribuer à un objet, représenté par son vecteur caractéristique ($X = [x_1 \quad x_2 \dots \quad x_N]$), une classe d'appartenance (C_i). Il réalise une fonction (F) de l'espace caractéristique dans l'espace de représentation des classes :

$$F: \Re^N \to \Omega \tag{1.1}$$

Le problème de la classification est généralement considéré comme un problème de réalisation d'un ensemble de fonctions discriminantes : $\mathcal{G} = \{g_1(X) \quad g_2(X) \cdots \quad g_K(X)\}$:

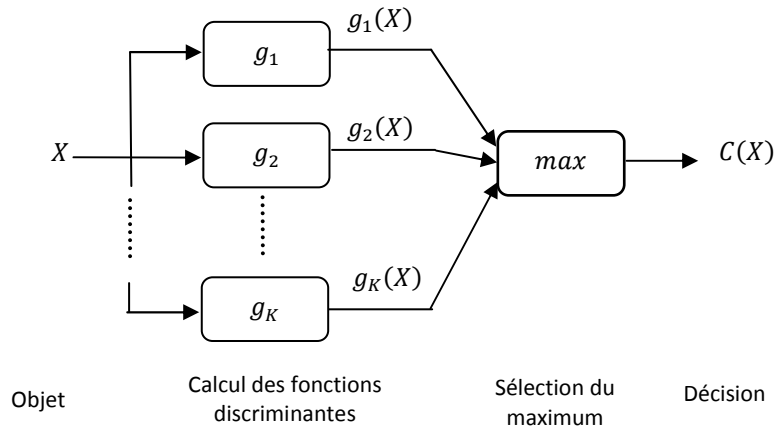


Fig. 1.3. Modèle canonique d'un classificateur

$$g_i: \Re^N \to \Re, \quad i = 1,2,\cdots,K \tag{1.2}$$

Chaque fonction g_i correspondant à une classe C_i. Typiquement, l'exemple X sera assigné à la classe dont la fonction discriminante produit la plus grande valeur (figure1.3). Cette méthode est appelée : la règle du maximum d'appartenance (maximum membership rule) donnée par :

$$X \in C_k \;\; si \; g_k(X) = \max_{k=1,..K}\{g_k(X)\} \tag{1.3}$$

Les fonctions discriminantes partitionnent l'espace caractéristique en K régions de décision (ou région de classification) notées : $\mathcal{R}_1, \mathcal{R}_2, \ldots \mathcal{R}_K$:

$$\mathcal{R}_k = \left\{X/X \in R^N, \; g_k(X) = \max_{k=1,..K}\{g_k(X)\}\right\} \tag{1.4}$$

La région de décision de la classe C_k se constitue de l'ensemble des points dont la $k^{ème}$ fonction discriminante ($g_k(X)$) correspond à la plus grande valeur. Les limites des régions de décision sont appelées limites de classification ou frontières de décision (classification

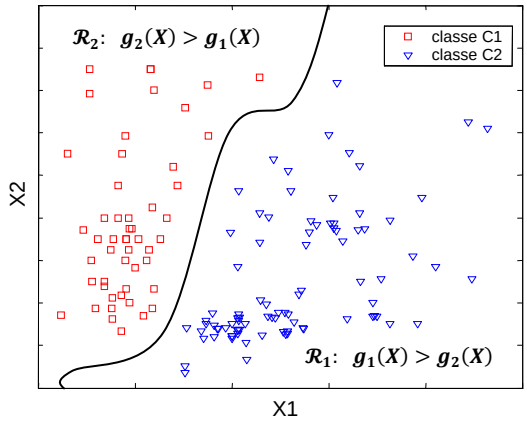


Fig. 1.4. Partitionnement de l'espace caractéristique

boundaries), ils contiennent les points pour lesquels les plus hautes fonctions discriminantes sont égales. La figure (1.4) illustre un exemple de frontières de décision d'un problème de classification bi-classes.

Si la région de décision $\mathcal{R}_i$ contient un exemple appartenant à la classe C_j, les classes C_i et C_j sont dites chevauchante (overlapped). Il est à noter que les classes chevauchante pour une partition particulière de l'espace caractéristique (défini par un classificateur F) ne sont pas chevauchante si l'espace caractéristique est partitionné d'une autre façon.

1.3.2 Apprentissage et adaptation

L'apprentissage fait référence à la procédure d'incorporation des informations à partir des exemples d'apprentissage lors de la conception d'un classificateur. La mise en ouvre des classificateurs implique donc la proposition à l'avance d'une forme générale du modèle et l'emploie d'une phase d'apprentissage pour estimer les paramètres inconnus de ce modèle.

L'apprentissage correspond à une certaine forme d'algorithme permettant de réduire l'erreur de classification sur les données d'apprentissage utilisées. L'apprentissage s'effectue selon plusieurs formes:

a) Apprentissage supervisé

Dans L'apprentissage supervisé, la conception du classificateur est basée sur un ensemble d'exemples préalablement classé par un superviseur (professeur). Ce dernier fournit une sortie désirée pour chaque exemple et durant l'apprentissage on cherche à réduire une fonction coût sur tous les exemples d'apprentissage.

b) Apprentissage non-supervisé

Dans l'Apprentissage non-supervisé (ou clustering), il n'y a aucun professeur explicite et le système forme des clusters (ou des groupements) des exemples d'entrée.

c) Apprentissage par renforcement

La façon la plus typique pour apprendre un classificateur est de lui présenter un exemple, de calculer sa tentative et d'utiliser la sortie désirée pour l'améliorer. Dans l'apprentissage par renforcement, l'adaptation du classificateur s'effectue seulement en lui indiquant si sa réponse est vraie ou fausse. Il s'agit donc de lui dire c'est juste ou c'est faut sans dire pourquoi.

1.3.3 Evaluation des classificateurs

Il est important de savoir à quel degré notre modèle s'en tire après, et même durant, la phase de conception. Afin d'évaluer le classificateur élaboré, plusieurs critères, dont le plus important est le taux de classification (classification accuracy), sont utilisés.

a) Taux de classification, Erreur de classification

Si nous pouvons tester le classificateur sur tous les exemples possibles, nous saurons exactement quel est son taux de classification. Malheureusement, il est généralement impossible d'effectuer de tels tests ; un estimé devra donc être utilisé à sa place. Pour un ensemble de Qt exemples de test, l'estimation du taux de classification est déterminée par la réalisation des tests sur tous les exemples disponibles. Il s'agit ensuite de calculer soit le rapport des exemples bien classés (N_{juste}), soit de celui des exemples mal classés (N_{faut}). Dans cette thèse nous utiliserons le taux de classification des exemples bien classés donné par :

$$taux = \frac{N_{juste}}{Qt} \tag{1.5}$$

Nous utiliserons aussi l'erreur quadratique totale sur l'ensemble des exemples d'apprentissage (EQT_{app}) ainsi que celle sur les exemples de test (EQT_{test}) :

$$EQT_{app} \sum_{q=1}^{Q} \sum_{k=1}^{K} \left\| T^{(q)} - Z^{(q)} \right\|^2 \tag{1.6}$$

$$EQT_{test} \sum_{q=1}^{Qt} \sum_{k=1}^{K} \left\| T^{(q)} - Z^{(q)} \right\|^2 \tag{1.7}$$

Où : $T^{(q)}$et $Z^{(q)}$sont respectivement la sortie calculée et la sortie désirée correspondantes au $q^{ème}$ exemple ; Q et Qt sont respectivement le nombre d'exemples d'apprentissage et le nombre d'exemples de test et $\|\cdots\|$ est la distance euclidienne.

b) Matrice de confusion

La matrice de confusion représente la façon de distribution de l'erreur de classification des exemples de test à travers les différentes classes. C'est une matrice carrée de dimension $K \times K$ (K est le nombre de classes). La composante a_{ij} de cette matrice indique le nombre d'exemples d'apprentissage dont leur véritable classe est C_i alors qu'ils sont assignés à la classe C_j. L'estimé du taux de classification est calculé comme la trace de la matrice de confusion divisée par le nombre total d'exemples.

c) Ensemble d'apprentissage et ensemble de test

Pour un ensemble donné d'exemples, il est naturel d'opter à l'emploi d'autant d'exemples pour un bon apprentissage du classificateur et également autant d'exemples pour un bon test. Cependant, si nous employons toutes les données pour l'apprentissage et les mêmes données pour le test, nous pourrons sur-apprendre (over-train) le classificateur de sorte qu'il apprend parfaitement les données disponibles et échoue sur les nouvelles données. C'est pourquoi il est important d'avoir un ensemble séparé sur lequel nous examinons le modèle

obtenu. Les principales solutions permettant d'effectuer une meilleure utilisation des exemples disponibles peuvent être récapitulées comme suit [3]:

1. Resubstitution (R-method) : Cette méthode est basée sur l'utilisation de la base de données entière à la fois pour l'apprentissage et pour le test.
2. hold-out (H-method) : La base de données est divisée en deux moitiés. Une partie est utilisée pour l'apprentissage et l'autre pour le test.
3. Validation croisée (cross validation) : Cette méthode est appelée aussi méthode de rotation (rotation method). Elle consiste à choisir un entier K (de préférence facteur de Q) et de diviser la base de données en K sous-ensembles de taille Q/K. L'un des sous-ensembles est ensuite employé pour le test du classificateur entraîné en utilisant l'union des $(K-1)$ sous-ensembles restants. Cette procédure est répétée K fois et les valeurs finales du taux et de l'erreur de classification seront donnés par la moyenne des K estimés.
4. Bootstrap : Cette méthode se base sur la génération aléatoire, à partir de l'ensemble entier, de L ensembles par remplacement. La moyenne des taux d'erreurs, réalisés sur ces ensembles, sera ensuite déterminée.

Le problème de définir la meilleure manière d'organiser les expériences Apprentissage /Test se pose depuis longtemps, et même avec les technologies de calcul modernes on n'arrive pas à s'en passer en raison de la croissance permanente de la taille des données rassemblées. La vieille méthode hold-out est de nouveau employée, d'abord parce que les autres méthodes pourront prendre trop de temps, et deuxièmement parce que la quantité de données pourra être si excessive que l'utilisation de petites parties suffira pour l'apprentissage et le test [3].

Par ailleurs et afin d'éviter le problème de sur-apprentissage, il est plus pratique d'utiliser trois ensembles au lieu de deux. Un de ces ensembles est utilisé pour l'apprentissage, un pour la validation et le troisième pour le test. Les exemples de l'ensemble de test sont toujours non vus pendant le processus d'apprentissage et l'ensemble de validation est utilisé comme un pseudo-test. Le processus d'apprentissage se poursuit jusqu'à l'amélioration des performances du classificateur sur l'ensemble de validation, il devra ainsi être cessé afin d'éviter le problème de sur-apprentissage.

1.4 Taxonomie des méthodes de classification

Les théories de classification sont abondamment évoquées par un grand nombre d'auteurs et il n'y a pas un consensus en une taxonomie unique de ces méthodes. Par exemple, certains auteurs catégorisent les méthodes de classification en méthodes basées sur l'approximation des densités de probabilité et méthodes basées sur l'approximation des fonctions discriminantes. D'autres auteurs considèrent un classement selon l'aspect paramétrique et non-paramétrique des méthodes. L'aspect global ou local est aussi à la base

des catégorisations données par d'autres chercheurs. Nous présentons ci-dessous trois exemples de taxonomies:

1ère taxinomie [4]

1. Théorie de la décision
    a) Théorie des graphes
    b) Statistique
    c) Par les règles
2. Représentation floue et par les réseaux de neurones
    a) Les réseaux de neurones non bouclés
    b) Les réseaux de neurones hybrides
    c) Les cartes auto-organisatrices
3. Structurelle

2ème taxinomie [5]

1. Approximation des densités de probabilité :
    a) Paramétrique (exp. LCD, QDC)
    b) Non-paramétrique (exp. K-nn, Parsen)
    c) Semi paramétrique
2. Approximation des fonctions discriminantes ou des frontières de décision :
    a) Structurel (exp. Arbre de décision, MLP, RBF)
    b) Fonctionnel
        - Linéaire (exp. LCD de Fisher, perceptron)
        - Non-linéaire (exp. Discriminant linéaire généralisé)

3ème taxinomie [6]

1. Classification basée sur l'estimation des densités :
    a) Paramétrique (exp. LCD, QDC)
    b) Non-paramétrique (exp. K-nn, RBF, méthode de kernel)
2. Classification basée sur la régression:
    a) Paramétrique (exp. régression linéaire, régression logistique, MLP)
    b) Non-paramétrique (exp. poursuite de projection, modèles additifs)
3. Autre classificateur (exp. : K-nn avec petites valeurs de K, LVQ basé sur les prototypes)

1.5 Théorie de Bayes

La théorie de décision Bayesienne constitue une approche fondamentale pour résoudre les problèmes de classification en permettant de déterminer les probabilités d'appartenance d'un objet à partir de son observation. Cette théorie est la méthode centrale des approches stochastiques où les problèmes de décision sont traités en termes de probabilités.

1.5.1 Règle de bayes

Soit un problème de classification d'un objet à l'une de K classes : $C_1, C_2, \dots, C_K$. Si l'on ne connait que les probabilités d'appartenance a priori $P(C_k)$ de ces classes et qu'on a été obligé de décider à quelle classe il appartient, on aura 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 3 billes rouges, le meilleur que l'on peut faire est d'assigner cette bille à la classe ayant la plus grande probabilité a priori. On décide donc toujours que c'est une bille bleue. Heureusement, on ne prend jamais de décision dans telle situation (avec si peu d'information). En fait, on dispose généralement de plus d'informations ; chaque objet étant représenté par un ensemble de caractéristiques qui aident à la décision en donnant des informations indispensables.

Soit X le vecteur caractéristique représentant cet objet et $p(X/C_k)$ la fonction de densité de probabilité conditionnelle d'observer X étant donné la classe C_k. Selon Bayes, en connaissant $p(X/C_k)$ et $P(C_k)$, on peut déterminer la probabilité a posteriori $P(C_k/X)$ que la classe de X soit C_k comme suit :

$$P(C_k/X) = \frac{p(X/C_k)P(C_k)}{p(X)} \tag{1.8}$$

La densité de probabilité inconditionnelle $P(X)$ est donnée par :

$$p(X) = \sum_{k=1}^{K} p(X/C_k)P(C_k) \tag{1.9}$$

Cette densité assure que la somme des probabilités a posteriori soit égale à l'unité :

$$\sum_{k=1}^{K} P(C_k/X) = 1 \tag{1.10}$$

La règle de bayes peut être formulé en langage parlé par [2] :

$$probabilité\ a\ posteriori = \frac{vraisemblance\ \times probabilité\ a\ priori}{facteur\ de\ normalisation} \tag{1.11}$$

L'importance de la théorie de Bayes réside dans le fait qu'elle permet d'exprimer les probabilités a posteriori en terme de quantités qui sont souvent faciles à calculer [2]. Les probabilités a priori sont toujours considérées comme connues et même s'il n'est pas le cas, elles peuvent être estimées à partir des proportions des données d'apprentissage appartenant à chaque classe. Les probabilités conditionnelles sont aussi considérées comme connues et elles peuvent être estimées à partir des données d'apprentissage.

1.5.2 Décision

La probabilité de mal-classer un exemple X est minimisée en choisissant la classe C_k ayant la plus grande probabilité a posteriori. Chaque exemple est ainsi classé selon la règle de décision suivante :

$$X \in C_k \quad si \quad P(C_k/X) > P(C_i/X) \ pour\ i = 1, \dots K\ et\ i \neq k \tag{1.12}$$

Nous pouvons utiliser Eq. 1.8 pour réécrire la règle précédente sous la forme :

$$X \in C_k \quad si \quad p(X/C_k)P(C_k) > p(X/C_i)P(C_i) \ pour\ i = 1, \dots K\ et\ i \neq k \tag{1.13}$$

Pour illustrer les différentes formes de densités de probabilité, considérons le problème de classification des exemples de la première et la deuxième classe de la base de données Iris (sétosas et versicolores). Nous admettons une représentation bidimensionnelle, chaque exemple étant représenté par deux caractéristiques : Largeur sépale / longueur sépale et largeur pétale / longueur pétale. En supposant que la probabilité a priori d'appartenance aux classes C_1 et C_2 soient égales ($P(C_1) = 0.5$ et $P(C_2) = 0.5$), les fonctions de densité de probabilité $P(C_1/X)$ et $P(C_2/X)$, ainsi que les probabilités a posteriori d'appartenance ($P(X/C_1)$ et $P(X/C_2)$) correspondantes à la première caractéristique, sont illustrées sur la figure (I.5).

1.5.3 Frontière de décision et erreur de classification

Du fait que la conception d'un classificateur consiste essentiellement à réaliser une règle pour assigner chaque point de l'espace caractéristique à une classe, l'espace caractéristique est ainsi séparé en un ensemble de régions de telle sorte que chaque point appartenant à une région est assigné à la classe correspondante. Pour définir le critère optimal pour le placement des frontières de décision entre ces régions, prenons un problème de

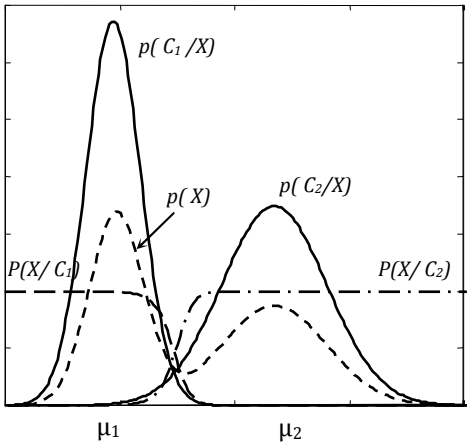


Fig. 1.5. Densités de probabilités correspondants à de la première caractéristique (largeur pétale/longueur pétale) des exemples de la 1ère et la 2ème classes de la base de donnée iris

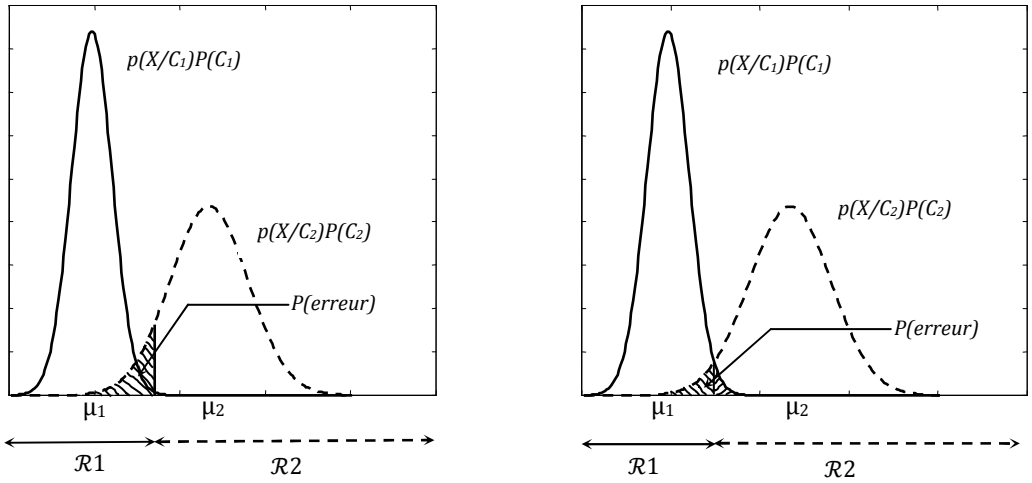


Fig. 1.6. Exemples de régions de décision générées

classification bi-classe (C_1, C_2). Une erreur de mal-classification se produit lorsqu'on assigne un exemple à la classe C_1 tandis qu'il appartient réellement à classe C_2, ou vice-versa. Nous pouvons calculer la probabilité totale d'erreur comme suit [2] :

$$\begin{aligned} P(erreur) &= P(X \in R_2, C_1) + P(X \in R_1, C_2) \\ &= P(X \in R_2/C_1)P(C_1) + P(X \in R_1/C_2)P(C_2) \\ &= \int_{R_2} p(X/C_1)P(C_1)\,dx + \int_{R_1} p(X/C_2)P(C_2)\,dx \end{aligned} \tag{1.14}$$

Où : $P(X \in R_2, C_1)$ est la probabilité jointe d'assigner X à C_2 or il appartient à C_1

Si $p(X/C_1)P(C_1) > p(X/C_2)P(C_2)$ pour un exemple X donné, nous devrons alors choisir R_1 et R_2 de telle sorte que X soit dans R_1. Cela correspond à la règle de décision de l'Eq. 1.13.

Le résultat précédent est illustré graphiquement sur la figure (1. 6). En choisissant une frontière de décision qui coïncide avec la valeur de X pour laquelle les deux distributions se croisent, on peut minimiser l'erreur de classification. Dans le cas multi-classes, la probabilité de classer correctement un nouvel exemple est donnée par [2]:

$$P(correct) = \sum_{k=1}^{K} \int_{R_k} p(X/C_k)P(C_k)\,dx \tag{1.15}$$

Cette probabilité est maximisée en choisissant les régions R_k de telle sorte que chaque exemple X soit assigné à la classe pour laquelle l'intégrale précédente est maximale, ce qui est équivalent aussi à la règle de l'Eq. 1.13.

1.5.4 Minimisation du risque

Dans la plus part des problèmes réels de classification, les risques d'erreur en assignant un exemple à une fausse classe ne sont pas le même pour toutes les classes. Par exemple, prenons le cas de la classification des cibles militaires ; il est plus dangereux de prendre une cible civile pour une cible militaire que l'inverse. Pour prendre de telles situations en considération, un élément s_{ij} est associé avec l'attribution d'un exemple à la classe C_j or il appartenant réellement à la classe C_i. L'espérance de perte est donnée par [2] :

$$o_k = \sum_{j=1}^{K} s_{kj} \int_{R_j} p(X/C_k)\, dx \tag{1.16}$$

L'espérance de perte totale est alors :

$$O = \sum_{k=1}^{K} o_k\, P(C_k)$$

$$= \sum_{j=1}^{K} \int_{R_j} \left\{ \sum_{k=1}^{K} s_{kj} p(X/C_k)\, P(C_k) \right\} dx \tag{1.17}$$

1.6 Classificateurs de base

1.6.1 Classificateurs linéaires et classificateurs quadratiques :

N'importe quel ensemble de fonctions discriminantes obtenues à partir d'une transformation monotone des probabilités a posteriori ($P(C_k/X)$) constituent un ensemble optimal en terme de minimisation d'erreur [1][2]. Nous pouvons alors prendre :

$$g_k(X) = log[p(X/C_k)P(C_k)], k = 1, \dots, K \tag{1.18}$$

En supposant que toutes les classes soient normalement distribuées avec une moyenne μ_k et une matrice de covariance Σ_k, l'Eq. (1.18) prend la forme :

$$g_k(X) = log[P(C_k)] + log\left\{ \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma_k|}} exp\left[-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k) \right] \right\}$$

$$= log[P(C_k)] - \frac{n}{2} log(2\pi) - \frac{n}{2} log(|\Sigma_k|) - \frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k) \tag{1.19}$$

En supposant également que les matrices de covariance soient les mêmes, nous obtiendrons un nouvel ensemble de fonctions discriminantes :

$$g_k(X) = log[P(C_k)] - \frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)$$

$$= w_{k0} + w_k^T X \tag{1.20}$$

Où : $w_{k0} \in R$ et $W_k \in R^N$ sont les coefficients de la fonction discriminante linéaire g_i

Ces fonctions peuvent être déterminées à partir des données en estimant la moyenne et la matrice de covariance, mais le classificateur obtenu n'est pas équivalant au classificateur de Bayes [3].

Si l'on suppose aussi que toutes les classes soient normalement distribuées, mais cette fois avec des matrices de covariance différentes, nous obtiendrons à partir de l'Eq. (1.19) l'équation du classificateur quadratique donnée par :

$$g_k(X) = w_{k0} + w_k^T X + X^T W_k \mathrm{X} \tag{1.21}$$

Avec :

$$w_{k0} = log[P(C_k)] - \frac{1}{2}\mu_k^T \Sigma_k^{-1} \mu_k - \frac{1}{2}\, log(|\Sigma_k|)$$

$$w_k = \Sigma_k^{1} \mu_k$$

$$W_k = -\frac{1}{2}\Sigma_k^{-1}$$

Les paramètres du classificateur linéaire et ceux du classificateur quadratique peuvent être déterminés à partir des données. Soit une base de données contenant Q exemples appartenant aux K classes $(C_1, C_2, \ldots, C_K)$ et soit $Q_k$le nombre d'exemples appartenant à la classe C_k. Les moyennes sont données par :

$$\hat{\mu}_k = \frac{Q_k}{Q} \tag{1.22}$$

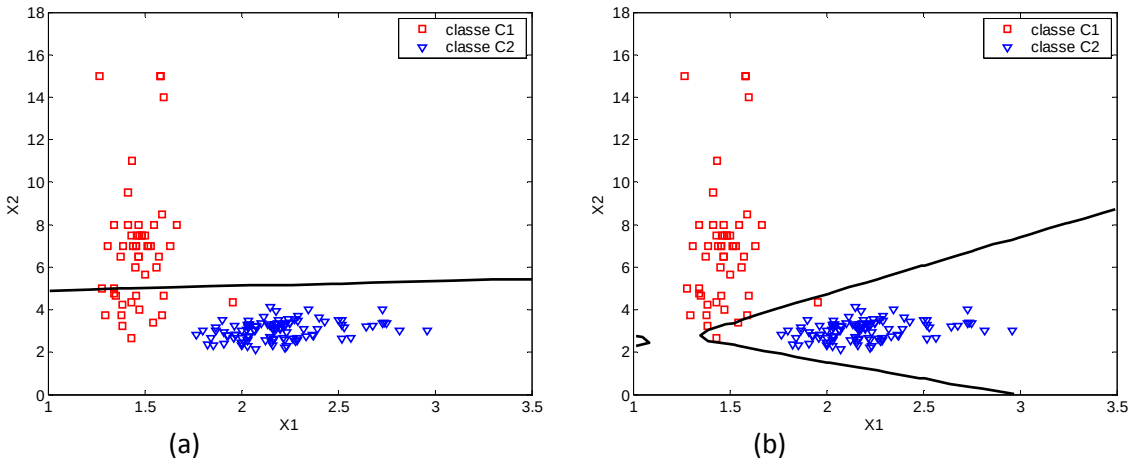


Fig. 1.7. Classification de la base de données Iris
(a) Classificateur linéaire
(b) Classificateur quadratique

Et la matrice de covariance est déterminée par :

$$\hat{\Sigma}_k = \sum_{X \in C_k} (X - \hat{\mu}_k)(X - \hat{\mu}_k)^T \tag{1.23}$$

Considérons de nouveau le problème bidimensionnel de classification des exemples de la première et la deuxième classe de la base de données Iris, et supposons aussi que la probabilité a priori d'appartenance aux classes C_1 et C_2 soient égales. Les frontières de décision générées par le classificateur linéaire et le classificateur quadratique sont illustrées dans la figure (1.7).

1.6.2 Classificateurs non paramétriques :

La construction des classificateurs précédents repose sur l'hypothèse d'une loi de distribution explicitement paramétrée pour chaque classe. En réalité, la forme de cette distribution est souvent difficile à choisir par manque d'un modèle suffisamment adéquat et complet. Dans la modélisation non paramétrique, on se libère de la supposition d'une loi paramétrique et l'on estime la distribution de densité de chaque classe au voisinage des observations. Les principales techniques non paramétriques se fondent sur le fait que la probabilité P qu'un vecteur X soit dans une région R est donnée par :

$$P = \int_R p(\acute{x}) d\,\acute{x} \tag{1.24}$$

Pour un ensemble de Q exemples générés par cette loi ($p(x)$), la probabilité que k éléments parmi ces exemples appartienne à la région R est donnée par la loi binomiale:

$$P_k = \binom{Q}{k} P^k (1 - P)^{Q-k} \tag{1.25}$$

La valeur P peut être estimée comme la proportion des points dans R par rapport au nombre total d'exemples [2] :

$$P \cong \frac{k}{Q} \tag{1.26}$$

Si nous supposons que $p(x)$ est continue et que la région R est si petite que $p(x)$ n'y change pas considérablement, nous pourrons écrire :

$$\int_R p(\acute{x}) d\,\acute{x} \cong p(x)V \tag{1.27}$$

Où : x est un point appartenant à R et V est le volume de R. En combinant les Eq. (1.26) et (1.27), nous arrivons à l'estimation suivante de $p(x)$ [1][3]:

$$p(x) \cong \frac{k}{QV} \tag{1.28}$$

Quand Q tend à l'infini et la région R se rétrécit à un point ($V \to O$), l'Eq 1.28 donne la valeur exacte de $p(x)$. Cela constitue un important point de départ de plusieurs méthodes non-paramétriques de classification [3].

a) Les Fenêtres de Parzen

Les fenêtres de Parzen [7], appelé aussi fonctions de Kernel, permettent d'estimer les densités de probabilité en se basant sur l'Eq. (1.28) : Q et V sont fixes tandis que k est définie à partir des données. Prenons une région R_k, supposée être un hyper-cube centré en x_k, de dimension d et de largeur de faces h_k. Le volume de cet hyper-cube est [1][2]:

$$V_k = h_k^d \tag{1.29}$$

Par définition, la fenêtre de Parzen est donnée par :

$$\varphi(U) = \begin{cases} 1 & si\ |u_j| < 1/2 \quad j = 1, \dots, d \\ 0 & autrement \end{cases} \tag{1.30}$$

Le nombre d'exemples dans cet hyper-cube est donné par :

$$N_k = \sum_{k=1}^{Q} \varphi\left(\frac{x - x_k}{h_k}\right) \tag{1.31}$$

En remplaçant ce résultat dans l'Eq. (1.28) nous obtiendrons:

$$p_k(x) = \frac{1}{Q} \sum_{k=1}^{Q} \frac{1}{V_k} \varphi\left(\frac{x - x_k}{h_k}\right) \tag{1.32}$$

La forme de cet estimé peut être lissée en choisissant de différentes formes de fonction de Kernel ($\varphi(u)$) dont la plus utilisée est la suivante [2] :

$$p_k(x) = \frac{1}{Q} \sum_{k=1}^{Q} \frac{1}{(2\pi h^2)^{d/2}} exp\left(\frac{\|x - x_k\|^2}{2{h_k}^2}\right) \tag{1.33}$$

D'une façon générale, si la fonction de kernel satisfait : $\varphi(u) \geq 0$ et $\int \varphi(u)du = 1$, alors l'estimé de l'Eq. (1.32) satisferait $p(x) \geq 0$ et $\int p(x)dx = 1$ [2]

Prenons comme exemple la troisième caractéristique de la base de données Iris. La figure (1. 8) représente l'histogramme correspondant et les estimations obtenues pour différentes valeurs de h (0.1, 0.5 et 1). Nous constatons que le paramètre h joue le rôle d'un lisseur et que son choix est important pour l'obtention d'une bonne estimation. Si h est choisi avec de

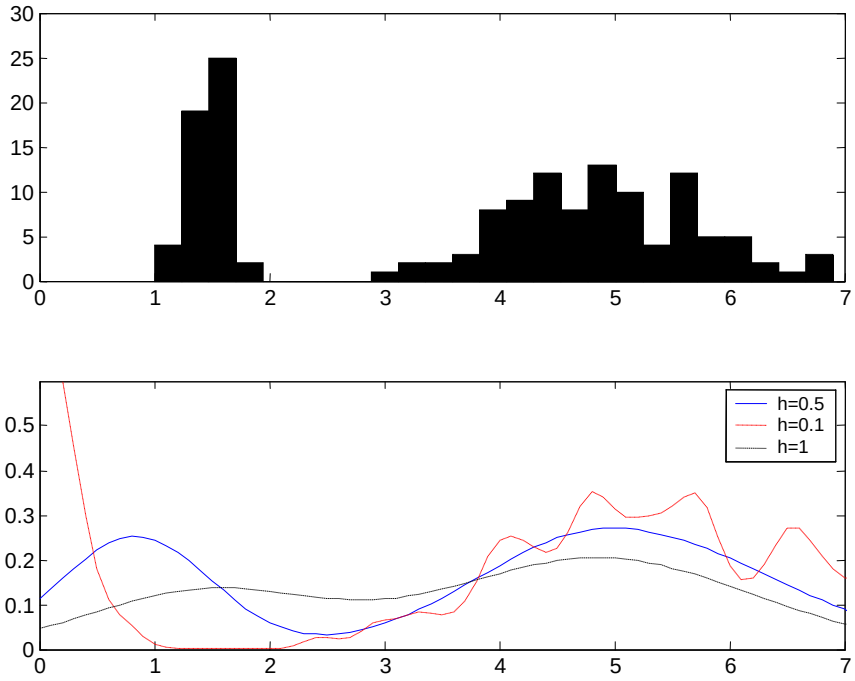


Fig. 1.8. Estimation non-paramétrique de la distribution de densité de la troisième caractéristique de la base de données Iris

grandes valeurs, l'estimation de la densité sera alors sur-lissé (over-smoothed) et la nature bimodale de la distribution sera perdue [2]. En revanche, si h est choisi avec de petites valeurs, l'ensemble des propriétés des données particulières sera plus considéré que la structure globale de la distribution sous-jacente.

b) Règle des K-plus proches voisins

Parmi les problèmes de la méthode des fenêtres de Parzen, c'est l'utilisation d'une longueur (h) fixe pour toutes les données d'apprentissage. La méthode des K-plus proches voisins tente de contourner ce problème en choisissant un volume variant. Cette méthode consiste à considérer une petite hyper-sphère centrée en x et à varier son rayon jusqu'à ou elle contienne exactement un nombre k d'exemples. L'estimé de la densité de probabilité est alors donnée par l'Eq. (1.28) ou V est le volume de l'hyper-sphère.

Soit un problème de classification multi-classes dans lequel nous avons Q exemples d'apprentissage. Prenons une hyper-sphère autour d'un point x qui regroupe k exemples sans tenir compte de leurs appartenances. Si cette hyper-sphère, de volume V, contient N_k exemple appartenant à la classe C_k, nous aurons une approximation des densités conditionnelles en utilisant l'Eq. (1.28):

$$p(x/C_k) \cong \frac{N_k}{Q_k V} \tag{1.34}$$

Où : Q_k le nombre d'exemples appartenant à la classe C_k. De la même façon, les densités non-conditionnelles peuvent-être estimées par :

$$p(x) \cong \frac{k}{QV} \tag{1.35}$$

Les probabilités a priori peuvent être estimées par :

$$P(C_k) \cong \frac{Q_k}{Q} \tag{1.36}$$

En utilisant la théorie de Bayes:

$$P(C_k/X) = \frac{p(X/C_k)P(C_k)}{p(X)} = \frac{N_k}{Q_k} \tag{1.37}$$

Le classificateur de Bayes basé sur les approximations ci-dessus assignera x à la classe avec la probabilité postérieure la plus élevée, c.-à-d., la classe la plus représentée parmi les k voisins les plus proches de x.

1.6.3 Les arbres de décision

Il est normal et intuitif de classifier en se basant sur une séquence de questions dans lesquelles la prochaine question posée dépendra de la réponse de la question courante. Cette approche est particulièrement utile dans le cas des données non-métriques du fait que toutes les questions peuvent être posées sous la forme de 'oui/non', 'vrai/faux' ou une réponse appartenant à un ensemble de valeurs finies dans un style qui n'exige aucune notion de métrique. Les arbres de décision peuvent être utilisés en tant que classificateurs dont la structure est basée sur une séquence de questions. De tels classificateurs ont trois caractéristiques importantes [3] :

1. Si tous les objets sont distinguables, c.-à-d. pas d'éléments identiques appartenant aux classes différentes, nous pourrons établir un arbre de décision avec une erreur de resubstitution nulle. Cela place les arbres de décision dans le groupe de classificateurs qui risquent d'apprendre les données d'apprentissage par cœur de sorte que des petits changements des données pourront mener à un classificateur structuré différemment.

2. Les arbres de décision sont intuitifs parce que le processus de décision peut être tracé comme une séquence de décisions simples.

3. Les caractéristiques binaires et les caractéristiques avec un nombre restreint de catégories conviennent parfaitement à la mise en œuvre des arbres de décision. En revanche pour les caractéristiques quantitatives, un point de division doit être établi pour les transformer en caractéristiques catégoriques ; les arbres de décision ne se fondent pas sur un concept de distance dans l'espace caractéristique.

L'arbre de décision de la figure (1. 9) représente un exemple de classification de fruits (reporté de [3]) en se basant sur la couleur, la taille, la forme et le goût. Ce graphe illustre un avantage des arbres par rapport à d'autres classificateurs tels que les réseaux de neurones ;

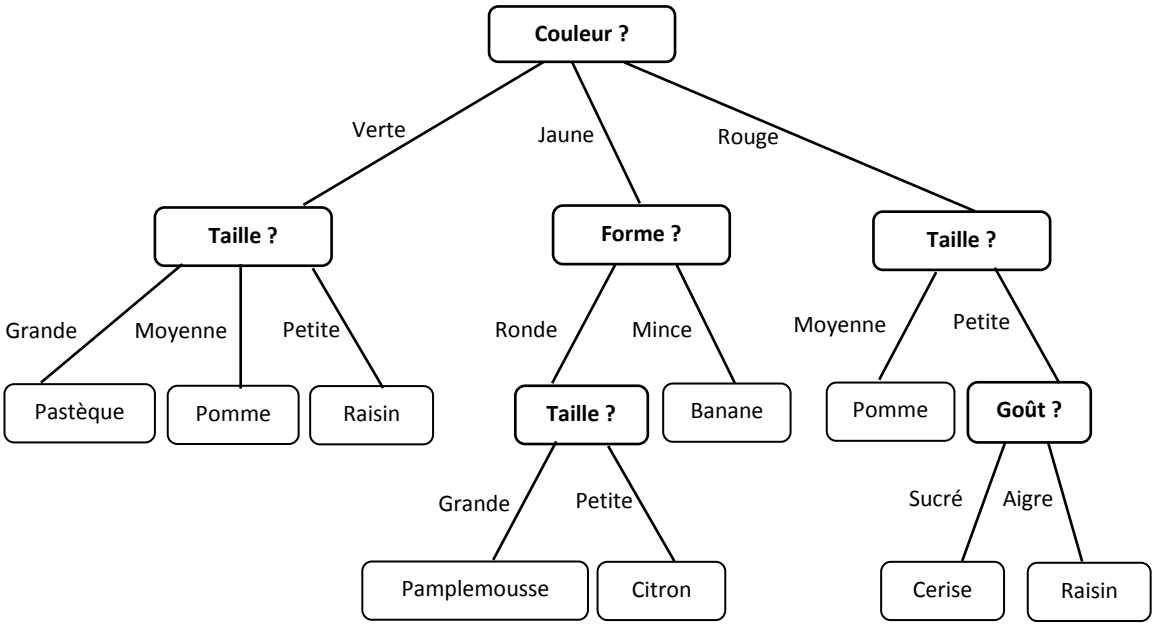


Fig. 1.9. Un arbre de décision permettant de classifier les fruit selon leurs couleur, taille, forme et goût

c'est la possibilité d'interprétation. Il est très facile de transformer les informations contenues dans un tel arbre comme des expressions logiques. D'abord, nous pouvons facilement interpréter la décision pour n'importe quel exemple particulier comme conjonction de décisions le long du chemin à son nœud correspondant. Par exemple, si les propriétés sont {goût, couleur, forme, taille}, le modèle X = {sucré, jaune, mince, moyen} est classifié comme banane. En second lieu, nous pouvons obtenir des interprétations claires des classes elles-mêmes en créant des descriptions logiques en utilisant des conjonctions et des disjonctions. Par exemple, cet arbre montre que : Pomme = (vert ET moyen) OU (rouge ET moyen).

1.7 Combinaison des classificateurs

1.7.1 Principes et objectifs

Les systèmes multi-classificateurs ont été introduits comme importantes alternatives aux méthodes classiques basées sur l'utilisation d'un seul classificateur, afin d'améliorer les performances de classification. Bien qu'il y ait beaucoup de questions sans réponse au sujet de l'adaptation des classificateurs aux problèmes réels, la combinaison des classificateurs est devenue rapidement répandant et appréciant beaucoup d'attention des communautés de la reconnaissance de formes et de l'apprentissage automatique. L'intérêt croissant de ces nouveaux systèmes consiste essentiellement dans les points suivants [3]:

1. Supposons que nous avons plusieurs classificateurs ayant de bonnes performances d'apprentissage pour un problème donné. Nous pouvons sélectionner l'un d'eux comme solution, mais nous risquons de faire un mauvais choix au niveau de la

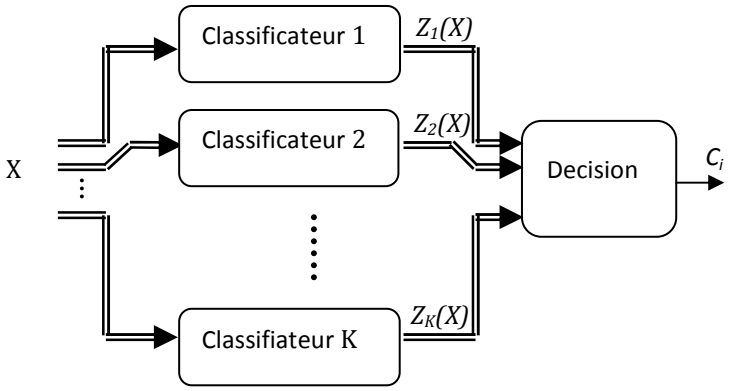


Fig.10. Un système multi-classificateurs

généralisation. Ainsi, une option plus sûre est d'employer tous les classificateurs et de faire la moyenne de leurs sorties.

2. Dans les méthodes classiques, il est difficile de choisir le classificateur approprié à un problème particulier parmi un ensemble de classificateurs considérés. Il est même possible que cet ensemble ne contienne pas le classificateur optimal. Par exemple pour la résolution d'un problème non linéaire, si nous considérons un ensemble de classificateurs linéaires, il est impossible d'y trouver le classificateur optimal. Toutefois, nous pouvons résoudre un problème non linéaire en combinant un ensemble de classificateurs linéaires.
3. La combinaison des classificateurs permet d'exploiter les informations complémentaires qui peuvent être obtenues en utilisant de différents classificateurs.
4. Certains algorithmes d'apprentissage effectuant une descente de gradient ou une recherche aléatoire peuvent conduire aux optimums locaux et l'utilisation de différents classificateurs, qui commencent à partir de différents points, pourrait surmonter ce problème.

Un système multi-classificateurs consiste généralement en un ensemble de classificateurs et une fonction de décision pour combiner leurs sorties (fig.1.10).

1.7.2 Combinaison des sorties

Les techniques de combinaison des sorties des classificateurs sont devenues elles même une importante direction de recherche et plusieurs approches ont été proposées à ce sujet. Les méthodes de base de combinaison sont : a règle du vote, la règle du produit, la règle de la moyenne et la règle du maximum. D'autres méthodes de combinaison sont basées sur la théorie de l'évidence de Dempster-Shafer [8][9][10], sur le calibre de décision (template decision) [11] ou sur l'intégrale floue [12] [13]. Considérons un problème de classification K-classes en utilisant J classificateurs, et notons $z_k^{(j)}(X)$ la $k^{ème}$ sortie du $j^{ème}$ classificateur recevant en entrée un exemple X. Les règle de base sont ainsi données par :

La règle de la moyenne :

$$F_{moyenne}(X) = \arg\max_k \left\{ \frac{1}{J} \sum_{j=0}^{J} z_k^{(j)}(X) \right\} \quad (1.38)$$

La règle du produit :

$$F_{produit}(X) = \arg\max_k \left\{ \frac{1}{J} \prod_{j=0}^{J} z_k^{(j)}(X) \right\} \quad (1.39)$$

La règle du maximum :

$$F_{max}(X) = \arg\max_k \left\{ z_k^{(j)}(X),\ \ j = 1 \dots J \right\} \quad (1.40)$$

1.8 Conclusion

Dans ce chapitre, nous avons essayé de fournir les bases nécessaires permettant de suivre les démarches faites dans les chapitres suivants. Nous nous sommes intéressés aux concepts de base de la classification que nous avons jugés indispensables pour la mise en œuvre et l'évaluation de la méthode proposée. Nous avons ainsi évoqué les techniques d'apprentissage, la théorie de Bayes, les méthodes d'évaluation des classificateurs et les principes de combinaison des classificateurs.

Dans le chapitre qui suit, nous aborderons les réseaux de neurones artificiels qui constituent de puissants outils d'intelligence artificielle, et qui ont prouvé leur efficacité dans le domaine de la classification.

Chapitre 2

Les réseaux de neurones

Du fait que l'objectif principal de notre travail est d'améliorer des performances de la classification par les réseaux de neurones, nous avons consacré ce chapitre à leur égard. Nous tenterons d'y cerner l'essentiel de ces outils mathématiques en évoquant leurs principes de modélisation, leurs apprentissages et leurs architectures. Bien que notre travail se limite à l'étude d'un type particulier de réseaux de neurones (le perceptron multicouche), nous présenterons à la fin de chapitre quelques modèles de base de ces systèmes.

2.1 Introduction

L'idée d'élaborer un modèle mathématique du cerveau humain, en quête de reproduire ses aptitudes intellectuelles, est à l'origine de la création des réseaux des neurones artificiels. Les premiers travaux sur ces systèmes ont été menés par les neurologues Warren McCulloch et Walter Pitts en 1943 [14]. Ces travaux ont été suivis en 1949 par le travail de Donald Hebb qui a proposé une simple règle d'apprentissage dans son ouvrage "The Organization of Behaviour" [15]. Le premier processus artificiel capable d'apprendre par expérience a été proposé par Franck Rosenblatt [16] en 1957, c'était le perceptron. Mais, Marvin Lee Minsky et Seymour Papert [17] ont publié un travail énonçant les limitations de ce dernier, et principalement l'impossibilité de traiter des problèmes non linéaires. Cette conclusion pessimiste a étendu tous les modèles des réseaux de neurones et a abaissé leurs intérêts pendant deux décennies. Il a fallu attendre l'apparition du perceptron multicouche, introduit par Rumelhart [18] en 1986, pour donnée renaissance aux réseaux de neurones. Cette découverte a été une vraie révolution qui a permis aux réseaux de neurones de connaitre un essor considérable. Actuellement, la littérature sur les réseaux de neurones est devenue énorme et ne cesse de croitre.

Malgré la réussite récente des réseaux de neurones artificiels, la communauté scientifique est toujours loin de la mise en œuvre de machines capables de reproduire les capacités de calcul des systèmes nerveux, même les plus simples. Il s'agit en fait d'une simple modélisation du comportement des neurones biologiques qui ne pourra, en aucune manière, de représenter leurs complexités de fonctionnement. En revanche, cette modélisation permet de mettre en œuvre des processus ayant de grandes capacités de calcul en se basant sur des simples cellules de base : les neurones formels.

2.2 Définition d'un réseau de neurones

Les réseaux de neurones possèdent une littérature énorme qui ne cesse de croitre de jour en jour et il n'y a pas de définition universellement tenue pour eux. Cependant, nous considérons quelques définitions qui sont, à notre avis, suffisantes et complémentaires. Selon Haykin [19], un réseau de neurones est un processeur massivement distribué en parallèle qui a une disposition naturelle pour stocker de la connaissance empirique et la rendre disponible à l'usage. Il ressemble au cerveau sur deux aspects. D'abord, la connaissance est acquise par le réseau au travers d'un processus d'apprentissage, et deuxièmement ; les connexions entre les neurones, connues sous le nom de poids synaptique, servent à stocker la connaissance. Selon Zurada [20], les systèmes de neurones artificiels, ou réseaux de neurones, sont des systèmes physiques cellulaires qui peuvent acquérir, stocker et utiliser de la connaissance empirique. Selon Nigrin [21], un réseau de neurones est un circuit composé d'un nombre très important d'unités de calcul simples basées sur des neurones. Chaque élément opère seulement sur l'information locale. Chaque élément opère de façon asynchrone; il n'y a donc pas d'horloge générale pour le système.

En nous basant sur les définitions précédentes, nous pouvons tirer la définition suivante : un réseau de neurones est un modèle mathématique qui tente de reproduire quelques fonctions du cerveau humain, telles que : le parallélisme, l'acquisition des connaissances au travers d'un processus d'apprentissage, le stockage des connaissances et la possibilité d'utilisation de ces connaissances.

2.3 Domaines d'applications des réseaux de neurones

L'essor des réseaux de neurones dans divers domaines actuels est certainement dû à leurs grandes capacités de calcul et à leurs hautes habilités d'apprentissage. De plus, l'estimation de leurs paramètres est indépendante de la complexité du problème traité ce qui leur permet d'être bien adaptés aux problèmes actuels qui ne cessent d'être de plus en plus complexes. Les applications des réseaux de neurones peuvent être récapitulées en trois grands domaines [22] :

- La modélisation : La plus part des problèmes industriels ou de recherche, que ce soit mécanique, physique, chimique ou même économique, nécessitent une représentation à l'aide d'un modèle mathématique permettant de reproduire le comportement du processus mis en œuvre. De telles tâches nécessitent des outils de calcul ayant de grandes capacités de calcul, d'apprentissage et surtout des outils dont leur conception est peu dépendante de la complexité et de la taille du problème traité. Les réseaux de neurones semblent être l'une des solutions les plus adéquates à ce type de problèmes.
- La commande : Commander un processus industriel consiste à concevoir un système permettant le calcul de la commande à appliquer à ce processus de manière à lui

assurer un comportement dynamique désiré. Les réseaux de neurones permettent de bonnes performances en tant que partie de commande à cause de leur souplesse d'auto adaptation.

- La classification : Une autre grande catégorie de problèmes industriels consiste à attribuer, de façon automatique, un objet à une classe parmi d'autres classes possibles. La résolution de ce type de problèmes demande de représenter les exemples à classifier à l'aide d'un ensemble de caractéristiques. Il s'agit ensuite de concevoir un système capable de classifier ces exemples en se basant sur leur représentation et les réseaux de neurones sont particulièrement bien adaptés à ce type de problème. La classification est d'ailleurs le domaine privilégié des réseaux de neurones.

2.4 Principes de modélisation des Réseaux de neurones

2.4.1 Le neurone biologique

Le neurone constitue l'unité de base de l'organisation du système nerveux. Ce sont des cellules spécialisées dans le traitement des signaux électriques et 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. La structure des neurones est parfaitement adaptée à leurs tâches. Un neurone se compose essentiellement de quatre parties : les dendrites, le corps cellulaire, l'axone et les synapses (fig.2.1).

- 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

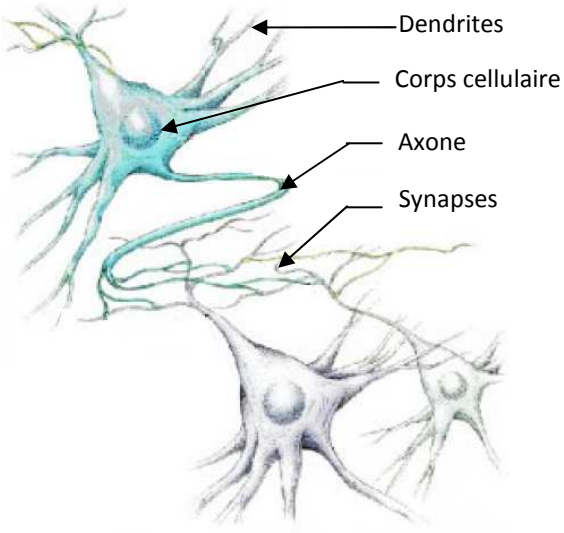


Fig. 2.1. Le neurone biologique (œuvre d'artiste)

neurone.

- Les dendrites : Ils constituent les entrées principales du neurone, ils captent les signaux envoyés vers eux. Leur longueur est de quelques dizaines de microns.
- L'axone : C'est la sortie du neurone qui conduit l'influx nerveux sous 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.
- Les synapses : Ce sont des jonctions entre deux neurones et qui sont essentielles dans le fonctionnement du système nerveux.

2.4.2 Fonctionnement du neurone biologique

La transmission de l'information s'effectue dans un seul sens : des dendrites vers les axones. Chaque neurone reçoit les informations provenant des autres neurones sous forme d'un signal électrique par ces dendrites. Si la somme de ces signaux est excitatrice, le neurone émettra à son tour un signal électrique qui se propage par les axones aux connecteurs terminaux, et ensuite aux dendrites des autres neurones. Selon Hebb [15], l'apprentissage des neurones se fait par la modification des résistances électriques des connexions dendrites.

2.4.3 Le neurone formel

La première modélisation du neurone remonte aux années 40 où McCulloch et Pitts [14] ont proposé le premier modèle de neurones formels. C'est un modèle mathématique qui réalise une somme pondérée des signaux qui lui parviennent et déclenche une réponse si cette somme dépasse un certain seuil. La figure (1.2 a) résume la chaîne de traitement développée par ce neurone. Ce modèle n'a pas possédé une règle d'apprentissage jusqu'à 1949 où Hebb

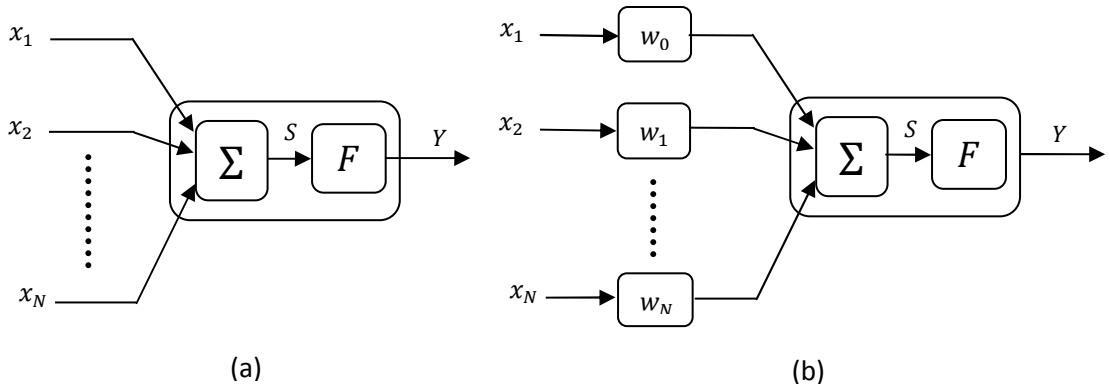


Fig. 2.2. Le neurone formel
(a) Modèle de McCulloch et Pitts
(b) Modèle avec capacité d'apprentissage

[15] a proposé un principe d'apprentissage sans avoir donné d'équations.

La valeur de la sortie (Y) résulte de la somme des entrées (x_i) pondérées par des coefficients (w_i) et du calcul d'une fonction d'activation (F) de cette somme pondérée. La formalisation mathématique de son comportement est donnée par :

$$Y = F(S) = F\left(w_0 + \sum_{n=1}^{N} w_n x_n\right) \tag{2.1}$$

Il est à noter que cette modélisation simplifiée est loin de fournir une explication exacte concernant la complexité de fonctionnement des neurones biologique. Malgré cela, cette formalisation permet d'étudier les connexions entre ces neurones dans d'autres processus plus complexes comportant plusieurs neurones interconnectés.

2.4.4 Les fonctions d'activation

La fonction d'activation peut avoir plusieurs formes différentes. Dans leur modèle d'origine, McCulloch et Pitts ont utilisé des fonctions d'activation à seuil. Les états des neurones avec de telles fonctions sont binaires et les ensembles de valeurs possible les plus couramment utilisées sont {-1, 1} ou {0, 1}. Les deux formes des fonctions à seuil sont :

$$Y_{seuil} = \begin{cases} 1 & si\ S > \theta \\ 0 & sinon \end{cases} \tag{2.2}$$

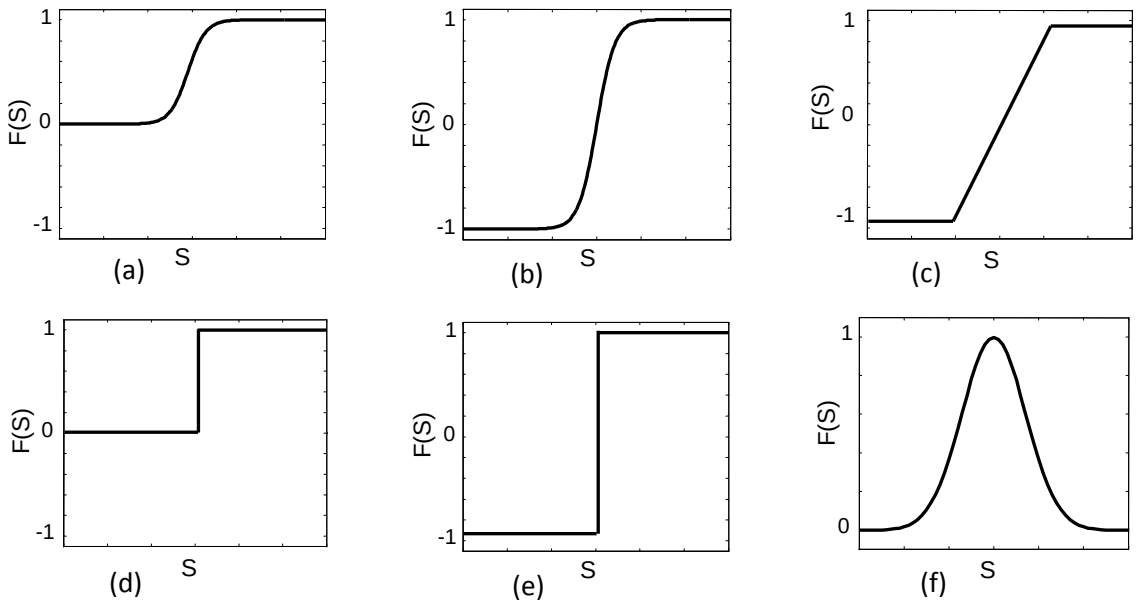


Fig. 2.3. Exemples de fonctions d'activation
(a) Fonction seuil (b) Fonction seuil bipolaire
(c) Fonction RBF (d) Fonction sigmoïde
(e) Fonction sigmoïde bipolaire (f) Fonction linéaire saturée

$$Y_{signe} = \begin{cases} +1 & si\ S > \theta \\ -1 & sinon \end{cases} \quad (2.3)$$

L'examen du comportement des cellules nerveuses par rapport aux signaux qui leur arrivent a donné lieu à l'utilisation d'un autre type très intéressant comme fonction d'activation : les fonctions sigmoïdales. Une fonction sigmoïde est donnée par :

$$F_{sigmoide}(S) = \frac{1}{1 + exp(-S)} \quad (2.4)$$

La figure (2.3) donne quelques exemples des fonctions d'activation. Les fonctions RBF forment une catégorie particulière des réseaux de neurones : Les réseaux RBF. Ces modèles seront discutés dans le paragraphe (§.2.7.3).

2.5 Apprentissage des réseaux de neurones

Reproduire les capacités d'apprentissage humaines est, sans doute, l'une des ambitions les plus importantes de la modélisation des réseaux de neurones artificiels. L'apprentissage est alors l'une de leurs propriétés fondamentales. C'est le processus permettant au réseau de se spécialiser sur un problème spécifique à partir de son expérience. Il consiste généralement à modifier les poids synaptiques jusqu'à ce que le réseau puisse effectuer la tâche désirée. Il s'agit de configurer les valeurs des poids synaptiques censés stocker les informations acquises. D'une façon générale, l'apprentissage se traduit par une modification dans la valeur des poids reliant les neurones du réseau. Chaque poids w_{ij} reliant un neurone i au un neurone j à l'itération (t) est modifié selon l'équation générale suivante :

$$w_{ij}(r) = w_{ij}(r-1) + \Delta w_{ij}(r-1) \quad (2.5)$$

Où : $w_{ij}(r)$ et $w_{ij}(r-1)$ sont respectivement les valeurs de ce poids à la $r^{ème}$ et la $(r-1)^{ème}$ itération et $\Delta w_{ij}(r-1)$ est le changement correspondant.

Selon que l'apprentissage est supervisé ou non, les méthodes d'apprentissage des réseaux de neurones peuvent être catégorisées en deux grandes familles. Dans la première catégorie, chaque exemple de la base d'apprentissage est couplé à une solution désirée. Cette dernière permet au réseau de connaître ces erreurs et de s'adapter à la présentation de chaque exemple d'apprentissage afin de se rapprocher du résultat souhaité. En revanche, dans la deuxième catégorie le réseau ne dispose pas de la solution désirée sur les exemples d'apprentissage pour l'aider à ajuster ses paramètres. Il doit chercher à représenter au mieux l'espace des exemples qui lui sont présentés. Les règles de base d'apprentissage des réseaux de neurones sont les suivantes :

2.5.1 La règle de Hebb

La règle de Hebb [15] est une règle d'apprentissage supervisé basée sur le renforcement des connexions entre neurones. Son principe est le suivant : si deux neurones d'une part et

d'autre d'une synapse sont activés de façon synchrone et répétée, la connexion synaptique sera alors renforcée. Bien que Hebb n'a donné aucune équation, le principe d'apprentissage Hebbien a eu une influence philosophique sur les travaux qui lui succédèrent. Il a été en premier lieu derrière l'idée d'incorporation des poids ajustables dans le modèle de McCulloch et Pitts. Ensuite, Il a été à la base de diverses approches d'apprentissage tel que le réseau de Hopfield (détaillé dans le paragraphe §.2.7.5) dont la règle d'apprentissage est donnée par :

$$\Delta w_{ij}(r-1) = \eta x_i^{(q)} x_j^{(q)} \tag{2.6}$$

2.5.2 La règle de Widrow-Hoff

La règle d'apprentissage de Widrow-Hoff [23], ou des moindres carrés (LMS, Least Square Sum), est une règle d'apprentissage supervisé basée sur la correction d'erreurs observées en sortie. Cette règle consiste à minimiser une fonction coût caractérisée par l'erreur quadratique moyenne. Pour un ensemble d'apprentissage contenant Q paires entrée/sortie désirée $\{(X^{(q)}/T^{(q)})\}, q = 1, \dots Q$ où $X^{(q)}$ et $T^{(q)}$ représentent respectivement la $q^{ème}$ entrée et la $q^{ème}$ sortie désirée, l'erreur ($e(t)$) à l'itération r est donnée par :

$$e(t) = T(r) - Y(r) \tag{2.7}$$

Où : $Y(r)$ est la sortie calculée du réseau. La fonction coût est :

$$F(X) = e^2(r) \tag{2.8}$$

L'apprentissage selon la règle LMS consiste à calculer le gradient à chaque présentation d'un exemple d'apprentissage. Le changement de poids est alors :

$$\begin{aligned} \Delta w_{ij}(t) &= -\eta \nabla F(X) \\ &= -\eta \frac{\partial e^2(r)}{\partial w_{ij}} \end{aligned} \tag{2.9}$$

Cette règle de correction permet donc aux neurones d'adapter leurs poids pour se rapprocher à une valeur désirée correspondante à chaque exemple présenté. Cette règle a été utilisée pour l'apprentissage de l'ADALINE [23] (détaillé dans le paragraphe §.3.2.2) dans lequel chaque neurone i corrige ses poids w_{ij} à l'itération r selon l'équation suivante :

$$\Delta w_{ij}(r) = \Delta w_{ij}(r-1) - \eta(t_i - y_i)x \tag{2.10}$$

Où : t_i et $y_i$sont respectivement la sortie désirée et la sortie calculée correspondantes au neurone i ; x est l'entrée et η est une constante positive appelée pas d'apprentissage.

2.5.3 Apprentissage compétitif

C'est un apprentissage non supervisé qui consiste à faire une compétition entre les neurones d'un réseau pour déterminer lequel sera actif à un instant donné. Contrairement aux autres types d'apprentissage, où la mise à jour des poids de tous les neurones d'effectue simultanément, ce mode d'apprentissage considère à chaque fois un neurone vainqueur, et parfois un ensemble de voisins du vainqueur, et seuls les poids de ces neurones seront adaptés.

Les poids d'un neurone vainqueur seront modifiés de telle sorte qu'ils se rapprochent de l'exemple X présenté en entrée et pour lequel ce neurone a gagné la compétition contre tous les autres neurones. En revanche, les poids d'un neurone qui ne gagne aucune compétition ne seront pas modifiés. La règle d'apprentissage est donnée par :

$$\Delta w_i = \begin{cases} \eta_1 (X - w_i) \; si \; le \; neurone \; est \; vainqueur \\ \eta_2 (X - w_i) \; si \; le \; neurone \; est \; voisin \; du \; vainqueur \\ 0 \qquad\qquad Autrement \end{cases} \tag{2.12}$$

Avec : $\eta_1$et η_2 sont les taux d'apprentissage et $\eta_1 < \eta_2$. Cette incrémentation reflète la distance entre la valeur courante du poids et la valeur d'entrée.

Ce type d'apprentissage caractérise une classe de réseaux de neurones tels que le réseau de Kohonen, qui sera présenté dans le paragraphe (§.2.7.5), où la décision est prise en déterminant quel neurone représente le mieux l'exemple d'entrée.

2.6 Architecture des réseaux de neurones

Bien que la modélisation du neurone formel n'est qu'une simple imitation du neurone biologique, de bonnes organisations de ces simples cellules de base permettent d'avoir des réseaux de neurones ayant de grandes capacités de calcul et d'apprentissage. Néanmoins, l'organisation topologique des réseaux de neurones n'est généralement le résultat d'aucunes imitations des structures neurobiologiques.

Un réseau de neurones artificiels est un ensemble interconnecté de neurones formels fonctionnant en parallèle. On peut distinguer entre les réseaux de neurones selon le fait qu'ils sont à couche ou non. Dans les réseaux à couche, les neurones appartenant à la même couche ne sont pas interconnectés et ils ont généralement les mêmes propriétés. Dans ces modèles, le calcul s'effectue de couche en couche (de l'entrée vers la sortie) et les neurones de la même couche opèrent simultanément.

D'autre part, selon qu'il y a des retours ou non (des neurones de sortie vers ceux d'entrée), on distingue deux grandes familles d'architecture : les réseaux de neurones non bouclés et les réseaux de neurones bouclés. Dans les réseaux du premier type, l'information se propage de l'entrée vers la sortie sans retour et les neurones de sortie n'influencent jamais sur les neurones d'entrée. Cette architecture est la plus utilisée dans la classification,

l'approximation des fonctions et la modélisation des procédés. En revanche, les réseaux du second type (appelés aussi les réseaux récurrents) contiennent des boucles ramenant la valeur d'une ou de plusieurs sorties vers l'entrée avec un retard. Ils peuvent alors être considérés comme des systèmes dynamiques. On trouve parmi les réseaux non bouclés: le perceptron, le Perceptron Multi-Couches, les réseaux RBF…etc. et parmi les réseaux récurrents : les cartes topographiques de Kohenen et les réseaux de Hopfield.

2.7 Modèles neuronaux de base

2.7.1 Le perceptron

Ayant été introduit par Rosenblatt [16][24] dans les années 50, le Perceptron est historiquement le premier réseau de neurones artificiels. Ce modèle comporte une seule couche de neurones recevant en entrée N valeurs $x_1, x_2, \dots x_N$ et calcule une sortie Y. L'apprentissage du perceptron est supervisé, il consiste à ajuster les valeurs des poids associés aux entrées des neurones qui forment ce perceptron.

Les perceptrons ont la capacité de résoudre seulement les problèmes linéairement séparable. Le célèbre exemple qui montre leur échec face aux problèmes non linéaires est celui du XOR logique dont le perceptron n'arrive jamais à le résoudre. L'apprentissage du perceptron ainsi que ses performances seront discutés dans le début du troisième chapitre de cette thèse.

2.7.2 Le perceptron multicouche

Le perceptron multicouche est le réseau de neurones le plus employé et le plus étudié. Ce modèle fait partie d'une génération de réseaux introduits comme importantes alternatives

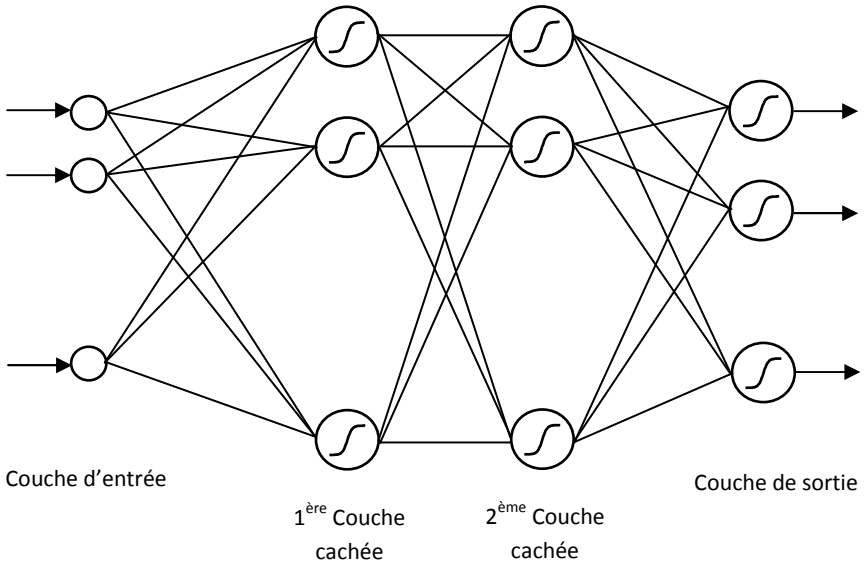


Fig. 2.4. Un perceptron multicouche avec deux couches cachées

des réseaux monocouches qui ont montré leur déficience face aux problèmes non linéaires.

Un perceptron multicouche se compose d'une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées. La figure (2.4) illustre un MLP avec deux couches cachées. Le MLP est un réseau de type feed-forward, l'information se propage donc dans un sens unique d'une couche à couche, des neurones de la couche d'entrée vers ceux de la couche de sortie. La sortie du $j^{ème}$neurone de la $i^{ème}$ couche ($O_{i,j}$) est donnée par :

$$O_{i,j} = f_{i,j}\left(\sum_{n=1}^{N_{i-1}} w_{n,j}\, O_{i-1,n}\right) \tag{2.13}$$

Où : $f_{i,j}$ est la fonction d'activation de ce neurone ; N_{i-1} est le nombre de neurones de la $(i-1)^{ème}$ couche et $w_{n,j}$ est le poids reliant le $n^{ème}$neurone de la couche précédente à ce neurone.

L'importance du MLP réside essentiellement dans ces capacités de former des frontières de décision complexe. Ces dernières sont déterminées par le nombre de couches cachées et le nombre de neurones de chaque couche. Vu l'importance du MLP, qui constitue l'application principale de notre méthode, nous lui réserverons le troisième chapitre de ce travail.

2.7.3 Les réseaux RBF

Les réseaux de fonctions à base radiale (réseaux RBF) sont utilisés pour la première fois par Broomhead & Lowe [25]. Leur modélisation tente d'imiter le fonctionnement de certains neurones biologiques à réponse de syntonisation locale, c'est à dire qui ne répondent qu'à une partie de l'espace du signal d'entrée. Une fonction à base radiale est une fonction qui fournit une sortie différente de zéro seulement pour des entrées qui se situent dans une région locale de l'espace des entrées (champ récepteur). La fonction RBF la plus courante est la fonction gaussienne donnée par :

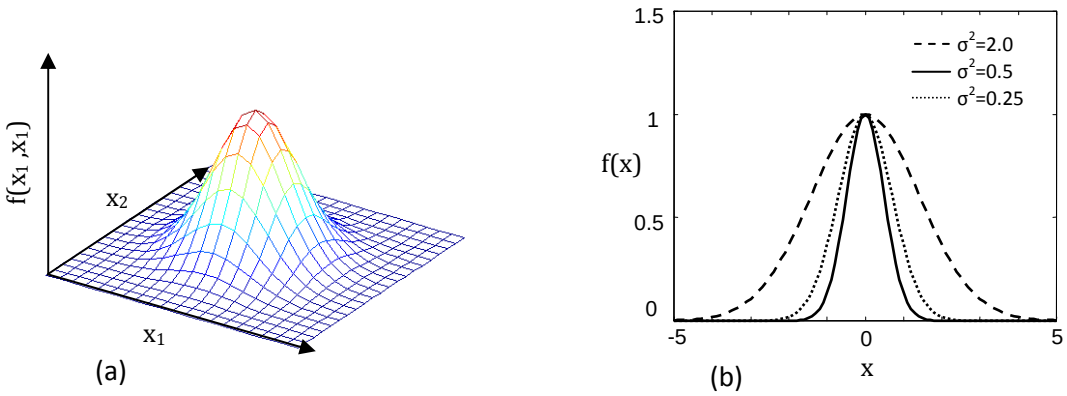


Fig. 2.5. Fonctions RBF
(a) Fonction RBF bidimensionnelle
(b) Fonction RBF unidimensionnelle pour différentes valeurs de σ

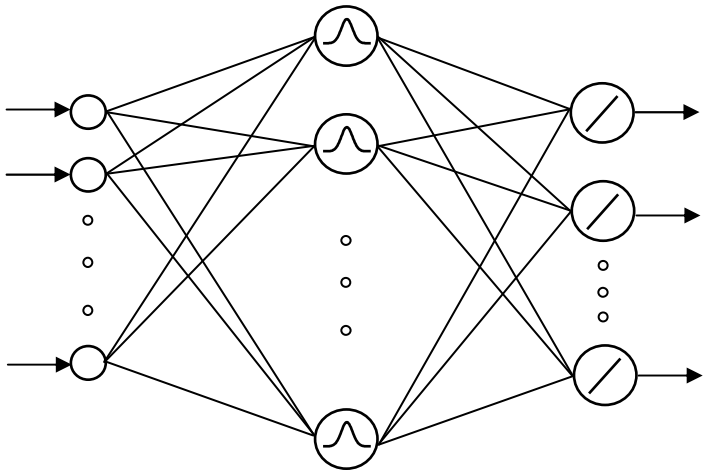


Fig. 2.6. Un réseau RBF

$$f(X) = exp\left(-\frac{\|X - V\|^2}{2\sigma^2}\right) \tag{2.14}$$

Où : V est le centre du champ récepteur (ayant une sortie maximale) et σ est le facteur d'adoucissement qui définit la taille du champ récepteur (figure 2.5)

Un neurone RBF calcule sa sortie en se basant sur la distance entre chaque entrée X et son vecteur prototype V. Sa sortie est d'autant plus grande 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 et Lowe [25], se compose de trois couches (figure 2.6) dont la couche cachée comporte un ensemble de neurones RBF connectés linéairement à la couche de sortie. Ces connexions sont les seules à être ajustées. Les neurones de la couche de sortie ont des fonctions d'activations linéaires.

2.7.4 Les réseaux de Hopfield

Les réseaux de Hopfield [26] sont apparus durant une période où les réseaux de neurones ont perdu considérablement de leurs l'intérêt et cette découverte a contribué à leur renaissance. Ce sont des réseaux récurrents constitués de neurones formels du type McCulloch et Pitts totalement connectés entre eux : ils sont tous à la fois neurone d'entrée et neurone de sortie du réseau. La figure (2.7) illustre un exemple de ces réseaux, chaque neurone est connecté à tous les autres, mais pas à lui-même.

Les sorties des neurones sont binaires et les connexions entre neurones sont symétriques (Pour chaque paire de neurones i et j : $w_{ij} = w_{ji}$). L'état du réseau est caractérisé par les états de tous les neurones. Le calcul du nouvel état du neurone i est donnée par :

$$O_i = \begin{cases} +1 \; si \; \sum\left(w_{ij}x_i + I_i - \tau_i\right) > 0 \\ -1 \; si \; \sum\left(w_{ij}x_i + I_i - \tau_i\right) \leq 0 \end{cases} \tag{2.15}$$

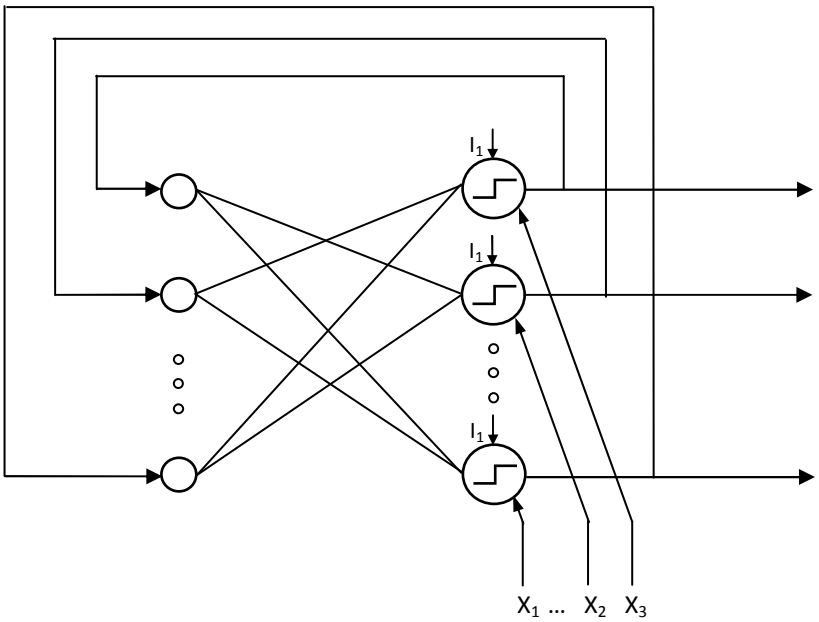


Fig. 2.7. Un réseau de Hopfield

Où : O_i est l'état du neurone i ; x_i est l'entrée de ce neurone ; τ_i est son seuil et I_i est une variable aléatoire.

A chaque présentation d'un exemple au réseau, tous les neurones calculent la somme pondérée de leurs entrées selon l'Eq. (2.15) et ce jusqu'à l'obtention d'un état stable du réseau. L'apprentissage de ces modèles est exceptionnel par rapport aux autres réseaux de neurones, il ne consiste pas à chercher une convergence basée sur des essais-erreurs. La connaissance de l'ensemble des états permet un apprentissage direct des poids. Il s'agit un apprentissage Hebbien donné par :

$$w_{ij} = \frac{1}{N}\sum_{q=1}^{Q} x_i^{(q)} x_j^{(q)} \tag{2.16}$$

Où : N est la dimension du vecteur d'entrée ; Q est le nombre d'exemple d'apprentissage et $x_i^{(q)}$ est la $q^{ème}$ entrée du neurone i.

2.7.5 Les Cartes auto-organisatrices

Les cartes auto-organisatrices sont inspirées des modèles d'auto-organisation constatés dans certaines aires biologiques. Ces cartes cherchent à représenter des données complexes et à extraire des régularités en utilisant un apprentissage supervisé. Le modèle de Kohonen [27][28] est le plus usuel des modèles auto-organisés. Ces réseaux ont pour objectif la représentation des informations spatiales des espaces caractéristiques en se basant sur un ensemble de neurones disposés suivant une topologie précise. Par leurs capacités de classification et d'extraction de caractéristiques avec une importante composante

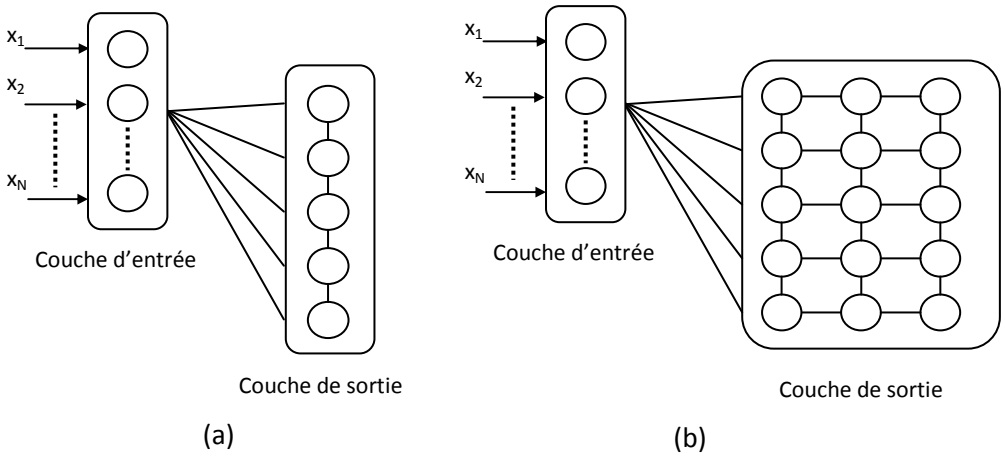


Fig. 2.8. Exemples de cartes auto-organisatrices
(a) Monodimensionnelle
(b) Bidimensionnelle

topologique, les cartes de Kohonen peuvent être utilisées pour la compression, l'analyse de données ou comme mémoire associative.

Une carte auto-organisatrice se compose de deux couches ; une couche d'entrée permettant de refléter les vecteurs d'entrée et une couche compétitive de sortie pouvant être en 1D, 2D ou 3D. A chaque neurone de la couche de sortie correspondent des poids (sous forme d'un vecteur de dimension N) reliant celui-ci au vecteur d'entrée (figure 2.8).

L'apprentissage des cartes de Kohonen est un apprentissage non supervisé. A chaque présentation d'un exemple $X^{(q)}$, tout neurone i de la carte détermine sa distance à celui-ci. Ces distances sont calculées entre le vecteur d'entrée et le vecteur des poids du neurone selon la métrique utilisée. Dans le cas de la distance euclidienne :

$$D_{qi} = \sum_{n=1}^{N} \left(x_n^{(q)} - v_n^{(i)} \right)^2 \tag{2.17}$$

Où : D_{qi} est la distance entre le $q^{ème}$exemple et le $i^{ème}$neurone ; $v_n^{(i)}$ est le $n^{ème}$poids du $i^{ème}$neurone et N est la dimension de l'espace caractéristique

Le neurone vainqueur (celui qui est le plus près du vecteur d'entrée) modifie ces poids afin de se rapprocher du vecteur d'entrée. Les neurones les plus proches voisins modifient leurs poids pour se rapprocher également du vecteur d'entrée, mais plus légèrement. En revanche, les poids des neurones les plus éloignés seront modifiés afin de s'éloigner encore de ce vecteur. Les neurones restants ne modifient pas leurs poids. La mise à jour des poids à l'itération (r) s'effectue selon la règle compétitive suivante :

$$v_n^{(i)}(r) = v_n^{(i)}(r-1) + \eta\left(X^{(q)} - v_n^{(i)} \right) \tag{2.18}$$

2.8 Conclusion

Les réseaux de neurones ont connu un essor considérable tant qu'en nouvelles architectures qu'en nouveaux algorithmes d'apprentissage et dans ce chapitre nous avons tenté de donner un simple survol sur ces importants outils mathématiques. Cette partie nous a permis de tirer quelques conclusions :

- Les réseaux de neurones semblent bien adaptés aux problèmes actuels qui ne cessent d'être de plus en plus complexes et qui nécessitent des outils mathématiques ayant de grandes capacités de calcul et d'apprentissage.
- Les réseaux de neurones artificiels fonctionnent en se basant sur un calcul parallèle effectué par un ensemble d'unité qui opère localement.
- D'un point de vue topologique, les réseaux de neurones offrent une grande gamme d'architectures permettant leur utilisation dans divers problèmes. Ainsi pour un problème donné, l'architecture employée doit être choisie en tenant compte de ces propriétés et en fonction des taches souhaitées.
- Dans le même contexte, il est à noter que les réseaux de neurones sont difficiles à paramétrer. Par exemple dans le cas du perceptron multicouche, il est difficile de définir le nombre de neurones dans les couches cachées.
- Les réseaux de neurones sont des boites noires qui ne permettent d'interpréter le modèle obtenu.

Partie II

Chapitre 3 : Classification en utilisant le perceptron multicouche

Chapitre 4 : Classification neuro-floue

Chapitre 3

Classification en utilisant le Perceptron multicouche

Nous consacrons ce chapitre à la classification par le perceptron multicouche qui est le réseau de neurones le plus utilisé, et qui constitue également l'application principale de la méthode proposée. Nous commençons par la classification en utilisant les réseaux monocouches, nous indiquons les limites de ces systèmes et nous verrons comment les réseaux multicouches surmontent ces limites. Nous abordons ensuite la classification par le perceptron multicouche en décrivant son architecture et son apprentissage. Nous donnons à la fin de ce chapitre un bref état de l'art concernant les techniques d'accélération et de stabilisation de la Rétro-propagation qui est l'algorithme de base d'apprentissage du MLP, et dont l'amélioration de ses performances constitue l'objectif principal de notre travail.

3.1 Introduction

Le Perceptron multicouche (MLP, Multi Layered Perceptron) fait partie d'une génération de réseaux introduits comme importants alternatifs de réseaux monocouches qui ont montré leur déficience face aux problèmes non linéaires. Le MLP est le réseau de neurones le plus employé, il a été introduit avec succès dans plusieurs applications de la reconnaissance des formes telles que la reconnaissance d'écriture, la lecture automatique, le traitement et la segmentation d'image…etc. Vu son importance, ce modèle a été largement étudié notamment en ce qui concerne son architecture. Cette dernière joue un rôle important dans les problèmes de classification par les réseaux de neurones. En effet, l'un des avantages de l'utilisation des réseaux de neurones réside dans le fait que les connaissances des problèmes réels traités, étant souvent de nature informelle ou heuristique, peuvent être facilement incorporée aux réseaux en choisissant le nombre de couches cachées, de neurones, des connexions des boucles de retour et ainsi de suite [1].

De même, l'apprentissage du MLP a été largement évoqué. Ces études se sont intéressées principalement à étudier la Rétro-propagation du gradient en tant qu'algorithme de base d'apprentissage du MLP. Parmi ces études, des travaux ont évoqué l'importance de ce processus en précisant ses capacités d'approximation et en lui donnant des interprétations

statistiques. En revanche, d'autres travaux se sont intéressés aux problèmes de ce processus, et plusieurs approches ont été proposées afin d'améliorer ces performances.

3.2 Classification par les réseaux monocouches

3.2.1 Classification en utilisant le perceptron

Le perceptron, ayant été introduit par Resemblat [16][24] en 1957, est historiquement le premier réseau de neurones. Comme l'indique son nom, le perceptron a été mis en œuvre essentiellement pour réaliser des tâches de classification. Ce réseau comporte une seule couche de neurones permettant de recevoir un vecteur d'entrées et de calculer une sortie (figure 3.1). Un perceptron est défini par un ensemble de poids synaptique et un biais. La fonction d'activation est une fonction à seuil qui prend ses valeurs dans {-1,1}.

Pour un perceptron avec N entrées $(x_1, x_2, \dots, x_N)$, sa sortie Y est donnée par :

$$Y = h(S) = h(w_1 x_1 + w_2 x_2 + \cdots + w_N x_N + w_0) \tag{3.1}$$

Où : $h(.)$ est la fonction d'activation, $w_1, w_2, \dots, w_N$ sont les poids synaptiques et w_0 est le biais.

La sortie du perceptron est $Y = 1$ pour les vecteurs d'entrée ayant $S > 0$ et $Y = -1$ pour les autres vecteurs. Le perceptron divise l'espace caractéristique en deux sous-espaces délimités par un hyperplan H. Ce dernier est formé par les vecteurs d'entrée dont $w_1 x_1 + w_2 x_2 + \cdots + w_N x_N + w_0 = 0$. L'espace caractéristique est partitionné donc en demi-espaces : un demi-espace gauche H^+ et demi-espace droit H^+ comme suit :

$$\begin{cases} Si\ w_1 x_1 + w_2 x_2 + \cdots + w_N x_N > w_0\ alors\ X \in\ H^+ \\ Si\ w_1 x_1 + w_2 x_2 + \cdots + w_N x_N < w_0\ alors\ X \in\ H^- \end{cases} \tag{3.2}$$

La figure (3.2) montre un exemple de partitionnement d'un espace caractéristique bidimensionnel. Il est divisé en deux sous-espaces H^- et H^+ par une droite constituant l'hyperplan dans le cas bidimensionnel. La règle d'apprentissage du perceptron consiste à

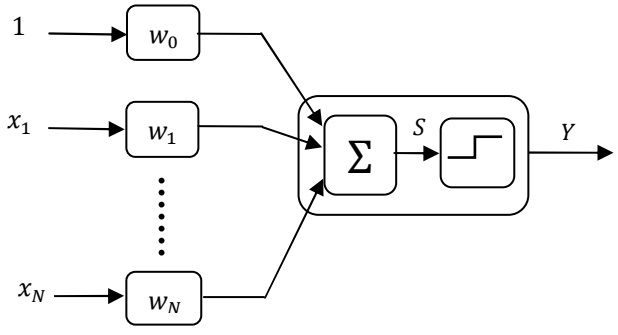


Fig. 3.1. Représentation d'un perceptron

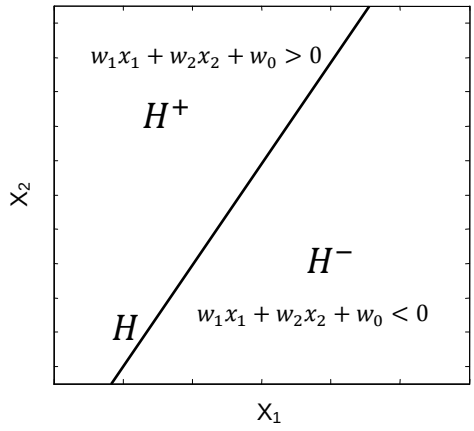


Fig. 3.2. Partitionnement de l'espace caractéristique par un perceptron

ajuster chaque poids w_n à l'itération $(r+1)$ comme suit :

$$w_n^{(r+1)} = \begin{cases} w_n^{(r)} & si\ y_n = t_n \\ w_n^{(r)} + \eta t_n x_n & si\ y_n \neq t_n \end{cases} \tag{3.3}$$

Où : x_n est la $n^{ème}$ entrée du réseau ;t_n est la sortie désirée ; y_n est la sortie calculée et η est le pas d'apprentissage (une constante positive choisie). Le principe d'apprentissage du perceptron est le suivant : S'il fournit la sortie désirée, il n'y aura pas de changement de poids. Sinon, chaque poids sera modifié avec $\Delta w = \eta t_n x_n$. Ce processus sera répété jusqu'à ce que le perceptron arrive à classifier correctement tous les exemples.

3.2.2 Classification en utilisant l'ADALINE

Ayant été introduit par Widrow [23], L'ADALINE (i.e. ADAptative LINear Element) est un modèle de réseaux de neurones fondé sur le travail de Mcculloch et Pitts. L'architecture de L'ADALINE est semblable à celle du perceptron, mais avec une fonction d'activation linéaire. Pour un problème avec N entrées, la sortie de L'ADALINE est donnée par :

$$Y = h(S) = \sum_{n=1}^{N} w_n x_n \tag{3.4}$$

L'apprentissage de l'ADALINE est effectué par la règle de Widrow [23] qui consiste à minimiser l'erreur quadratique (E) donnée par :

$$E = \sum_{q=1}^{Q} \left(T^{(q)} - Y^{(q)}\right)^2 \tag{3.5}$$

Où : Q est le nombre totale d'exemples d'apprentissage ; $T^{(q)}$ et $Y^{(q)}$ sont respectivement la sortie désirée et la sortie calculée correspondantes au $q^{ème}$ exemple $(X^{(q)})$

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

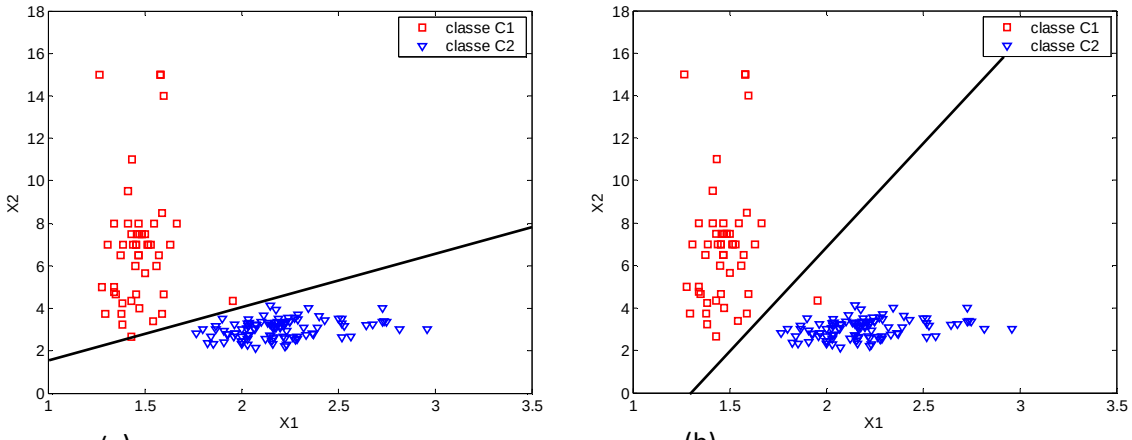


Fig.3.3. Frontière de décision de la base de données Iris
(a) Classification par l'ADALINE
(b) Classification par le perceptron

$$w_n^{(r+1)} = w_n^{(r)} - \eta \frac{\partial E}{\partial w_n}$$

$$= w_n^{(r)} - \eta(t_n - y_n)x_n \qquad (3.6)$$

L'apprentissage de l'ADALINE consiste à ajuster ses poids jusqu'à l'obtention d'une erreur assez faible entre la sortie désirée et la sortie calculée, c'est-à-dire contrairement au perceptron où l'apprentissage s'achève quand tous les exemples seront correctement classés.

3.2.3 Le perceptron contre l'ADALINE

Pour comparer les performances de classification du perceptron et de l'ADALINE, considérons par exemple le problème de classification de la première et la deuxième classe de la base de données Iris. Pour apercevoir les frontières de décisions nous admettant une représentation bidimensionnelle, chaque exemple étant représenté par deux caractéristiques ; largeur sépale / longueur sépale et largeur pétale / longueur pétale. Nous effectuons la classification avec un perceptron et une ADALINE ayant deux entrées.

La figure (3.3) illustre les exemples d'apprentissage ainsi que les frontières de décisions obtenues. D'après cette figure 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 au fait que l'ADALINE effectue la classification en minimisant l'erreur quadratique entre les sorties désirées et les sorties calculées contrairement au perceptron qui consiste à classifier tous les exemples correctement.

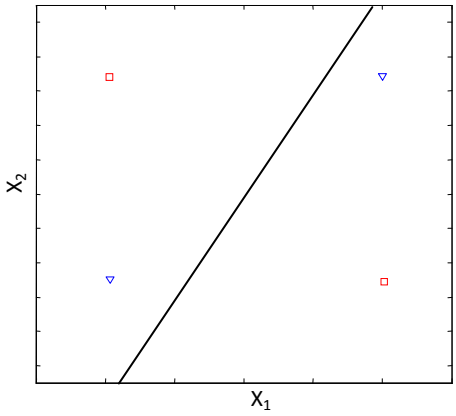


Fig. 3.4. Le problème du XOR logique

3.2.4 Limitation des réseaux à couche

La fameuse publication de Minsiky-Paper en 1960 [17] montrant l'échec du perceptron à résoudre le XOR logique a été derrière la diminution d'intérêt de ce réseau pendant deux décennies. Ils ont énoncé la limitation des réseaux de neurones de l'époque face aux problèmes non linéairement séparables. La figure (3.4) illustre ce problème ; il est apercevable que c'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.

3.3 Classification par le MLP

3.3.1 Performance des réseaux multicouches

Contrairement aux réseaux monocouches, qui sont limités par leur incapacité à produire des frontières de décision complexes, les réseaux multicouches ont de grandes capacités de

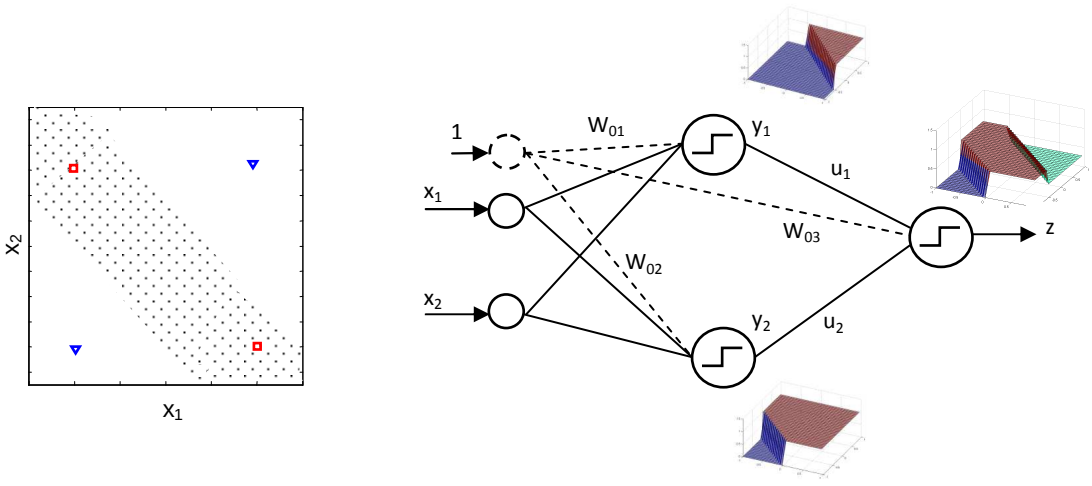


Fig. 3.5. Classification du XOR en utilisant un réseau multi couche

calcul et peuvent former des régions de décision arbitrairement complexes [4].

Le problème du XOR logique est le problème de classification non linéaire le plus simple dont les réseaux monocouches n'arrivent pas à résoudre. Pour indiquer la possibilité de résoudre ce problème avec un réseau multicouche, considérons le réseau formé de trois couches de la figure (3.5). C'est le même exemple qui a été présenté dans [1]. La couche d'entrée de ce réseau comporte deux neurones dont le rôle est de refléter les composantes du vecteur présenté, la couche cachée contient deux neurones avec des fonctions d'activation à seuil tandis que la couche de sortie comprend un seul neurone ayant également une fonction d'activation à seuil. Les sorties des deux neurones cachées sont: $y_1 = f(x_1 + x_2 + w_{01})$ et $y_2 = f(x_1 + x_2 + w_{02})$. Ces neurones opèrent comme des perceptrons. La frontière de décision formée par le premier neurone est donnée par : $x_1 + x_2 + w_{01} = 0$, sa sortie est alors $y_1 = 1$ pour les exemples dont $x_1 + x_2 + w_{01} \geq 0$ et elle est $y_1 = 0$ pour les autres exemples. De la même façon, le deuxième neurone forme la frontière de décision donnée par $x_1 + x_2 + w_{02} = 0$. Finalement, la sortie du réseau est donnée par: $z = f(u_1 y_1 + u_2 y_2 + w_{03})$. Avec un choix convenable des valeurs des poids et des bais, ce réseau donne la solution appropriée du problème XOR considéré (les valeurs utilisées dans [1] sont : $u_1 = 0.7, u_2 = -0.4, w_{01} = 0.5, w_{01} = -1.5, w_{01} = -1$).

3.3.2 Architecture du MLP

Le MLP est une réalisation en série de perceptrons. Ce modèle comprend en plus de la couche d'entrée et la couche de sortie, une ou plusieurs couches cachées. Dans ce réseau, 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.

La figure (3.6) illustre un MLP, avec une seule couche cachée, comportant N neurones d'entrée, M neurones cachés et J neurones de sortie. Le $n^{ème}$ neurone d'entrée est relié

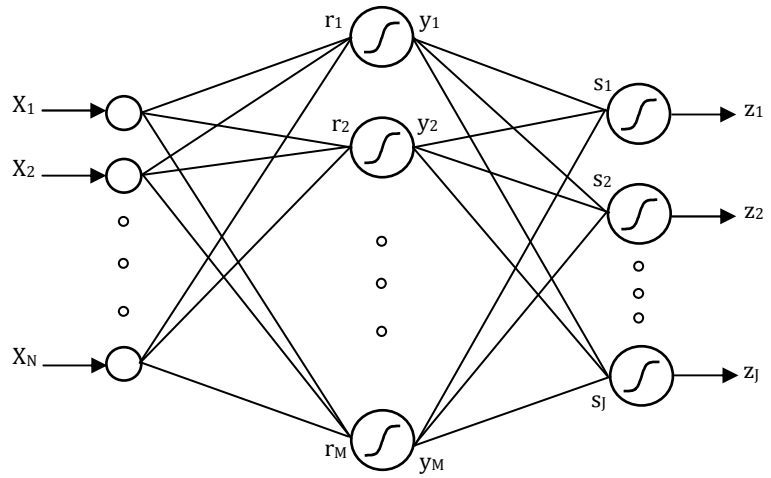


Fig. 3.6. MLP avec une seule couche cachée contenant N neurones d'entrée, M neurones cachés et J neurones de sortie.

avec le $m^{ème}$ neurone caché par le poids w_{nm} et le $m^{ème}$ neurone caché est relié avec le $j^{ème}$ neurone de sortie par le poids u_{mj}. A chaque présentation d'un exemple X $(x_1, x_2, \dots, x_N)$, les composants de son vecteur caractéristique seront transmis aux neurones de la couche cachée. Les sorties de ces neurones $(y_1, y_2, \dots, y_M)$ seront à leur tour transmis aux neurones de la couche suivante (la couche de sortie). La sortie du $m^{ème}$ neurone caché est donnée par :

$$y_m = h(r_m) = h\left(\sum_{m=1}^{M} x_n w_{nm}\right) \tag{3.7}$$

Où : $h(.)$ est la fonction d'activation des neurones de la couche cachée.

Les neurones de la couche de sortie constituent la sortie du réseau. La sortie du $j^{ème}$ neurone est donnée par :

$$z_j = g(s_j) = g\left(\sum_{j=1}^{J} y_m u_{mj}\right) \tag{3.8}$$

Où : $g(.)$ est la fonction d'activation des neurones de sortie.

Dans les problèmes de classification, le nombre de neurones de la couche d'entrée du MLP est égal à la dimension de l'espace caractéristique parce que le rôle de cette couche est de présenter chaque composante du vecteur caractéristique aux neurones de la couche suivante. Le nombre de neurones de la couche de sortie égale le nombre de classes, soit un neurone par classe (la sortie de ce neurone est 1 pour les exemples appartenant à la classe correspondante et les autres sont à 0).

En tant que classificateur, le MLP doit donc fournir une sortie Z $(z_1, z_2, \dots, z_J)$ correspondante à la classe de l'exemple X $(x_1, x_2, \dots, x_N)$ présenté en entrée.

3.3.3 Nombre de couches cachées

Les capacités d'approximations du MLP ont été évoquées dans plusieurs travaux [2][29][30][31]. Dans ce contexte, il a été démontré qu'un réseau de neurones avec une seule couche cachée peut approximer n'importe quelle fonction. Par exemple, Cybenko [29] a démontré 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 quelle transformation continue représentée par cet exemple.

En absence d'un problème spécifique qui nécessite plusieurs couches cachées, il est alors plus convenable de procéder en utilisant une seule couche cachée, surtout en tenant compte du fait qu'il a été constaté empiriquement que les réseaux avec de multiples couches cachées sont plus disposés à tomber dans les minima locaux indésirables [1]. Par

ailleurs, il a été démontré par plusieurs auteurs que les sorties d'un MLP avec une seule couche cachée peuvent être interprétées comme les probabilités a posteriori [2][32][33][34]. Ces travaux seront présentés dans le paragraphe (§3.7).

3.4 Apprentissage du MLP par la retro-propagation

La rétro-propagation du gradient (Back Propagation) [18] est l'une des méthodes les plus simples et les plus utilisées pour l'apprentissage des réseaux de neurones. C'est une extension de la règle d'apprentissage de Widrow [23] appliquée aux réseaux monocouches. La RP (rétro-propagation) consiste donc à minimiser la distance entre la sortie calculée $Z^{(q)}$ et la sortie désirée $T^{(q)}$ correspondantes à chaque exemple d'apprentissage $X^{(q)}$. L'erreur quadratique est souvent employée comme étant la fonction coût de la RP. Pour un ensemble de Q exemples d'apprentissage, l'erreur quadratique totale est donnée par :

$$E = \sum_{q=1}^{Q} \sum_{j=1}^{J} \left(t_j^{(q)} - z_j^{(q)} \right)^2 \tag{3.9}$$

L'algorithme de la RP est basé sur la modification des poids du réseau de façon à effectuer une descente de gradient sur la surface d'erreur. Au début de l'apprentissage, les poids sont initialisés avec des valeurs aléatoires et modifiés ensuite dans une direction qui réduira l'erreur. La modification Δw d'un poids w est donnés par :

$$\Delta w = -\eta \frac{\partial E}{\partial w} \tag{3.10}$$

Où : η est une constante positive appelée pas d'apprentissage permettant de définir la taille des modifications des poids.

Il s'agit donc de prendre un pas dans l'espace des poids permettant de réduire la fonction coût. Chaque poids est modifié à l'itération $(r+1)$ en fonction de sa valeur à l'itération (r) par :

$$w^{(r+1)} = w^{(r)} + \Delta w^{(r)} \tag{3.11}$$

Pour un MLP avec une couche cachée (fig. 3.6), la mise à jour des poids de la couche cachée et ceux de la couche de sortie est donnée par :

$$u_{mj}^{(r+1)} = u_{mj}^{(r)} - \eta \frac{\partial E^{(r)}}{\partial u_{mj}} \tag{3.12}$$

$$w_{nm}^{(r+1)} = w_{nm}^{(r)} - \eta \frac{\partial E^{(r)}}{\partial w_{nm}} \tag{3.13}$$

Le développement de l'Eq. (3.12) et l'Eq. (3.13) donne les équations d'adaptation suivantes :

$$u_{mj}^{(r+1)} = u_{mj}^{(r)} + \eta_1 (t_j - z_j) \dot{g}(s_j) y_m \tag{3.14}$$

$$w_{nm}^{(r+1)} = w_{nm}^{(r)} + \eta_2 \left(\sum_{j=1}^{J} (t_j - z_j) \dot{g}(s_j) u_{mj} \right) \acute{h}(r_m) x_n \tag{3.15}$$

Où : $\dot{g}(s_j)$ et $\acute{h}(r_m)$ sont respectivement les dérivées des fonctions d'activation des neurones cachés et des neurones de sortie. Dans le cas des fonctions sigmoïde, les dérivées sont données par :

$$\dot{g}(s_j) = z_j(1 - z_j) \tag{3.16}$$

$$\grave{h}(r_m) = y_m(1 - y_m) \tag{3.17}$$

L'algorithme de la RP est donné par :

1. **begin**
2. Initialisation : $M, \varepsilon, \eta, R, r \leftarrow 0$.
3. **do** $r \leftarrow r + 1$
4. $q \leftarrow 0$;
5. **do** $q \leftarrow q + 1$
6. Select $X^{(q)}$
7. $\Delta w_{nm} \leftarrow -\eta \partial E / \partial w_{nm}$
8. $\Delta u_{mj} \leftarrow -\eta \partial E / \partial u_{mj}$
9. $u_{mj} \leftarrow u_{mj} + \Delta u_{mj}$
10. $w_{nm} \leftarrow w_{nm} + \Delta w_{nm}$
11. **Until** $q = Q$
12. **Until** $r = R$ **or** $E \leq \varepsilon$
13. **end**

3.5 Paramètres et performances du MLP

3.5.1 Les fonctions d'activation

Bien que la RP fonctionne pratiquement avec n'importe quelle fonction d'activation, étant données quelques conditions simples telles que la continuité de cette fonction ainsi que sa dérivée, il y a un certain nombre de propriétés souhaitables [1]: D'abord, la fonction d'activation doit être non linéaire. Deuxièmement, la fonction d'activation doit être saturable ; sa sortie doit être limitée par une valeur maximale et une valeur minimale. Troisièmement, la fonction d'activation doit être continue et lisse, c.-à-d. la fonction et sa dérivée doivent être définies sur tout l'intervalle d'entrée. En outre, il est préférable, mais pas nécessaire, que la fonction d'activation soit monotone afin que les dérivées aient le même signe dans tout l'intervalle d'entrée. En effet, si ce n'est pas le cas, des minimas locaux additionnels peuvent se présenter dans la surface d'erreur [1].

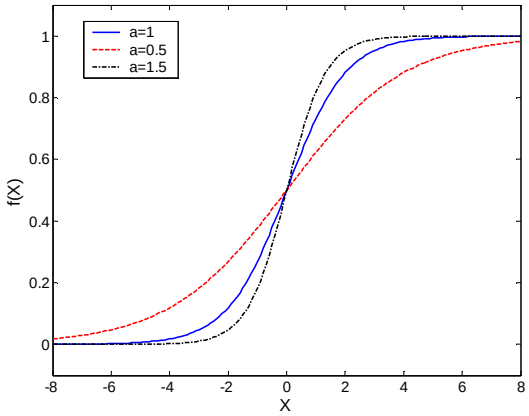


Fig. 3.7. La fonction sigmoïde avec différentes valeurs de α

La fonction sigmoïde possède toutes les propriétés précédentes (fig.3.7) ; elle est lisse, différentiable, non-linéaire, et saturable. De plus, la dérivée de la sigmoïde peut être facilement exprimée en fonction de la sigmoïde elle-même.

Par ailleurs, une couche cachée de neurones sigmoïdales permet une représentation distribuée (ou globale) des entrées. Chacune de celles-ci est susceptible donc d'activer tous les neurones cachés. Selon Duda et Al. [1], les représentations distribuées sont préférables quand il y a peu d'exemples d'apprentissage parce que les données influencent les sorties à n'importe quelle région d'entrée.

3.5.2 Nombre de neurones cachés

Dans les problèmes de classification, le nombre de neurones d'entrée et celui des neurones de sortie d'un MLP sont simplement déterminés par la dimension des caractéristiques et le nombre de classes. Malheureusement, ce n'est pas le cas pour le nombre de neurones cachés dont son choix s'effectue généralement d'une façon empirique. Le nombre de neurones cachés influe considérablement sur les performances du MLP et détermine la forme des frontières de décision générées. En effet, si les exemples à classifier sont linéairement séparables, quelques neurones cachés suffirent, et inversement, si les exemples d'apprentissage sont très chevauchés, il sera nécessaire d'utiliser un grand nombre de neurones.

La figure (3.8) illustre quelques exemples de frontières de décisions générées avec un MLP pour un problème de classification bidimensionnel à deux classes. Ces frontières sont obtenues avec différents nombres de neurones cachés. Nous pouvons constater que les frontières de décision générées dépendent énormément du nombre de neurones cachés. Pour ce problème, le MLP permet d'avoir des résultats satisfaisants à partir de 4 neurones cachés.

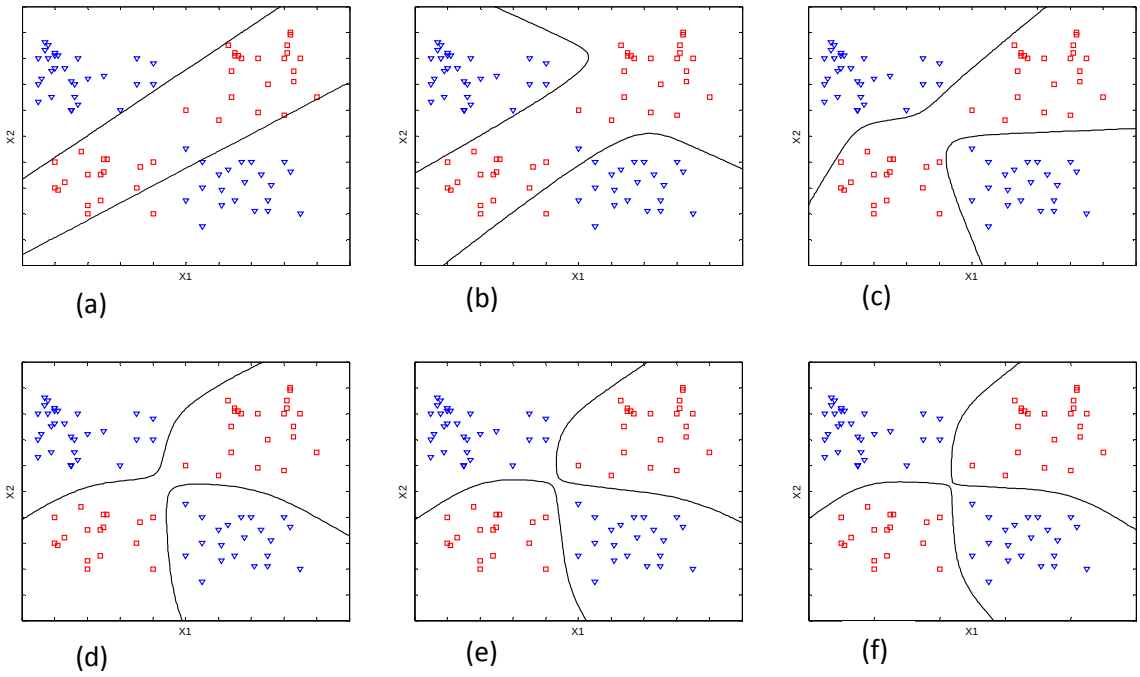


Fig. 3.8. Frontières de décision générées pour un MLP ayant de différents nombres de neurones cachés :
(a) 2 neurones (b) 3 neurones (c) 3 neurones
(d) 4 neurones (e) 8 neurones (f) 8 neurones

3.5.3 Les poids initiaux

La phase d'initialisation des poids s'effectue après avoir fixé la topologie du réseau. Dans cette phase, nous avons intérêt à choisir des valeurs de sorte que l'apprentissage soit rapide et uniforme, c.-à-d. les valeurs finales d'équilibre des poids seront atteintes simultanément [1]. Pour ce faire, les poids doivent être initialisés aléatoirement selon une distribution uniforme autour d'une certaine valeur moyenne ($W_{min} < W < W_{max}$) [4]. Ces deux paramètres (W_{min} et W_{max}) sont choisis avec des valeurs petites afin que l'activation des neurones cachés soit petite. En revanche, s'ils sont grands, les neurones cachés risquent de se saturer même avant le début d'apprentissage [1].

3.5.4 Le pas d'apprentissage

Dans le processus d'apprentissage du MLP par la RP, le pas d'apprentissage définit la taille de la descente sur la surface de l'erreur et, par conséquent, la vitesse d'apprentissage. Si ce paramètre est choisi avec une petite valeur, l'adaptation des poids sera avec des petits pas et l'apprentissage sera lent. En revanche s'il est choisi grand, le réseau risquera de dépasser le minimum global et risquera aussi d'avoir des oscillations.

Théoriquement, le pas d'apprentissage optimal est celui qui mène au minimum d'erreur dans une seule époque d'apprentissage (Fig. 3. 9). La détermination d'un gain optimale est l'un des problèmes de la RP.

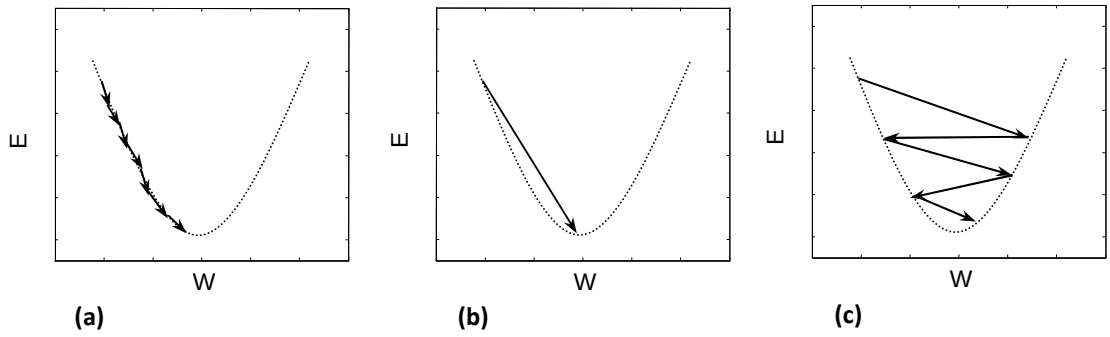


Fig. 3.9. Descente de gradient dans un problème unidimensionnel avec différents valeur du pas d'apprentissage :
(a) $\eta < \eta_{opt}$
(b) $\eta = \eta_{opt}$
(c) $\eta > \eta_{opt}$

3.5.5 La surface d'erreur

Du fait que la RP est basée sur une descente de gradient, nous pourrons avoir une idée sur cet algorithme en étudiant les surfaces d'erreur de la fonction coût employée en fonction des poids. Bien que ces surfaces dépendent du problème traité, il y a quelques propriétés générales qui caractérisent la plus part des problèmes réels de reconnaissance des formes [1]. Le problème le plus important concerne les minima locaux ; si la surface d'erreur admet plusieurs minima locaux, il sera peu probable que le réseau atteint le minimum global, et par conséquent cela pourra conduire à une mauvaise performance. Un autre problème concerne la présence des régions-plateaux où l'erreur varie légèrement en fonction des poids ; si de tels plateaux sont nombreux, l'apprentissage se fera très lentement.

Pour visualiser un exemple de ces surfaces, considérons le problème unidimensionnel de la figure (3.10). C'est un simple problème de classification à deux classes linéairement séparables. Pour classifier les exemples de ce problème, nous utilisons un simple MLP avec deux neurones à l'entrée, deux neurones cachés et un neurone à la sortie. La surface d'erreur illustrée sur la figure indique la présence d'un seul minimum global. La frontière optimale de décision, un point situant en $X = 5$, sépare parfaitement les deux classes. Au cours de l'apprentissage, le MLP atteindra facilement le minimum global et le problème sera rapidement résolu.

Appliquons maintenant le même réseau pour la résolution d'un autre problème unidimensionnel, mais qui n'est pas linéairement séparable (figure 3.11). Nous pouvons noter en premier lieu que, globalement, la surface d'erreur est légèrement plus élevée que dans le cas précédent. Il est également clair que la surface d'erreur dans cet exemple est plus complexe et comporte beaucoup de minimas locaux et de région-plateaux ce qui complique la tâche d'apprentissage.

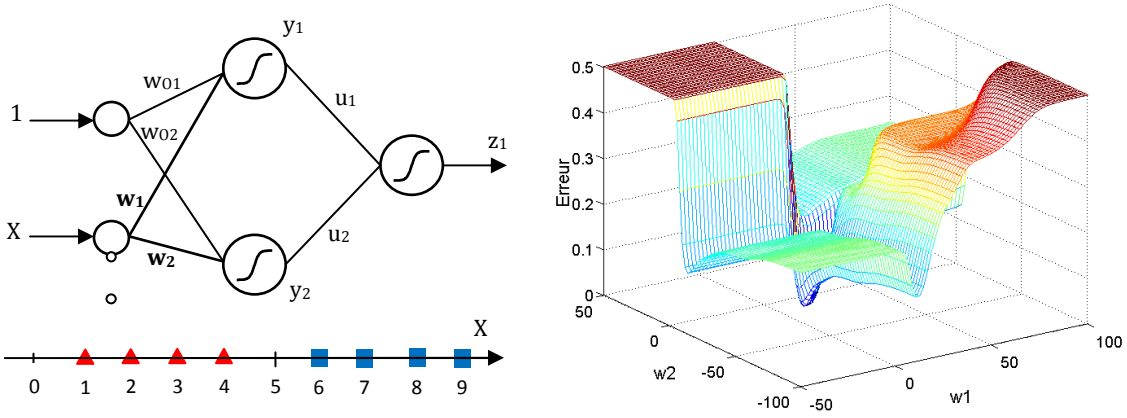


Fig.3.10. Surface d'erreur en fonction des poids dans un problème de classification unidimensionnel linéairement séparable

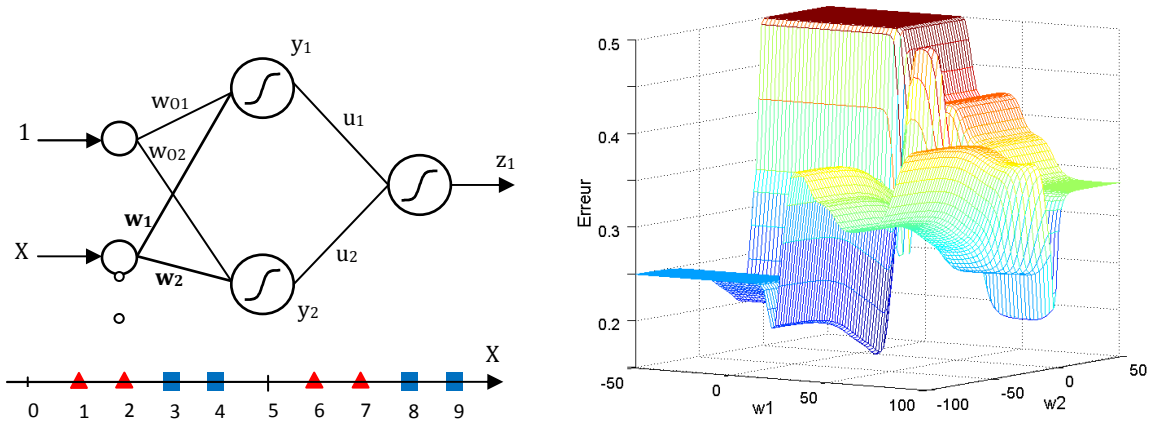


Fig.3.11. Surface d'erreur en fonction des poids dans un problème de classification unidimensionnel non linéairement séparable

Les deux exemples précédents donnent une idée concernant la surface d'erreur et comment la tâche du MLP devient plus difficile dans les problèmes non linéairement séparables. Cela va surement être de plus en plus difficile dans les problèmes multi-classes et à grande dimension.

En effet et selon Duda et Al. [1], l'intuition que nous gagnons en considérant des surfaces d'erreur pour les petits réseaux ne donne que des notes sur qui se passe dans les grands réseaux. Dans ces problèmes l'erreur varie peu à peu avec chaque changement d'un poids unique, ce qui rend la présence des minimas locaux abondante.

3.6 Exemple de classification

Pour apercevoir les performances d'un MLP entrainé par la RP, considérons le problème de classification de la figure (3.12). Il s'agit d'un problème bidimensionnel à deux classes (C_1et C_2) non linéairement séparables.

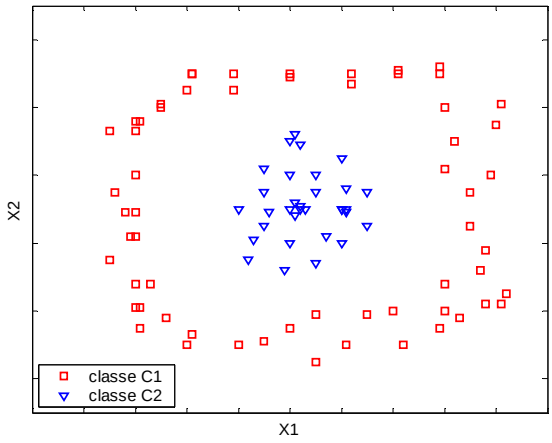


Fig. 3.12. Un problème de classification à 2 classes.

Nous utilisons un MLP avec trois neurones d'entrée, quatre neurones cachés et deux neurones de sorties. Pour l'apprentissage de cet MLP nous employons l'algorithme de la RP ; les pas d'apprentissage utilisés sont : $\eta_1 = 0.5$ et $\eta_2 = 0.5$.

Les figures (3.13.a) et (3.13.b) illustrent respectivement l'évolution de l'erreur et celle du taux de classification au cours de l'apprentissage. Le MLP parvient à classifier les exemples de ce problème après environ 70 itérations.

La figure (3.14.a) représente le MLP utilisé et les formes de sorties des neurones cachés et des neurones de sortie.

Finalement, la figure (3.14.b) met en évidence la forme de la frontière de décision générée par ce MLP. Nous constatons qu'il réalise une frontière de décision permettant de bien classifier tous les exemples d'apprentissage.

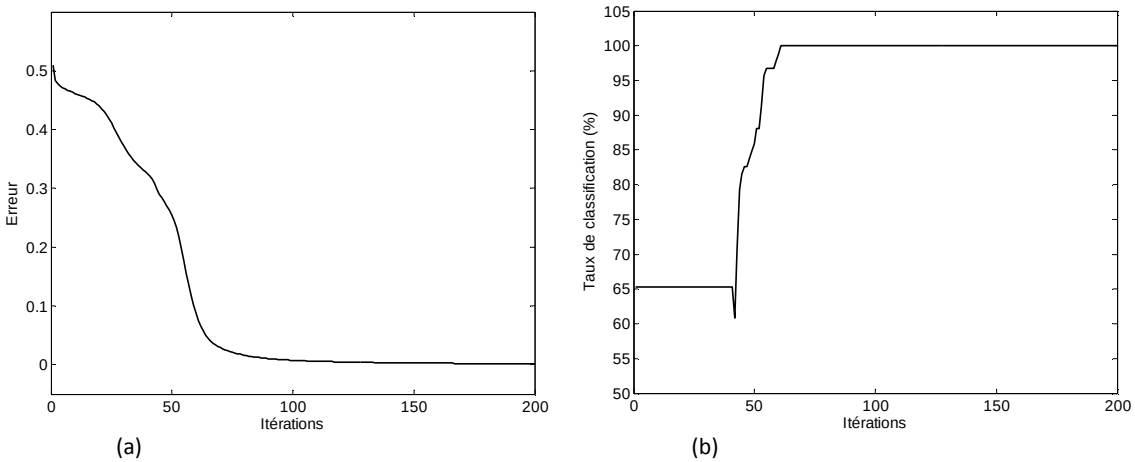


Fig. 3.13. Résultats de la classification du problème de la figure (3.12) avec un MLP :
(a) Evolution de l'erreur
(b) Evolution du taux de classification

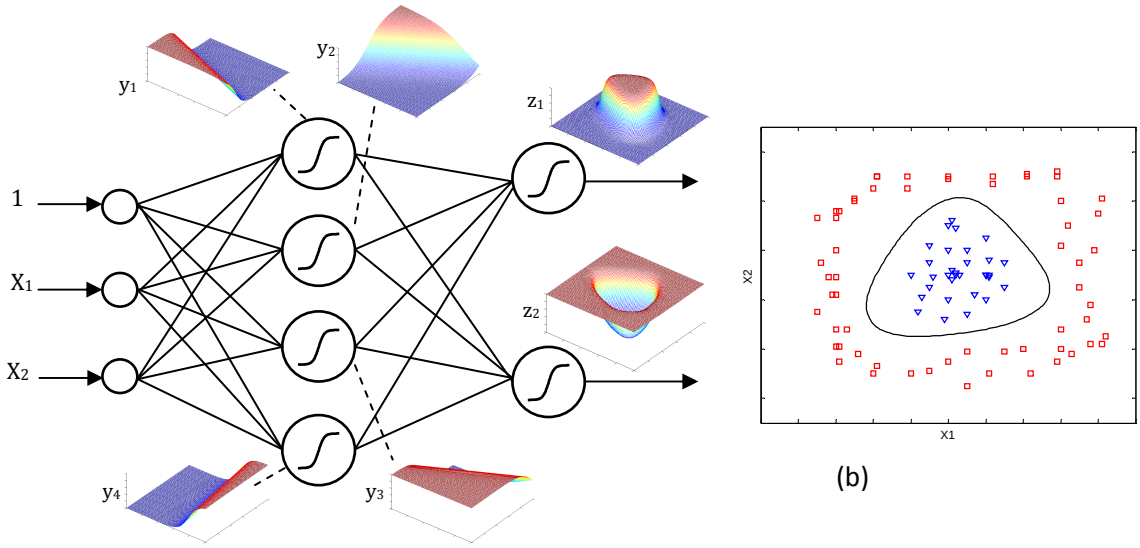


Fig. 3.14. Classification du problème de la figure (3.12) avec un MLP contenant 3 neurones d'entrée, 4 neurones cachés et 2 neurones de sortie.
(a) Formes des sorties des neurones cachés et des neurones de sortie
(b) La frontière de décision générée

3.7 Le MLP et la statistique : Interprétation des sorties

Dans le domaine de la classification, il a été démontré par plusieurs auteurs [2][32][33][34]que les sorties d'un MLP avec une seule couche cachée peuvent être interprétées comme les probabilités a posteriori. Dans cette partie, nous présentons les étapes suivies par ces auteurs :

Pour un problème de classification à K classes $(C_1, C_2, \dots, C_K)$, supposons que le réseau entrainé comporte K sorties $(z_1, z_2, \dots, z_K)$ permettant de coder les sorties désirées comme suit :

$$t_k = \begin{cases} 1 \; si \; X \in C_k \\ 0 \; autrement \end{cases} \tag{3.18}$$

En notant l'ensemble des poids du réseau par W, la fonction coût correspondante à chaque sortie z_k est :

$$F(W) = [z_k(X, W) - t_k]^2 \tag{3.19}$$

$$F(W) = \sum_{X \in C_k} [z_k(X, W) - 1]^2 + \sum_{X \notin C_k} [z_k(X, W) - 0]^2 \tag{3.20}$$

$$F(W) = Q \left\{ \frac{Q_k}{Q} \frac{1}{Q_k} \sum_{X \in C_k} [z_k(X, W) - 1]^2 + \frac{Q - Q_k}{Q} \frac{1}{Q - Q_k} \sum_{X \notin C_k} [z_k(X, W) - 0]^2 \right\} \tag{3.21}$$

Où : Q est nombre total d'exemples et Q_k est celui des exemples appartenant à la classe C_k.

Dans la limite d'une infinité de données, nous pouvons utiliser l'équation de Bayes pour exprimer l'Eq. (3.21) comme suit :

$$\lim_{Q\to\infty}\frac{1}{Q}F(W) = P(C_k)\int[z_k(X,W)-1]^2\,p(X/C_k)dx + P(C_{i\neq k})\int[z_k(X,W)]^2\,p(X/C_{i\neq k})dx \tag{3.22}$$

$$= \int z_k^2(X,W)p(X)dx - 2\int z_k^2(X,W)p(X,C_k)dx + \int p(X,C_k)dx \tag{3.23}$$

$$= \int[z_k(X,W)-P(C_k/X)]^2\,p(X)dx + \int P(C_k/X)P(C_{i\neq k}/X)\,p(X)dx \tag{3.24}$$

Dans l'équation précédente (Eq. 3.24) le deuxième terme est indépendant du réseau. La RP modifie donc les poids pour minimiser :

$$\int[z_k(X,W)-P(C_k/X)]^2\,p(X)dx \tag{3.25}$$

Du fait que ceci est vrai pour chaque classe C_k, la RP minimise la somme suivante :

$$\sum_{k=1}^{K}\int[z_k(X,W)-P(C_k/X)]^2\,p(X)dx \tag{3.26}$$

A la limite d'une infinité de données, les sorties d'un MLP entrainé permettront donc d'approximer les probabilités a posteriori :

$$z_k(X,W) \cong P(C_k/X) \tag{3.27}$$

Il est à noter que le résultat précédent ne serait garanti que si le nombre de neurones caché est suffisant, que le réseau ne tombe pas dans les minima locaux et que le nombre d'exemples d'apprentissage soit suffisant [1][2] [32][33][34].

3.8 Amélioration des performances de la RP

La RP est un algorithme puissant, utile et relativement facile à comprendre. C'est un algorithme simple même pour les modèles complexes ayant des centaines ou des milliers de paramètres [1], mais la convergence de cet algorithme est lente. Pour corriger ses problèmes, plusieurs travaux ont été proposés dans la littérature. Ces auteurs se sont investis sur plusieurs axes :

3.8.1 Normalisation des données

Dans certaines applications réelles de la RDF, il est possible que les caractéristiques soient de rangs très différents ce qui influencera mal le rendement du MLP utilisé. Dans de telles situations, le réseau ajustera les poids à partir des caractéristiques de plus grand rang bien davantage qu'aux caractéristiques de rangs inférieurs. Cela est dû au fait que l'erreur dépendra à peine des valeurs d'entrée très petite. Afin d'éviter de telles difficultés, les caractéristiques devront être normalisées de sorte qu'elles soient du même rang. Ce processus est effectué une seule fois avant la phase d'apprentissage proprement dite.

3.8.2 Méthodes basées sur les poids initiaux

L'initialisation des poids a été largement considérée en tant qu'une des approches les plus efficaces pour l'accélération de l'apprentissage des réseaux de neurones, et plusieurs travaux s'inscrivent dans ce contexte. Parmi ces travaux, nous citons :

Russo [35] a proposé une règle heuristique permettant de générer des poids initiaux dont la possibilité d'avoir des saturations prématurées est peu probable ; il suggère de choisir les poids initiaux avec des amplitudes inférieures à $2.4/M$ (M étant le nombre des entrées du neurone dont réside le poids) :

$$-\frac{2.4}{M} < {u_{mj}}^{(i)} < \frac{2.4}{M} \tag{3.28}$$

Nguyen et Widrow [36] ont mis au point une méthode d'initialisation qui consiste à distribuer les poids initiaux dans la couche cachée de telle sorte que chaque exemple présenté permette d'accorder un apprentissage efficace à tous les neurones cachés. Ils ont posé : $\beta = 0.7(M^{1/N})$ où M est le nombre des neurones cachés et N est la dimension de l'espace caractéristique. Ensuite et après avoir initialisé aléatoirement les poids (${w_{nm}}^{(i)}$) entre -1 et $+1$, chaque poids initial w_{nm} sera calculé selon la relation suivante :

$${w_{nm}}^{(0)} = \frac{\beta}{\|W^{(i)}\| {w_{nm}}^{(i)}} \tag{3.29}$$

Où : $W^{(i)} = ({w_{11}}^{(i)}, \dots, {w_{NM}}^{(i)})$. L'ensemble (${w_{nm}}^{(0)}$) est pris pour être les poids initiaux de la couche cachée. Selon les hauteurs de cette approche, cela aide à éviter la saturation prématurée des neurones de la couche cachée.

Shepanski [37] a considéré l'apprentissage d'un MLP comme un problème d'estimation. L'ensemble optimal de poids est déterminé en utilisant le critère des moindres carrés employé avec une technique de matrice pseudo-inverse.

Master [38] a employé aussi la méthode des moindres carrés comme partie de son algorithme d'initialisation des poids. Pour un réseau avec une couche cachée, il a suggéré l'utilisation des algorithmes génétiques pour l'initialisation des poids de la couche cachée.

Yam et Chow ont proposé deux méthodes d'initialisation des poids basées également sur les moindres carrés [39][40]. Dans [39], le système est considéré linéaire et les paramètres reliant l'entrée et la sortie sont obtenus par une méthode des moindres carrés linéaire. Dans [40], les sorties des neurones cachés sont des valeurs assignées dans la région de non-saturation et les poids initiaux optimaux entre l'entrée et les couches cachées sont évalués par une méthode algébrique linéaire. Ces auteurs ont proposé par la suite une autre approche [41] basée sur l'inégalité de Cauchy et l'utilisation d'une méthode algébrique linéaire.

3.8.3 Méthodes basées sur le gain d'apprentissage

Afin d'optimiser le temps d'apprentissage, plusieurs travaux ont été basés sur la modification de ce paramètre d'une façon automatique durant l'apprentissage. Par exemple, l'une des méthodes de base consiste à l'augmenter si l'erreur décroît, le réduire si l'erreur augmente et le laisser inchangé si cette dernière change légèrement [4].

Dans ce même contexte, Jacobs [42] a introduit une règle d'apprentissage appelée la règle delta bar delta. Cette technique est basée sur l'utilisation d'un pas d'apprentissage différent pour chaque poids et l'ajustement itératif de ces pas. L'adaptation du poids w_{nm} (ou u_{mj}) à l'itération $(r+1)$ est de la forme :

$$w_{nm}^{(r+1)} = w_{nm}^{(r)} - \eta_{nm} \frac{\partial E^{(r)}}{\partial w_{nm}} \tag{3.30}$$

Cette technique consiste à ajuster chaque poids en se basant sur la forme locale de l'erreur définie par les signes de ses gradients. L'ajustement de chaque pas d'apprentissage dépend de la variation de l'erreur en fonction du poids associé ; il devra être augmenté si les dérivées de plusieurs modifications ont le même signe et il devra être réduit s'ils changent de signes. La modification du pas d'apprentissage η_{nm} s'effectue comme suit :

On détermine pour chaque gain :

$$\Delta_{nm}^{(r)} = \beta\left(\Delta_{nm}^{(r-1)}\right) + (\beta - 1)\frac{\partial E^{(r)}}{\partial w_{nm}} \tag{3.31}$$

Ce gain sera ensuite ajusté selon :

$$\eta_{nm}^{(r-1)} = \eta_{nm}^{(r-1)} + \delta \quad si \quad \Delta_{nm}^{(r-1)} \frac{\partial E^{(r)}}{\partial w_{nm}} > 0 \tag{3.32a}$$

$$\eta_{nm}^{(r-1)} = \theta \eta_{nm}^{(r-1)} \quad si \quad \Delta_{nm}^{(r-1)} \frac{\partial E^{(r)}}{\partial w_{nm}} < 0 \tag{3.32b}$$

$$\eta_{nm}^{(r-1)} = \eta_{nm}^{(r-1)} \quad si \quad \Delta_{nm}^{(r-1)} \frac{\partial E^{(r)}}{\partial w_{nm}} = 0 \tag{3.32c}$$

Où : δ, β et θ sont des petites constantes positives pré-spécifiées par l'utilisateur, Jacobs [42] a posé $\delta = 0.05, \theta = 0.3$ et $\beta = 0.7$. La valeur initiale des $\eta_{nm}^{(0)}$ est mise à (0.1).

3.8.4 L'ajout d'un terme d'inertie : Le moment

Cette technique a été proposée par Rumelhart et al. [18], elle consiste à ajouter un terme d'inertie aux corrections apportées à chaque poids. L'adaptation du poids w_{nm} (ou u_{mj}) à l'itération $(r+1)$ sera de la forme :

$$w_{nm}^{(r+1)} = w_{nm}^{(r)} + \eta \nabla E\left(w_{nm}^{(r)}\right) + \mu \Delta\left(w_{nm}^{(r-1)}\right)$$

$$= w_{nm}^{(r)} - \eta \frac{\partial E^{(r)}}{\partial w_{nm}} + \mu\left(w_{nm}^{(r)} - w_{nm}^{(r-1)}\right) \quad (3.33)$$

L'introduction du moment tente de corriger les surfaces d'erreur qui comportent souvent des plateaux (régions dans lesquelles la pente *dj(w)/dw* est très petite).

3.8.5 Méthodes basées sur les fonctions d'activation

Le paramètre (α) agit sur la forme des sigmoïdes (figure3.7), et par conséquent sur la réponse des neurones. Une modification de ce paramètre améliore donc la convergence de l'algorithme de l'apprentissage [1][4][43][44]. Parmi les méthodes basées sur ce paramètre, la technique de Zurada et Yamada [43] consiste à traiter ce paramètre comme un poids additionnel et à l'ajuster itérativement. Chaque $\alpha^{(r)}$ étant ajusté à l'itération $(r+1)$ comme suit :

$$\alpha_m^{(r+1)} = \alpha_m^{(r)} - \eta \frac{\partial E^{(r)}}{\partial \alpha_m} \quad (3.34)$$

Les paramètres des neurones de la couche de sortie sont modifies par:

$$\frac{\partial E^{(r)}}{\partial \alpha_m} = \frac{\partial E^{(r)}}{\partial z_j} \frac{\partial z_j}{\partial \alpha_m}$$

$$= -2\left(t_j - z_j\right) \acute{g}\left(s_j\right) s_j \quad (3.35)$$

Les paramètres des sigmoïdes des neurones cachés sont modifies par:

$$\frac{\partial E^{(r)}}{\partial \alpha_m} = \frac{\partial E^{(r)}}{\partial \alpha_m} \frac{\partial z_j}{\partial \alpha_m}$$

$$= \left[-2 \sum_{j=1}^{J} \left(t_j - z_j\right) \acute{g}\left(s_j\right) u_{jmj}\right] \acute{h}(r_m) r_m \quad (3.36)$$

Dans le même contexte, Eom et al. [45] ont suggéré l'utilisation d'un gain ajustable pour les fonctions d'activation. Ils ont proposé un système basé sur la logique floue pour l'ajustement

automatique de ce gain. Les entrées de ce système sont les erreurs mesurées dans la couche cachée et la couche de sortie tandis que la sortie de ce système est le gain.

3.8.6 Autres approches

Nous pouvons citer également le travail de Zweiri [46][47] qui consiste à ajouter, en plus du gain et du moment, un troisième terme à l'équation de mise à jour des poids. Selon les auteurs, le terme ajouté (le facteur proportionnel) permet d'accélérer le processus d'apprentissage. La formule de mise à jour proposée est:

$$w_{nm}^{(r+1)} = w_{nm}^{(r)} + \eta \nabla E\left(w_{nm}^{(r)}\right) + \mu \Delta w_{nm}^{(r-1)} + \delta e\left(w_{nm}^{(r)}\right) \tag{3.37}$$

Wang et al. [48] ont noté que le problème des minima locaux dans l'algorithme de la RP est habituellement provoqué par un désaccord de mise à jour entre les poids de la couche cachée et ceux de la couche de sortie. Pour résoudre ce problème, ils ont proposé une fonction d'erreur modifiée. Cette fonction vise l'harmonisation de la mise à jour des poids par l'ajout d'un terme à l'erreur conventionnelle. La fonction d'erreur proposée est :

$$E = \frac{1}{2}\sum_{q=1}^{Q}\sum_{j=1}^{J}\left(t_j^{(q)} - z_j^{(q)}\right)^2 + \frac{1}{2}\sum_{q=1}^{Q}\left(\sum_{j=1}^{J}\left(t_j^{(q)} - z_j^{(q)}\right)^2\right)\left(\sum_{m=1}^{M}\left(y_m^{(q)} - 0.5\right)^2\right) \tag{3.38}$$

3.9 Conclusion

Dans ce chapitre, nous avons évoqué la classification par le MLP, et décrit son apprentissage en utilisant la RP qui est l'algorithme le plus utilisé pour l'entrainement de ce type de réseaux. C'est un algorithme puissant, facile à mettre en œuvre et simple même pour les modèles complexes. Cela a permis au MLP d'être un approximateur universel et un bon classificateur permettant de former des frontières de décision complexe. De plus, ses sorties forment une approximation des probabilités a posteriori ce qui permet leur utilisation dans les systèmes d'aide à la décision. En contrepartie, la convergence de cet algorithme est lente, et risque souvent d'être prisonnier des minima locaux. Ces problèmes ont été abondamment évoqués dans la littérature des réseaux de neurones. Nous avons effectué un bref survol sur les méthodes qui s'inscrivent dans ce contexte.

A partir de ce chapitre, nous concluons qu'un MLP entrainé par la RP constitue un processus important ayant de bonnes propriétés, mais qui admet aussi l'inconvénient de la lenteur de son apprentissage, voire le risque de ne pas converger. Cela constitue la motivation principale de notre travail : proposer une méthode qui exploite les propriétés du MLP et dont l'objectif est d'améliorer son apprentissage. La méthode proposée, la classification étiquetée, sera présentée dans le cinquième chapitre de ce manuscrit.

Chapitre 4

Classification neuro-floue

Ce chapitre est voué à la classification par les systèmes neuro-flous dont leur mise en œuvre se base sur l'intégration des réseaux de neurones et des systèmes flous. En premier lieu, nous donnons une introduction à la logique floue et aux systèmes flous. Nous présentons ensuite les systèmes neuro-flous, et décrivons leurs objectifs, leurs architectures et leurs apprentissages. Finalement, nous abordons l'utilisation de ces systèmes dans le domaine de la classification, et analysons leurs performances à travers deux exemples synthétiques.

4.1 Introduction

On assiste de nos jours à une croissance technologique permanente comportant des systèmes de plus en plus complexes. Ces systèmes nécessitent le traitement de différents problèmes dont chacun exige un type de calcul distinct. Ceci a donné lieu à la naissance de nouveaux systèmes hybrides combinant les réseaux de neurones, la logique floue, les systèmes experts et les algorithmes génétiques. Chacune de ces techniques d'intelligence artificielle possède des propriétés particulières lui permettant d'être bien adaptée à résoudre certains types de problèmes. L'objectif de l'hybridation consiste alors à concevoir des systèmes plus robustes permettant d'exploiter les avantages offerts par chacune des techniques utilisées.

La conception des systèmes neuro-flous constitue un exemple important de cette tendance actuelle d'intégration de différents outils dans des architectures hybrides. Ces systèmes mettent à profit les propriétés de deux importantes méthodes de l'intelligence artificielle ; les réseaux de neurones et la logique floue. Ceci leur a permis de prouver leur efficacité dans une variété de problèmes, notamment dans le domaine de la reconnaissance des formes et du contrôle industriel.

4.2 Notions de base la logique floue

Le concept de la logique floue a été introduit en 1965 par Lotfi Zadeh, professeur de l'université de Californie de Barkley (USA), en publiant son article intitulé : Ensembles flous (fuzzy sets) [49]. Au début, ce nouveau concept 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

de nombreux produits industriels utilisant cette théorie. Ces produits ont été portés à large public d'où l'essor de cette théorie.

Au lieu d'utiliser des variables numériques précises, la logique floue fait appel à des variables floues comme : faible, grand, petit…etc. Elle tente de formaliser les phénomènes traités de la même façon que les humains le font. Face à la plupart des situations, nos connaissances sont en effet imparfaites et les informations qu'on manipule sont souvent imprécises et/ou incertaines. Par exemple pour choisir un appartement, il nous suffit de savoir qu'elle est proche de notre travail sans une approche exacte de la distance. 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, si le feu de réglementation est rouge et s'il n'est 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 » !

Il convient donc d'utiliser la logique floue lorsque les connaissances dont nous disposons sur le système traité sont entachées d'incertitudes et/ou d'imprécision.

4.2.1 Concept d'un sous-ensemble flou

Le concept de sous-ensemble flou a été introduit essentiellement pour répondre au besoin de représenter les connaissances imprécises. Cette théorie est basée sur la notion d'appartenance partielle, et évite les limites rigides entre les catégories en autorisant des éléments à n'appartenir ni à une catégorie ni à l'autre et à appartenir partiellement à chacune. Cela correspond à de nombreuses situations dans le monde réel où il est difficile de décider d'une manière précise si quelque chose appartient ou non à une classe spécifique.

Cependant, l'utilisation des sous-ensembles flous semble bien adaptée aux problèmes nécessitant le traitement des catégories aux limites mal définies, les situations intermédiaires entre le tout et le rien, le passage progressif d'une propriété à l'autre et le traitement des valeurs approximatives.

4.2.2 Définition d'un sous-ensemble flou

Dans la logique classique (boolienne), un sous-ensemble A d'un ensemble de référence X est défini par une fonction caractéristique μ_A qui prend la valeur 1 pour les éléments de X appartenant à A et la valeur 0 pour ceux n'appartenant pas à A. Chaque élément x de l'ensemble X soit qu'il appartienne à A où qu'il ne le soit pas :

$$\mu_A(x): A \rightarrow \{0,1\} \tag{4.1}$$

La logique floue est basée sur la notion d'appartenance partielle ; un élément peut alors appartenir partiellement à plusieurs sous-ensembles. Un sous-ensemble flou B est défini par une fonction d'appartenance μ_B qui associe à chaque élément x un degré $\mu_B(x)$ compris entre 0 (non-appartenance) et 1 (appartenance totale) :

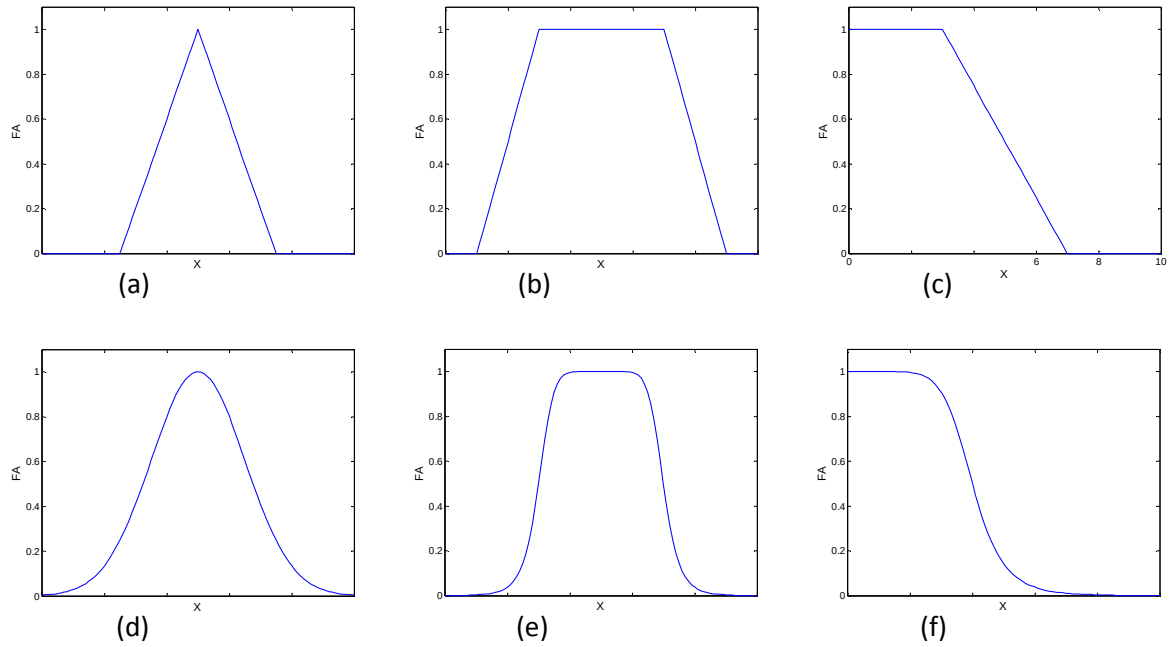


Fig. 4.1. Exemples de fonctions d'activation
(a) Fonction triangulaire (b) Fonction trapézoïdale
(c) Fonction linéaire saturable (d) Fonction gaussienne
(e) Fonction cloche généralisée (f) Fonction sigmoïdale

$$\mu_B(x): B \rightarrow [0,1] \tag{4.2}$$

Par conséquent, plus $\mu_B(x)$ tend vers 1, plus le degré d'appartenance de x à B sera élevé et vice-versa. La figure (4.1) représente quelques exemples de fonctions d'appartenance.

4.2.3 Caractéristiques d'un sous-ensemble flou

Soit un sous-ensemble flou A d'un référentiel X.

Le support de A, noté $supp(A)$, est la partie de X sur laquelle la fonction d'appartenance de A n'est pas nulle :

$$supp(A) = \{x \in X / \mu_A(x) \neq 0\} \tag{4.3}$$

La hauteur de A, noté $h(A)$, est la plus grande valeur prise par sa fonction d'appartenance :

$$h(A) = sup_{x \in X}\{\mu_A(x)\} \tag{4.4}$$

A est normalisé si $h(A) = 1$.

Le noyau de A, noté $noy(A)$, est la partie de X sur laquelle la fonction d'appartenance de A est égale à 1:

$$noy(A) = \{x \in X / \mu_A(x) = 1\} \tag{4.5}$$

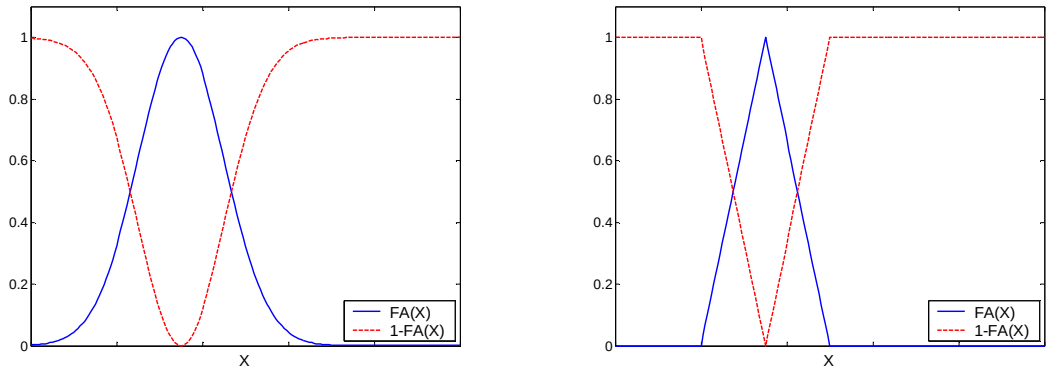


Fig. 4.2. Exemples de l'opérateur **NON**
(a) Fonction Gaussienne
(b) Fonction triangulaire

Lorsque X est fini, A peut être caractérisé par sa cardinalité $|A|$:

$$|A| = \sum_{x \in X} \mu_A(x) \tag{4.6}$$

4.2.4 Opérateurs flous :

a) L'opérateur Négation (Complément)

Soit A un sous-ensemble d'un référentiel X : L'ensemble complémentaire de A, noté $\bar{A}$, est défini par les éléments de X qui n'appartiennent pas à A. La fonction d'appartenance de l'ensemble complémentaire est donnée par:

$$\forall x \in X: \mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{4.7}$$

La figure (4.2) illustre des exemples des fonctions d'appartenances du complément.

b) L'opérateur ET (intersection)

Soit A et B deux sous-ensembles flous, 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 :

$$\forall x \in X: \mu_C(x) = min\{\mu_A(x), \mu_B(x)\} \tag{4.8}$$

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

$$\forall x \in X: \mu_C(x) = \mu_A(x)\mu_B(x) \tag{4.9}$$

La figure (4.3) illustre un exemple de l'intersection de deux fonctions d'appartenances gaussienne en utilisant ces deux opérateurs.

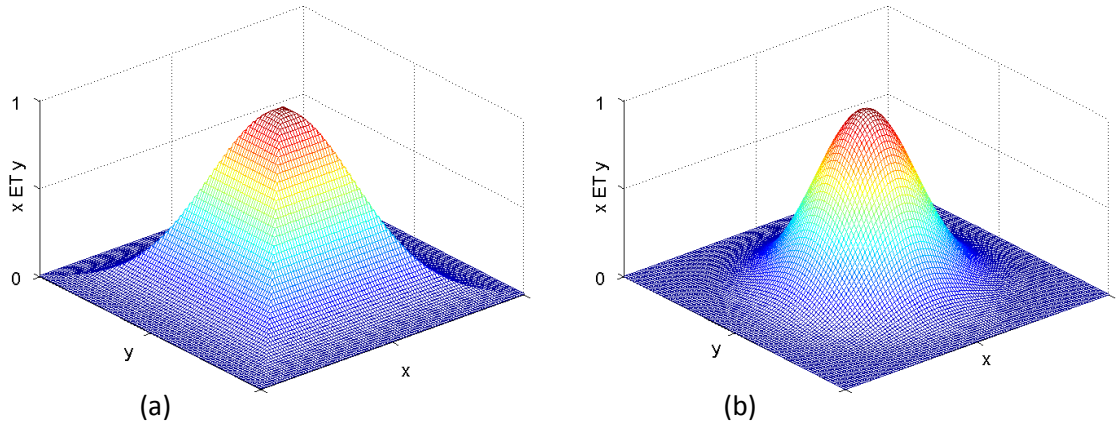


Fig. 4.3. Exemple de l'opérateur **ET** de deux fonctions d'appartenance gaussienne
(a) Opérateur min: $min\{\mu_A(x), \mu_B(x)\}$
(b) Opérateur produit: $\mu_A(x)\mu_B(x)$

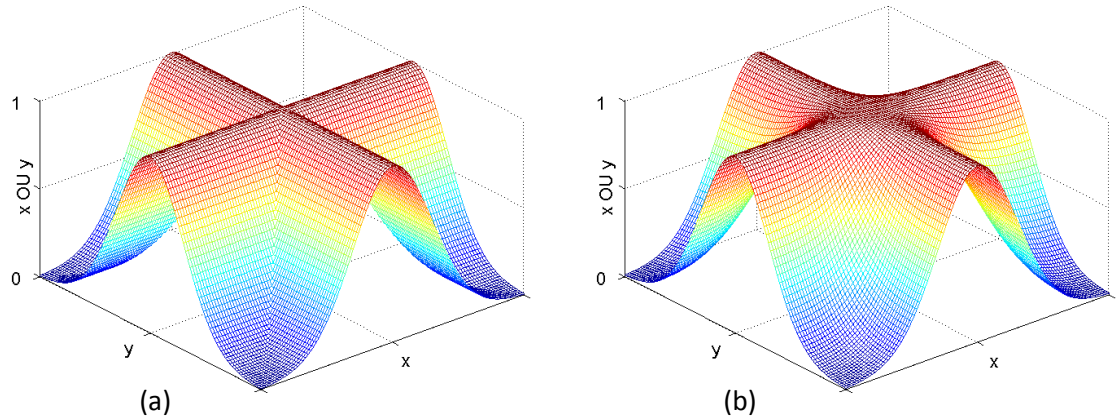


Fig. 4.4. Exemple de l'opérateur **OU** de deux fonctions d'appartenance gaussienne
(a) Opérateur max : $max\{\mu_A(x), \mu_B(x)\}$
(b) Opérateur probabiliste : $\mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$

c) L'opérateur OU (union)

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

$$\forall x \in X: \mu_C(x) = max\{\mu_A(x), \mu_B(x)\} \tag{4.10}$$

L'opérateur OU peut également être réalisé par :

$$\forall x \in X: \mu_C(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x) \tag{4.11}$$

La figure (4.4) illustre un exemple de l'intersection de deux fonctions d'appartenances gaussienne en utilisant les deux opérateurs précédents.

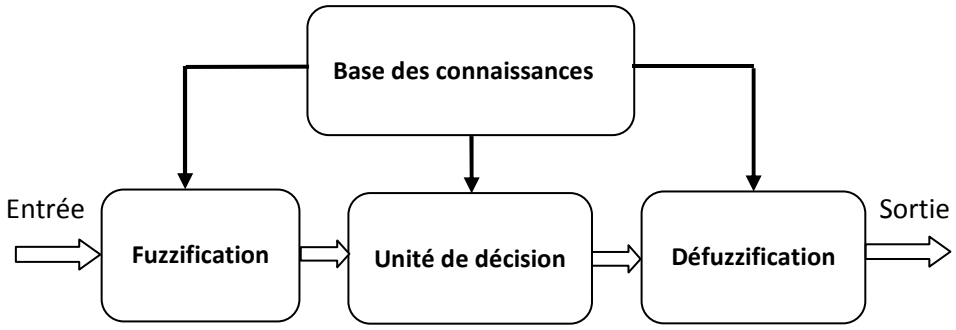


Fig. 4.5. Système d'Inférence Flou (SIF)

4.3 Systèmes d'inférence flous

Les systèmes d'inférence flous (SIF), appelés aussi les systèmes experts flous [50] (Fuzzy expert systems), les modèles flous [51] [52](Fuzzy Models), ou simplement systèmes flous [53] (Fuzzy systems), sont des structures de calcul basées sur le concept de la théorie des ensembles flous, des règles si-alors floues et du raisonnement flou [50]. En raison de leur caractère pluridisciplinaire, ils ont été appliqués avec succès dans une grande variété d'applications, tels que l'automatique, la robotique, la reconnaissance des formes ainsi que les systèmes d'aide à la décision [53].

Un SIF se compose essentiellement de trois éléments conceptuels (figure 4.5): une base de règles qui contient une collection de règles floues, une base de données qui définit les fonctions utilisées dans les règles floues, et un mécanisme de raisonnement qui effectue la procédure d'inférence en se basant sur les règles. Ce processus s'effectue en trois phases:

- Fuzzification : Cette étape consiste à déterminer le degré d'appartenance de chaque variable d'entrée aux sous-ensembles flous utilisés dans les prémisses des règles.
- Inférence : Il s'agit de la détermination des degrés d'activations des règles par la combinaison des propositions de leurs prémisses et du calcul des sorties.
- Défuzzification : Le rôle de cette étape est de fournir des valeurs précises des variables de sorties du système afin qu'elle soit exploitable. Par exemple dans le cas de la commande industriel, la sortie du système constitue le signal envoyé au processus.

4.3.1 Les règles Si-Alors floues

Les règles Si-Alors floues (appelées aussi règles floues) admettent généralement la forme suivante :

$$\boldsymbol{Si}\ x\ est\ A\ \boldsymbol{et}\ y\ est\ B\ \boldsymbol{Alors}\ z\ est\ C \tag{4.12}$$

Où : A,B et C sont des variables linguistiques définies par des sous-ensembles flous.

Une règle floue se compose de deux parties :

- La prémisse : formée par une combinaison de prépositions liées entre elles par les opérateurs Et, Ou et Non. Dans l'exemple précédent, la proposition «$x\ est\ A\ \boldsymbol{et}\ y\ est\ B$ » constitue la prémisse de cette règle.
- La conclusion : c'est la partie qui définit la conséquence de la règle. Dans la règle précédente, la conclusion est la proposition «$z\ est\ C$ ».

La forme de ces règles est très employée dans notre expression quotidienne, par exemple :

$$\boldsymbol{Si}\ la\ Résistance\ est\ Grande\ \boldsymbol{Alors}\ le\ Courant\ est\ Faible$$

Où $le\ Courant$ et $la\ Résistance$ sont des variables floues et $Faible$ et $Grande$ sont des termes linguistiques caractérisés par des fonctions d'appartenances.

Une autre forme de règles Si-Alors floues, proposée par Sugeno et Takagi [51][52], comporte une conclusion numérique. Dans ces règles, les sous-ensembles flous sont introduits seulement dans les prémisses. Ce type de règles admet la forme :

$$\boldsymbol{Si}\ x\ est\ A\ \boldsymbol{et}\ y\ est\ B\ \boldsymbol{Alors}\ z = f(x, y) \tag{4.13}$$

La fonction $f(x, y)$ est généralement un polynôme de variables d'entrée x et y, mais elle peut être n'importe quelle fonction tant qu'elle permet de décrire convenablement la sortie du modèle en se basant sur les régions floues définie par les prémisses des règles [53]. Un exemple de telles règles est donné par :

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

La conclusion ($Tension$) est numérique.

En se basant sur un ensemble de règles floues, un SIF met en œuvre une application non linéaire de l'espace d'entrée à l'espace de sortie. Chacune des règles utilisées décrit le comportement local de cette application ; sa prémisse définit une région de l'espace d'entrée et sa conclusion précise la sortie dans cette région [53].

4.3.2 Système d'Inférence Flou de Mamdani

Le Système d'Inférence Flou de Mamdani [54] a été proposé en premier lieu comme une tentative de contrôle d'une machine à vapeur et de chaudières. Ce système est basé sur l'utilisation d'un ensemble de règles linguistiques obtenue à partir de l'expérience humaine.

Ce système s'effectue en se basant sur six étapes :

1. Détermination d'un ensemble des règles floues.
2. Fuzzification des entrées en utilisant les fonctions d'appartenances des entrées correspondantes.
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 sortie.

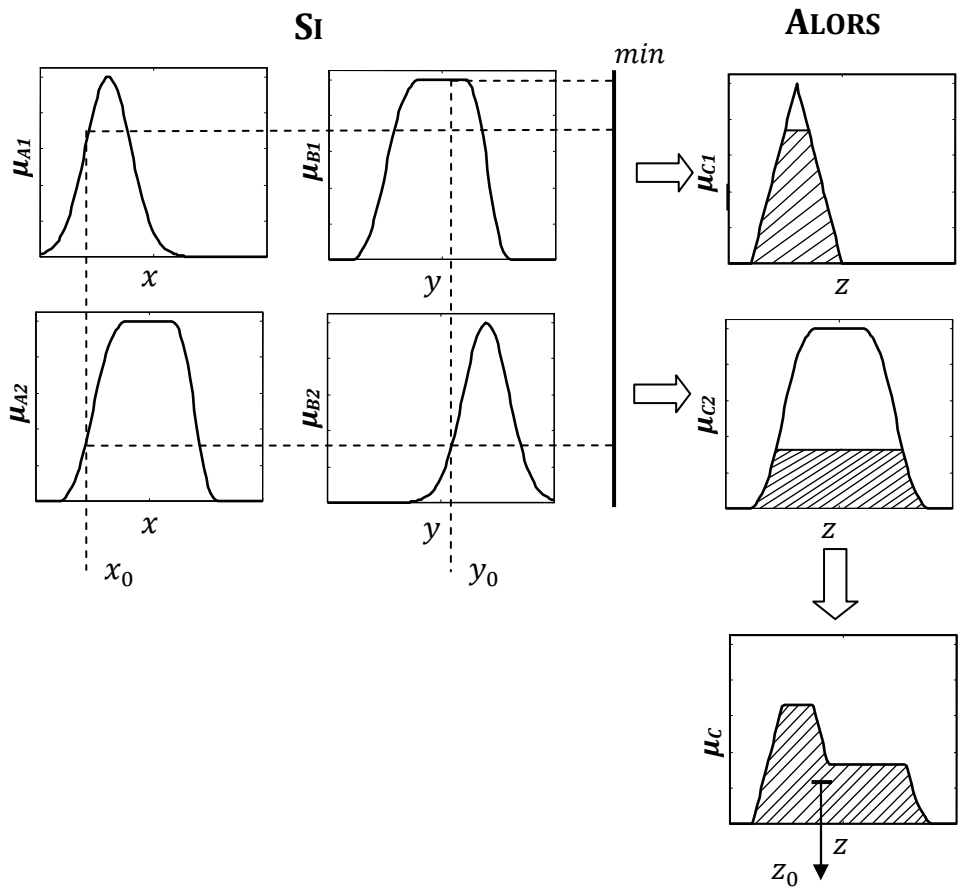


Fig. 4.6. Système d'inférence de Mamdani

La figure (4.6) illustre 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 la somme des conséquences de règles.

4.3.3 Système d'Inférence Flou de Sugeno et Takagi

Le système de Sugeno et Takagi [51] [52] a été proposé afin de concevoir un système de génération des règles floues en se basant sur les entrée/sortie. Ce système emploie des règles de la forme donnée par l'Eq. 4.13 . Il se diffère ainsi du système de Mamdani sur la façon de détermination de la sortie. Le système de Sugeno et Takagi détermine les conséquences des règles par une combinaison linéaire des entrées avec des constantes, la sortie finale sera ensuite calculée par la moyenne pondérée des conséquences des règles.

Un exemple à deux variables d'entrée et deux règles est illustré sur la figure (4.7). Les deux règles sont données par :

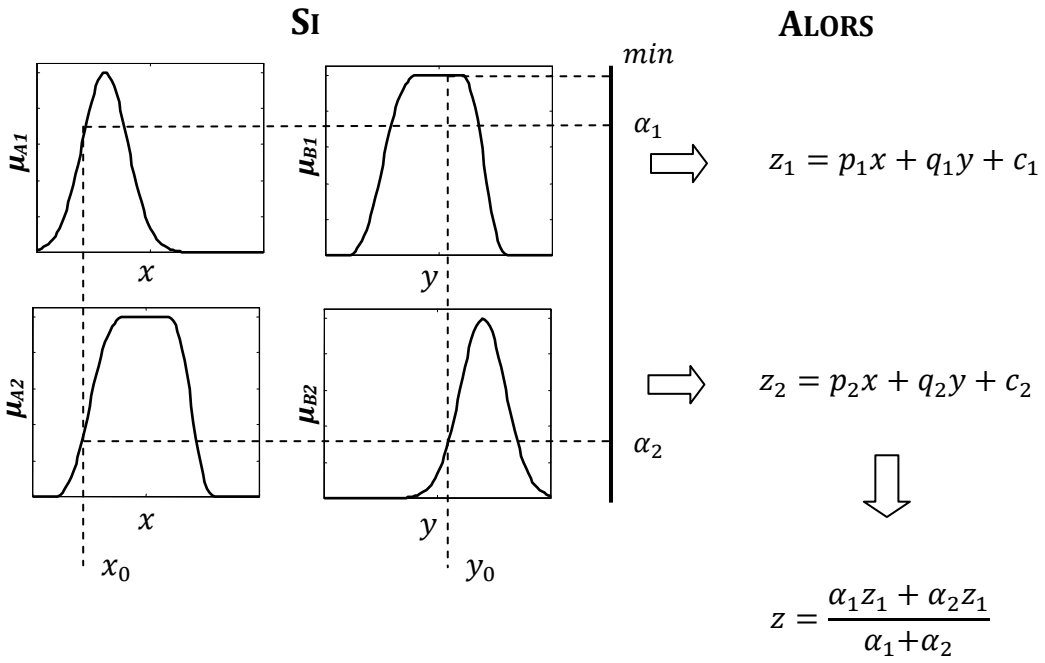


Fig. 4.7. Système d'inférence de Sugeno et Takagi

$$\begin{cases} \mathcal{R}_1 : Si\ x\ est\ A_1\ et\ \ y\ est\ B_1\ Alors\ z_1 = p_1x + q_1y + c_1 \\ \mathcal{R}_2 : Si\ x\ est\ A_2\ et\ \ y\ est\ B_2\ Alors\ z_2 = p_2x + q_2y + c_2 \end{cases} \tag{4.14}$$

Les degrés d'activation de ces deux règles sont donnés par :

$$\begin{cases} \alpha_1 = \mu_{A1}(x_0)\ ET\ \mu_{B1}(y_0) \\ \alpha_2 = \mu_{A2}(x_0)\ ET\ \mu_{B2}(y_0) \end{cases} \tag{4.15}$$

Les sorties sont :

$$\begin{cases} z_1 = p_1x + q_1y + c_1 \\ z_2 = p_2x + q_2y + c_2 \end{cases} \tag{4.16}$$

La sortie finale du système est donnée par :

$$z = \frac{\alpha_1 z_1 + \alpha_2 z_1}{\alpha_1 + \alpha_2} \tag{4.17}$$

Selon Melin et Castillo [53] ; le système de sugeno, et contrairement au modèle de mamdani, ne permet pas de suivre strictement les règles d'inférence dans leur mécanisme de raisonnement flou et cela pose quelques difficultés dans le cas des entrées floues.

4.4 Les systèmes neuro-flous

4.4.1 Objectifs

Bien que les réseaux de neurones et la logique floue aient prouvé leur efficacité dans divers problèmes de l'intelligence artificielle, néanmoins leurs utilisations soufrent de quelques

inconvénients comme la difficulté d'initialisation et d'interprétation des réseaux de neurones et le manque de techniques d'adaptation des paramètres des systèmes flous. La mise œuvre des systèmes neuro-flous ne vise pas seulement l'exploitation de leurs avantages, mais aussi de contourner leurs inconvénients en exploitant leurs propriétés qui semblent très complémentaires sur plusieurs aspects. Par exemple, la logique floue, avec ses capacités de modélisation des connaissances, pourra surmonter les difficultés d'initialisation et d'interprétation des réseaux de neurones. En outre, les capacités d'apprentissage des réseaux de neurones peuvent être employées pour l'ajustement des paramètres des systèmes d'inférences flous, ce qui pourra réduire l'effort fourni lors de leur élaboration et leur évaluation. Le tableau (4.1) récapitule les avantages et les inconvénients de ces deux méthodes.

Tableau (4.1) : Avantages et inconvénients des RNA et SIF

	Avantages	Inconvénients
RNA	Capacités d'apprentissage Capacités de généralisation Calcul parallèles	Difficulté d'interprétation Manque de techniques d'initialisation Impossibilité d'utilisation des connaissances a priori
SIF	Interprétables possibilité d'utilisation des connaissances a priori	Manque de techniques d'adaptation

4.4.2 Architecture

La combinaison des réseaux des neurones et de la logique floue peut être effectué sous différentes formes et à plusieurs niveau. Généralement, les approches d'intégration se basent sur les stratégies suivantes [55]:

- incorporation du concept du flou dans la structure des RNA : fuzzification des entrées, utilisation des classes floues aux exemples d'apprentissage, fuzzification de la procédure d'apprentissage et obtention des sorties des RNA en terme de sous-ensembles flous.
- Conception d'une structure neuronale basée sur le mécanisme des SIF.
- Changement des caractéristiques de base des neurones : les neurones sont conçus pour accomplir de diverses opérations floues utilisées pour la manipulation des sous-ensembles flous.
- Utilisation d'une mesure floue pour modéliser la fonction erreur du réseau.
- Rendre les neurones flous : les entrées et les sorties des neurones sont des sous-ensembles flous et leur activation est aussi une procédure floue.

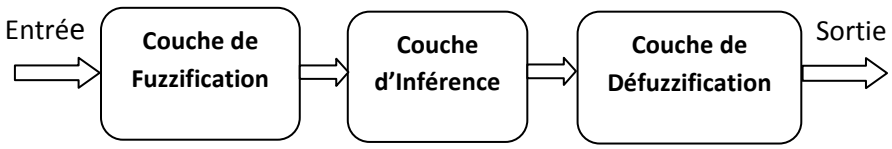


Fig. 4.8. Architecture de base d'un système neuro-flou

Les systèmes neuro-flous sont généralement basés sur l'incorporation des systèmes flous dans la structure des réseaux de neurones, ainsi chaque étape des SIF sera effectuée par une couche de neurones (figure 4.8). Le système obtenu est entraîné en utilisant les algorithmes d'apprentissage des réseaux de neurones.

4.4.3 Systèmes basés sur le modèle de mamdani

Ces systèmes se composent de cinq couches [56][57] (fig. 4.9):

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

2ème couche : Chaque neurone de cette couche correspond à un terme linguistique caratérisé par une fonction d'appartenance. Leur rôle est de déterminer les degrés d'appartenances des variables d'entrée aux différents sous-ensembles flous (étape de fuzzification).

3ème couche : Chaque neurone de cette couche correspond à une règle, il détermine son degré d'activation en effectuant le Et des variables qui lui arrivent. Généralement, l'opérateur ET est réalisé par le produit.

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

5ème couche : le rôle de cette couche est de combiner toutes les conséquences des règles

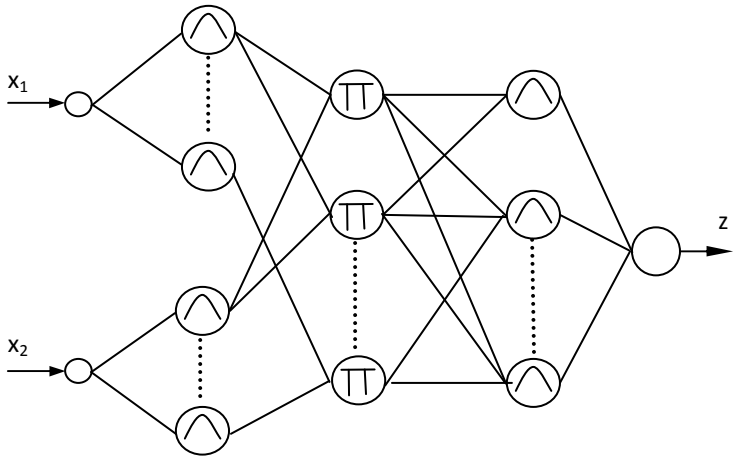


Fig. 4.9. Système neuro-flou basé sur le modèle de Mamdani

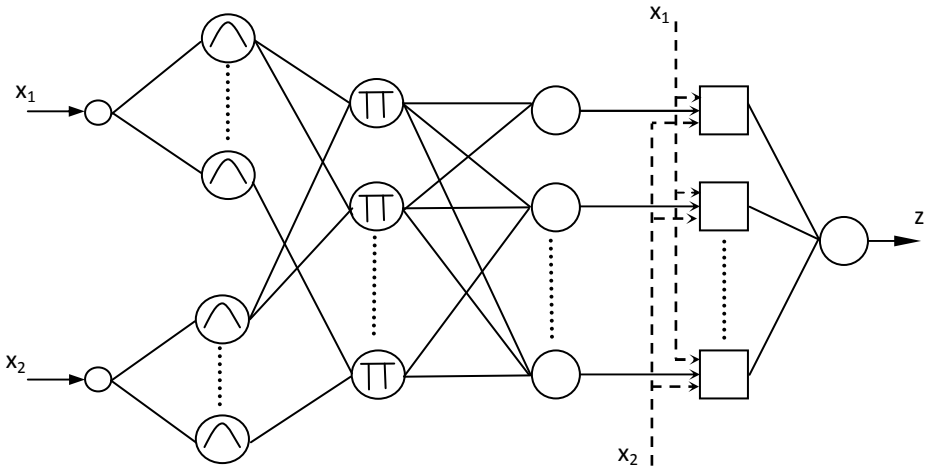


Fig. 4.10 Système neuro-flou basé sur le modèle de Sugéno et Takagi

et de fournir les sorties déffuzzifieés.

4.4.4 Systèmes basés sur le modèle de Takagi et Sugeno

Ce modèle se compose de six couches (fig. 4.10) dont les trois premières sont similaires à celle du modèle précédent, le rôle des trois couches restantes est comme suit [56][58] :

4ème couche : le rôle de cette couche est la normalisation des degrés d'activation des règles (calculés dans la troisième couche). Chaque neurone i détermine le rapport de α_i par rapport à la somme des degrés d'activation de toutes les règles.

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

$$s_i = p_1 x_1 + q_1 x_2 + c_1 \tag{4.18}$$

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

$$z = \frac{\sum \alpha_i s_i}{\sum \alpha_i} \tag{4.19}$$

4.5 Classification neuro-floue

Lors de la conception d'un classificateur basé sur les règles floues, il est très important de définir la façon de développer un ensemble optimal de fonctions d'appartenance et de règles floues à partir des données d'apprentissage. Ces règles et ces fonctions sont normalement fixées par des experts en se basant sur leurs connaissances et leurs expériences. Malheureusement, cela est très difficile, voire impossible, à cause des ambiguïtés et des complexités des systèmes traités. Par conséquent, il est nécessaire d'introduire des algorithmes d'apprentissage afin de générer et d'ajuster automatiquement les règles floues et les fonctions d'appartenance. Cela pourra être parfaitement réalisé par

les systèmes neuro-flous qui exploitent les capacités d'apprentissage des réseaux de neurones pour l'ajustement des paramètres des SIF.

4.5.1 Architecture

Les systèmes neuro-flous ont été largement utilisés pour effectuer des tâches de classification, et plusieurs modèles ont été proposés [60]…[65]. Un CNF (classificateur neuro-flou) admet généralement une architecture permettant d'incorporer dans sa structure les règles Si-Alors floues de la forme:

$$Si\ \ x_1\ est\ A_1\ et\ \ x_2\ est\ B_2 \quad Alors\ la\ classe\ de\ X\ \ est\ C_1 \tag{4.20}$$

Où x_1 et x_2 sont les variables d'entrée ; A_1 et B_2 sont des termes linguistiques caractérisés par leurs fonctions d'appartenance; X est l'exemple représenté par x_1 et x_2 ; et C_1 est sa classe.

Une autre forme de règles floues très utilisées dans les problèmes de classification comprend dans sa conclusion un degré de confiance (CD, certainly degree) [66][67][68] :

$$Si\ \ x_1\ est\ A_1\ et\ \ x_2\ est\ B_2 \quad Alors\ la\ classe\ de\ X\ est\ C_1\ avec\ CD \tag{4.21}$$

Où : CD est le degré de confiance de cette règle qui représente sa puissance.

La figure (4.11) représente un classificateur neuro-flou avec deux variables d'entrée x_1 et x_2 et deux sorties z_1 et z_2 correspondent à deux classes C_1 et C_2. La première variable d'entrée (x_1) est fuzzifiée en utilisant trois variables linguistiques A_1, A_2 et A_3 tandis que x_2 est fuzzifiée en utilisant B_1 et B_2. Ce modèle se compose de quatre couches permettant d'établir un système de classification basé sur les règles précédentes.

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

2ème couche : Chaque neurone de cette couche correspond à une variable linguistique représentée par une fonction d'appartenance. Dans le cas des fonctions gaussiennes, la sortie de ces neurones est de la forme :

$$\mu_{Ai}(x_n) = exp\left(-\left(\frac{x_n - m_{Ai}}{\sigma_{Ai}}\right)^2\right) \tag{4.22}$$

Où : x_n est la composante du vecteur d'entrée correspondante à ce neurone; m_{Ai} et σ_{Ai} sont les paramètres de la fonction d'appartenance μ_{Ai}.

3ème couche : Les neurones de cette couche établissent le ET flou des signaux arrivant (les sorties des neurones de la deuxième couche). En utilisant le produit, les sorties de ces neurones sont de la forme:

$$y_m = \mu_{Ai}(x_1)\mu_{Bj}(x_2) \tag{4.23}$$

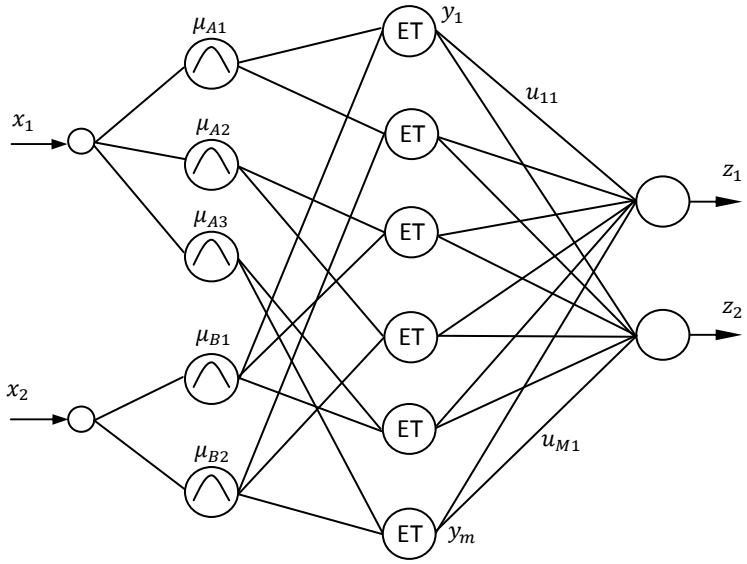


Fig. 4.11. Architecture de base d'un CNF avec deux variables d'entrée et deux sorties

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 flous caractérisant A_i et B_j.

4ème couche : Chaque classe est représentée par un neurone de cette couche. Le nombre de ces neurones est ainsi égal à celui des classes présentes. La sortie du $k^{ème}$ neurone est généralement donnée par [60][61][68]:

$$z_k = \frac{(\sum_{m=1}^{M} u_{mk} y_m)}{(\sum_{m=1}^{M} y_m)} \tag{4.24}$$

Où : u_{mk} est le poids reliant cette le $m^{éme}$ neurone de la deuxième couche avec le $k^{éme}$ neurone de la couche de sortie.

Dans le modèle de Jang [59], il a utilisé dans cette couche des neurones avec des fonctions d'activation sigmoïdale. La sortie du $k^{ème}$ neurone est ainsi donnée par :

$$z_k = \frac{1}{1 + e^{-s_k}} \tag{4.25}$$

Où : $s_k = \sum_{m=1}^{M} u_{mk} y_m$

4.5.2 Apprentissage

La RP a été largement utilisée pour l'apprentissage des classificateurs neuro-flous [59][60][61][62][68]. L'objectif de l'apprentissage est de minimiser l'erreur quadratique totale EQT donnée par :

$$EQT = \sum_{q=1}^{Q} \sum_{j=1}^{J} \left(t_j^{(q)} - z_j^{(q)} \right)^2 \tag{4.26}$$

Où $t_j^{(q)}$ et $z_j^{(q)}$ sont respectivement les $j^{ème}$ composantes de la sortie désirée et la sortie calculée correspondantes au $q^{ème}$ exemple.

L'adaptation du poids u_{mj} à l'itération $(r+1)$ est comme suit :

$$u_{mj}^{(r+1)} = u_{mj}^{(r)} - \eta \frac{\partial E^{(r)}}{\partial u_{mj}}$$

$$= u_{mj}^{(r)} - \eta \left(t_j - z_j \right) y_m \Big/ \sum_{m=1}^{M} y_m \tag{4.27}$$

Dans le modèle de Jang [57], l'adaptation est donnée par :

$$u_{mj}^{(r+1)} = u_{mj}^{(r)} - \eta \left(t_j - z_j \right) \dot{h}\left(s_j \right) y_m \tag{4.28}$$

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

$$m_{Ai}^{(r+1)} = m_{Ai}^{(r)} - \eta_m \frac{\partial E^{(r)}}{\partial m_{Ai}} \tag{4.29}$$

$$m_{Ai}^{(r+1)} = m_{Ai}^{(r)} - \eta_m (t_i - z_i)(x_i - m_{Ai}) \sum_{m=1}^{M(Ai)} \left(u_{mj} - z_i \right) y_m \Big/ (\sigma_{Ai})^2 \sum_{m=1}^{M} y_m \tag{4.30}$$

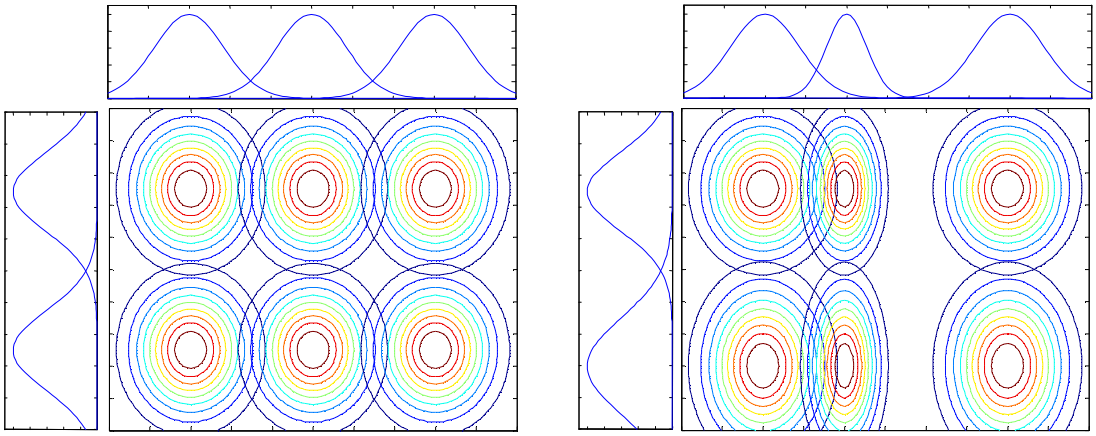
$$\sigma_{Ai}^{(r+1)} = \sigma_{Ai}^{(r)} - \eta_\sigma \frac{\partial E^{(r)}}{\partial \sigma_{Ai}} \tag{4.31}$$

$$\sigma_{Ai}^{(r+1)} = \sigma_{Ai}^{(r)} - \eta_\sigma (t_i - z_i)(x_i - m_{Ai})^2 \sum_{m=1}^{M(Ai)} \left(u_{mj} - z_i \right) y_m \Big/ (\sigma_{Ai})^3 \sum_{m=1}^{M} y_m \tag{4. 32}$$

Où ; η, η_m et η_σ sont les pas d'apprentissage et $M(Ai)$ est le nombre de règles qui dépendent de A_i.

4.5.3 Propriétés des classificateurs neuro-flous entrainés par la RP

L'utilisation de la RP pour l'adaptation des CNF permet une simple formulation de l'algorithme d'apprentissage et l'emploi de différentes fonctions d'appartenance pour chaque variable. L'apprentissage de chacune d'elles s'effectue indépendamment de celui des autres, ce qui permet de bonnes capacités d'apprentissage et de généralisation, même si le nombre d'exemples d'apprentissage n'est pas assez grand [69]. Néanmoins, après la phase d'apprentissage les fonctions d'appartenance pourront changer leurs formes de sorte qu'il soit possible d'avoir des régions dans l'espace des entrées n'étant pas considérées dans les



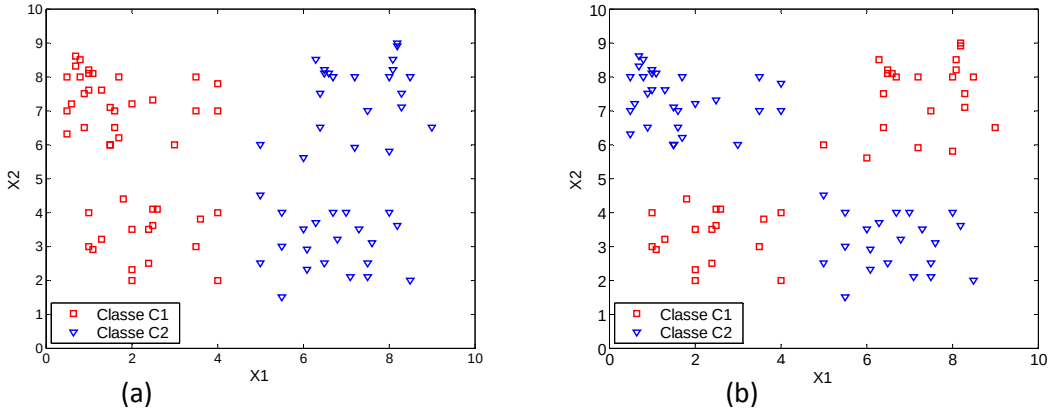
(a) (b)

Fig. 4.12. Exemple de régions formées dans l'espace des entrées avec différentes fonctions d'appartenance
(a) Avant l'apprentissage
(b) Après l'apprentissage

règles d'inférence (figure 4.12). Cela pourra dégrader les capacités de généralisation du classificateur [62][66][68]. Par ailleurs, les problèmes de classification avec un grand nombre de caractéristiques nécessitent un grand nombre de fonctions d'appartenance et de règles, et par conséquent un grand nombre de paramètres à ajuster ce qui entraine une lenteur d'apprentissage du CNF.

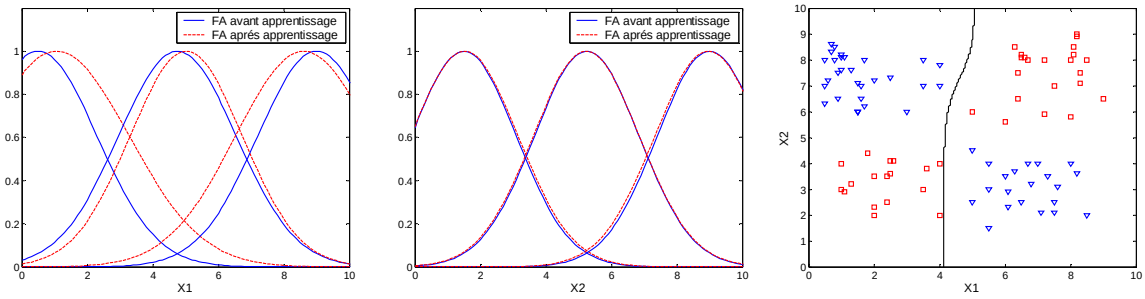
4.5.4 Exemples de classification

Pour apercevoir les capacités de classification en utilisant un système neuro-flou, considérons les deux problèmes de la figure (4.13). Il s'agit de deux problèmes bidimensionnels à deux classes. Les classes du premier problème sont linéairement séparables tandis que celles du deuxième problème ne le sont pas.



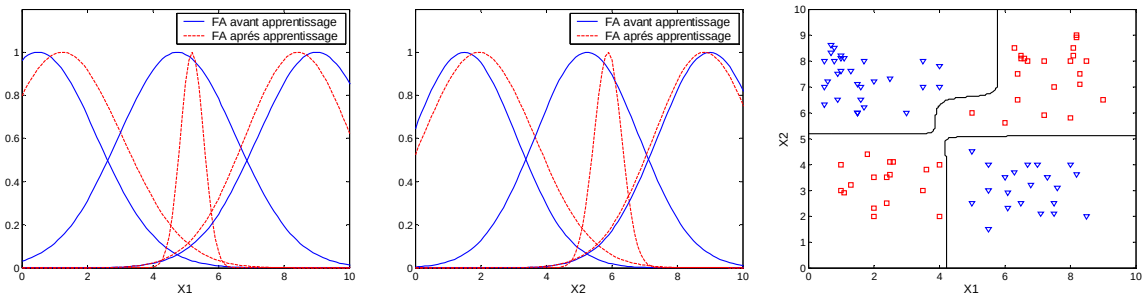
(a) (b)

Fig. 4.13. Problèmes de classification à deux classes.
(a) Exemple 1 : problème linéairement séparable
(b) Exemple 2 : problème non linéairement séparable



(a) (b) (c)

Fig. 4.14. Résultats de classification du problème de la figure (4.13.a)
(a) Fonctions d'appartenance de la première caractéristique (X_1)
(b) Fonctions d'appartenance de la première caractéristique (X_2)
(c) Frontière de décision



(a) (b) (c)

Fig. 4.15. Résultats de classification du problème de la figure (4.13.b) :
(a) Fonctions d'appartenance de la première caractéristique (X_1)
(b) Fonctions d'appartenance de la première caractéristique (X_2)
(c) Frontière de décision

Nous utilisons un classificateur neuro-flou avec deux variables d'entrée (x_1 et x_2). Pour la fuzzification, nous employons trois termes linguistiques pour chaque variable. Les fonctions d'appartenance correspondantes sont des gaussiennes ayant comme centres : le minimum, la moyenne et le maximum de chacune des variables d'entrée. Le classificateur utilisé a l'architecture suivante : Une première couche ayant 2 neurones : chacun d'eux correspond une variable d'entrée. Une deuxième couche avec 6 neurones dans lesquels les trois premiers correspondent à x_1 et les autres pour x_2. La troisième couche comprend 9 neurones. Finalement, la couche de sortie comporte deux neurones dont chacun correspond à une classe. Les gains d'apprentissage utilisés sont : $\eta = 0.5$, $\eta_m = 0.02$ et $\eta_\sigma = 0.02$

La figure (4.14) illustre la forme des fonctions d'appartenance avant et après l'apprentissage ainsi que la frontière de décision obtenue dans le cas du premier problème. Nous constatons que le classificateur parvient à bien classifier tous les exemples d'apprentissage avec des petites modifications des fonctions d'appartenance. De même, la figure (4.15) illustre la forme des fonctions d'appartenance et la frontière de décision obtenue pour le deuxième problème. Le classificateur parvient également à bien classifier tous les exemples

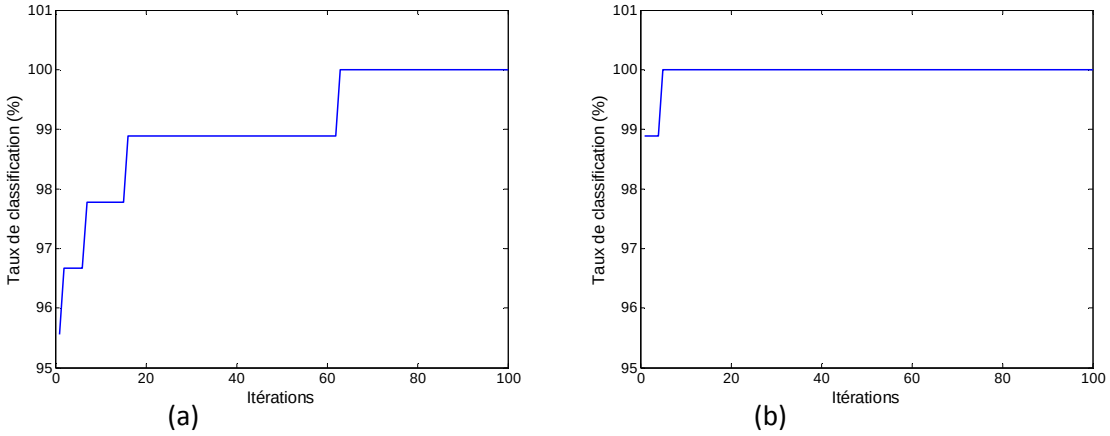


Fig. 4.16. Evolution du taux de classification durant l'apprentissage :
(a) Premier problème (fig. 4.13.a)
(b) Deuxième problème (fig. 4.13.b)

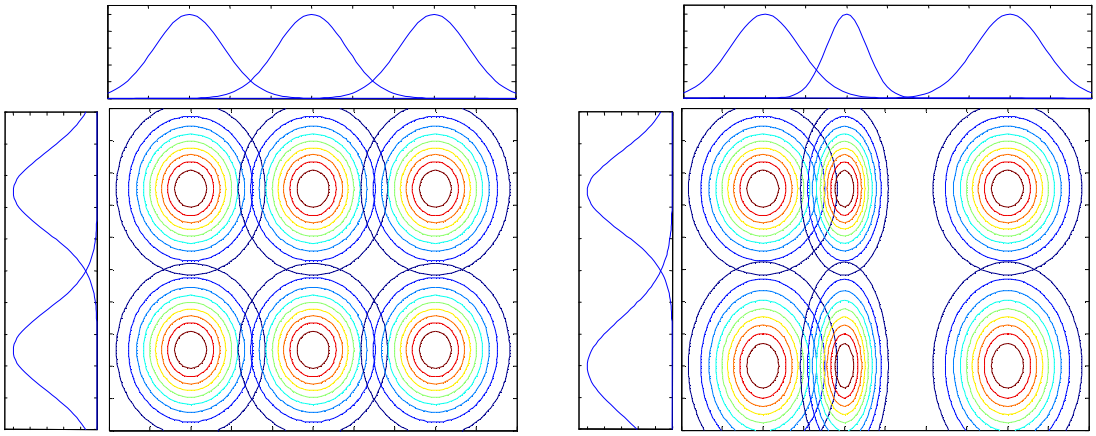
d'apprentissage, mais cette fois avec des modifications considérables des fonctions d'appartenance.

D'après les deux exemples, nous constatons que le classificateur neuro-flou parvient à résoudre les deux problèmes avec succès. Néanmoins, nous remarquons que la tâche est plus simple dans le premier exemple où les classes sont linéairement séparables. La classification est réalisée à la fois avec de simples modifications des fonctions d'appartenance et avec moins d'itérations. Ceci est illustré par les graphes de la figure (4.16) ; le classificateur nécessite seulement 5 itérations pour avoir un taux de classification égale à 100% dans le premier problème, mais pour le deuxième ce n'est possible qu'après plus de 60 itérations.

4.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la classification par les systèmes neuro-flous en mettant l'accent sur leurs avantages dans le domaine de la classification. Ces modèles permettent, en combinant les réseaux de neurones et les SIF, la mise en œuvre de classificateurs incorporant les différentes étapes des SIF dans une structure neuronale interprétable. Les capacités d'apprentissage des RNA sont exploitées pour le réglage des poids et des fonctions d'appartenance ce qui permet d'avoir de bonnes performances de classification.

Toutefois, nous avons remarqué que ces systèmes souffrent toujours de la lenteur d'apprentissage de la RP, notamment dans les problèmes de grande dimension et les problèmes à classes non linéairement séparables. Cela nous a motivé à introduire la deuxième application de la méthode proposée qui sera présentée dans la partie suivante.



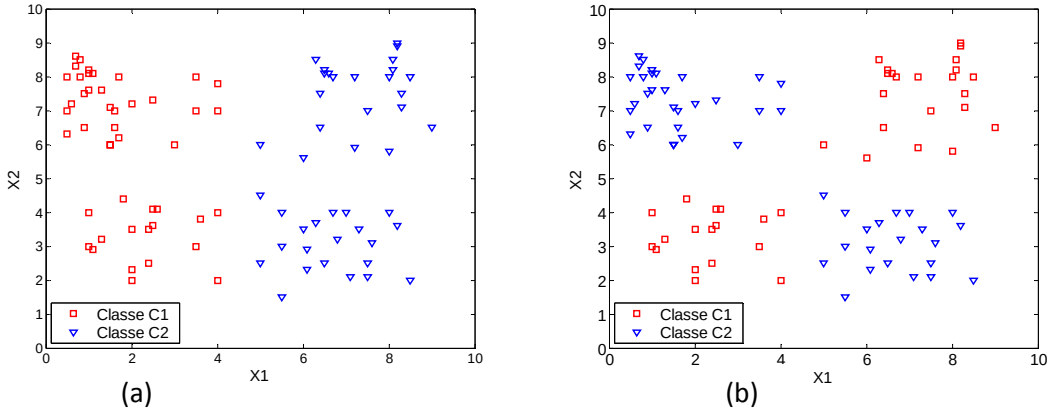
(a) (b)

Fig. 4.12. Exemple de régions formées dans l'espace des entrées avec différentes fonctions d'appartenance
(a) Avant l'apprentissage
(b) Après l'apprentissage

règles d'inférence (figure 4.12). Cela pourra dégrader les capacités de généralisation du classificateur [62][66][68]. Par ailleurs, les problèmes de classification avec un grand nombre de caractéristiques nécessitent un grand nombre de fonctions d'appartenance et de règles, et par conséquent un grand nombre de paramètres à ajuster ce qui entraine une lenteur d'apprentissage du CNF.

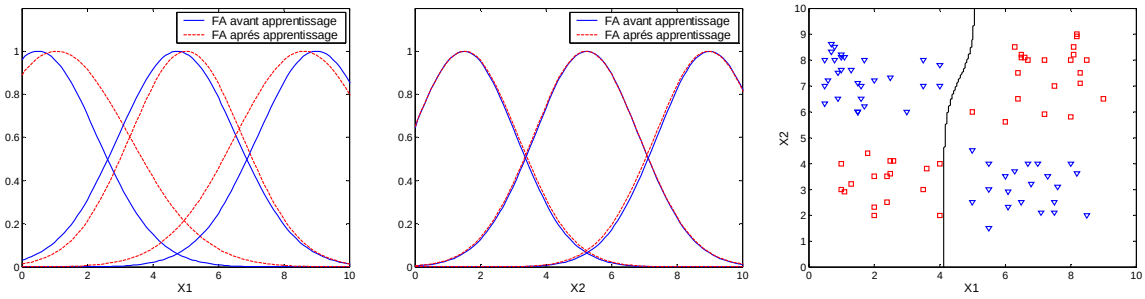
4.5.4 Exemples de classification

Pour apercevoir les capacités de classification en utilisant un système neuro-flou, considérons les deux problèmes de la figure (4.13). Il s'agit de deux problèmes bidimensionnels à deux classes. Les classes du premier problème sont linéairement séparables tandis que celles du deuxième problème ne le sont pas.



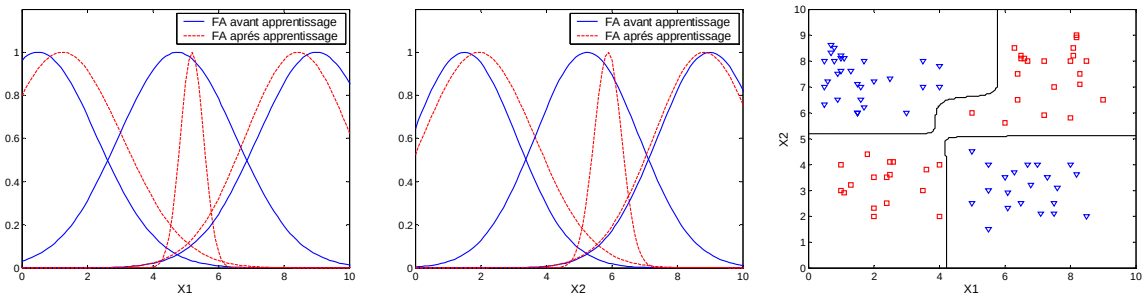
(a) (b)

Fig. 4.13. Problèmes de classification à deux classes.
(a) Exemple 1 : problème linéairement séparable
(b) Exemple 2 : problème non linéairement séparable



(a) (b) (c)

Fig. 4.14. Résultats de classification du problème de la figure (4.13.a)
(a) Fonctions d'appartenance de la première caractéristique (X_1)
(b) Fonctions d'appartenance de la première caractéristique (X_2)
(c) Frontière de décision



(a) (b) (c)

Fig. 4.15. Résultats de classification du problème de la figure (4.13.b) :
(a) Fonctions d'appartenance de la première caractéristique (X_1)
(b) Fonctions d'appartenance de la première caractéristique (X_2)
(c) Frontière de décision

Nous utilisons un classificateur neuro-flou avec deux variables d'entrée (x_1 et x_2). Pour la fuzzification, nous employons trois termes linguistiques pour chaque variable. Les fonctions d'appartenance correspondantes sont des gaussiennes ayant comme centres : le minimum, la moyenne et le maximum de chacune des variables d'entrée. Le classificateur utilisé a l'architecture suivante : Une première couche ayant 2 neurones : chacun d'eux correspond une variable d'entrée. Une deuxième couche avec 6 neurones dans lesquels les trois premiers correspondent à x_1 et les autres pour x_2. La troisième couche comprend 9 neurones. Finalement, la couche de sortie comporte deux neurones dont chacun correspond à une classe. Les gains d'apprentissage utilisés sont : $\eta = 0.5$, $\eta_m = 0.02$ et $\eta_\sigma = 0.02$

La figure (4.14) illustre la forme des fonctions d'appartenance avant et après l'apprentissage ainsi que la frontière de décision obtenue dans le cas du premier problème. Nous constatons que le classificateur parvient à bien classifier tous les exemples d'apprentissage avec des petites modifications des fonctions d'appartenance. De même, la figure (4.15) illustre la forme des fonctions d'appartenance et la frontière de décision obtenue pour le deuxième problème. Le classificateur parvient également à bien classifier tous les exemples

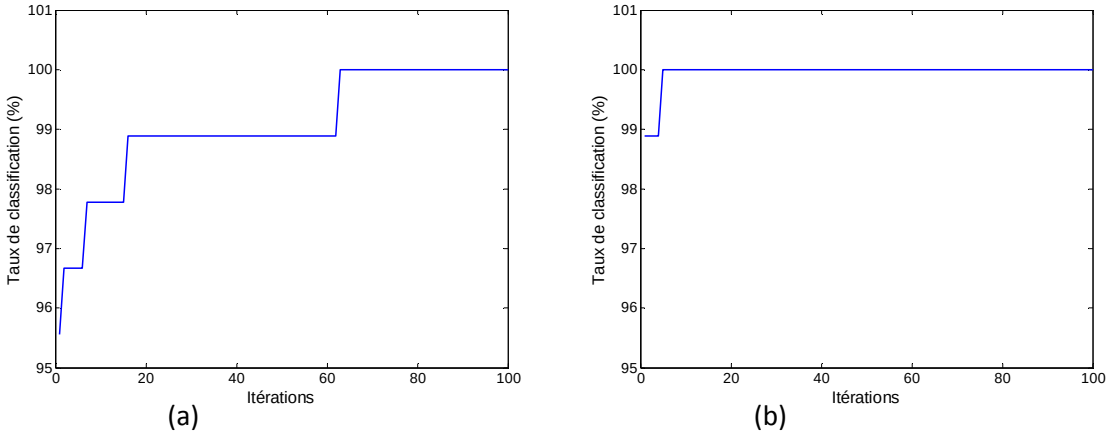


Fig. 4.16. Evolution du taux de classification durant l'apprentissage :
(a) Premier problème (fig. 4.13.a)
(b) Deuxième problème (fig. 4.13.b)

d'apprentissage, mais cette fois avec des modifications considérables des fonctions d'appartenance.

D'après les deux exemples, nous constatons que le classificateur neuro-flou parvient à résoudre les deux problèmes avec succès. Néanmoins, nous remarquons que la tâche est plus simple dans le premier exemple où les classes sont linéairement séparables. La classification est réalisée à la fois avec de simples modifications des fonctions d'appartenance et avec moins d'itérations. Ceci est illustré par les graphes de la figure (4.16) ; le classificateur nécessite seulement 5 itérations pour avoir un taux de classification égale à 100% dans le premier problème, mais pour le deuxième ce n'est possible qu'après plus de 60 itérations.

4.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la classification par les systèmes neuro-flous en mettant l'accent sur leurs avantages dans le domaine de la classification. Ces modèles permettent, en combinant les réseaux de neurones et les SIF, la mise en œuvre de classificateurs incorporant les différentes étapes des SIF dans une structure neuronale interprétable. Les capacités d'apprentissage des RNA sont exploitées pour le réglage des poids et des fonctions d'appartenance ce qui permet d'avoir de bonnes performances de classification.

Toutefois, nous avons remarqué que ces systèmes souffrent toujours de la lenteur d'apprentissage de la RP, notamment dans les problèmes de grande dimension et les problèmes à classes non linéairement séparables. Cela nous a motivé à introduire la deuxième application de la méthode proposée qui sera présentée dans la partie suivante.

Partie III

Chapitre 5 : La classification étiquetée

Chapitre 6 : Les systèmes étiquetés multi-réseaux de neurones

Chapitre 7 : Tests et résultats

Chapitre 5

La classification étiquetée

Dans ce chapitre nous introduisons une nouvelle méthode de classification, appelée *la classification étiquetée,* qui vise l'amélioration des performances des classificateurs entrainés par la RP. En nous basant sur cette méthode, nous proposons trois modèles de classification dont les deux premiers seront présentés dans ce chapitre. Il s'agit du *MLP étiqueté*, qui constitue l'application principale de la méthode proposée, et le *classificateur neuro-flou étiqueté*.

5.1 Introduction

La RP est un algorithme très utilisé dans l'apprentissage des systèmes neuronaux et neuro-flous et son importance a été largement évoquée. En revanche, ses problèmes ont fait l'objet de plusieurs travaux et l'amélioration de ses performances a été abondamment traitée. Dans ce contexte, certains auteurs se sont intéressés à l'initialisation des poids [35]…[41], d'autres ont opté pour l'introduction d'un ajustement itératif du pas d'apprentissage [42] ou des paramètres des fonctions d'activation[43]…[45], d'autres auteurs ont ajouté des termes à l'équation de mise à jour [46][47] …etc.

Notre travail se situe aussi dans cet axe de recherche, mais notre approche se diffère de ces méthodes du fait qu'elle ne requiert aucune modification de l'algorithme d'apprentissage. La méthode proposée s'articule essentiellement sur la modification de la représentation des exemples d'apprentissage, par l'ajout des *étiquettes*, et la réalisation de plusieurs tests pour classer les nouveaux exemples.

5.2 Fondement de la classification étiquetée

5.2.1 Idée de base

La lenteur de la RP est généralement expliquée par le fait que c'est une méthode de décente de gradient. Toutefois, une importante raison de la lenteur de convergence de la RP est l'occurrence du phénomène de la saturation prématurée des neurones [43][44][46][70]. En effet, lorsqu'une sigmoïde comporte une pente près de zéros (figure 5.1), les poids risquent d'entrer dans des régions de saturation de l'espace des poids [4]. Dans de telles situations, l'incrémentation d'un poids reste petite même si l'erreur est relativement grande. Cela est

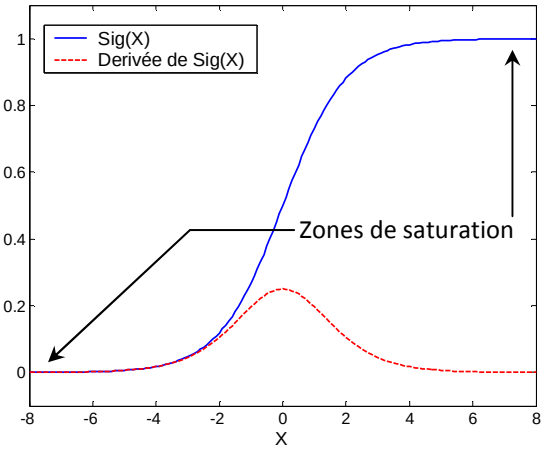


Fig. 5.1. Zones de saturation d'une fonction sigmoïde.

caractérisé par la présence des plateaux et des minima locaux dans la surface d'erreur. Le réseau prendra ainsi un grand nombre d'itérations pour s'en sortir, et parfois ne jamais s'échapper.

La mise en œuvre de la classification étiquetée tire profit du fait que les problèmes précédents apparaissent moins dans les problèmes où les classes sont linéairement séparables. L'idée de base est de rendre les exemples d'apprentissage linéairement séparables par l'ajout d'une caractéristique additionnelle : les *étiquettes*. Ces dernières doivent être identiques pour les exemples appartenant aux mêmes classes afin d'assurer leur séparation linéaire. Après la phase d'apprentissage, chaque nouvel exemple sera classifié en se basant sur les sorties du classificateur obtenues avec toutes les étiquettes.

5.2.2 Processus

Rappelons que le rôle d'un classificateur est de fournir une sortie $Z(z_1\ z_2\ \dots\ z_K)$ pour chaque vecteur caractéristique $X(x_1\ x_2\ \dots\ x_N)$ présenté à l'entrée. La sortie Z correspond à la classe C_i de cet exemple. L'application de la classification étiquetée avec un classificateur

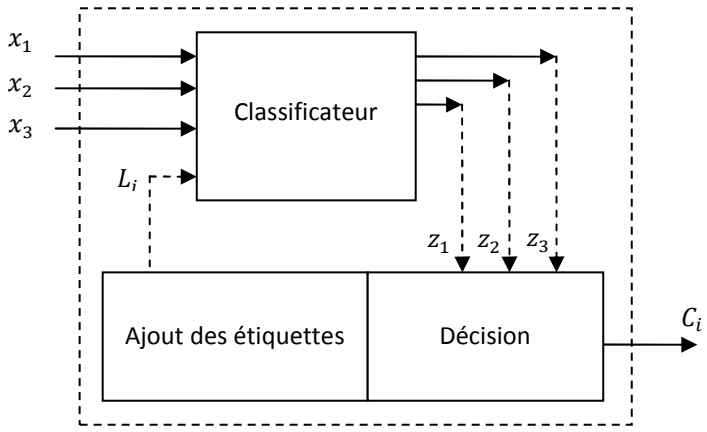


Fig. 5.2. Structure d'un classificateur étiqueté.

donné consiste à ajouter les étiquettes à son entrée et à réaliser une décision en se basant sur ses sorties. La figure (5.2) représente un classificateur étiqueté avec trois entrées et trois sorties.

La classification étiquetée s'effectue en deux phases (figure 5.3):

- Ajout des étiquettes et réalisation de l'apprentissage selon deux modes : apprentissage complet (full training) et apprentissage simple (simple training).
- Réalisation des tests avec toutes les étiquettes pour classifier chaque nouvel exemple.

Pour chaque classe C_k on fait correspondre une étiquette L_k et tout exemple d'apprentissage $X(x_1\ x_2\ \dots\ x_N)$ appartenant à C_k sera représenté par $X(x_1\ x_2\ \dots\ x_N\ L_k)$. La représentation de tous les exemples d'apprentissage sera modifiée de la même façon. Après la phase d'apprentissage, chaque exemple de test sera testé avec toutes les étiquettes et il sera classifié selon la règle de décision suivante :

$$F(X) = \arg\min_k \{Er_k(X)\} \tag{5.1}$$

Où : Er_k est l'erreur quadratique entre la sortie désirée T_k correspondante à la classe C_k et la sortie calculée $Z\left(X^{(q)}, L_k\right)$ en utilisant l'étiquette L_k. Er_k est donnée par :

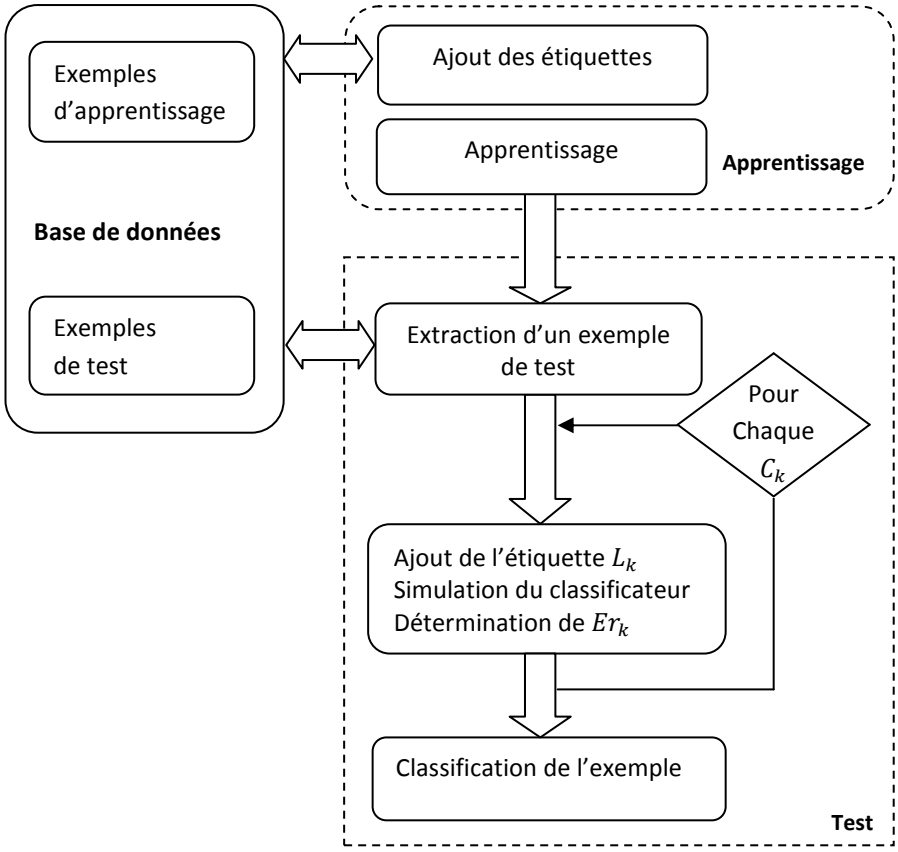


Fig. 5.3. Processus de la classification étiquetée.

$$Er_k(X) = \|T_k - Z(X, L_k)\|^2 \tag{5.2}$$

5.3 Apprentissage dans la classification étiquetée

L'apprentissage dans la classification étiquetée peut être réalisé selon deux modes.

5.3.1 Premier mode : Apprentissage simple

Ce mode est basé sur l'ajout d'une seule étiquette à chaque exemple d'apprentissage. On ajoute à chaque exemple ($X^{(q)}$) appartenant à la classe C_i l'étiquette L_i. Le processus d'ajout des étiquettes s'effectue avant la phase d'apprentissage. La fonction coût est l'erreur quadratique totale donnée par :

$$E_{Simple} = \sum_{q=1}^{Q} \left\| T^{(q)} - Z\left(X^{(q)}, L_i\right) \right\|^2 \tag{5.3}$$

Où :$T^{(q)}$et $Z\left(X^{(q)}, L_i\right)$ sont respectivement la sortie désirée et la sortie calculée correspondantes au $q^{ème}$ exemple et Q est le nombre total d'exemples d'apprentissage.

5.3.2 Deuxième mode : Apprentissage complet

Dans ce mode, l'apprentissage est effectué en minimisant l'erreur entre la sortie désirée et les sorties du classificateur obtenues avec toutes les étiquettes (figure 5.4). C'est-à-dire,

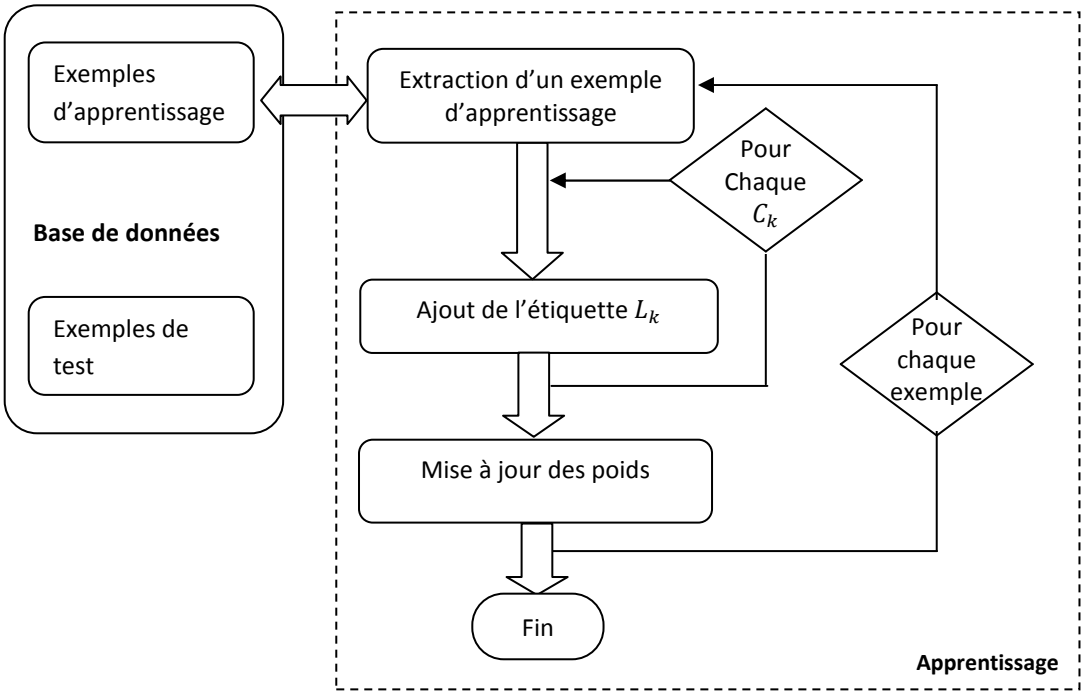


Fig. 5.4. Processus de l'apprentissage complet.

pour chaque exemple présenté ; le classificateur devra être simulé et mis à jour en utilisant toutes les étiquettes. La fonction coût est :

$$E_{Full} = \sum_{q=1}^{Q} \sum_{k=1}^{K} \left\{ \left\| T^{(q)} - Z\left(X^{(q)}, L_k\right) \right\|^2 \right\} \tag{5.4}$$

Où :$T^{(q)}$et $Z\left(X^{(q)}, L_k\right)$ sont respectivement la sortie désirée et la sortie calculée correspondante au $q^{ème}$ exemple ; Q et le nombre total des exemples d'apprentissage et K est le nombre de classes .

5.4 Le perceptron multicouche étiqueté

5.4.1 Motivations

Le perceptron multicouche est le modèle neuronal le plus employé et le plus étudié. Ce réseau constitue également l'application principale de la classification étiquetée. L'application de la classification étiquetée avec ce modèle repose sur les motivations suivantes :

- L'apprentissage d'un MLP entrainé par la RP est lent et il est très important de l'accélérer.
- Après l'étape d'apprentissage, le MLP permet une classification rapide et par conséquent il permet une exécution rapide de la phase de reconnaissance de la classification étiquetée qui nécessite plusieurs exécutions.
- Les sorties du MLP sont des estimés des probabilités a posteriori et ils peuvent être utilisés dans d'autres étapes d'aide à la décision.

5.4.2 Architecture

L'application de la classification étiquetée avec le MLP consiste à l'entraîner en utilisant des exemples ayant une caractéristique additionnelle (les étiquettes). Cela nécessite donc de réserver un neurone de la couche d'entrée à cette caractéristique (figure 5.4). La caractéristique ajoutée sera traitée comme les autres caractéristiques et le vecteur d'entrée $X(x_1\ x_2\ \dots\ x_N)$ sera remplacé par $X(x_1\ x_2\ \dots\ x_N\ L_k)$. La sortie du $m^{ème}$ neurone de la couche cachée sera :

$$y_m = h\left(\sum_{m=1}^{M} x_n w_{nm} + L_k w_{N+1,m}\right) \tag{5.5}$$

La sortie du $j^{ème}$ neurone de la couche de sortie sera :

$$z_j = g\left(\sum_{m=1}^{M} \left\{ h\left(\sum_{m=1}^{M} x_n w_{nm}\right) u_{mj} \right\}\right) + g\left(\sum_{m=1}^{M} \left\{ h\left(L_k w_{N+1,m}\right) u_{mj} \right\}\right) \tag{5.6}$$

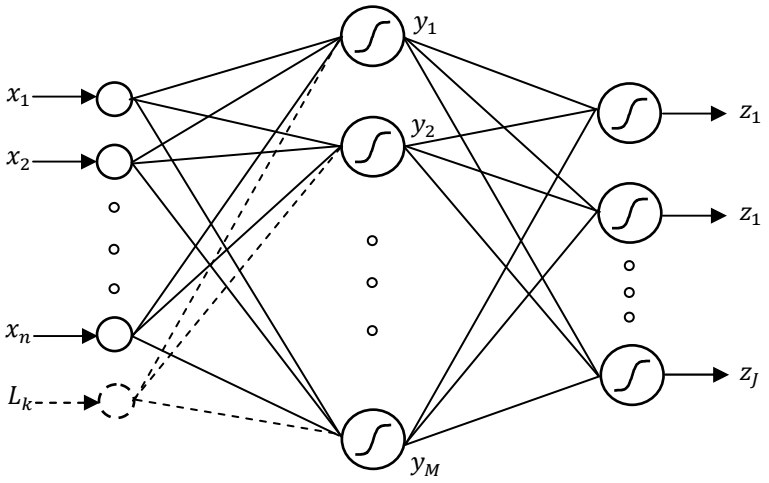


Fig.5.5. Un perceptron multicouche étiqueté contenant N neurone d'entrée, M neurones cachés et J neurones de sortie.

Le deuxième terme de l'équation précédente montre l'impact de la caractéristique ajoutée sur la sortie du réseau.

5.4.3 Surface d'erreur

Pour apercevoir la différence entre la surface d'erreur d'un MLP simple et celle d'un MLP étiqueté, considérons de nouveau le problème unidimensionnel traité dans le chapitre 3.

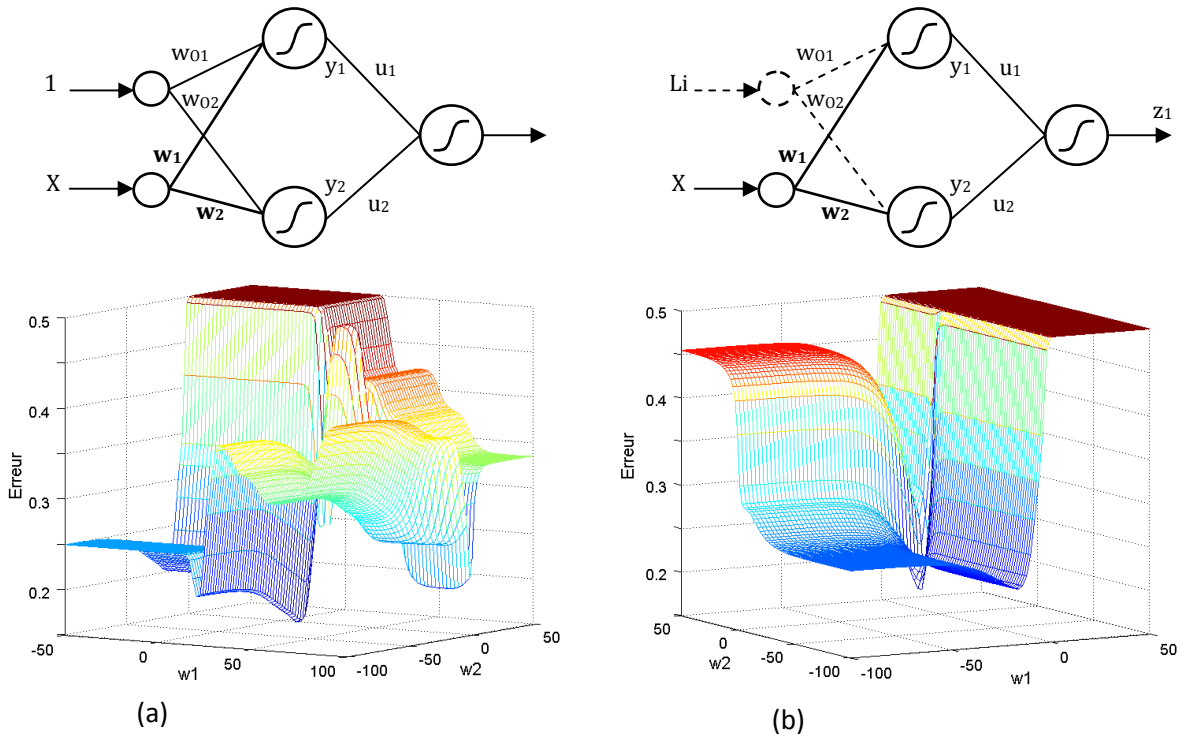


Fig.5.6. Surface d'erreur en fonction des poids dans un problème de classification unidimensionnel non linéairement séparable
a) MLP simple
b) MLP étiqueté

Nous modifions la représentation des exemples de ce problème par l'ajout des étiquettes : $L_1 = 4.5$ et $L_2 = 5.5$ aux exemples appartenant respectivement à la classe C_1 et C_2.

La figure (5.6) illustre les surfaces d'erreur en fonction des poids w_1 et w_2 pour un MLP simple et un MLP étiqueté ayant les mêmes architectures ; 2 neurones à l'entrée, 2 neurones cachés et un neurone de sortie.

Nous remarquons que la surface d'erreur correspondante au MLP étiqueté (figure 5.6.b) est plus simple et qu'elle n'admet qu'un seul minimum. En effet, l'objectif principal de rendre le problème linéairement séparable par l'ajout des étiquettes est d'éviter au maximum les problèmes que peut rencontrer le MLP en parcourant la surface d'erreur. Ces problèmes se caractérisent essentiellement par les minimas locaux et les plateaux. L'apprentissage d'un réseau étiqueté s'avère ainsi moins compliqué.

5.4.4 Apprentissage du MLP étiqueté

L'apprentissage du MLP étiqueté peut être effectué selon les deux modes de la classification étiquetée :

Apprentissage simple

L'apprentissage selon ce mode consiste à ajouter les étiquettes une seule fois avant l'apprentissage. Cette caractéristique est traitée comme tous les autres caractéristiques et l'apprentissage s'effectue en minimisant l'erreur quadratique totale donnée par :

$$E_{Simple} = \sum_{q=1}^{Q} \sum_{j=1}^{J} \left(t_j^{(q)} - z_k\left(X^{(q)}, L_k\right) \right)^2 \tag{5.7}$$

Où : $t_j^{(q)}$et $z_j^{(q)}$ sont respectivement les $j^{ème}$ composantes de la sortie désirée et la sortie calculée correspondantes au $q^{ème}$ exemple et J est le nombre de neurones de sortie.

La mise à jour des poids s'effectue par :

$$u_{mj}^{(r+1)} = u_{mj}^{(r)} + \eta_1 \left(t_j - z_j\right) \grave{g}\left(s_j\right) y_m \tag{5.8}$$

$$w_{nm}^{(r+1)} = w_{nm}^{(r)} + \eta_2 \left(\sum_{j=1}^{J} \left(t_j - z_j\right) \grave{g}\left(s_j\right) u_{mj} \right) \acute{h}(r_m) x_n \tag{5.9}$$

Où : $\grave{g}\left(s_j\right)$ et $\acute{h}(r_m)$ sont respectivement les dérivées des fonctions d'activation des neurones cachés et des neurones de sortie.

L'algorithme du MLP étiqueté avec apprentissage simple est donné par:

1. **begin**
2. Initialisation : $M, \varepsilon, K, \eta, R, r \leftarrow 0$.
3. Initialisation : $L_k, k = 1 \ldots K$.
4. **do** $q \leftarrow q + 1$
5. $k \leftarrow 0$
6. **do** $k \leftarrow k + 1$
7. **If** class($X^{(q)}$) = k
8. **do** $X^{(q)} \leftarrow [X^{(q)}; L_k\ ]$
9. **Until** $k = K$
10. **Until** $q = Q$
11. **do** $r \leftarrow r + 1$
12. $q \leftarrow 0$;
13. **do** $q \leftarrow q + 1$
14. Select $X^{(q)}$
15. $\Delta w_{nm} \leftarrow -\eta \partial E / \partial w_{nm}$
1. $\Delta u_{mj} \leftarrow -\eta \partial E / \partial u_{mj}$
2. $u_{mj} \leftarrow u_{mj} + \Delta u_{mj}$
3. $w_{nm} \leftarrow w_{nm} + \Delta w_{nm}$
4. **Until** $q = Q$
5. **Until** $r = R$ **or** $E \leq \varepsilon$
6. **end**

Apprentissage complet

L'apprentissage selon ce mode consiste à entrainer le MLP en utilisant toutes les étiquettes avec chaque exemple d'apprentissage. Cette caractéristique est également traitée comme toutes les autres caractéristiques et l'apprentissage s'effectue en minimisant l'erreur quadratique totale donnée par :

$$E_{Full} = \sum_{q=1}^{Q} \sum_{k=1}^{K} \left\{ \sum_{j=1}^{J} \left(t_j^{(q)} - z_j\left(X^{(q)}, L_k\right) \right)^2 \right\} \tag{5.10}$$

Où : $t_j^{(q)}$et $z_j\left(X^{(q)}, L_k\right)$ sont respectivement les $j^{ème}$ composantes de la sortie désirée et la sortie calculée correspondante au $q^{ème}$ exemple ; Q et le nombre des exemples d'apprentissage et K est le nombre de classes.

Les équations de la mise à jour des poids sont les mêmes que Eq. (5.8) et Eq. (5.9).

L'algorithme d'apprentissage du MLP étiqueté avec apprentissage complet est le suivant:

1. **begin**
2. Initialisation : $M, \varepsilon, K, \eta, R, r \leftarrow 0$.
3. Initialisation : $L_k, k = 1 \dots K$.
4. **do** $r \leftarrow r + 1$
5. $q \leftarrow 0$
6. **do** $q \leftarrow q + 1$
7. Select $X^{(q)}$; $k \leftarrow 0$
8. **do** $k \leftarrow k + 1$
9. $X^{(q)} \leftarrow [X^{(q)}; L_k\ ]$
10. $\Delta w_{nm} \leftarrow -\eta \partial(E)/\partial w_{nm}$
11. $\Delta u_{mj} \leftarrow -\ \eta \partial(E)/\partial \Delta u_{mj}$
12. $w_{nm} \leftarrow w_{nm} +\ \Delta w_{nm}$
13. $u_{mj} \leftarrow u_{mj} +\ \Delta u_{mj}$
14. **Until** $k = K$
15. **Until** $q = Q$
16. **Until** $r = R$ **or** $E \leq \varepsilon$
17. **end**

5.4.5 Interprétation des sorties du MLP étiqueté

Dans cette partie nous montrons que les sorties d'un MLP étiqueté peuvent également estimer les probabilités a posteriori. A cet effet, nous suivons la même procédure utilisée dans [1][2][32][33] et qui a été présentée dans le paragraphe ($3.7) . Pour un problème de classification à K classes, considérons un MLP étiqueté avec K sorties permettant de coder les sorties désirées comme suit :

$$t_k = \begin{cases} 1\ si\ X \in C_k \\ 0\ autrement \end{cases} \tag{5.11}$$

Chaque sortie z_k du réseau sera en fonction des poids du réseau, des caractéristiques initiales et aussi de l'étiquette ($z_k(X, W, L_k)$). La fonction coût correspondante à chaque sortie z_k est donnée par:

$$F(W) = [z_k(X, W, L_k) - t_k]^2 \tag{5.12}$$

En suivant les mêmes étapes du paragraphe ($3.7), nous obtiendrons :

$$\begin{aligned} \lim_{Q\to\infty} \frac{1}{Q} F(W) = &\int [z_k(X, W, L_k) - P(C_k/X\ , L_k)]^2\ p(X)dx \\ &+ \int P(C_k/X\ , L_k) P(C_{i\neq k}/X\ , L_k)\ p(X)dx \end{aligned} \tag{5.13}$$

Dans l'équation précédente (Eq. 5.13) le deuxième terme est indépendant du réseau. La RP modifie donc les poids du MLP étiqueté pour minimiser :

$$\int [z_k(X, W, L_k) - P(C_k/X\,, L_k)]^2\, p(X) dx \tag{5.14}$$

Du fait que ceci est vrai pour chaque classe C_k, la RP minimise la somme suivante :

$$\sum_{k=1}^{K} \int [z_k(X, W, L_k) - P(C_k/X\,, L_k)]^2\, p(X) dx \tag{5.15}$$

A la limite d'une infinité de données, les sorties d'un MLP étiqueté permettront d'approximer les probabilités a posteriori :

$$z_k(X, W, L_k) \cong P(C_k/X\,, L_k) \tag{5.16}$$

La sortie z_k d'un MLP étiqueté peut être considérée comme un estimé de la probabilité a posteriori d'appartenance à la classe C_k étant donnée l'étiquette L_k.

Comme il a été mentionné dans le paragraphe (\$3.7), le fait que les sorties du MLP constituent les probabilités a posteriori ne serait garanti que si le nombre de neurones caché est suffisant, que le critère à minimiser est quadratique et que le réseau ne tombe pas dans les minima locaux. L'application de la classification étiquetée ne modifie pas ces critères puisque cette méthode ne change pas la structure et la stratégie d'apprentissage du classificateur utilisé.

5.4.6 Choix des étiquettes

La classification étiquetée se base essentiellement sur l'ajout des étiquettes, et par conséquent leurs valeurs influencent directement sur performances du MLP étiqueté. 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, tandis qu'un choix de valeurs proches permet de le garder, mais n'accélère pas l'apprentissage. Par ailleurs, les valeurs des étiquettes ne doivent pas être loin des valeurs des autres caractéristiques (qui s'échelonnent généralement entre 0 et 1). On propose donc de choisir un ensemble d'étiquettes autour de 0.5 avec un petit écart (σ) entre eux. Par exemple pour deux classes les étiquettes 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 $L_3 = 0.5 + \sigma/2$, et ainsi de suite.

5.4.7 Exemple illustratif

Pour analyser le fonctionnement du MLP étiqueté, nous considérons l'exemple synthétique de la figure (5.7). C'est un problème bidimensionnel à deux classes (C_1et C_2) non linéairement séparables.

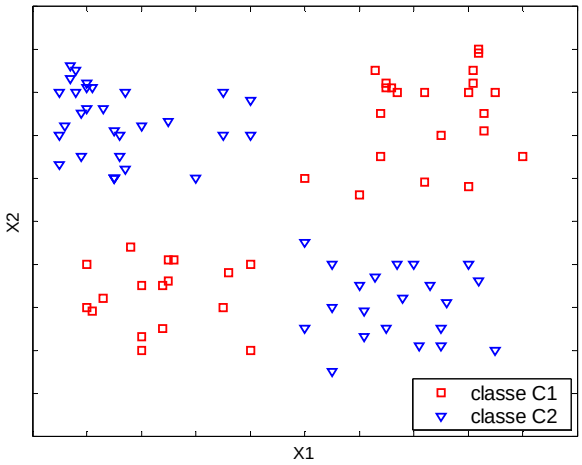
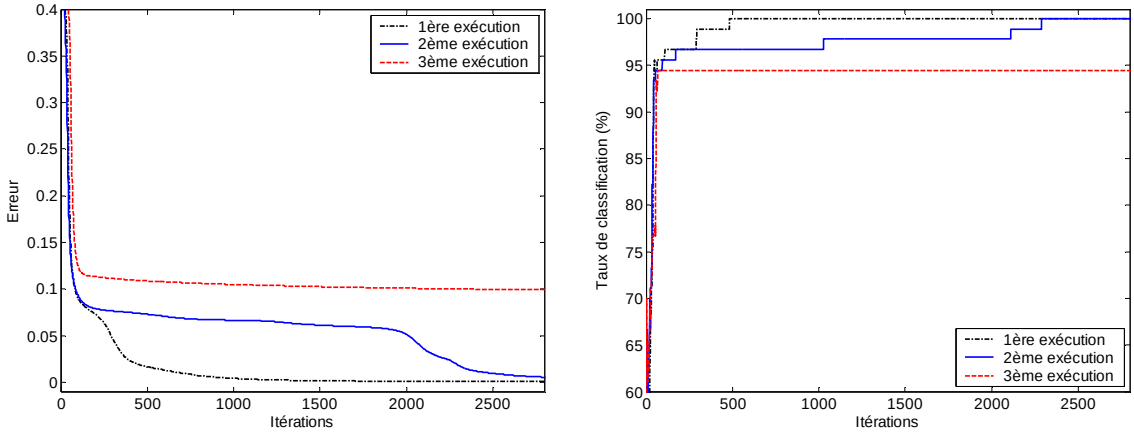


Fig. 5.7. Un problème de classification à deux classes non linéairement séparables

Nous utilisons un MLP avec trois neurones à l'entrée, quatre neurones cachés et deux neurones de sortie. Dans ce genre de problème, le MLP entrainé par la RP risque souvent d'être prisonnier dans des régions de saturations. La figure (5.8) représente l'évolution de l'erreur et du taux d'apprentissage au cours de l'apprentissage, et illustre des exemples des trois cas possibles. La première exécution correspond au cas de convergence. La deuxième exécution représente un exemple où les poids entrent dans une région de saturation et dans lequel le MLP prend un grand nombre d'itérations pour s'en sortir. La troisième exécution représente le cas où le MLP n'arrive pas à s'échapper.

Bien que l'objectif de la classification étiquetée consiste à corriger les problèmes de la RP, nous ne nous limitons pas à traiter les cas où le MLP n'arrive pas à classifier les exemples de ce problème. Nous considérons également le cas de convergence du MLP afin d'analyser les



(a) (b)

Fig. 5.8. Résultats de la classification du problème de la figure (5.7) avec un MLP pour trois exécutions différentes :
(a) Evolution de l'erreur
(b) Evolution du taux de classification

performances du MLP étiqueté même dans le cas où le MLP parvient à résoudre le problème traité.

a) Cas de convergence retardée

Pour évaluer les améliorations apportées par la classification étiquetée, nous traitons essentiellement les cas où le MLP échoue à résoudre ce problème. Nous considérons d'abord le cas de la deuxième exécution où le MLP prend du temps pour sortir des régions de saturation. Dans ce cas (figure 5.8.b) le MLP n'arrive pas à avoir un taux de classification égal à 100% qu'après 2200 itérations, tandis que dans la première exécution il y arrive après seulement 500 itérations.

Les réponses des neurones cachés et des neurones de sortie ainsi que les frontières de décision générées après 1000 itérations sont représentées sur la figure (5.9). Les frontières de décision obtenues ne permettent pas une classification convenable de ces exemples ; on remarque la saturation des troisième et quatrième neurones cachés.

Nous utilisons un MLP étiqueté ayant la même architecture ; 2 neurones d'entrée (en plus d'un neurone réservé aux étiquettes), 4 neurones cachés et 2 neurones de sortie. Nous employons les mêmes poids initiaux et les mêmes paramètres d'apprentissage (pas d'apprentissage et moment). L'évolution des taux de classification correspondants aux MLP simple et MLP étiqueté est illustrée sur la figure (5.10). Le MLP étiqueté entrainé par le premier mode parvient à bien classifier tous les exemples d'apprentissage dans moins de 500 itérations tandis que dans le cas du deuxième mode il y parvient après environ 200 itérations seulement.

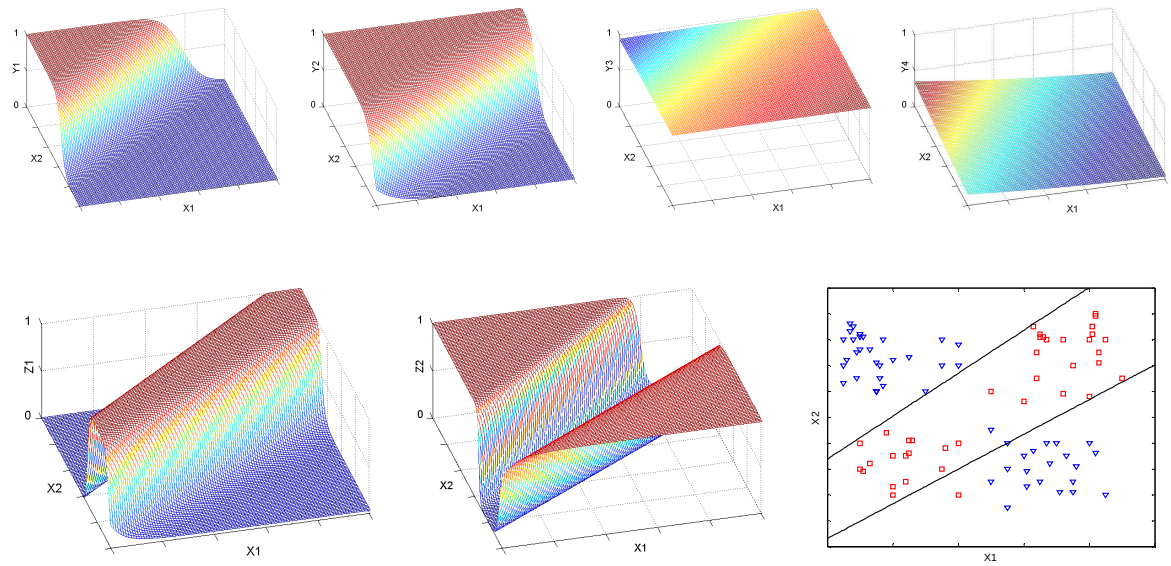


Fig. 5.9. Résultats de classification du problème de la figure (5.7) avec un MLP pour la deuxième exécution (cas de convergence retardée). Les figures de dessus représentent les réponses des neurones cachés. Les figures de dessous représentent les réponses des neurones de sortie ainsi que les frontières de décision générées

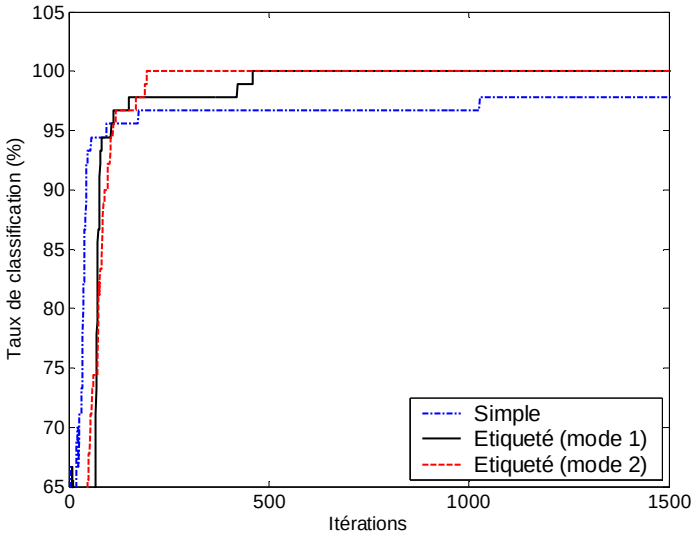
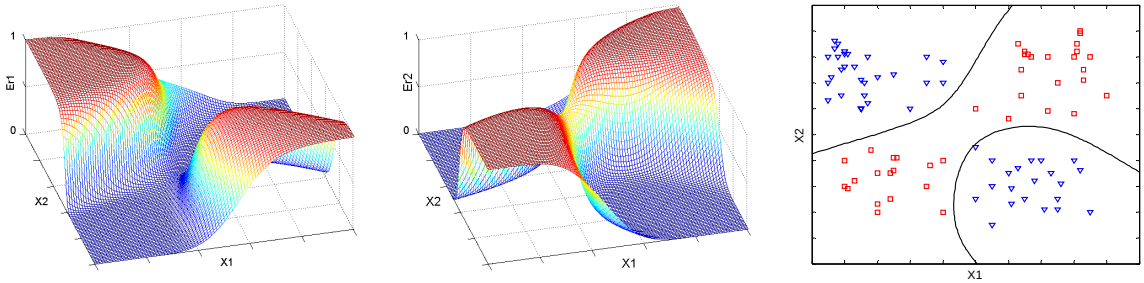


Fig. 5.10. Evolution du taux de classification pour un MLP simple et un MLP étiqueté pour le problème de la figure (5.7) : cas de convergence retardée.

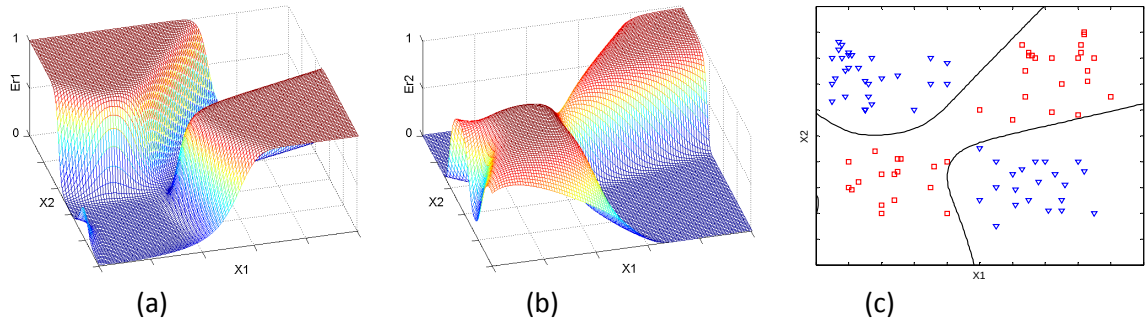
Les autres résultats obtenus en utilisant les deux modes d'apprentissage de la classification étiqueté sont illustrés sur la figure (5.11) et (5.12). Celles-ci représentent la forme des erreurs correspondantes à chacune des deux classes ainsi que les frontières de décision obtenues. Nous remarquons que dans les deux modes, l'erreur correspondante à chaque classe est nulle pour les exemples appartenant à cette classe et elle est égale à 1 pour les autres exemples. L'application de la classification étiquetée a permis au MLP de bien résoudre ce problème.

Pour analyser les performances du MLP étiqueté en fonction des étiquettes nous réalisons des tests avec différentes valeurs. Le tableau (5.1) reporte les résultats obtenus. Ce tableau indique que dans les deux modes d'apprentissage, le MLP étiqueté permet d'avoir un taux de classification égale à 100% pour $\sigma \leq 0{,}080$. Ce tableau indique aussi que le deuxième



(a) (b) (c)

Fig. 5.11. Résultats de classification du problème de la figure (5.7) avec un MLP étiqueté entrainé en utilisant le mode1 (dans le cas d'une convergence retardée).
(a) Erreur correspondante à la classe C1
(b) Erreur correspondante à la classe C2
(c) Frontières de décision obtenues.



(a) (b) (c)

Fig. 5.12. Résultats de classification du problème de la figure (5.7) avec un MLP étiqueté entrainé en utilisant le mode2 (dans le cas d'une convergence retardée).
(a) Erreur correspondante à la classe C1
(b) Erreur correspondante à la classe C2
(c) Frontières de décision obtenues.

mode accorde moins de dépendance aux valeurs des étiquettes : pour $\sigma \leq 0{,}080$ les résultats sont quasiment les mêmes.

Tableau (5.1) : Effet des étiquettes (cas de convergence retardée)

Méthode	σ	Etiquettes	Itérations	Taux de classification
MLP Simple	-	-	2291	100 %
MLP étiqueté (mode 1)	0.002	[0.499 0.501]	755	100 %
	0.010	[0.495 0.505]	690	100 %
	0.030	[0.485 0.515]	215	100 %
	0.050	[0.475 0.525]	82	100 %
	0.080	[0.460 0.540]	275	100 %
	0.100	[0.450 0.550]	336	91 %
MLP étiqueté (mode 2)	0.002	[0.499 0.501]	182	100 %
	0.010	[0.495 0.505]	182	100 %
	0.030	[0.485 0.515]	185	100 %
	0.050	[0.475 0.525]	198	100 %
	0.080	[0.460 0.540]	274	100 %
	0.100	[0.450 0.550]	1176	100 %

b) Cas de non-convergence

Dans cette partie nous analysons le cas où le MLP ne converge pas (cas de la troisième exécution). Dans de telles situations, le MLP n'arrive pas à s'échapper des régions de saturation.

La figure (5.13) illustre les réponses des neurones cachés, des neurones de sortie ainsi que les frontières de décision. Dans cette figure nous remarquons la saturation du quatrième neurone caché et nous remarquons également, à travers les frontières de décision obtenues, l'incapacité du MLP à classifier les exemples de ce problème. En revanche, un MLP étiqueté

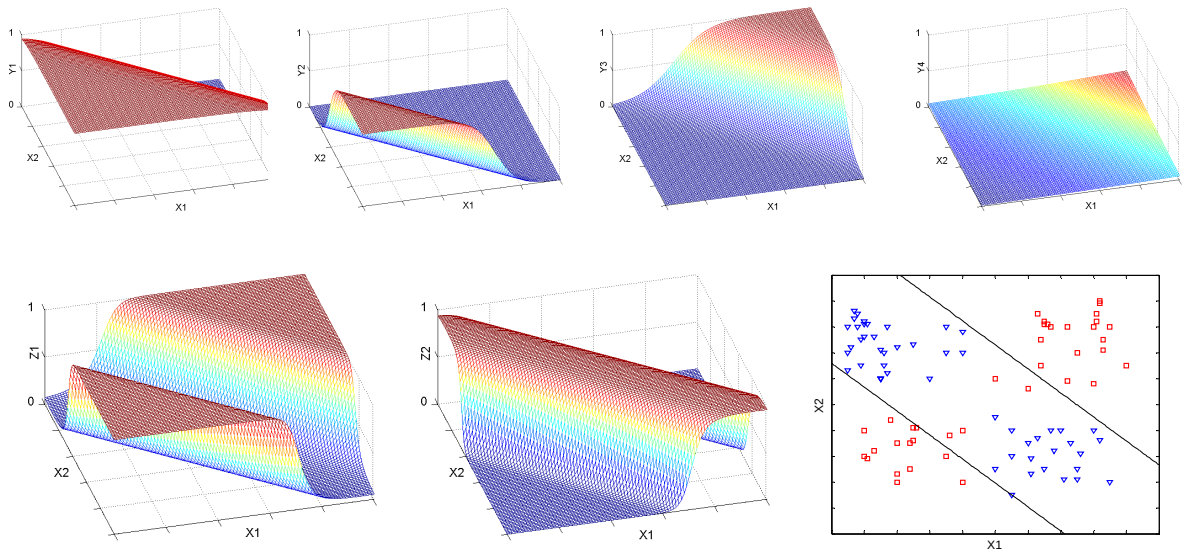


Fig. 5.13. Résultats de classification du problème de la figure (5.6) avec un MLP pour la deuxième exécution (cas de non convergence). Les trois figures de dessus représentent les réponses des neurones cachés. Les figures de dessous représentent les réponses des neurones de sortie ainsi que les frontières de décision générées

ayant la même architecture, les mêmes paramètres et entrainé avec les mêmes poids initiaux s'en tire bien.

La figure (5.14) représente l'évolution du taux de classification au cours de l'apprentissage dans le cas d'un réseau simple et d'un réseau étiqueté. Un MLP étiqueté entrainé en utilisant le deuxième mode d'apprentissage de la classification étiqueté à permet d'avoir un taux de classification égal à 100% après moins de 400 itérations tandis qu'en utilisant le premier mode cela nécessite environs 700 itérations.

Les figures (5.15) et (5.16) illustrent respectivement les résultats obtenus avec un MLP

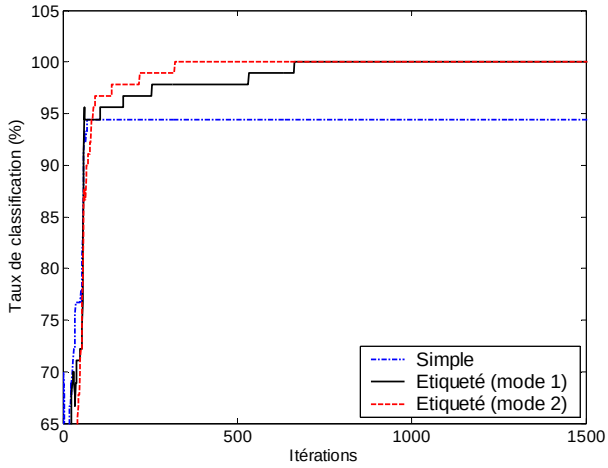
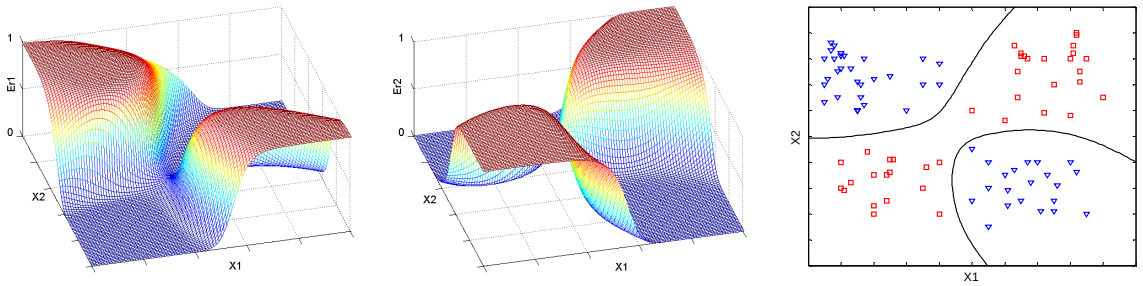
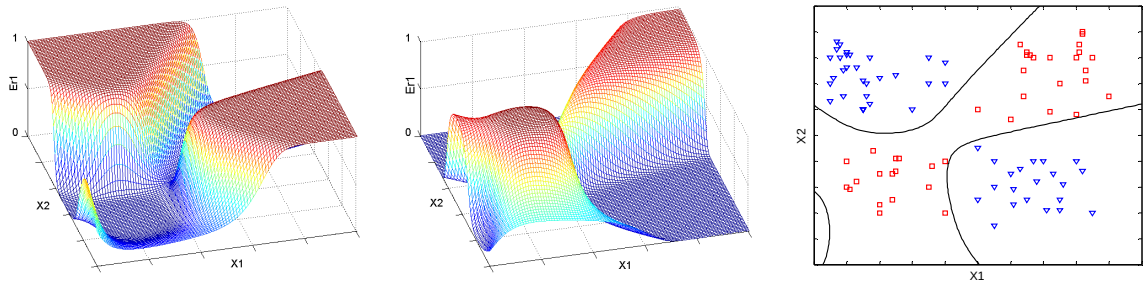


Fig. 5.14. Evolution du taux de classification pour un MLP simple et un MLP étiqueté pour le problème de la figure (5.7) : cas de non convergence



(a) (b) (c)

Fig. 5.15. Résultats de classification du problème de la figure (5.7) avec un MLP étiqueté entrainé en utilisant le mode1 (dans le cas de non convergence).
(a) Erreur correspondante à la classe C1
(b) Erreur correspondante à la classe C2
(c) Frontières de décision obtenues.



(a) (b) (c)

Fig. 5.16. Résultats de classification du problème de la figure (5.7) avec un MLP étiqueté entrainé en utilisant le mode2 (dans le cas de non convergence).
(a) Erreur correspondante à la classe C1
(b) Erreur correspondante à la classe C2
(c) Frontières de décision obtenues.

étiqueté entrainé en utilisant le premier mode de la classification étiqueté et en utilisant le deuxième mode. Les frontières de décision obtenues avec les deux modes montrent de bonnes capacités de généralisation.

Pour évaluer les performances du MLP étiqueté en fonction des étiquettes nous effectuons des tests avec différentes valeurs. Le tableau (5.2) illustre les résultats obtenus.

A partir de ce tableau, nous pouvons constater que le premier mode permet d'obtenir des résultats satisfaisants pour des étiquettes ayant $\sigma \leq 0{,}050$, tandis que pour le deuxième mode il suffit que $\sigma \leq 0{,}100$. Nous pouvons noter également qu'en deuxième mode et pour $\sigma \leq 0{,}050$ les résultats sont presque identiques.

Tableau (5.2) : Effet des étiquettes (cas de non-convergence)

Méthode	σ	Etiquettes	Itérations	Taux de classification
MLP Simple	-	-	70	94,44 %
MLP étiqueté (mode 1)	0.002	[0.499 0.501]	664	100 %
	0.010	[0.495 0.505]	510	100 %
	0.030	[0.485 0.515]	553	100 %
	0.050	[0.475 0.525]	105	100 %
	0.080	[0.460 0.540]	120	91,11 %
	0.100	[0.450 0.550]	126	90 %
MLP étiqueté (mode 2)	0.002	[0.499 0.501]	259	100 %
	0.010	[0.495 0.505]	261	100 %
	0.030	[0.485 0.515]	277	100 %
	0.050	[0.475 0.525]	321	100 %
	0.080	[0.460 0.540]	698	100 %
	0.100	[0.450 0.550]	1471	100 %
	0.200	[0.400 0.600]	5000	98.89 %

c) Cas de convergence

Dans les deux parties précédentes, nous avons évalué les améliorations apportées par la classification étiquetée dans les cas où le MLP trouve des difficultés. Dans cette partie, et afin d'analyser les propriétés du MLP étiqueté dans tous les cas possibles, nous traitons le cas où le MLP converge (cas de la première exécution). La figure (5.17) illustre les réponses

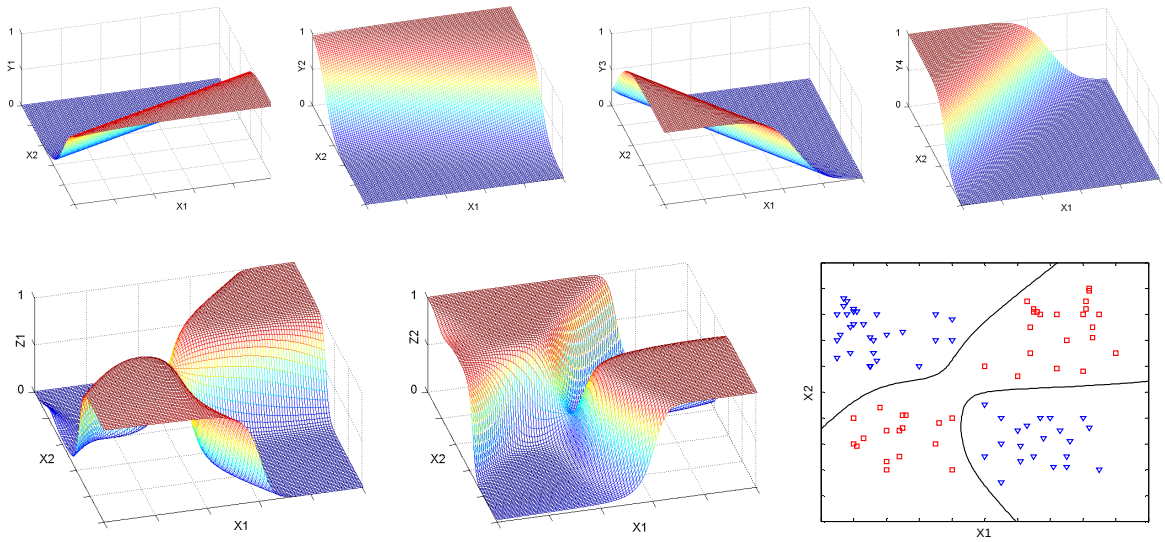


Fig. 5.17. Résultats de classification du problème de la figure (5.7) avec un MLP pour la première exécution (cas de convergence). Les figures de dessus représentent les réponses des neurones cachés. Les figures de dessous illustre les réponses des neurones de sortie ainsi que les frontières de décision générées

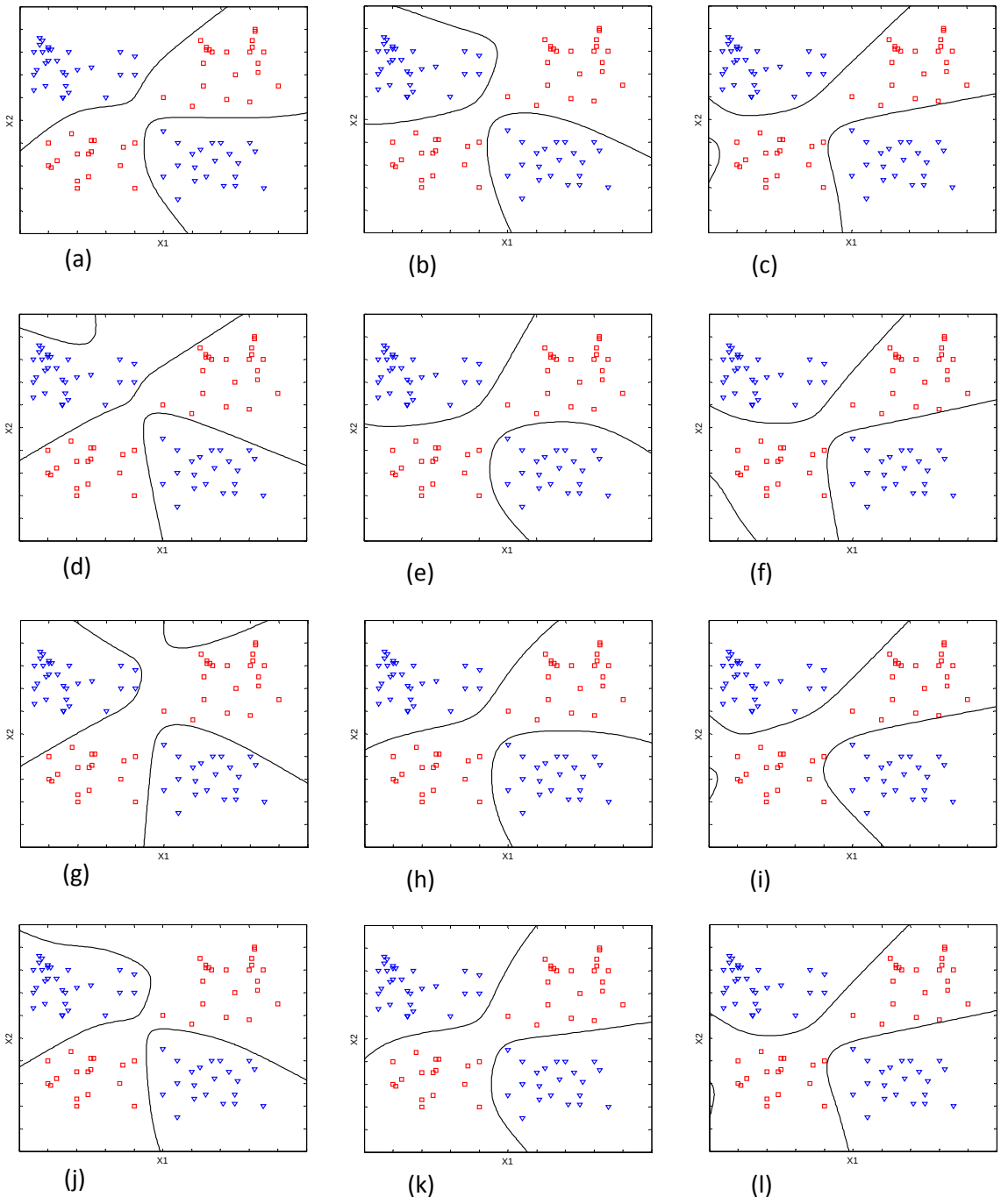


Fig. 5.18. Exemples de frontières de décision générées dans le cas de convergence :
(a), (d), (g) et (j): MLP simple
(b), (e), (h) et (k): MLP étiqueté (mode1)
(c), (f), (i) et (l): MLP étiqueté (mode2)

des neurones cachés, des neurones de sortie ainsi que les frontières de décision. Les frontières de décision générée par le MLP dans ce cas montrent qu'il a bien classifié tous les exemples. Nous ne remarquons aucune saturation au niveau des neurones cachés.

Bien que dans ce cas le MLP parvienne à bien classifier tous les exemples de ce problème, nous y examinons aussi le MLP étiqueté de s'assurer que l'application de la classification

étiquetée ne dégrade pas les capacités du MLP quand il parvient à résoudre un problème. La figure (5.18) illustre les frontières de décision générées par un MLP simple et un MLP étiqueté dans quelques exemples de convergence. A partir de cette figure, nous pouvons noter que le MLP étiqueté permet aussi une bonne classification de tous les exemples d'apprentissage. De plus, nous constatons une amélioration dans certains cas. Par exemple dans le deuxième et le troisième cas de cette figure ; les frontières de décision obtenues par l'application de la classification étiquetée sont plus appropriées à ce problème notamment en ce qui concerne les capacités de généralisation.

Il est à noter également que les frontières de décision générées par le MLP étiqueté ont presque les mêmes formes dans tous les cas (même dans le cas de non-convergence et le cas de convergence retardée).

5.5 Classificateur neuro-flou étiqueté

5.5.1 Principe de base et motivations

Dans cette partie nous introduisons une deuxième application de la classification étiquetée ; nous tentons la mise en œuvre d'un classificateur étiqueté en nous basant les systèmes neuro-flous. La conception de ce modèle, le CNF (Classificateur neuro-flou) étiqueté, est menée suite aux motivations suivantes :

- L'apprentissage des classificateurs neuro-flous par la RP est toujours lent, notamment l'adaptation des paramètres des fonctions d'appartenance.
- Les problèmes linéairement séparables sont plus faciles à résoudre.
- Les sorties des CNF peuvent également être interprétées comme des probabilités a posteriori [60].

Nous tirons ainsi profit du fait que les problèmes linéairement séparables sont plus faciles à résoudre et que les sorties des classificateurs neuro-flous fournissent des estimés des probabilités a posteriori afin d'appliquer la classification étiquetée. En outre, notre objectif est d'élaborer un classificateur neuro-flou dans lequel les fonctions d'appartenance ne sont pas ajustées. Cela devrait alléger considérablement le processus d'apprentissage.

L'application de la classification étiquetée s'effectue de la même façon qu'avec les réseaux de neurones, il s'agit d'ajouter les étiquettes à tous les exemples d'apprentissage et d'effectuer des tests avec ces étiquettes pour classer les nouveaux exemples. L'ajout des étiquettes entraine le remplacement des règles floues incorporées dans les classificateurs neuro-flous ordinaires qui sont de la forme :

$$\boldsymbol{Si}\ x_1\ est\ A_1\ et\ x_2\ est\ B_2\ \boldsymbol{Alors}\ X \in C_1\ avec\ CD_1\ et\ X \in C_2\ avec\ CD_2 \qquad (5.17)$$

Par des règles ayant dans leurs prémisses une proposition additionnelle correspondante aux étiquettes :

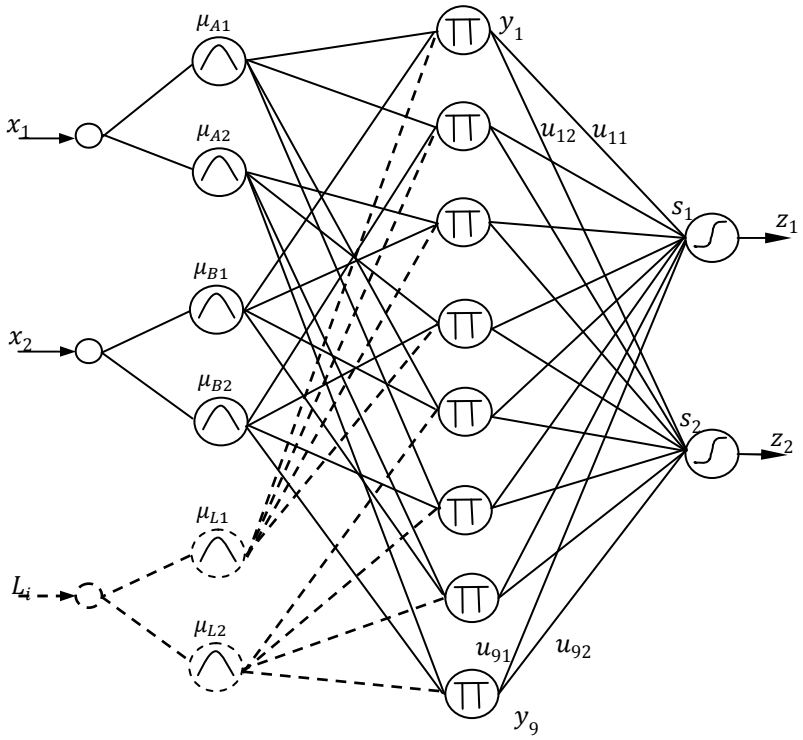


Fig. 5.19 Classificateur Neuro-Flou étiqueté bidimensionnel à deux classes et avec neuf règles

$$\boldsymbol{Si}\ x_1\ est\ A_1\ et\ x_2\ est\ B_2\ et\ x_3\ est\ L_1 \boldsymbol{Alors}\ X \in C_1\ avec\ CD_1\ et\ X \in C_2\ avec\ CD_2 \quad (5.18)$$

Ou sous forme de langage naturel : si x1 est petit et x2 est grand et son étiquette est L1, alors cet exemple appartient à la classe C1 avec un degré de croyance égale à 0.9 et à la classe C2 avec un degré de croyance égale à 0.3

5.5.2 Architecture

La première étape de cette méthode consiste à ajouter les étiquettes à tous les exemples d'apprentissage, et par conséquent à ajouter au CNF 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. La structure d'un CNF étiqueté est la même que celle d'un CNF ordinaire. La caractéristique ajoutée (les étiquettes) est considérée comme les autres caractéristiques.

La figure (5.19) illustre un CNF étiqueté pour un problème de classification bidimensionnel à deux classes (C_1 et C_2). Il se compose de quatre couches :

1ère couche : Le rôle des neurones de cette couche est d'émettre les entrées aux neurones de la couche suivante.

2ème couche : Chaque neurone de cette couche correspond à une fonction d'appartenance. Le rôle de cette couche est la fuzzification des variables d'entrée. Dans le cas d'utilisation de fonctions gaussiennes, la sortie du neurone correspondant à la fonction μ_{Ai} est de la forme :

$$\mu_{Ai}(x_n) = exp\left(-\left(\frac{x_n - m_{Ai}}{\sigma_{Ai}}\right)^2\right) \quad (5.19)$$

Où : x_n est la composante du vecteur d'entrée correspondante à ce neurone; m_{Ai} et σ_{Ai} sont les paramètres de la fonction d'appartenance μ_{Ai}.

Dans le figure (5.19) : A_1 et A_2 sont les termes linguistiques utilisés pour la fuzzification de x_1 et qui sont caractérisées par les fonctions d'appartenances μ_{A1} et μ_{A2} ; B_1 et B_2 corresponds à x_2 et elles sont caractérisés par μ_{B1} et μ_{B2}. L_1 et L_2 sont respectivement les étiquettes correspondantes aux classes C_1 et C_2 et qui sont caractérisées par μ_{L1} et μ_{L2}. Ces dernières ont la valeur d'étiquettes comme centre. En employant des fonctions gaussiennes, elles sont de la forme :

$$\begin{aligned}\mu_{Li}(L_j) &= exp\left(-\left(\frac{L_j - m_{Li}}{\sigma_{Li}}\right)^2\right) \\ &= exp\left(-\left(\frac{L_j - L_i}{\sigma_{Li}}\right)^2\right)\end{aligned} \quad (5.20)$$

3ème couche : Chaque neurone de cette couche correspond à une règle ; il détermine le degré d'activation de cette règle en calculant le produit de ses entrées. Les sorties de ces neurones sont de la forme:

$$y_m = \mu_{Ai}(x_1)\mu_{Bj}(x_2)\mu_{Lp}(L_q) \quad (5.21)$$

4ème couche : Chaque classe est représentée par un neurone de cette couche. Nous utilisons, de la même manière que le modèle de Jang [59], des neurones avec des fonctions d'activation sigmoïdale. La sortie du $k^{ème}$ neurone est donnée par :

$$\begin{aligned}z_k &= h(s_k) \\ &= h\left(\sum_{m=1}^{M} u_{mk} y_m\right)\end{aligned} \quad (5.22)$$

Où : $h(.)$ est la fonction sigmoïde et u_{mk} est le poids reliant le $m^{éme}$ neurone de la troisième couche avec le $k^{éme}$ neurone de la couche de sortie.

On notant NN_i le nombre de neurones de la $i^{ème}$ couche, nous aurons:

- $NN_1 = N$; Il est égal au nombre de caractéristiques.
- $NN_2 = (\sum_{n=1}^{N} D_n) + K$. Où D_n est le nombre des sous-ensembles flous utilisés pour la fuzzification de la $n^{ème}$ caractéristique (x_n) et K est le nombre de classes
- $NN_3 = \prod_{n=1}^{N} D_n$, ce nombre représente le nombre de règles.
- $NN_4 = N$, Il est égal au nombre de classes.

5.5.3 Apprentissage :

L'apprentissage du CNF étiqueté s'effectue uniquement par l'adaptation des poids de la couche de sortie. Les sous-ensembles flous utilisés pour la fuzzification ne sont pas modifiés. Dans les règles floues incorporées dans le CNF étiqueté (Eq. 5.18) les degrés de croyance des règles sont représentés par les poids qui sont reliés au neurone correspondant. Les poids reliés au $m^{ème}$ neurone: $u_{m1}, u_{m2}, \dots u_{mK}$ représentent les degrés de croyance de la $m^{ème}$ règle et correspondent respectivement aux classes : $C_1, C_2, \dots, C_K$.

L'adaptation du poids u_{mk} à l'itération $(r+1)$ est donnée par :

$$u_{mk}^{(r+1)} = u_{mk}^{(r)} - \eta(t_k - z_k)\dot{h}(s_k)y_k \tag{5.23}$$

Où : $\dot{h}(.)$ est la dérivée de la fonction sigmoïde utilisée dans les neurones de sortie.

L'apprentissage du CNF étiqueté consiste donc à ajuster les degrés de croyance des règles en maintenant les fonctions d'appartenance fixes. En effet, l'importance de ces paramètres a été évoquée dans d'autres travaux [66][67][68], et il a été noté que leur ajustement permet d'avoir de différentes formes de frontières de décision sans modification des fonctions d'appartenance. Il a été également noté que l'effet de ces paramètres est plus important dans le cas d'utilisation des partitions floues grossières.

5.5.4 Choix des étiquettes

Les degrés d'activation des règles établies par la troisième couche sont affectés par les fonctions d'appartenances des étiquettes plutôt que par les étiquettes elles-mêmes. C'est à

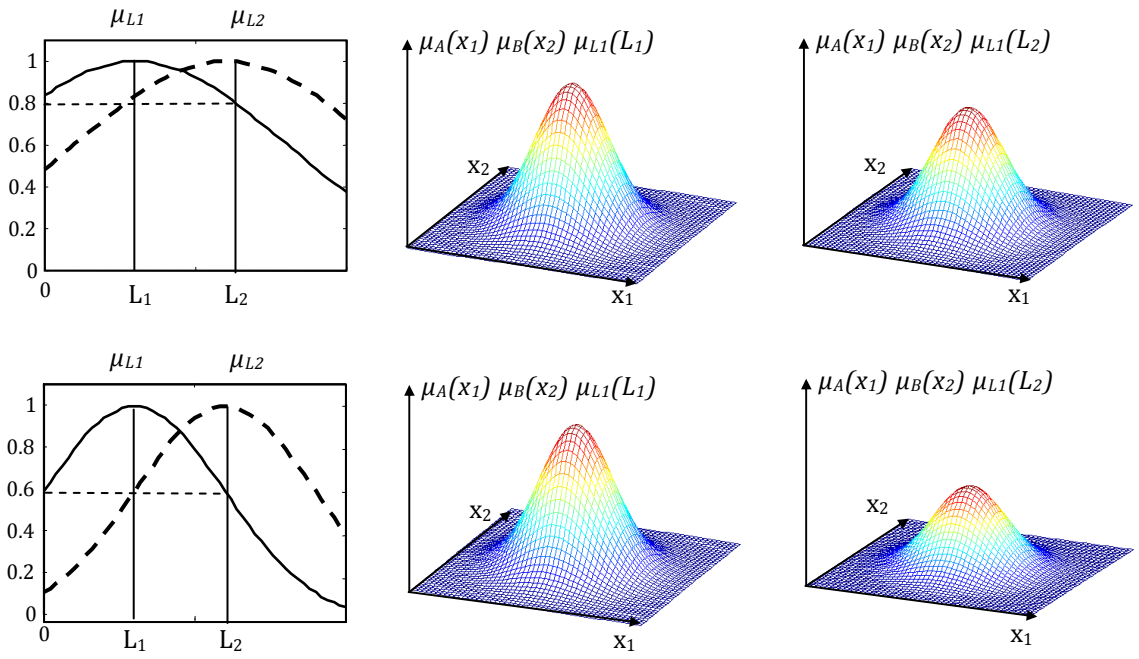


Fig. 5.20 Effet des étiquettes sur la sortie de la troisième couche

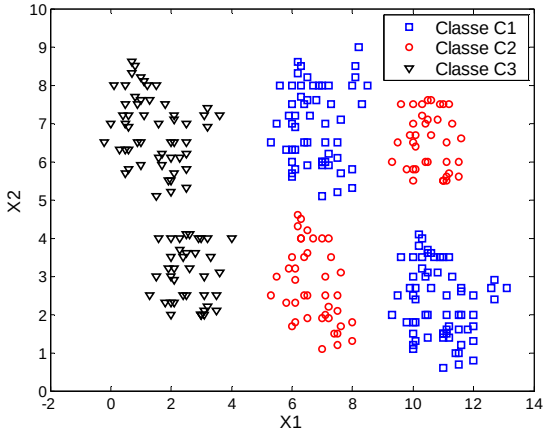


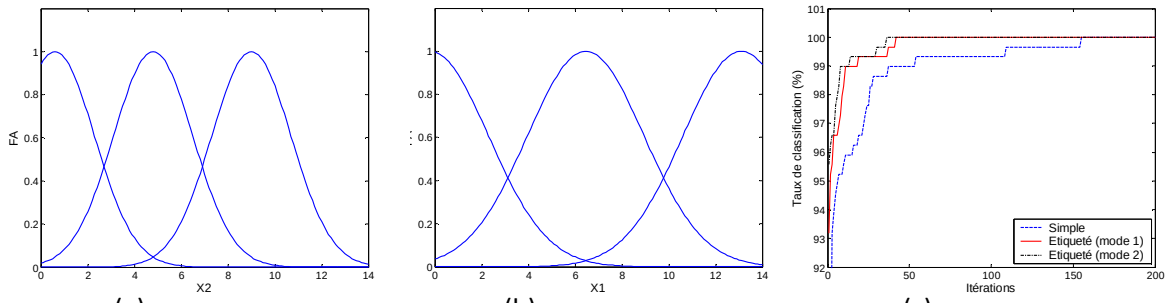
Fig. 5.21. Un problème de classification bidimensionnel à trois classes

dire, et contrairement au cas du MLP où les performances de la classification sont influencées directement par les valeurs des étiquettes, les performances du CNF étiqueté sont influencées par les fonctions d'appartenances des étiquettes. Plus précisément, par la valeur attribuée par la fonction d'appartenance μ_{Li} d'une étiquette L_i aux autres étiquettes $L_j, j = 1, \dots K, j \neq i$ (fig. 5.20). Nous posons : $\beta = \mu_{Li}(L_j)$

5.5.5 Exemple de classification

Pour évaluer les performances de l'application de la classification étiquetée avec les classificateurs neuro-flous, considérons l'exemple de la figure (5.21). C'est un problème bidimensionnel à 3 classes (C_1, C_2et C_3).

Pour analyser les améliorations apportées par la méthode proposée nous analysons les réponses d'un CNF ordinaire et d'un CNF étiqueté ayant les mêmes structures et les mêmes paramètres. Nous utilisons 3 sous-ensembles flous pour la fuzzification des variables d'entrée (figure 5.22.a et 5.22.b). Le pas d'apprentissage employé est $\eta = 0.9$.



(a) (b) (c)

Fig. 5.22. Résultats de classification du problème de la figure (5.21) en utilisant un CNF simple et un CNF étiqueté :
(a) Fonctions d'appartenance correspondantes à x_1
(b) Fonctions d'appartenance correspondantes à x_2
(c) Evolution du taux de classification durant l'apprentissage.

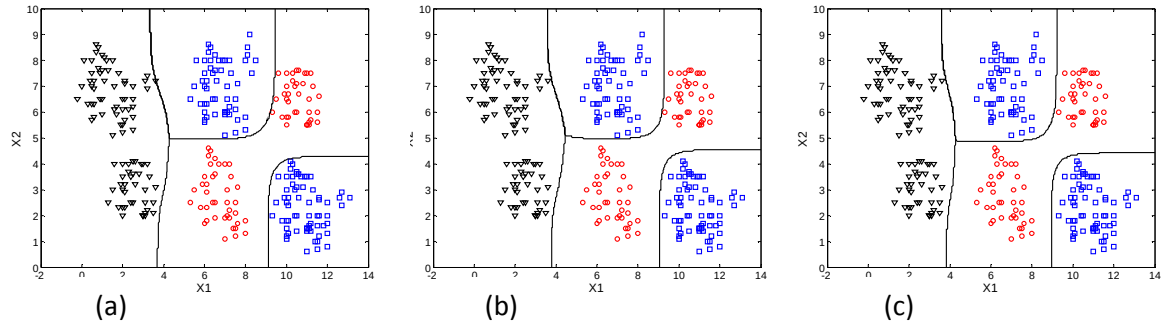


Fig. 5.23. Frontières de décision générées pour le problème de la figure (5.21) :
(a) CNF simple
(b) CNF étiqueté (mode1)
(c) CNF étiqueté (mode2)

Les taux de classification du CNF simple, du CNF étiqueté entrainé par le 1er mode et du CNF étiqueté entrainé par le 2ème mode sont illustrés sur la figure (5.22.c). Les graphes indiqués sur cette figure montrent une rapidité de convergence du CNF étiqueté par rapport au CNF ordinaire. Le classificateur étiqueté parvient à classifier tous les exemples dans moins de 40 itérations (en utilisant les deux modes) tandis que le CNF n'y parvient qu'après 155 itérations. Les frontières de décision générées par les trois modèles sont presque les mêmes (fig. 5.23). La classification étiquetée a permis donc une accélération de l'apprentissage sans dégradation des capacités de généralisation.

Pour analyser les performances du CNF en fonction des étiquettes, nous réalisons des tests, en utilisant les deux modes d'apprentissage, pour différentes valeurs des étiquettes. Les résultats obtenus sont reportés sur le tableau (5.3).

Tableau (5.3) : Effet des étiquettes

Méthode	$\boldsymbol{\beta}\left(\boldsymbol{\mu_{Li}(L_j)}\right)$	**Itérations**	**Taux de classification**
CNF Simple		155	100 %
CNF étiqueté (mode 1)	0.1	160	98.63 %
	0.3	120	98.98 %
	0.5	71	100 %
	0.7	38	100 %
	0.8	37	100 %
	0.9	42	100 %
CNF étiqueté (mode 2)	0.1	106	100 %
	0.3	58	100 %
	0.5	36	100 %
	0.7	24	100 %
	0.8	30	100 %
	0.9	37	100 %

Le tableau (5.3) montre que le CNF étiqueté entrainé en utilisant le premier mode donne de bons résultats pour $\beta \geq 0.5$. En revanche, le deuxième mode d'apprentissage permet d'avoir un taux de classification égal à 100 % pour toutes les valeurs et il y parvient avec un petit nombre d'itérations pour $\beta \geq 0.3$. À travers ces résultats nous constatons que le deuxième mode d'apprentissage offre plus de souplesse dans le choix des étiquettes, mais il est plus compliqué.

5.6 Conclusion

Dans ce chapitre nous avons présenté une nouvelle méthode de classification, appelée la classification étiquetée, dont l'objectif est d'améliorer les performances des classificateurs entrainés par la RP. Nous avons proposé deux modèles de classification basés sur l'application de cette méthode : le MLP étiqueté et le CNF étiqueté. La mise en œuvre des deux modèles s'articule essentiellement sur l'idée de rendre le problème de classification linéairement séparable afin d'alléger le processus d'apprentissage. L'application de cette méthode tire profit des capacités de ces deux classificateurs de fournir des estimés des probabilités a posteriori afin d'élaborer la décision finale.

Dans le chapitre qui suit, nous proposons un troisième modèle de classification qui se base sur l'application de classification étiquetée avec les systèmes multi-réseaux de neurones.

Chapitre 6

Les systèmes étiquetés multi-réseaux de neurones

Ce chapitre introduit un autre concept d'application de la classification étiqueté. Nous proposons un modèle basé sur l'emploi de cette méthode avec les systèmes multi-réseaux de neurones. Notre objectif consiste à concevoir un système comportant dans une structure hybride des réseaux simples et des réseaux étiquetés destinés à résoudre les parties difficiles du problème. Pour décrire nos apports, nous présentons d'abord les systèmes multi-réseaux de neurones et les stratégies les plus utilisées pour la décomposition des problèmes de classification multi-classes : un-contre-tous et un-contre-un. Nous décrirons ensuite les deux modèles proposés qui se basent sur ces deux stratégies de décomposition.

6.1 Introduction

La notion de modularité dans les systèmes informatiques s'appuie sur des imitations d'origines biologiques et psychologiques [71]. D'un point de vue biologique, la modularité se présente dans le cerveau sur différents niveaux. Cela se caractérise, à petite échelle, dans les relations entre neurones et, à grande échelle, dans les différentes zones du cerveau. Chacune de ces zones comporte un certain nombre de neurones responsables d'une tâche spécifique. Par ailleurs, plusieurs processus psychologiques naturels chez les humains ont servi de modèles pour la conception des systèmes modulaire, comme l'apprentissage sur différentes phases et selon différentes formes, la décomposition des tâches et la collaboration de différents participants pour la prise de décisions.

Dans le domaine la reconnaissance des formes, les problèmes de classification, qui sont généralement multi-classes, sont décomposés en un ensemble de sous-problèmes afin de simplifier la complexité du problème entier. Ils peuvent être ensuite traités avec un système multi-réseaux de neurones en assignant à chaque réseau la tâche de résoudre l'un de ces sous-problèmes. Toutefois, même si de tels problèmes sont décomposés, quelques parties restent difficiles à résoudre [72]. Dans cette partie, nous proposons un système de mise en œuvre des problèmes multi-classes basé sur une mixture de réseaux de neurones : ceux qui ont la tâche de résoudre les parties difficiles seront renforcés par l'application de la classification étiquetée. En nous basant sur les deux méthodes les plus utilisées pour la

décomposition des problèmes de classification multi-classes, nous introduisons deux versions de l'application de la classification étiquetée. Les systèmes proposés permettent de fournir une décision issue d'une combinaison des sorties de tous les réseaux intégrés. A cet effet, nous faisons appel aux méthodes de combinaison appliquées dans les systèmes multi-classificateurs.

6.2 Classification par les systèmes multi réseaux de neurones

6.2.1 Principes et objectifs

Afin de simplifier les problèmes de classification multi-classes, ils peuvent être divisés en plusieurs parties et mis en œuvre ensuite en utilisant un système multi-réseaux de neurones. Dans ce système, chaque réseau est entraîné indépendamment pour résoudre une partie du problème entier. De telles implémentations se basent généralement sur le fameux principe d'informatique "diviser pour régner " (divide and conquer). Ce principe consiste à diviser une tâche en sous-tâches plus petits et moins complexes, de faire apprendre chaque tâche par de différents modules, puis à combiner les résultats pour résoudre le problème entier.

Ces systèmes comportent un module de décision utilisé pour générer une décision finale par la combinaison des résultats de tous les réseaux (fig. 6.1). Pour ce faire, plusieurs méthodes ont été proposées dans la littérature, par exemple : la méthode de vote; la méthode du "Winner Takes All" ; méthodes basées sur la logique floue [12][13] ; les méthodes statistiques [73] [74][75].

La mise en œuvre des systèmes mutli-réseaux de neurones vise les objectifs suivants [3][53][71][76] :

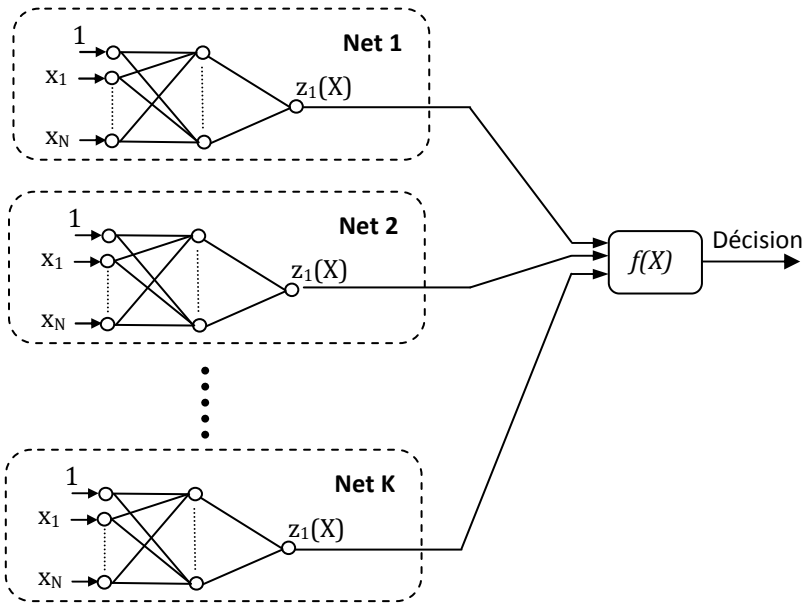


Fig. 6.1. Un système multi-réseaux de neurones

- Utilisation de réseaux ayant de différentes architectures (correspondantes aux différentes tâches attribuées) ce qui permet de réduire la taille des réseaux effectuant les tâches les plus simples.
- Utilisation de différents espaces caractéristiques pour chacun des réseaux et la possibilité de modélisation d'un comportement complexe en utilisant de différents types de connaissances.
- Un apprentissage indépendant de chaque réseau et par conséquent, la possibilité d'utilisation de plusieurs ordinateurs.
- Accélération de l'apprentissage.
- Amélioration des performances de généralisation.
- Eviter les interférences qui peuvent influencer l'aspect global des réseaux de neurones.
- La détermination de l'activité effectuée dans chaque partie du système, et à comprendre le rôle que joue chacun des réseaux dans le système complet.
- La modification des réseaux touchés par les changements affectant le processus sans modification de l'ensemble du système.

6.2.2 Stratégies de décomposition

La première étape lors de l'élaboration d'un système modulaire consiste à diviser le problème entier en un ensemble de sous-problèmes. Dans le domaine de classification, la décomposition des problèmes multi-classes s'effectue selon différentes stratégies [76][77] :

1. Un-contre-tous (OAA, One-Against-All) ; chaque sous-problème consiste à classifier une classe contre toutes les autres classes.
2. Un-contre-un (OAO, One-Against-One) : Chaque sous-problème consiste à différencier un couple de classes.
3. P-contre-Q (PAQ, P-Against-Q) : Le principe de cette méthode consiste à représenter les classes par des mots binaires de même taille, les "mots codes " (code word). Chaque exemple est ainsi attribué à la classe qui correspondant au mot le plus proche de ce vecteur au sens de la distance de Hamming.

Dans ce travail, nous nous intéressons aux deux premières méthodes (OAA et OAO) qui sont les plus utilisées [76].

6.2.3 Systèmes basés sur la méthode OAA

La méthode OAA divise un problème de classification K-classes en K sous-problèmes. Chacun de ces derniers a pour objectif la classification des exemples d'une classe contre ceux de toutes les autres classes [76][77][78].

La classification par un système multi-réseaux de neurones, modélisé par cette méthode, se réalise avec K réseaux de neurones ($Net_1, Net_2, \dots Net_K$). Chaque réseau Net_K est entraîné à classifier les exemples de la classe C_K des autres exemples, il a une sortie $z_K(X)$ permettant d'indiquer si l'exemple X appartient à la classe C_K ou non. Tous ces réseaux sont entraînés en utilisant l'ensemble de tous les exemples ; les données d'apprentissage sont les mêmes, mais les sorties désirées sont différentes.

Pour classifier un exemple, il sera présenté à tous les réseaux et la décision s'obtient en utilisant toutes leurs réponses. La règle de décision la plus utilisée est celle basée sur le maximum [76][78]. Cette fonction (F_{max}) assigne l'exemple présenté à la classe correspondante au réseau qui fournit la sortie la plus grande :

$$F_{max}(X) = \arg\max_k \{z_k(X), \quad k = 1 \dots K\} \tag{6.1}$$

6.2.4 Exemple de classification en utilisant un système basé sur l'approche OAA

Pour analyser les performances des systèmes multi-réseaux basés sur la méthode OAA, considérons le problème de classification à 3 classes de la figure (6.2). Comme le montre cette figure, les exemples de la troisième classe (C_3) sont linéairement séparables de ceux des autres classes, tandis que ceux de C_1 et C_2 ne sont pas linéairement séparables.

Nous utilisons trois réseaux dont chacun réalise la classification des exemples d'une classe de ceux des autres classes. Le premier réseau classifie les exemples de C_1, le deuxième classifie ceux de C_2 et le troisième s'en charge des exemples de C_3. Ce dernier aura donc la tâche la plus facile. Chaque réseau comporte deux neurones à l'entrée et un neurone à la sortie. Les premier et deuxième réseaux comportent quatre neurones cachés alors que le troisième comporte seulement deux vu la simplicité de sa tâche.

Après l'apprentissage de ces trois réseaux, ils génèrent les frontières de décision indiquées sur la figure (6.3). Comme le montre celle-ci, les frontières générées par les premier et

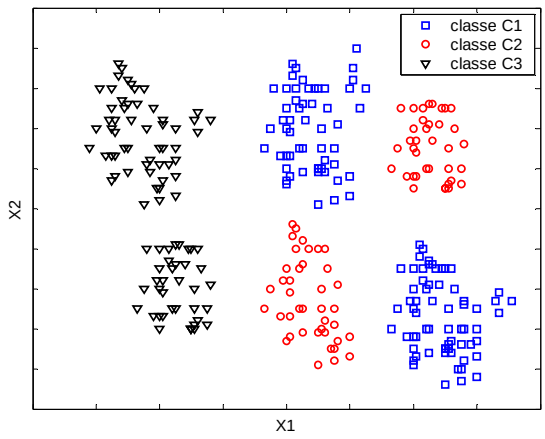


Fig. 6.2. Un problème de classification à 3-classes

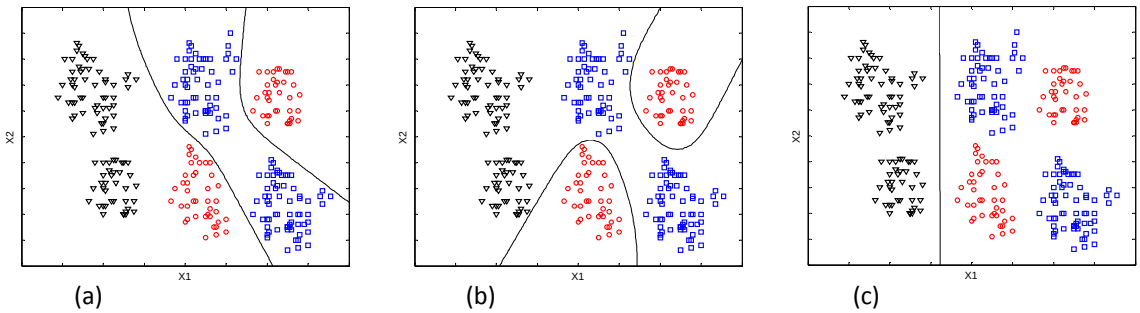


Fig. 6.3. Les frontières de décision générées par les trois réseaux
(a) Le premier réseau : C1 contre C2 et C3
(b) Le deuxième réseau : C2 contre C1 et C3
(c) Le troisième réseau : C3 contre C1 et C2

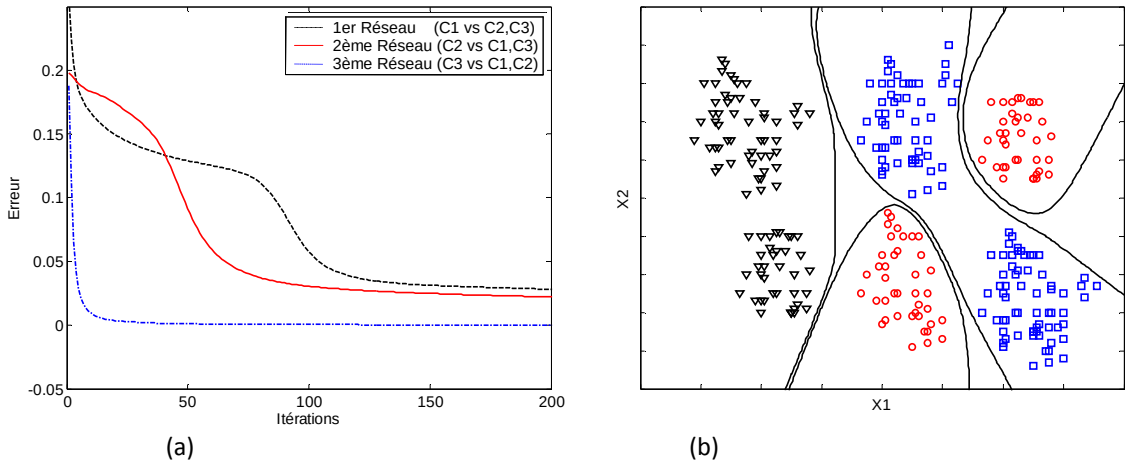


Fig. 6.4. Résultats de classification en utilisant un système basé sur l'approche OAA.
(a) Evolution de l'erreur des trois réseaux au cours de l'apprentissage
(b) Frontière de décision finale

deuxième réseaux sont plus complexes que celle réalisée par le troisième réseau. La complexité de ces deux réseaux peut être également constatée à partir de la figure (6.4.a) qui représente l'évolution de l'erreur de ces trois réseaux durant les 200 premières itérations d'apprentissage. Les graphes indiqués sur cette figure montrent que le troisième réseau parvient à converger plus rapidement ; dans moins de 10 itérations, tandis que les autres n'y arrivent pas même après 200 itérations.

Les frontières de décision établies par le système complet sont illustrées sur la figure (6.4.b). La décision finale est formée par la combinaison des résultats des trois réseaux. Nous pouvons constater que le système arrive à bien classifier tous les exemples.

6.2.5 Systèmes basés sur la méthode OAO

L'approche OAO, appelée aussi "pairwise method" [75], ou encore "round robin method" [77], se base sur la classification des exemples appartenant à chaque classe contre les exemples appartenant à chacune des autres classes. La méthode OAO transforme un

problème à K classes en $K(K-1)/2$ sous-problèmes [73][74][76][77] dont chacun correspond à un couple de classes.

La classification par un système multi-réseaux de neurones, modélisé par la méthode OAO, s'effectue en utilisant $K(K-1)/2$ réseaux de neurones : $Net_{ij}, i = 1 \dots K-1, j = i+1, \dots K$. Chaque réseau Net_{ij} est entraîné pour distinguer les exemples de la classe C_i de ceux de la classe C_j. Il comporte une sortie z_{ij} permettant d'indiquer si l'exemple X appartient à la classe C_i ou à la classe C_j.

Pour classer un exemple, la fonction de décision du système est formulée en utilisant les sorties de tous les réseaux. Le vote est la méthode la plus utilisée pour combiner les comparaisons duelles obtenues [75][76][79] ; la classe gagnante est celle qui gagne le plus grand nombre de comparaisons duelles. La règle du vote est :

$$F_{vote}(X) = \arg\max_k \left\{ \sum_{k,j \neq k} I\left(z_{kj}(X) > z_{jk}(X)\right) \right\} \tag{6.2}$$

Où : $I(.)$ est la fonction d'indication: $I(X) = 1$ si X est vrai, et égale à 0 autrement

D'après Friedman [79], cette règle est équivalente à la règle de Bayes lorsque les probabilités postérieures sont connues. Plusieurs extensions de cette méthode ont été proposées. L'une de ces méthodes, reportée dans [76], considère une valeur de confidence pour chaque exemple :

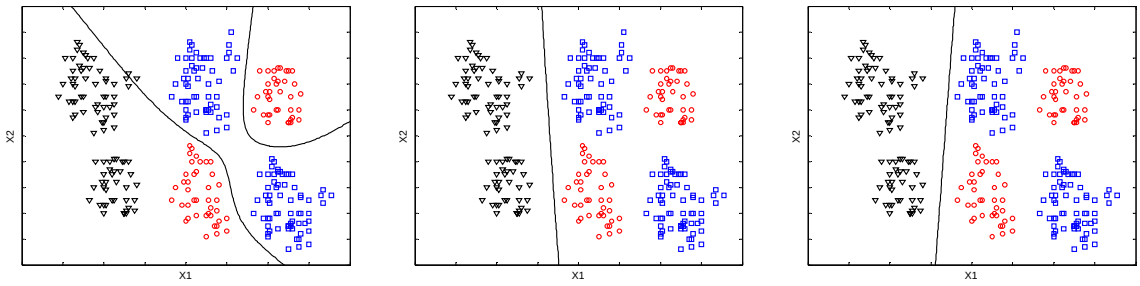
$$F(X) = \arg\max_k \left\{ \sum_{j=k+1}^{K} z_{kj}(X) + \sum_{j=1}^{k-1} \left(1 - z_{jk}(X)\right) \middle| k = 1, \dots, K \right\} \tag{6.3}$$

Où : $z_{ij}(X)$ est la valeur de confidence de X appartenant à C_i

6.2.6 Exemple de classification en utilisant un système basé sur l'approche OAO

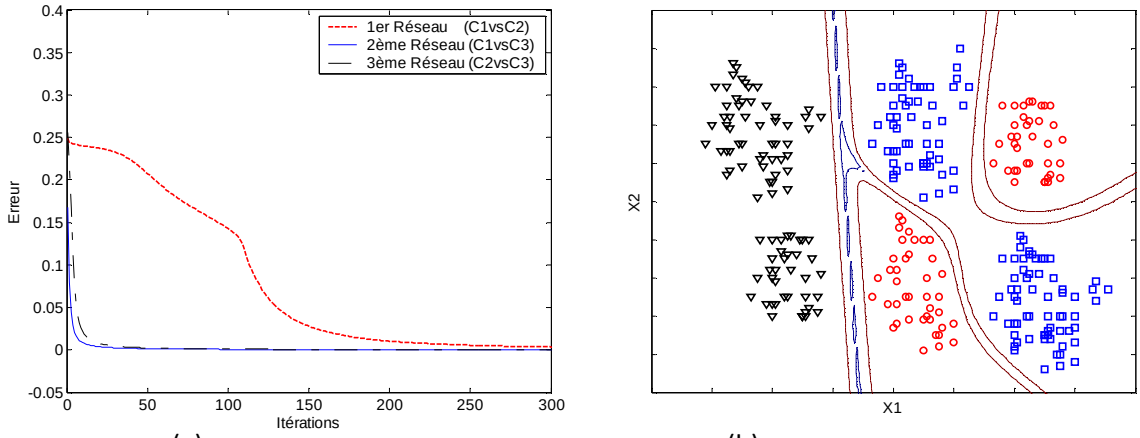
Pour analyser les performances des systèmes multi-réseaux basés sur la stratégie OAO, considérons le problème de classification de la figure (6.2). Nous utilisons également trois réseaux ; le premier classifie les exemples de la classe C_1 de ceux de la classe C_2, le deuxième classifie les exemples de la classe C_1 contre ceux de C_3 et le troisième réseau classifie les exemples de C_2 de ceux de C_3. Cette fois-ci, le premier réseau est le seul qui a la tâche la plus difficile.

La figure (6.5) illustre les frontières de décision établies par les trois réseaux. Celle-ci montre que la frontière de décision du premier réseau est la plus complexe. Cela est dû au fait que les classes C_1 et C_2 ne sont pas linéairement séparables.



(a) (b) (c)

Fig. 6.5. Les frontières de décision générées par les trois réseaux
(a) Le premier réseau : C1 contre C2
(b) Le deuxième réseau : C1 contre C3
(c) Le troisième réseau : C2 contre C3



(a) (b)

Fig. 6.6. Résultats de classification en utilisant un système basé sur l'approche OAO.
(a) Evolution de l'erreur au cours de l'apprentissage
(b) Frontière de décision finale

La figure (6.6.a), qui illustre l'évolution de l'erreur de ces réseaux, montre que le deuxième et le troisième convergent plus rapidement que le premier. Ces deux réseaux convergent après moins de 10 itérations tandis que le premier nécessite 300.

Les frontières de décision générées par le système entier sont illustrées sur la figure (6.6.b). Ce système permet, en se basant sur les réponses de tous les réseaux, de bien classifier les exemples de ce problème et permet de bonnes capacités de généralisations.

6.3 Les systèmes étiquetés multi-réseaux de neurones

La décomposition d'un problème de classification multi-classes en plusieurs petites parties vise la simplification du problème entier. En effet, une règle de classification de K-classes tend à être plus simple pour $K = 2$ que pour $K \geq 2$ [75]. Néanmoins, même si le problème est divisé, quelques parties restent difficiles à résoudre. Par exemple, dans le problème de la figure (6.2), la classe C_3 peut être facilement séparée des autres classes tandis que C_1 et C_2

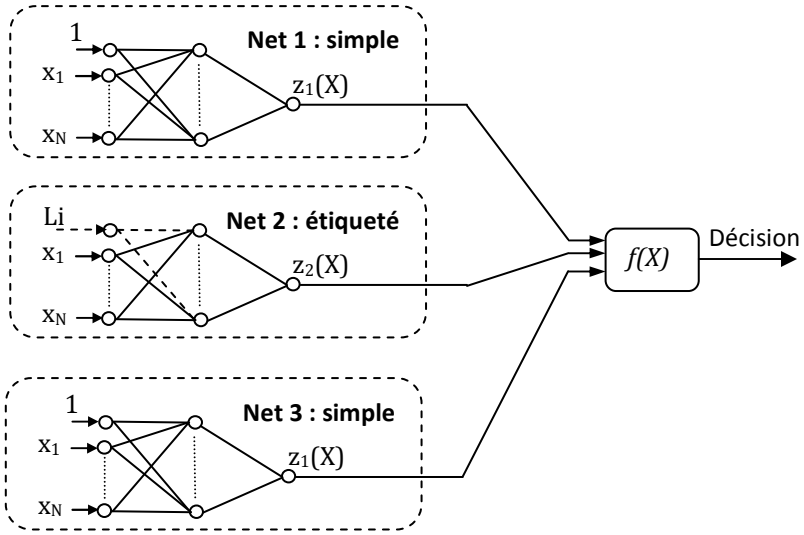


Fig. 6.7. Un système étiqueté multi-réseaux de neurones avec un deuxième réseau étiqueté

qui ne sont pas linéairement séparable ce qui rend leur classification plus difficile. De telles situations se présentent beaucoup dans les problèmes réels de classification.

Ce problème a été discuté dans [72], les auteurs ont suggéré de diviser encore le problème entier afin de le simplifier. Dans ce travail nous proposons un système multi-MLP qui tient compte de la complexité du problème attribué à chaque réseau. Notre contribution consiste à l'application de la classification étiquetée avec les réseaux ayant les tâches complexes afin de les renforcer. Nous nous basons sur le fait que les réseaux de neurones fournissent une estimation des probabilités a postériori et qu'un MLP étiqueté préserve cette propriété (paragraphe §5.4.5). Nous tirons profit de cette propriété pour la réalisation d'une décision finale dans le système proposé qui comporte dans une même structure deux types de réseaux. La figure (6.7) illustre un système étiqueté composé de trois réseaux de neurones, dans lequel le deuxième est étiqueté.

En nous basant sur les deux approches les plus utilisées pour la décomposition des problèmes multi-classes, à savoir : OAA et OAO, le modèle proposé s'étend sur deux aspects. Nous proposons donc deux types de systèmes étiquetés multi-réseaux de neurones.

6.3.1 Système étiqueté basé sur l'approche Un-Contre-Tous

Dans un système multi-réseaux de neurones modélisé par la méthode OAA, chaque réseau Net_K est entraîné à classer les exemples de la classe C_k des autres exemples. L'application de classification étiquetée avec un réseau Net_K de base sur l'emploi de deux étiquettes L_1 et L_2. Elle consiste à ajouter l'étiquette L_1 à tous les exemples appartenant à la classe C_k et à ajouter l'étiquette L_2 aux autres exemples. L'apprentissage d'un réseau étiqueté peut être effectué selon les deux modes d'apprentissage de la classification étiquetée.

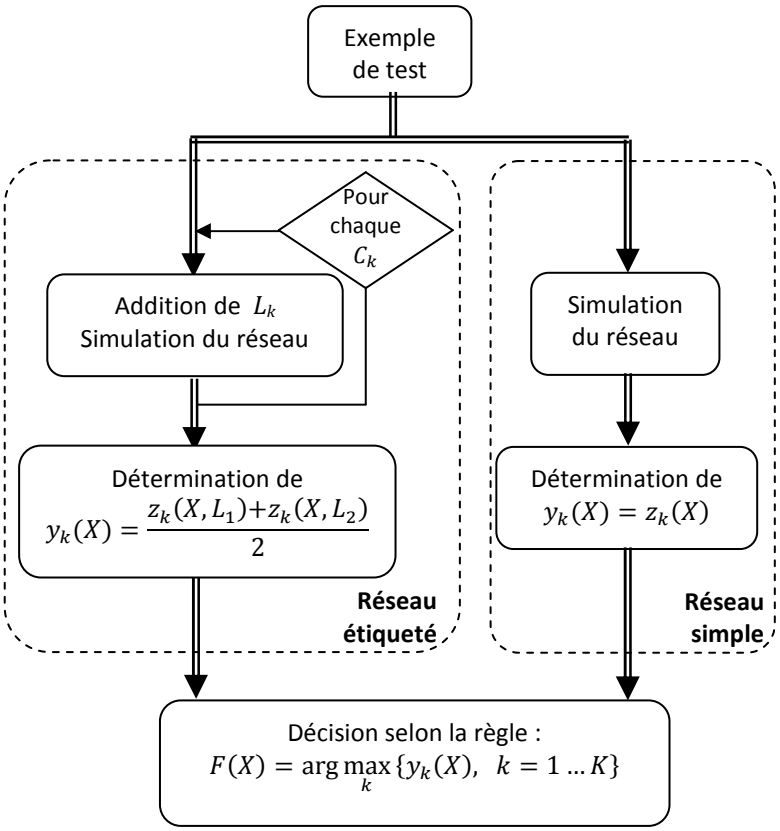


Fig. 6.8. Processus de classification dans les systèmes basés sur l'approche OAA

Dans la phase de classification, les sorties de tous les réseaux doivent être déterminées. Pour les réseaux étiquetés, elles sont calculées avec L_1 et L_2. Les sorties correspondantes ($z_k(X, L_1)$ et $z_k(X, L_2)$) peuvent être considérées comme les probabilités a posteriori. Nous définissons donc :

$$P_1(C_k/X) = z_k(X, L_1) \tag{6.4a}$$

$$P_2(C_k/X) = z_k(X, L_2) \tag{6.4b}$$

respectivement comme la probabilité que X appartient à C_k étant donné l'étiquette L_1 et la probabilité que X appartient à C_k étant donné l'étiquette L_2.

Pour l'intégration de ces deux sorties, nous utilisons la méthode de la moyenne qui est la plus utilisée pour la combinaison des sorties de différents classificateurs. Selon Tax et al. [80], cette méthode est simple à mettre en œuvre et permet de bonnes performances malgré le fait que le calcul de la moyenne n'est pas basé sur une fondation Baysienne solide. Dans leur étude comparative, ils ont noté que le calcul de la moyenne des probabilités a posteriori est préférable dans le cas les probabilités ne sont pas bien estimées.

Pour classer un exemple, la moyenne de ces deux probabilités sera donc combinée avec les sorties des autres réseaux pour produire la décision finale du système. Celle-ci est donnée par :

$$F(X) = \arg\max_k \{y_k(X),\ \ j = 1 \ldots J\} \tag{6.5}$$

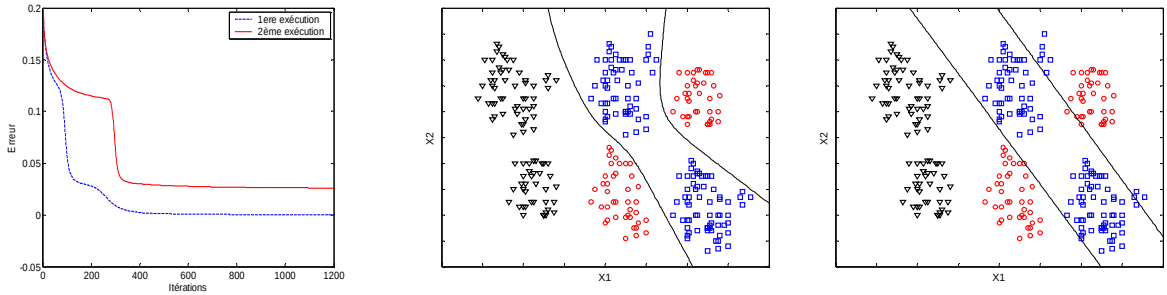
Avec :

$$y_k = \begin{cases} z_k(X) & Si\ Net_K\ est\ simple \\ \dfrac{z_k(X, L_1) + z_k(X, L_2)}{2} & Si\ Net_K\ est\ étiqueté \end{cases}$$

La figure (6.8) schématise le processus de classification dans un système étiqueté basé sur l'approche Un-Contre-Tous.

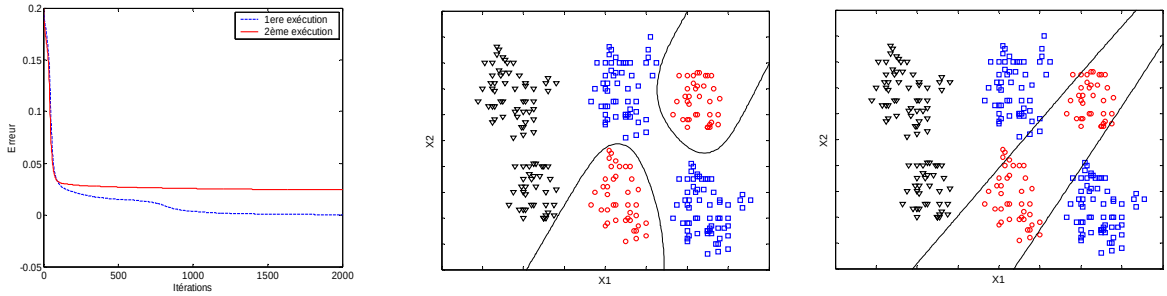
6.3.2 Exemple de classification en utilisant un système étiqueté basé sur l'approche OAA

Pour évaluer le modèle proposé, nous considérons de nouveau l'exemple de la figure (6.2). Dans le paragraphe (§6.4.2) nous avons étudié la résolution de ce problème en utilisant un système multi-MLP basé sur la méthode OAA. Dans ce paragraphe, nous avons noté que les tâches attribuées aux premier et deuxième réseaux sont plus complexes que celle du troisième (figure 6.4.a). En résolvant ces deux sous-problèmes (la classification des exemples



(a) (b) (c)

Fig. 6.9. Réponses du premier réseau pour deux exécutions différentes
(a) Evolution de l'Erreur au cours de l'apprentissage
(b) Frontière de décision pour la première exécution (cas de convergence)
(c) Frontière de décision pour la deuxième exécution (cas de non-convergence)



(a) (b) (c)

Fig. 6.10. Réponses du deuxième réseau pour deux exécutions différentes
(a) Evolution de l'Erreur au cours de l'apprentissage
(b) Frontière de décision pour la première exécution (cas de convergence)
(c) Frontière de décision pour la deuxième exécution (cas de non-convergence)

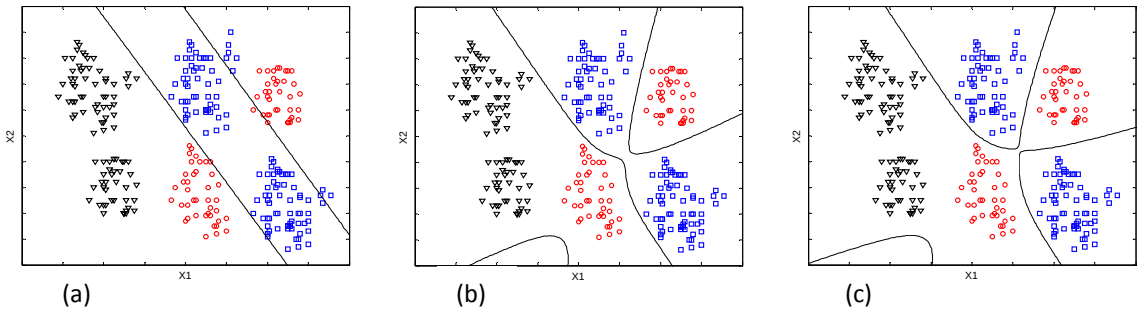


Fig. 6.11. Frontières de décision générées par le premier réseau
(a) Réseau simple
(b) Réseau étiqueté (mode1)
(c) Réseau étiqueté (mode2)

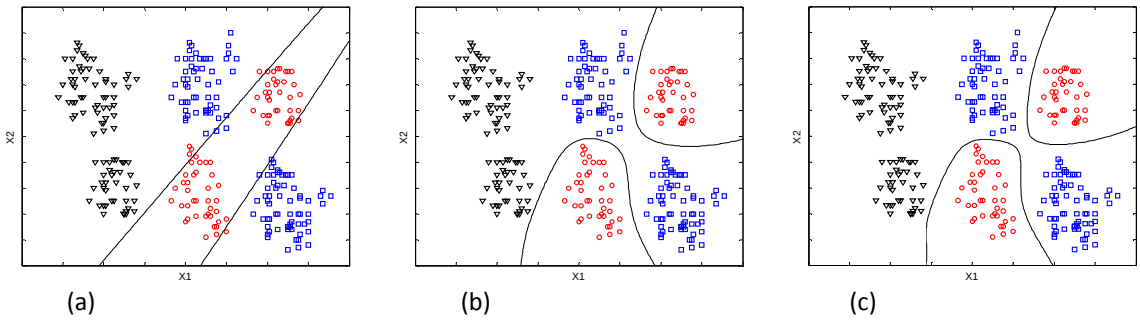


Fig. 6.12. Frontières de décision générées par le deuxième réseau
(a) Réseau simple
(b) Réseau étiqueté (mode1)
(c) Réseau étiqueté (mode2)

de C_1 et la classification de ceux de C_2), le MLP risque souvent d'être prisonnier dans des régions de saturations.

Les figures (6.9) et (6.10) illustrent les performances des premier et deuxième réseaux dans les deux cas : convergence et non-convergence. Le cas de convergence a été traité dans le paragraphe (§6.2.2). Dans cette partie nous traitons le cas où ces deux réseaux n'arrivent pas à résoudre leurs tâches, et montrons comment le modèle proposé pourrait corriger ce problème.

Nous utilisons donc un système étiqueté dans lequel les premier et deuxième réseaux sont étiquetés et le troisième est simple. Pour évaluer les améliorations obtenues, nous considérons les poids initiaux pour lesquels les réseaux ne convergent pas et nous utilisons les mêmes paramètres et les mêmes architectures. Nous employons pour chaque réseau les deux modes de la classification étiquetée : l'apprentissage simple et l'apprentissage complet.

La figure (6.11) illustre les frontières de décision du premier réseau, nous pouvons noter que l'application de la classification étiquetée, par ses deux modes d'apprentissage, a permis d'améliorer les performances de ce réseau. De même, la figure (6.12) qui illustre les

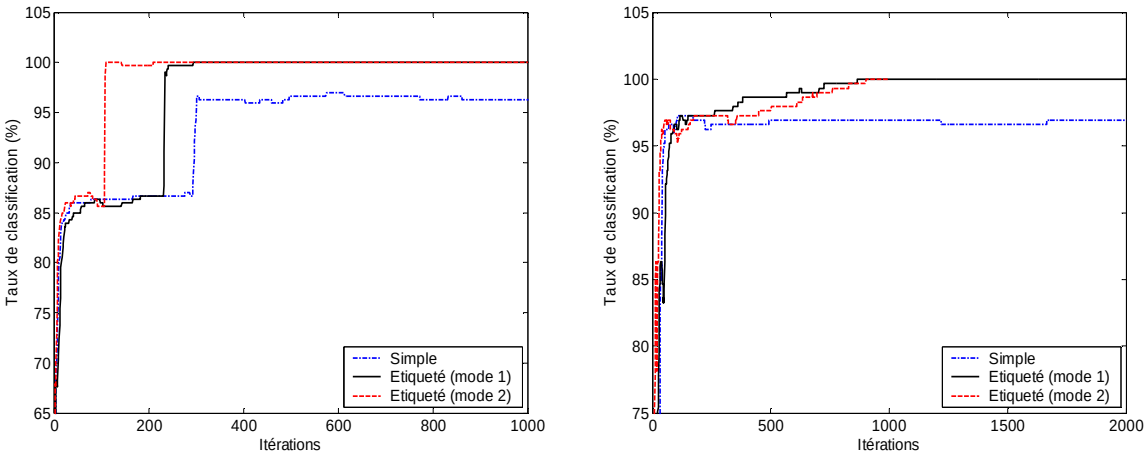


Fig. 6.13. Evolution du taux de classification dans le cas d'un réseau simple et d'un réseau étiqueté
(a) Premier réseau
(b) Deuxième réseau

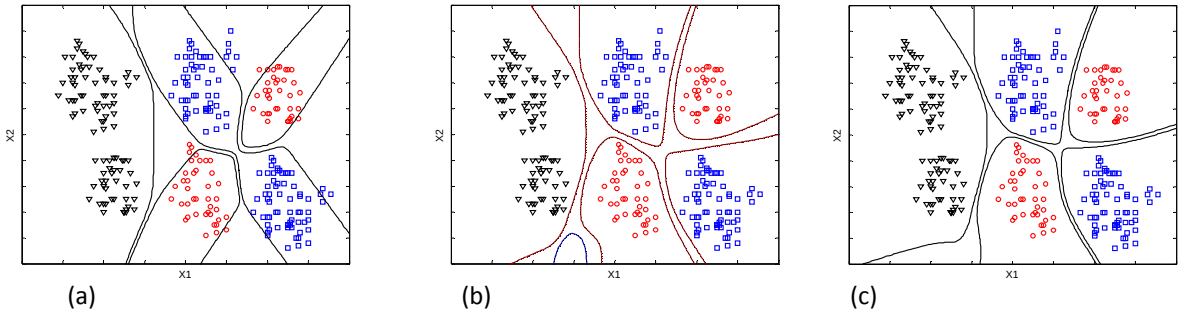


Fig. 6.14. Frontières de décision générées par le système entier
(a) Système simple
(b) Système étiqueté (mode1)
(c) Système étiqueté (mode2)

frontières de décision du deuxième réseau, indique les améliorations apportées par la classification étiquetée.

Les améliorations apportées par la classification étiquetée peuvent également être constatées à partir des graphes de la figure (6.13) qui illustrent l'évolution des taux de classification des premier et deuxième réseaux. L'utilisation de la classification étiquetée avec le premier réseau (figure 6.13.a) a permis d'avoir un taux de classification égal à 100% après 250 itérations en mode 1 et après 100 itérations en mode 2 tandis qu'un réseau simple ne dépasse pas le seuil de 96% même après 1000 itérations. De la même façon, la figure (6.13.b) représente les taux de classification correspondant au deuxième réseau.

Finalement, la figure (6.14) met en évidence les frontières de décision générées par le système entier. A la lecture de deux parties de cette figure, nous constatons que l'utilisation de la classification étiquetée, par ses deux modes d'apprentissage, a permis au système de

bien classer les exemples d'apprentissage et elle lui a permis d'avoir de bonnes capacités de généralisation.

6.3.3 Système étiqueté basé sur l'approche Un-Contre-Un

Dans un système de classification multi-réseaux de neurones modélisé par la méthode OAO, chaque réseau Net_{ij} est entraîné pour classer les exemples de la classe C_i contre les exemples de la classe C_j. L'application de la classification étiquetée avec cette méthode nécessite également l'utilisation de deux étiquettes : L_1 et L_2. L'apprentissage d'un réseau Net_{ij} étiqueté s'effectué après l'ajout de L_1 aux exemples appartenant à la classe C_i et L_2 aux exemples de la classe C_j.

De la même manière que le cas du OAA, la classification d'un nouvel exemple s'effectue par la combinaison des sorties de tous les réseaux. Les sorties des réseaux étiquetés sont calculées avec L_1 et L_2. Nous définissons les probabilités suivantes :

$$P_1(C_i/X) = z_{ij}(X, L_1), P_1\big(C_j/X\big) = 1 - P_1(C_i/X) \tag{6.6a}$$

$$P_2(C_i/X) = z_{ij}(X, L_1), P_2\big(C_j/X\big) = 1 - P_2(C_i/X) \tag{6.6b}$$

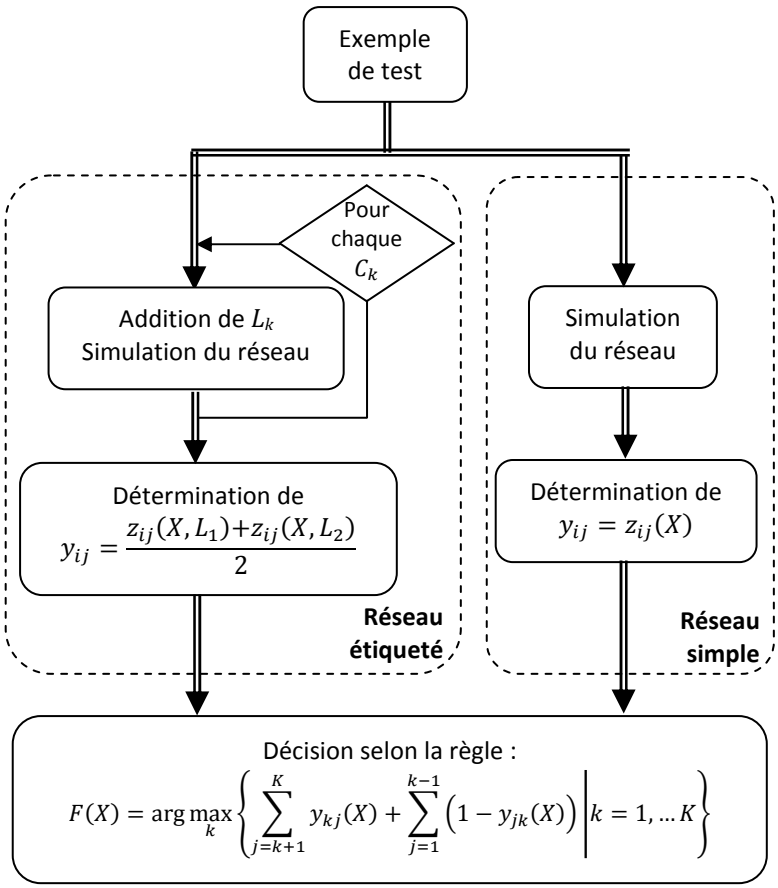


Fig. 6.15. Processus de classification dans les systèmes basés sur l'approche Un-Contre-Un

Où : $P_k(C_i/X)$ est la probabilité que X appartient a la classe C_i étant donné l'étiquette L_k. Pour chaque réseau, la moyenne des deux probabilités correspondantes est introduite dans la décision finale comme suit :

$$F(X) = \arg\max_k \left\{ \sum_{j=k+1}^{K} y_{kj}(X) + \sum_{j=1}^{k-1} \left(1 - y_{jk}(X)\right) \middle| k = 1, \dots K \right\} \tag{6.7}$$

Avec :

$$y_{ij} \begin{cases} z_{ij}(X) & Si\ Net_{ij}\ est\ simple \\ \dfrac{z_{ij}(X, L_1) + z_{ij}(X, L_2)}{2} & Si\ Net_{ij}\ est\ étiqueté \end{cases}$$

La figure (6.15) décrit la phase de classification dans un système étiqueté basé sur l'approche Un-Contre-Un.

6.3.4 Exemple de classification en utilisant un système étiqueté basé sur l'approche OAO

Pour évaluer le modèle proposé, nous l'examinons aussi en utilisant l'exemple de la figure (6.2). Dans le paragraphe (2.4) nous avons étudié ce problème en le décomposant par la méthode OAO. Dans ce paragraphe, nous avons noté que la tâche attribuée au premier réseau est la plus difficile (figure 6.6.a). Dans ce réseau, les poids risquent souvent d'être prisonniers dans des zones de saturation. La (figure 6.16.a) représente l'évolution de l'erreur au cours de l'apprentissage, elle illustre des exemples de trois cas possibles. La première exécution correspond au cas de convergence (le cas traité dans le paragraphe §6.2.6). La deuxième exécution représente un exemple où les poids entrent dans une région de saturation et dans lequel le MLP prend un grand nombre d'itérations pour en sortir (cas de convergence retardée). La troisième exécution représente le cas où le MLP n'arrive pas à sortir des zones de saturation (cas de non-convergence).

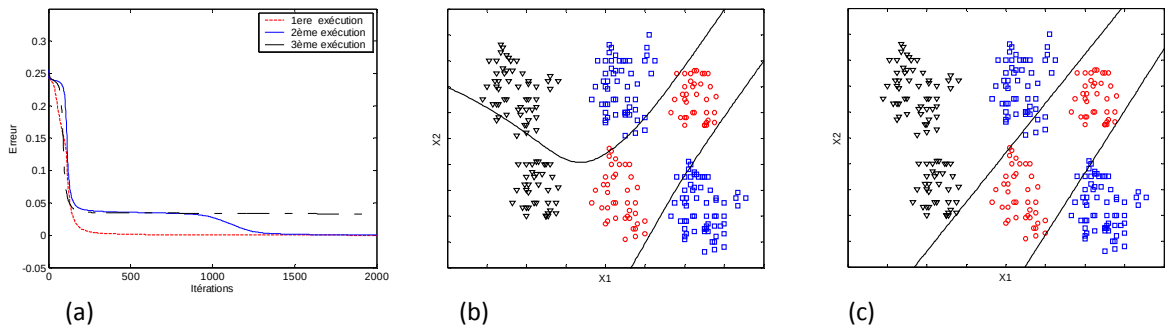


Fig. 6.16. Réponse du premier réseau (C1 contre C2) pour trois exécutions différentes
(a) Evolution de l'Erreur au cours de l'apprentissage
(b) Frontière de décision pour la 2ème exécution (cas de convergence retardée)
(c) Frontière de décision pour la 3ème exécution (cas de non convergence)

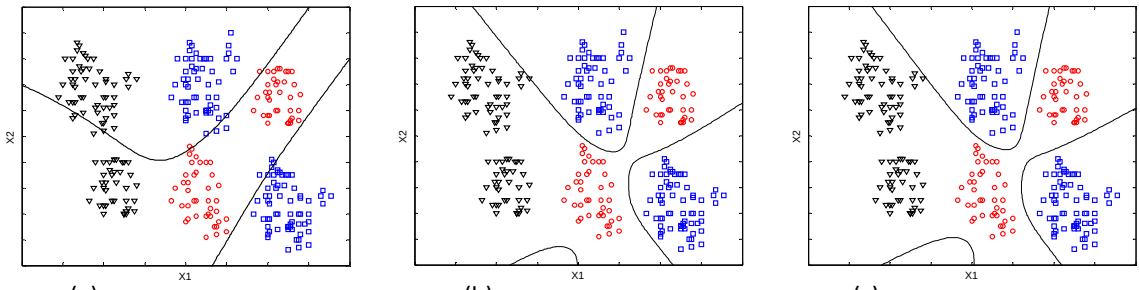


Fig. 6.17. Réponses du premier réseau (C1 contre C2) pour la deuxième exécution
(a) Réseau simple
(b) Réseau étiqueté (mode1)
(c) Réseau étiqueté (mode2)

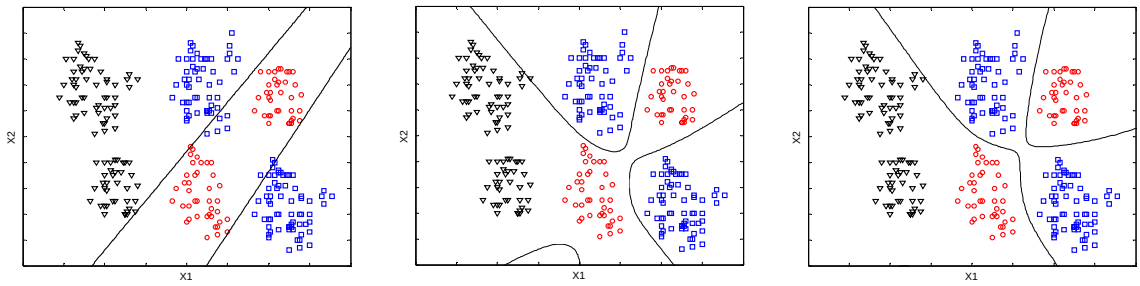


Fig. 6.18. Réponses du premier réseau (C1 contre C2) pour la troisième exécution
(a) Réseau simple
(b) Réseau étiqueté (mode1)
(c) Réseau étiqueté (mode2)

La figure (6.16.b) et la figure (6.16.c) illustrent les régions de décisions établies par le premier réseau pour les deuxième et troisième exécutions respectivement.

Nous utilisons donc un système étiqueté dans lequel le premier réseau est le seul réseau étiqueté. Pour évaluer les améliorations proposées, nous considérons les poids initiaux qui correspondent aux deux exécutions précédentes (convergence retardée et non-convergence) et nous utilisons les mêmes paramètres et les mêmes structures.

La figure (6.17) illustre les frontières de décision de ce réseau pour la deuxième exécution. A la lecture de cette figure, nous pouvons constater que l'application de la classification étiquetée, par ces deux modes d'apprentissage, a permis d'améliorer les performances de ce réseau. De même, la figure (6.18), qui illustre les frontières de décision pour la troisième exécution, indique les améliorations apportées par la classification étiquetée. L'évolution des taux de classification correspondants à ces deux cas est illustrée sur la figure (6.19). L'introduction de la classification étiquetée a permis au MLP de sortir des régions de saturation. Il parvient à avoir un taux de classification égale à 100% après environ 500 itérations en mode1 et après moins de 100 itérations en mode 2.

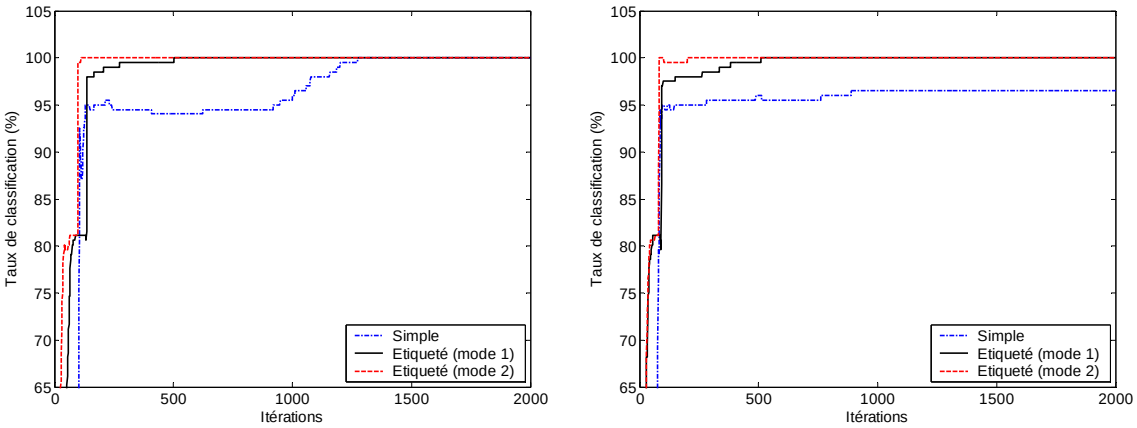


Fig. 6.19. Evolution du taux de classification du premier réseau (C1 contre C2) dans le cas d'un réseau simple et d'un réseau étiqueté
(a) Cas de la deuxième exécution (convergence retardée)
(b) Cas de la troisième exécution (non convergence)

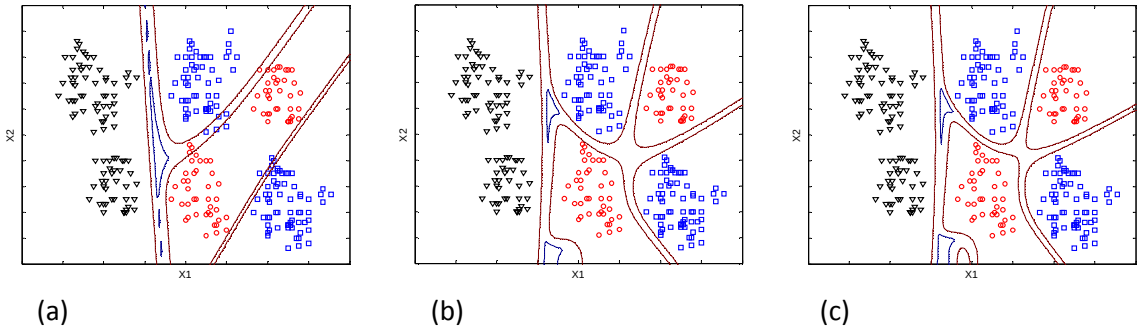


Fig. 6.20. Frontières de décision générées par le système (cas de convergence retardée)
(a) Réseau simple
(b) Réseau étiqueté (mode1)
(c) Réseau étiqueté (mode2)

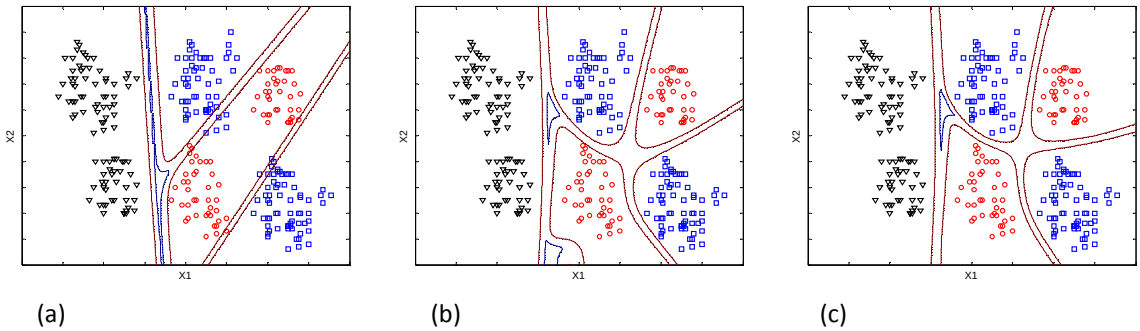


Fig. 6.21. Frontières de décision générées par le système (cas de non convergence)
(a) Système simple
(b) Système étiqueté (mode1)
(c) Système étiqueté (mode2)

Les réponses du système entier pour les deux cas sont illustrées dans les figures (6.20) et (6.21). A partir de ces figures, nous pouvons constater que l'utilisation de la classification

étiquetée, par ses deux modes d'apprentissage, a permis au système de bien classifier les exemples d'apprentissage et d'avoir de bonnes capacités de généralisation.

6.4 Conclusion

Dans ce chapitre, nous avons présenté le troisième modèle proposé dans cette thèse. C'est un processus de mise en œuvre des problèmes de classification multi-classes en utilisant de différents réseaux de neurones. La classification étiquetée a été introduite dans ce modèle pour renforcer les réseaux ayant les tâches les plus complexes. Ce modèle incorpore donc dans une même structure deux types de réseaux, et combine leurs réponses pour la réalisation de la décision finale. En nous basant sur les deux approches les plus utilisées pour la décomposition des problèmes multi-classes (OAA et OAO), nous avons proposé deux versions de systèmes étiquetés.

Afin d'évaluer les performances de ces systèmes, nous avons réalisé des tests de classification sur un problème synthétique bidimensionnel. Nous avons analysé les améliorations apportées en nous basant sur les frontières de décision générées par les réseaux et par les systèmes entiers. A travers ces tests, nous avons constaté que l'application de ces systèmes a permis d'améliorer les performances de classification à la fois au niveau des réseaux et au niveau des systèmes complets.

Dans le chapitre qui suit, nous analyserons les performances des trois modèles proposés en utilisant de différentes bases de données.

Chapitre 7

Tests et résultats

Ce chapitre évalue la méthode proposée et les trois modèles de classification qui en découlent. Pour cela nous utilisons quatre bases de données : Iris, cuisse humaine, vin et Texture. Afin d'apprécier nos apports, nous comparons les modèles proposés, d'une part, avec leurs versions originales et, d'autre part, avec d'autres approches.

7.1 Introduction

Dans le cadre de cette thèse, nous avons introduit trois modèles de classification : le perceptron multicouche étiqueté (LMLP, Labeled Multi Layered Perceptron); le classificateur neuro-flou étiqueté (LNFC, labeled Neuro-Fuzzy Classifier) et les systèmes étiquetés multi-réseaux de neurones (LSMulNet, Labeled Systems of Multiple Neural Networks). Afin d'évaluer ces modèles, nous effectuons des tests sur différents types de données : des mesures de longueur dans la base de données Iris ; des couleurs dans la base de données de la cuisse humaine ; des calculs de différentes corrélations locales dans la base de données de la texture et les propriétés physico-chimiques dans la base de données du vin. Nous analysons d'abord les améliorations proposées par une comparaison des modèles proposés avec leurs versions originales. En nous basant sur les résultats obtenus, nous discutons ensuite les avantages et les inconvénients de la méthode proposée et comparons nos résultats avec ceux d'autres travaux.

7.2 Classification de la base de données Iris

La base de données Iris de Fisher [81] est l'une des bases de données les plus connues et les plus utilisées en reconnaissance des formes ; elle est souvent utilisée comme référence pour évaluer les performances des classificateurs. Cette base de données comporte 150 échantillons de fleurs réparties en trois classes : sétosa, versicolor et virginia. Chaque fleur est décrite par quatre caractéristiques : la longueur et la largeur du sépale, et la longueur et la largeur du pétale. Nous utilisons tous les exemples de cette base de données pour l'apprentissage des classificateurs étudiés.

7.2.1 Classification en utilisant le MLP et le MLP étiqueté

Pour la classification de cette base de données, nous utilisons un MLP et un MLP étiqueté ayant 5 neurones à l'entrée, 8 neurones cachés et 3 neurones à la sortie. La figure (7.1) illustre l'évolution du taux de classification pendant la phase d'apprentissage du MLP ainsi que ceux d'un LMLP entrainé en utilisant le premier mode de la classification étiquetée (LMLP1) et d'un LMLP entrainé avec le deuxième mode (LMLP2). Les trois modèles ont les mêmes architectures, les mêmes paramètres d'apprentissage et ils sont initialisés par les mêmes poids. Les étiquettes utilisées sont : $L_1 = 0.475$, $L_2 = 0.5$ et $L_3 = 0.525$ ($\delta = 0.025$).

Les graphes de la figure (7.1) indiquent les améliorations obtenues par la classification étiquetée. Le LMLP1 et le LMLP2 ont permis d'obtenir un taux de classification égal à $98\ \%$

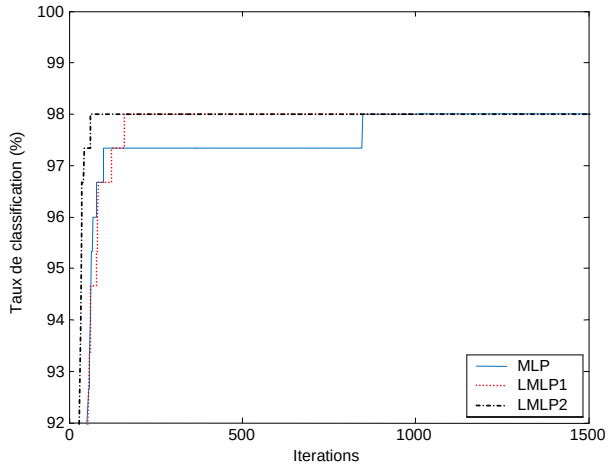


Fig. 7.1. Classification de la base de données Iris en utilisant un MLP et un MLP étiqueté

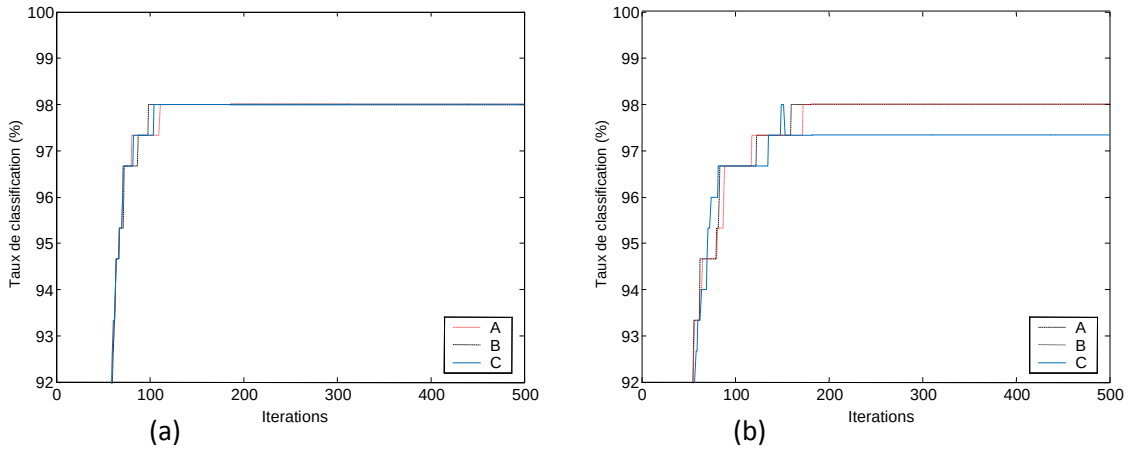


Fig. 7.2. Classification de la base de données Iris en utilisant un MLP étiqueté pour différentes étiquettes
(a) Apprentissage simple (mode1)
(b) Apprentissage complet (mode2)

dans moins de 200 itérations, tandis que le MLP n'y parvient qu'après plus de 750 itérations. Nous remarquerons que le MLP prend 700 itérations pour dépasser le seuil de 97,33%. Cela peut être expliqué par une zone de saturation où le MLP s'est trouvé prisonnier, et où il a nécessité toutes ces itérations pour s'en sortir.

La figure (7.2.a) illustre l'évolution du taux de classification pour différentes valeurs d'étiquettes dans le cas de l'apprentissage simple. Le graphe A correspond *à* $\delta = 0.015$ *(*$L_1 = 0.485$, $L_2 = 0.5$ et $L_3 = 0.515$*)*, le graphe B correspond *à* $\delta = 0.025$ et le graphe C correspond *à* $\delta = 0.050$. Nous pouvons noter que le LMLP entrainé avec ce mode donne des résultats acceptables pour $\delta \leq 0.025$. La figure (7.2.b) indique l'effet des étiquettes dans le cas de l'apprentissage complet ; le LMLP permet d'avoir les mêmes résultats pour les trois ensembles d'étiquettes. Cela montre que le deuxième mode accorde plus de souplesse dans le choix des étiquettes.

7.2.2 Classification en utilisant un NFC et un NFC étiqueté

Afin d'évaluer le LNFC sur cette base de données, nous comparons ses performances avec un NFC ayant les mêmes paramètres d'apprentissage et initialisés par les mêmes poids. Nous utilisons trois variables linguistiques (figure 7.3) pour la fuzzification des caractéristiques.

La figure (7.4) illustre l'évolution du taux de classification pendant l'apprentissage du NFC, LNFC1 et NFC2. Les fonctions d'appartenance utilisée pour les étiquettes satisfont $\mu_i(L_i) = 1$ et $\mu_i(L_j) = 0.85$. Cette figure indique les améliorations obtenues grâce à classification étiquetée ; le LNCF1 permet d'avoir un taux de classification égal à $99.33\,\%$ après 60 itérations et le LNFC2 donne $98.67\,\%$ après 38 itérations, tandis que le NFC

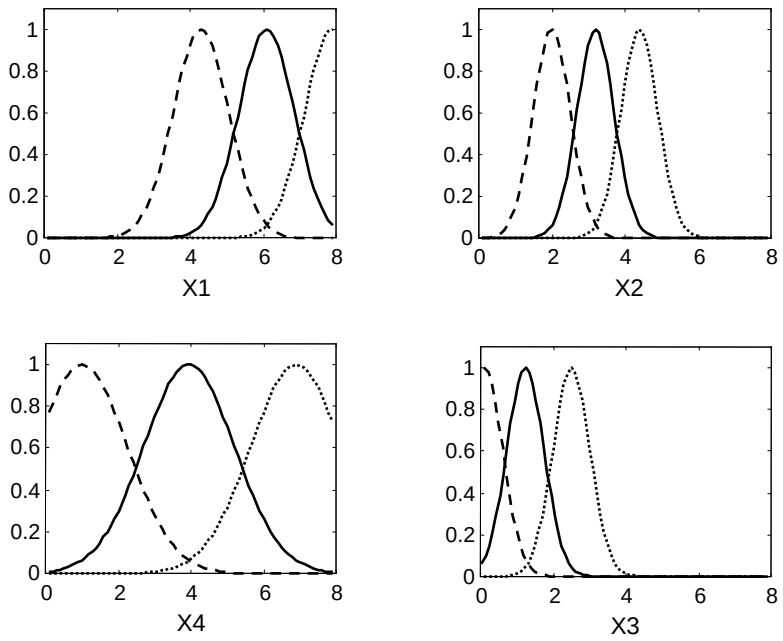


Fig. 7.3. Fonctions d'appartenance utilisées pour la fuzzification des caractéristiques de la base de données Iris.

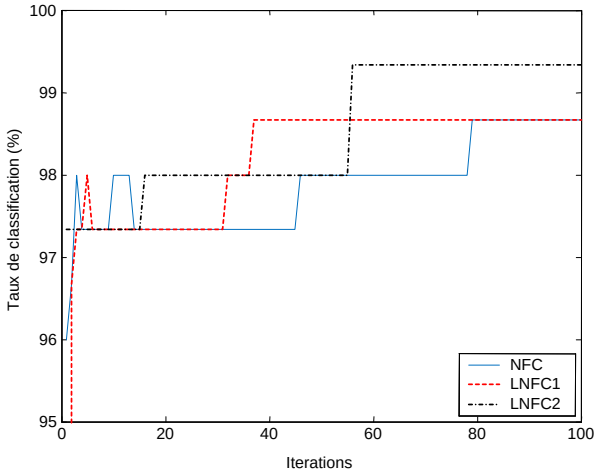


Fig. 7.4. Classification de la base de données Iris en utilisant un NFC et un NFC étiqueté

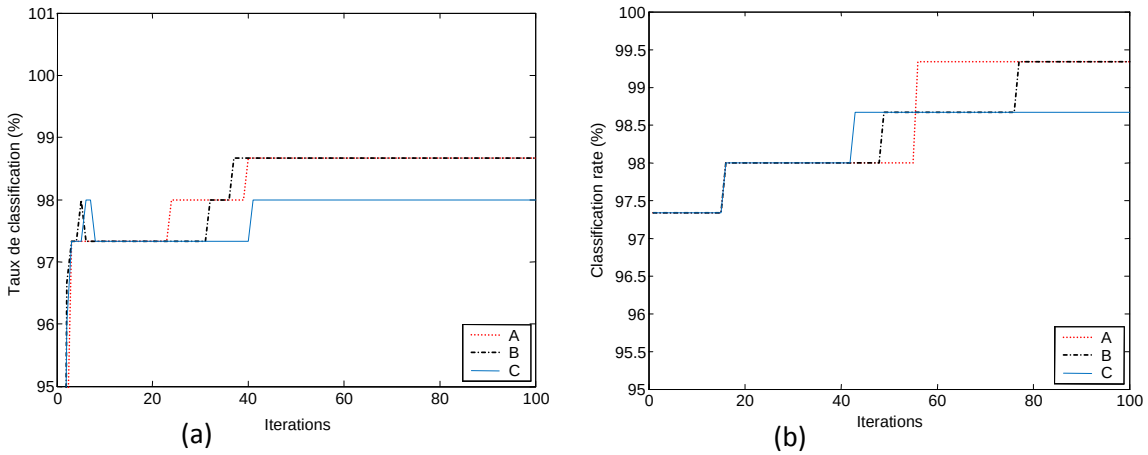


Fig. 7.5. Classification de la base de données Iris en utilisant un NFC étiqueté pour différentes valeurs des étiquettes :
(a) Apprentissage simple (mode1)
(b) Apprentissage complet (mode2)

ordinaire ne parvient à avoir ce dernier taux qu'après 80 itérations. Ces résultats montrent les améliorations obtenues grâce à la classification étiquetée en matière d'accélération d'apprentissage et d'augmentation du taux de classification. .

Comme il été mentionné dans le paragraphe (§5.5.4), les performances de classification du LNFC dépendent des fonctions d'appartenance des étiquettes plutôt que des étiquettes elles-mêmes. Plus précisément, les performances du LNFC sont influencées par le paramètre $(\beta = \mu_i(L_j), j \neq i)$ qui indique la valeur attribuée par la fonction μ_{Li} aux étiquettes L_j. Pour évaluer les performances du LNFC en fonction de ce paramètre, nous effectuons des tests avec différentes valeurs. La figure (7.5.a) montre l'effet du choix du paramètre β dans le cas du premier mode d'apprentissage de la classification étiquetée. Le graphe A correspond à $\beta = 0.9$, le graphe B correspond à $\beta = 0.8$ et le graphe C correspond à $\beta = 0.7$. Nous pouvons noter que le LNFC1 donne un taux de classification acceptable (98.67%) pour

$\mu_i(L_j) \geq 0.8$. Les résultats obtenus pour différentes valeurs de β dans le cas du deuxième mode sont illustrés dans la figure (7.5.b) ; Le LNFC2 permet l'obtention d'un taux de classification égal à $99.33\ \%$ pour $\mu_i(L_j) \geq 0.8$.

7.2.3 Classification en utilisant les systèmes multi-réseaux de neurones

Dans la base de données Iris, les exemples de la première classe peuvent être séparés facilement des autres exemples, mais il est très difficile de distinguer les exemples de la classe 2 de ceux de la classe 3.

La classification de cette base de données avec un système multi-réseaux modélisé par OAA nécessite l'utilisation de trois réseaux dont chacun se charge de classifier l'une des classes contre les deux restantes. Nous employons des réseaux avec 5 neurones d'entrée, 4 neurones cachés et 1 neurone de sortie. Le premier réseau, entrainé pour séparer la classe 1 contre les classes 2 et 3, permet d'obtenir un taux de classification égal à 100% après quelques itérations (moins de 20 itérations). En revanche, le deuxième et le troisième réseaux ne peuvent pas avoir un taux de 100% même s'ils sont entrainés pour un grand nombre d'itérations. La décomposition de ce problème par la méthode OAA a suscité donc trois sous-problèmes dans lesquels le deuxième et le troisième sont plus difficiles que le premier. Nous utilisons par conséquent un système étiqueté comportant deux réseaux étiquetés : le deuxième et le troisième. La Figure (7.6) illustre l'évolution du taux de classification d'un MLP, LMLP1 et LMLP2 destiné à résoudre ces deux sous-problèmes. Nous constatons des améliorations dans les deux cas, notamment au niveau du premier où le LMLP2 a permis d'avoir un taux de classification de 98.67% ; le LMLP1 a permis d'avoir 98% tandis que le MLP conventionnel ne donne que 96.67%.

Pour examiner l'effet des étiquettes, nous avons effectué des essais en utilisant différentes valeurs. Le tableau 7.1 montre les résultats obtenus ; nous pouvons noter que

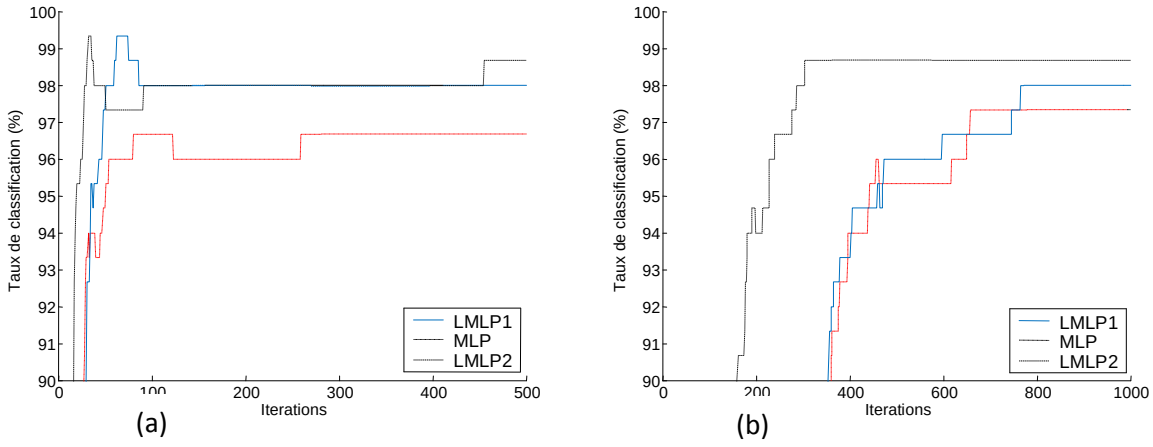


Fig. 7.6. Evolution du taux de classification d'un MLP et un MLP étiqueté durant l'apprentissage de la base de données Iris en mode OAA:
(a) 2ème réseau
(b) 3ème réseau

l'apprentissage simple donne des résultats acceptables pour $\delta \leq 0.010$, tandis que l'apprentissage complet permet d'obtenir des résultats satisfaisants avec les différentes étiquettes.

Tableau 7.1. Effet des étiquettes dans le mode OAA

Classificateur	δ	Etiquettes	Itérations	Taux de classification (%)
MLP Simple	-	-		97,33
MLP étiqueté Apprentissage simple	0.005	0.497, 0.503	820	98,00
	0.010	0.495, 0.505	680	98,00
	0.025	0.487, 0.513	810	97,33
MLP étiqueté Apprentissage complet	0.005	0.497, 0.503	265	98,67
	0.010	0.495, 0.505	270	98,67
	0.025	0.487, 0.513	280	98,67

La division de ce problème par la méthode OAO, nécessite un système multi-réseaux comportant également trois réseaux. Nous employons des MLP ayant l'architecture 5-4-1. Les premier et deuxième réseaux sont entrainés pour séparer respectivement la classe 1 contre la classe 2 et la classe 1 contre la classe 3. Ils permettent d'obtenir un taux de classification égal à $100\ \%$ après seulement quelques itérations (moins de 10 itérations), mais le troisième n'y parvient pas même s'il est entrainé pour un très grand nombre d'itérations. Dans ce cas-ci, le MLP risque souvent d'entrer dans des régions de saturation. Il nécessite ainsi un grand nombre d'itérations pour s'échapper, voire ne jamais sortir. Un exemple d'une telle situation est illustré dans figure (7.7a).

Nous utilisons par conséquent un système étiqueté comportant un seul réseau étiqueté destiné à résoudre le troisième sous-problème. La figure (7.7) illustre l'évolution du taux de classification pendant l'apprentissage d'un MLP, LMLP1 et LMLP2. Nous avons traité le cas

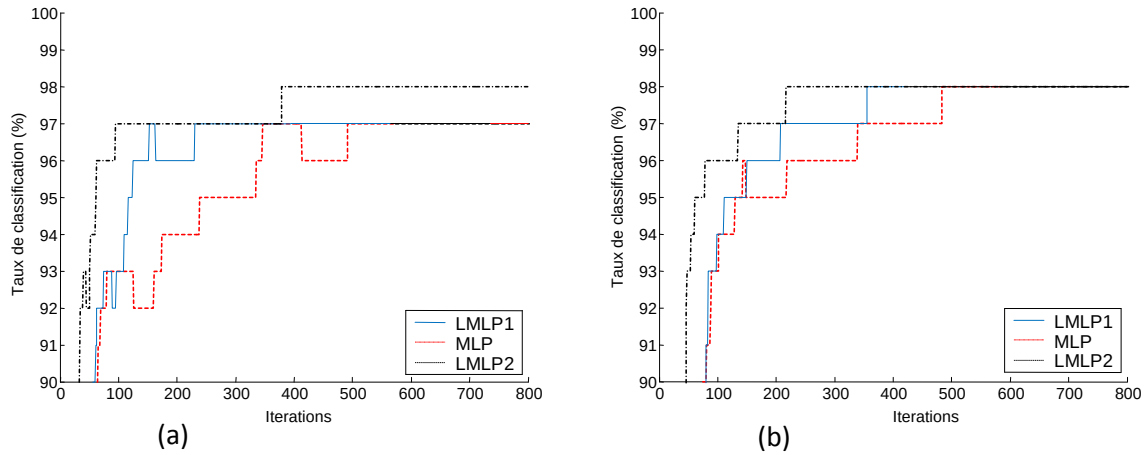


Fig. 7.7. Evolution du taux de Classification durant l'apprentissage du troisième sous-problème (C2 contre C3) de la base de données Iris en mode OAO:
(a) 1ère exécution (non convergence)
(b) 2ème exécution (convergence)

où le MLP ne peut pas dépasser le seuil de 97% (fig. 7.7.a) et le cas où il parvient à avoir 98% (fig. 7.7.b). Ces figures indiquent les développements obtenus par l'utilisation de la classification étiquetée. Dans le premier cas, le LMLP2 a permis de dépasser le seuil de 97%, tandis que dans le deuxième cas, les améliorations se sont limitées à l'accélération de l'apprentissage.

Pour examiner l'effet des étiquettes dans le cas de la modélisation OAO, nous avons effectué des tests avec différentes valeurs pour les deux modes d'apprentissage de la classification étiquetée. Les résultats obtenus sont illustrés dans le tableau 7.2 : l'apprentissage complet permet d'obtenir des résultats satisfaisants pour une gamme d'étiquettes plus grande que celle de l'apprentissage simple.

Tableau 7.2. Effet des étiquettes dans le mode OAO

Classificateur	δ	Etiquettes	Itérations	Taux de classification (%)
Apprentissage simple	0.005	$0.497, 0.503$	350	$98{,}00$
	0.010	$0.495, 0.505$	450	$98{,}00$
	0.025	$0.487, 0.513$	200	$97{,}00$
Apprentissage complet	0.005	$0.497, 0.503$	200	$98{,}00$
	0.010	$0.495, 0.505$	205	$98{,}00$
	0.025	$0.487, 0.513$	250	$98{,}00$

7.2.4 Comparaison des résultats

La base de données iris a été abondamment utilisée pour l'évaluation des nouvelles approches de classification. Le tableau (7.1) illustre nos résultats ainsi que quelques résultats obtenus sur cette base de données ; nous avons choisi des exemples de méthodes statistiques, floues et neuro-flous.

Tableau 7. 3. Différents résultats de classification de la base de données Iris

Nos résultats	MLP	98.00	LMLP1	98.00	LMLP2	98.00
	NFC	98.67	LNFC1	98.67	LNFC2	99.33
	MultiNet OAA	97.33	LMultiNet1 OAA	98.00	LMultiNet1 OAA	98.67
	MultiNet OAO	98.00	LMultiNet1 OAO	98.67	LMultiNet2 OAO	98.67
Méthode statistiques	LDA [82] RDA [82]	98.00 98.00	Quadrat. [82]	97.33	Quadrat. [83]	98.00
Flous et AG-flouS	GA-flou[84]	98.67	Flou [85]	98.67	GA-flou [86]	100
Neuro-flous et AG- neuro-flous	AG-neuro-flou[62]	97.33	FugeNeSys[87]	100		

A la lecture du tableau (7.3), nous constatons en premier lieu que, dans notre travail, le NFC donne le meilleur taux de classification et l'application de la classification a permis d'améliorer davantage ses performances. Le système multi-réseaux modélisé par OAO est plus performant que celui modélisé par OAA, et l'application de la méthode proposée a permis d'améliorer les performances de ce dernier.

Par ailleurs, le tableau (7.3) indique que les systèmes flous et systèmes neuro-flous combinés avec les algorithmes génétiques sont généralement plus performants que les méthodes statistiques et les méthodes floues, et que nos modèles rejoignent les meilleurs résultats obtenus sur cette base de données.

7.3 Classification de la base de données du vin

La base de données du vin [88] a été obtenue après une analyse chimique des vins cultivés dans la même région, mais dérivés de trois cultivars différents. Cette base de données contient 178 exemples représentés par 13 caractéristiques, et appartenant à 3 classes. La première classe comporte 59 exemples, la seconde comporte 71 exemples et la troisième comporte 48. Nous utilisons la base entière pour l'apprentissage et le test. Cette base de données, considérée facile, est souvent utilisée pour valider les nouvelles méthodes de classification. Bien que les trois modèles étudiés parviennent à classifier parfaitement tous les exemples de cette base de données, nous appliquons la classification étiquetée pour s'assurer de ses performances.

7.3.1 Classification en utilisant un MLP et un MLP étiqueté

Les caractéristiques de cette base de données sont de rangs très différents ce qui dégrade le rendement du MLP utilisé. Dans de telles situations, le réseau ajuste les poids à partir des caractéristiques de plus grand rang bien davantage qu'aux caractéristiques de rangs inférieurs. Nous procédons donc à une normalisation des données afin d'éviter ces

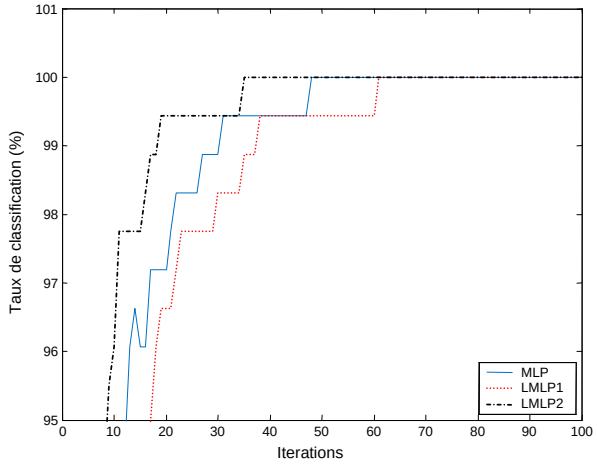


Fig. 7.8. Evolution du taux de classification durant l'apprentissage d'un MLP et un MLP étiqueté pour la classification de la base de données du vin.

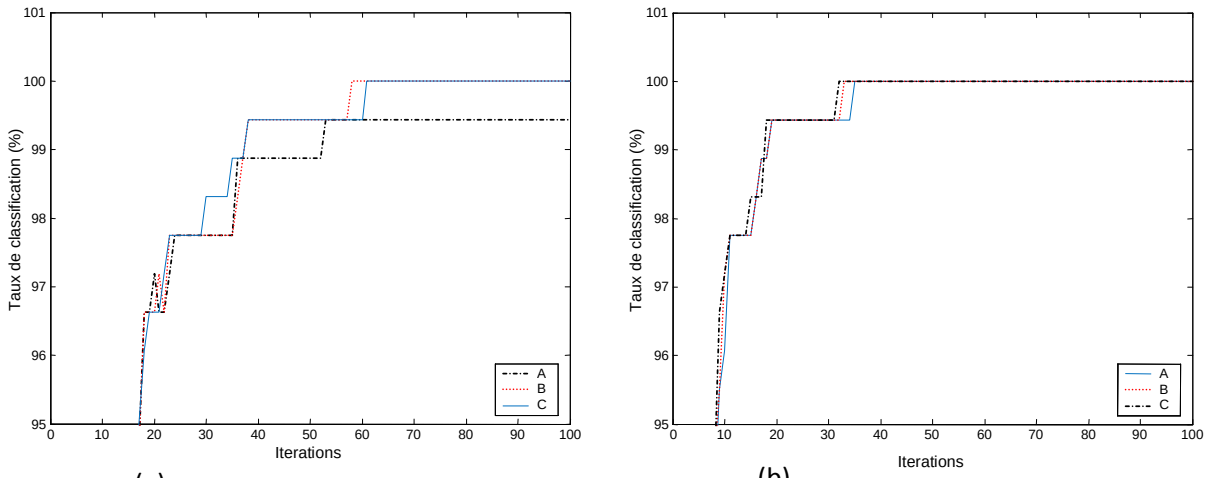


Fig. 7.9. Classification de la base de données du vin en utilisant un MLP étiqueté pour différentes valeurs des étiquettes :
(a) Apprentissage simple
(b) Apprentissage complet

difficultés. Cela permet au MLP de bien classifier tous les exemples de cette base de données.

La figure (7.8) illustre l'évolution du taux de classification pendant l'apprentissage d'un MLP et d'un MLP étiqueté. Les deux modèles ont la même architecture (6 neurones cachés), les mêmes poids initiaux et les mêmes paramètres. Les étiquettes utilisées sont $L_1 = 0.49$, $L_2 = 0.5$ et $L_3 = 0.51$ ($\delta = 0.010$). Les graphes de cette figure indiquent les améliorations obtenues par la classification étiquetée ; le MLP2 permet d'obtenir un taux de classification égal à $100\ \%$ après 35 itérations tandis que le MLP donne ce taux après 50 itérations. Les améliorations apportées se limitent donc à une simple accélération de l'apprentissage.

La figure (7.9.a) illustre l'effet des étiquettes dans le cas du premier mode d'apprentissage de la classification étiquetée. Le graphe A correspond *à* δ=0.050 ($L1$=0.45, $L2$=0.5 et $L3$=0.55), le graphe B correspond *à* δ=0.025 et le graphe C correspond à δ=0.010. Ces graphes montrent que le LMLP1 donne des résultats acceptables pour $\delta \leq 0.025$. La figure (7.9.b), qui illustre l'effet des étiquettes dans le cas d'apprentissage complet, montre que Le LMLP2 donne les mêmes résultats pour ces différentes étiquettes.

7.3.2 Classification en utilisant un NFC et un NFC étiqueté

Pour la fuzzification cette base de données, nous utilisons deux variables linguistiques à chacune des caractéristiques (fig. 7. 10).

L'évolution du taux de classification pendant l'apprentissage du NFC et du NFC étiqueté est montrée dans la Fig. (7. 11). Dans les deux cas, les poids initiaux et les paramètres d'apprentissage sont les mêmes. Les fonctions d'appartenance utilisées pour les étiquettes satisfont $\mu_i(L_i) = 1$ et $\mu_i(L_j) = 0.85$. Les graphes représentés sur cette figure indiquent

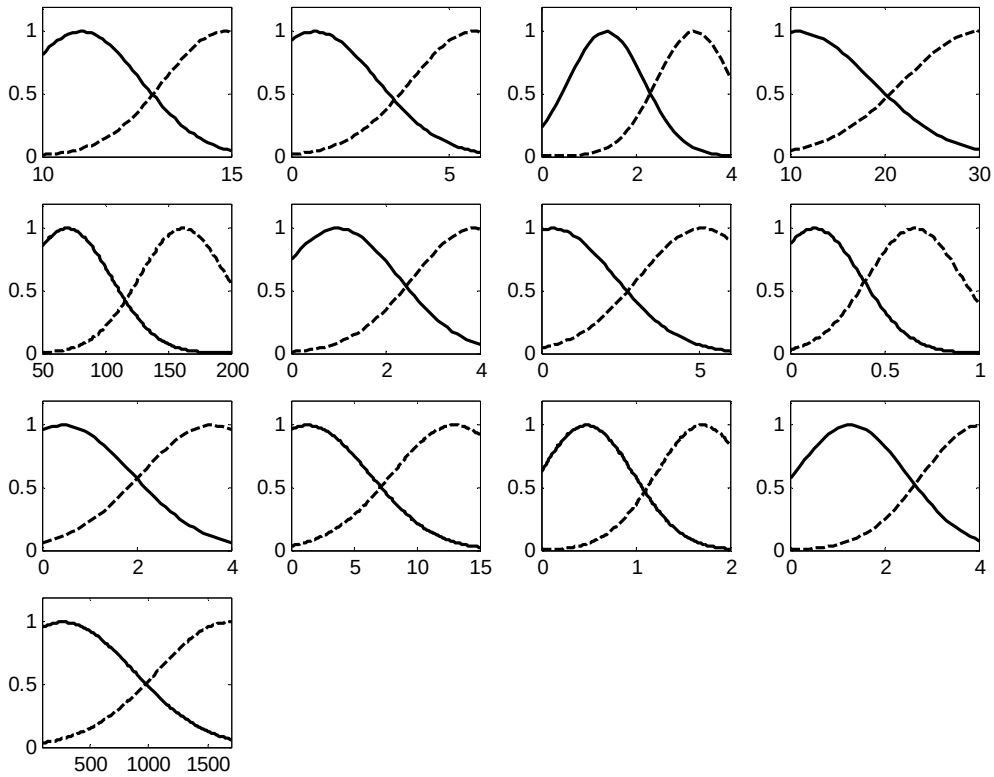


Fig. 7.10 Fonctions d'appartenance utilisées pour la fuzzification des caractéristiques de la base de données du vin

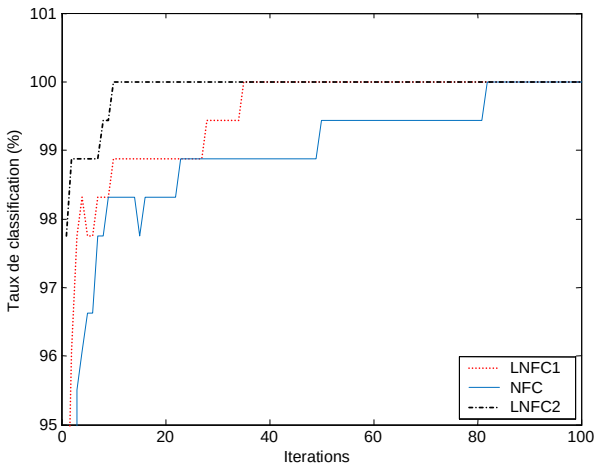


Fig. 7.11. Evolution du taux de classification durant l'apprentissage d'un NFC et un NFC étiqueté pour la classification de la base de données du vin

les améliorations obtenues par la classification étiquetée. Celle-ci permet d'obtenir un taux de classification égal à $100\ \%$ après 10 itérations en utilisant LNFC2 et après 40 itérations par le LNFC1, tandis que le NFC simple donne ce taux après 80 itérations.

Pour apercevoir l'effet des fonctions d'appartenance des étiquettes, nous avons effectué des tests pour différentes valeurs du paramètre β. Les résultats obtenus, en utilisant les deux

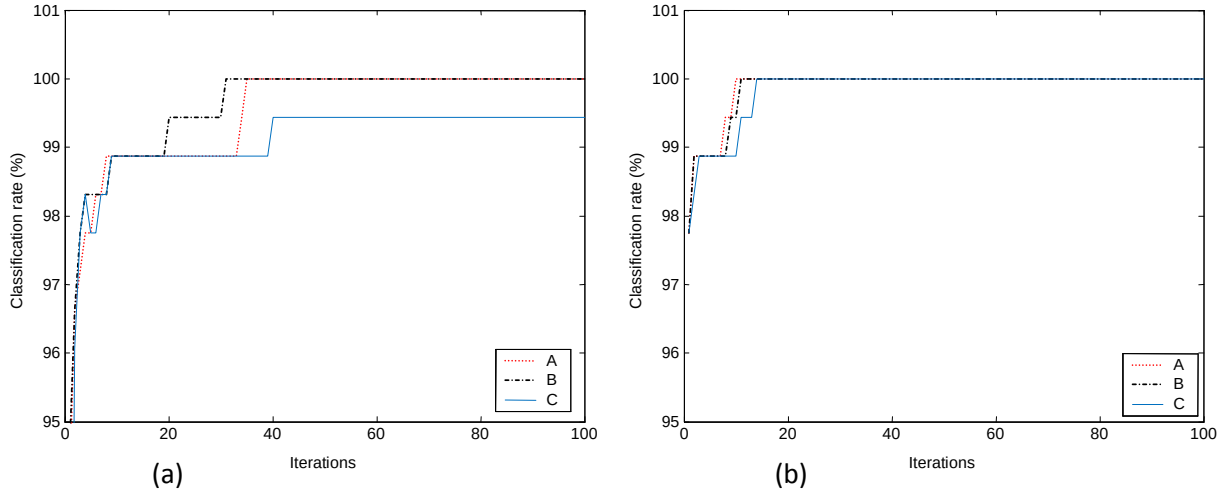


Fig. 7.12. Résultats de classification de la base de données du vin en utilisant un NFC étiqueté pour différentes valeurs des étiquettes :
(a) Apprentissage simple
(b) Apprentissage complet

modes de la classification étiquetée, sont illustrés dans la figure (7.12). Dans les deux parties de celle-ci, les graphes A correspondent à $= 0.9$ ($\mu_i(L_i) = 1$ et $\mu_i(L_j) = 0.9$), les graphes B correspondent à $\beta = 0.8$ et les graphes C correspondent à $\beta = 0.7$. Ces résultats montrent que le LNFC1 permet d'avoir des résultats acceptables pour $\mu_i(L_j) \geq 0.8$, tandis que LNFC2 le permet pour toutes ces étiquettes.

7.3.3 Classification en utilisant les systèmes multi-réseaux de neurones

La classification de cette base de données se réalise facilement par les systèmes multi-réseaux en utilisant les deux méthodes de décomposition : OAA et OAO. La figure (7.13.a)

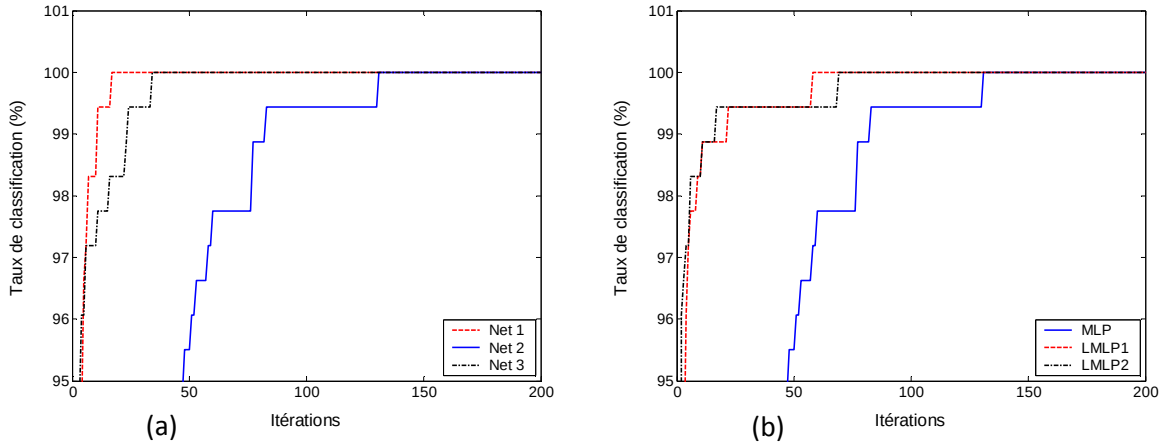


Fig. 7.13. Classification de la base de données du vin par un système multi-réseaux basé sur OAA:
(a) Evolution du taux de classification pour les 3 sous-problèmes avec des MLP simples
(b) Evolution du taux de classification pour le troisième sous-problème en utilisant un MLP et un MLP étiqueté

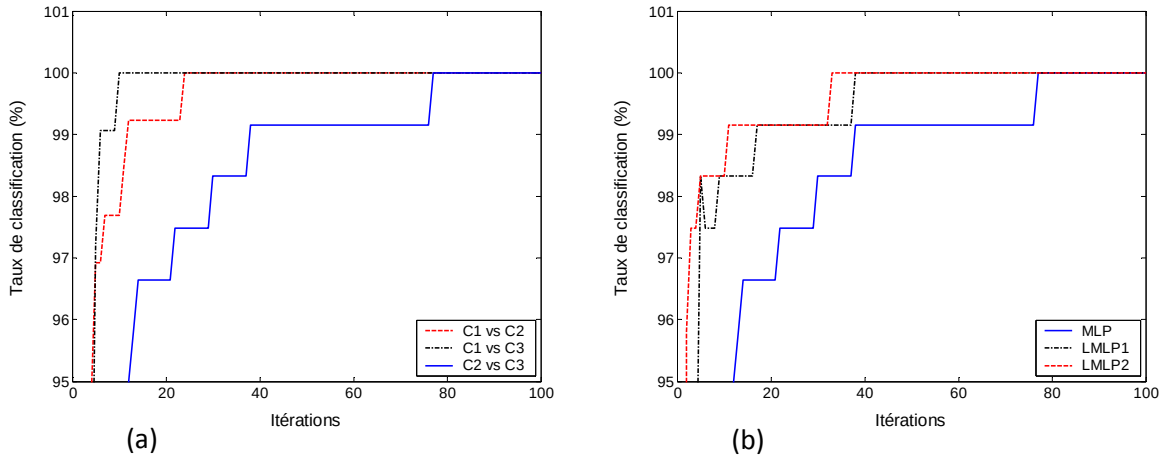


Fig. 7.14. Classification de la base de données du vin par un système multi-réseaux basé sur OAO:
(a) Evolution du taux de classification pour les 3 sous-problèmes avec des MLP simples
(b) Evolution du taux de classification pour le troisième sous-problème par un MLP et un MLP étiqueté

illustre l'évolution des taux de classification dans le cas du OAA : les trois réseaux permettent l'obtention d'un taux de classification de 100%. Toutefois, nous remarquons que le troisième réseau nécessite un nombre d'itérations un peu plus important que les autres. Nous employons ainsi un système étiqueté ayant un troisième réseau étiqueté. La figure (7.13.b) représente l'évolution des taux de classification d'un MLP simple et un MLP étiqueté : nous constatons une accélération d'apprentissage en utilisant les deux modes d'apprentissage de la classification étiquetée.

Les résultats obtenus en utilisant un système multi-réseaux basé sur la méthode OAO sont illustrés sur la figure (7.14.a). Les trois réseaux permettent également l'obtention d'un taux de classification égal à 100%, néanmoins le troisième nécessite plus d'itérations. Nous employons ainsi un système étiqueté ayant un troisième réseau étiqueté. L'évolution des taux de classification d'un MLP simple et un MLP étiqueté, chargés de cette partie du problème, est illustrée à la figure (7.14.b) qui montre une accélération de l'apprentissage en utilisant les deux modes de la classification étiquetée.

7.3.4 Comparaison des résultats

La classification de la base de données du vin est relativement facile et les trois modèles étudiés parviennent à classifier parfaitement tous les exemples. Les classificateurs proposés y parviennent aussi et les améliorations obtenues se limitent à l'accélération de l'apprentissage. Cette base de données a été bien classifiée dans plusieurs autres travaux [62][87] [89] [90]. Néanmoins, quelques approches n'ont permis que l'obtention d'un taux de classification égal à 99.4% [91][92].

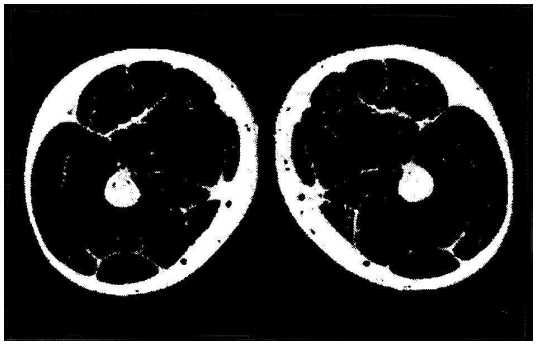


Fig.7.15. Image d'une cryosection de cuisses humaines

7.4 Classification de la base de données Cuisse humaine

L'image de la figure (7.15), prise du site "National Library of Medicine", est acquise par la photographie couleur de cryosection d'une cuisse humaine. Ce mode consiste à immerger le corps dans un gel, le congeler puis le découper en couches (de 1 mm pour l'homme et de 0.33 pour la femme) qui sont arasées avant d'être photographiées. Cette image a une taille de $670*415$ pixels dont chacun est caractérisé en RVB (couleur rouge, vert et bleu). Le niveau de variation des couleurs est entre 0 et 255. Une classification manuelle a été faite par un expert (médecin anatomiste) et quatre tissus ont été identifiés (graisse, os, moelle et muscle). Chacun de ces tissus correspond à une classe comportant 300 pixels. L'échantillon obtenu est ainsi constitué de 1200 pixels.

Pour évaluer les performances de généralisation des classificateurs étudiés, nous utilisons une validation croisée d'ordre 4. Quatre ensembles d'apprentissage, dont chacun se constitue de 900 pixels, sont ainsi obtenus. Les ensembles de test correspondants contiennent 300 pixels. Les tests effectués 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 [93][94]. Chaque pixel sera donc représenté par cinq caractéristiques : les trois couleurs RVB et les deux composantes de sa position géométrique.

7.4.1 Classification en utilisant un MLP et un MLP étiqueté

Pour la classification de cette base de données, nous utilisons un MLP et un MLP étiqueté ayant 6 neurones à l'entrée, 8 neurones cachés et 4 neurones de sortie. Ces deux réseaux ont les mêmes paramètres d'apprentissage et initialisés avec les mêmes poids. Le tableau (7.4) illustre le les résultats obtenus (les moyennes des 4 bases de données issues de la validation croisée). Ce tableau montre que les performances du MLP conventionnel sont meilleures que ceux du MLP étiqueté ce qui indique l'échec de la classification étiquetée dans ce problème.

Tableau 7.5. Résultats de classification de la base de données de la cuisse humaine par un MLP et un MLP étiqueté

Classificateur	Etiquettes	Taux de classification (%)
MLP		98.17
LMLP1 (apprentissage simple)	0.425 0.475 0.525 0.575 (δ = 0.050)	97.83
	0.485 0.495 0.505 0.515 (δ = 0.010)	97.92
LMLP2 (apprentissage complet)	0.470 0.490 0.510 0.530 (δ = 0.020)	97.92
	0.485 0.495 0.505 0.515 (δ = 0.010)	97.92

7.4.2 Classification en utilisant un NFC et un NFC étiqueté

Pour la fuzzification de cette base de données, nous employons trois variables linguistiques pour chaque caractéristique (figure 7.16). Le tableau (7.6) illustre les résultats obtenus en utilisant un NFC et un NFC étiqueté entrainés avec les mêmes paramètres d'apprentissage et initialisés avec les mêmes poids.

Tableau 7.6. Résultats de classification de la base de données de la cuisse humaine par un NFC et un NFC étiqueté

Classificateur	Etiquettes	Taux de classification (%)
NFC		97.92
LNFC1 (Apprentissage simple)	μi (Li) =1 μi (Lj)=0.7	97.92
	μi (Li) =1 μi (Lj)=0.9	98.08
LNFC2 (Apprentissage complet)	μi (Li) =1 μi (Lj)=0.7	98.08
	μi (Li) =1 μi (Lj)=0.9	98.08

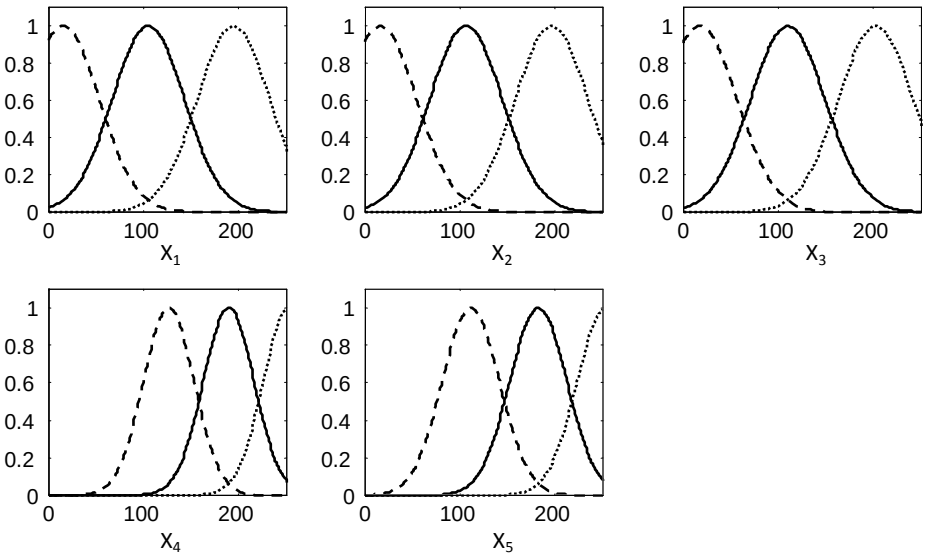


Fig. 7.16. Fonctions d'appartenance utilisées pour la fuzzification des caractéristiques de la base de données de la cuisse humaine

Nous constatons à partir du tableau (7.6) une amélioration des performances de classification en utilisant les deux modes d'apprentissage de la classification étiquetée. Nous constatons également que le deuxième mode accorde plus de souplesse dans le choix des étiquettes.

7.4.3 Classification en utilisant les systèmes multi-réseaux de neurones

La décomposition du problème de classification de cette base de données en utilisant la méthode OAA nécessite l'utilisation d'un système ayant 4 réseaux, tandis qu'en utilisant la méthode OAO cela nécessite un système ayant 6 réseaux.

Dans cette base de données, il est très difficile de séparer les exemples de la classe 1 de ceux de la classe 2. Il s'en suit que dans la méthode OAA, le troisième et le quatrième réseau séparent facilement les exemples de la troisième et la quatrième classe 3 des autres exemples. Ces deux réseaux parviennent à classifier l'ensemble de tous les exemples d'apprentissage après moins de 50 itérations, tandis que le premier et le second n'y parviennent pas même s'ils sont entrainés pour un grand nombre d'itérations. Nous employons par conséquent un système qui contient deux réseaux étiquetés correspondants aux premier et deuxième sous-problèmes.

Dans le mode OAO, le premier sous-problème (séparant les exemples de la classe 1 de ceux de la classe 2) est le plus difficile. Le système étiqueté que nous employons comprend donc un seul réseau étiqueté chargé de ce sous-problème.

Le tableau (7.7) récapitule les résultats obtenus par des systèmes conventionnels et des systèmes étiquetés basés sur les deux méthodes de décompositions : OAA et OAO.

Tableau 7.7. Résultats de classification de la base de données de la cuisse humaine par les systèmes multi-réseaux

Méthode			Taux d'apprentissage (%)	Taux de test (%)
OAA 4 réseaux	Système simple		97.88	97.58
	Système étiqueté (2 réseaux étiquetés)	mode 1	99.38	98.42
		mode 2	99.58	98.83
OAO 6 réseaux	Système simple		99.28	98.75
	Système étiqueté (1 réseau étiqueté)	mode 1	99.38	98.41
		mode 2	99.44	98.83

A partir du tableau (7.7), nous remarquons en premier lieu que le système conventionnel modélisé par OAO donne des résultats plus importants que ceux du système basé sur OAA. Nous constatons aussi que la classification étiquetée améliore les performances du système modélisé par OAA, et lui permet d'avoir des résultats comparables avec ceux du système basé sur l'approche OAO. En revanche, l'application de la classification étiquetée avec le système basé sur la méthode OAO n'accorde qu'une légère amélioration en utilisant son

deuxième mode d'apprentissage, et même une dégradation des performances en utilisant son premier mode.

7.4.4 Comparaison des résultats

Dans cette section nous discutons les résultats obtenus en utilisant les trois modèles proposés, ainsi que ceux du système d'apprentissage supervisé par génération de règles (SUCRAGE, Supervised Classification by Rule Automatic Generation). Le tableau (7.8), qui illustre les différents résultats, montre que le système multi-réseaux de neurones modélisé par OAO donne des résultats meilleurs que ceux du MLP et du NFC. L'application de la classification avec le système multi-réseaux modélisé par OAA a permis d'améliorer ses performances en lui permettant d'avoir un taux de classification égal à celui obtenu avec le système basé sur OAO. ce tableau montre également que les performances du SUCRAGE sont bien meilleures que ceux du MLP, du NFC et des systèmes multi-réseaux mêmes avec l'application de la classification étiquetée.

Tableau 7.8. Différents résultats de classification de la base de données cuisse humaine

Nos résultats	MLP	98.17	LMLP1	97.92	LMLP2	97.92
	NFC	97.92	LNFC1	98.08	LNFC2	98.08
	MultiNet OAA	97.58	LMultiNet1 OAA	98.42	LMultiNet1 OAA	98.83
	MultiNet OAO	98.75	LMultiNet1 OAO	98.41	LMultiNet2 OAO	98.83
SUCRAGE	Numérique [93]	99.08	Symbolique [94]	99.08		

7.5 Classification de la base de données Texture

La figure (7.17) est une image constituée de deux microtextures différentes. Un prétraitement (calcul des différentes corrélations locales) de l'image initiale a donné

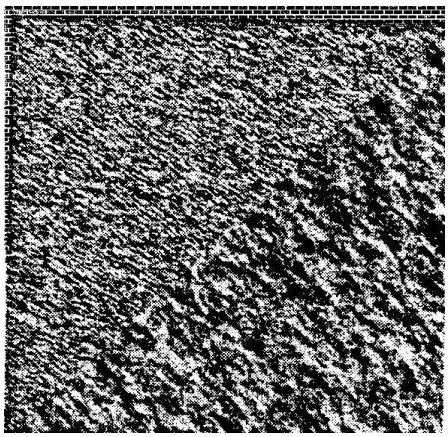


Fig. 7.17. Image des textures

naissance à une série de 8 images dont chacune est le résultat d'une détection d'un attribut particulier [95]. Chaque pixel est ainsi décrit par un vecteur à 8 attributs et chacune des deux classes comprend 400 pixels. 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.

7.5.1 Classification en utilisant un MLP et un MLP étiqueté

Le MLP et le LMLP utilisés pour la classification de cette base de données se composent de 9 neurones à l'entrée, 8 neurones cachés et 2 neurones de sortie. Les poids initiaux et les paramètres d'apprentissage employés sont les mêmes. Les résultats obtenues, présentés dans le tableau (7.9), indique des améliorations importantes grâce à l'application de la classification étiquetée. Ces améliorations sont plus importantes dans le cas du deuxième mode d'apprentissage ; le MLP2 a permis de classifier tous les exemples de tests.

Tableau 7.9. Résultats de classification de la base de données texture par un MLP et un MLP étiqueté

Classificateur	Etiquettes	Taux de test (%)
MLP	-	99.125
LMLP apprentissage simple	0.475 0.525 (δ = 0.050)	99.375
	0.490 0.510 (δ = 0.020)	99.50
LMLP apprentissage complet	0.475 0.525 (δ = 0.050)	100
	0.490 0.510 (δ = 0.020)	100

7.5.2 Classification en utilisant un NFC et un NFC Etiqueté

Nous effectuons la fuzzification de cette base de données en utilisant deux variables linguistiques pour chaque caractéristique (figure 7.17). Le NFC et le LNFC utilisés sont entrainé par les mêmes paramètres d'apprentissage et avec les mêmes poids initiaux. Le tableau (7.10) illustre les résultats obtenus. Nous constatons de simples améliorations en utilisant les deux modes d'apprentissage de la classification étiquetée.

Tableau 7.10. Résultats de classification de la base de données texture par un NFC et un NFC étiqueté

Classificateur	Etiquettes	Taux de test (%)
NFC		99.125
LNFC apprentissage simple	$\mu i\ (Li)$ =1 $\mu i\ (Lj)$=0.8	99.25
	$\mu i\ (Li)$ =1 $\mu i\ (Lj)$=0.9	99.25
LNFC apprentissage complet	$\mu i\ (Li)$ =1 $\mu i\ (Lj)$=0.7	99.25
	$\mu i\ (Li)$ =1 $\mu i\ (Lj)$=0.9	99.25

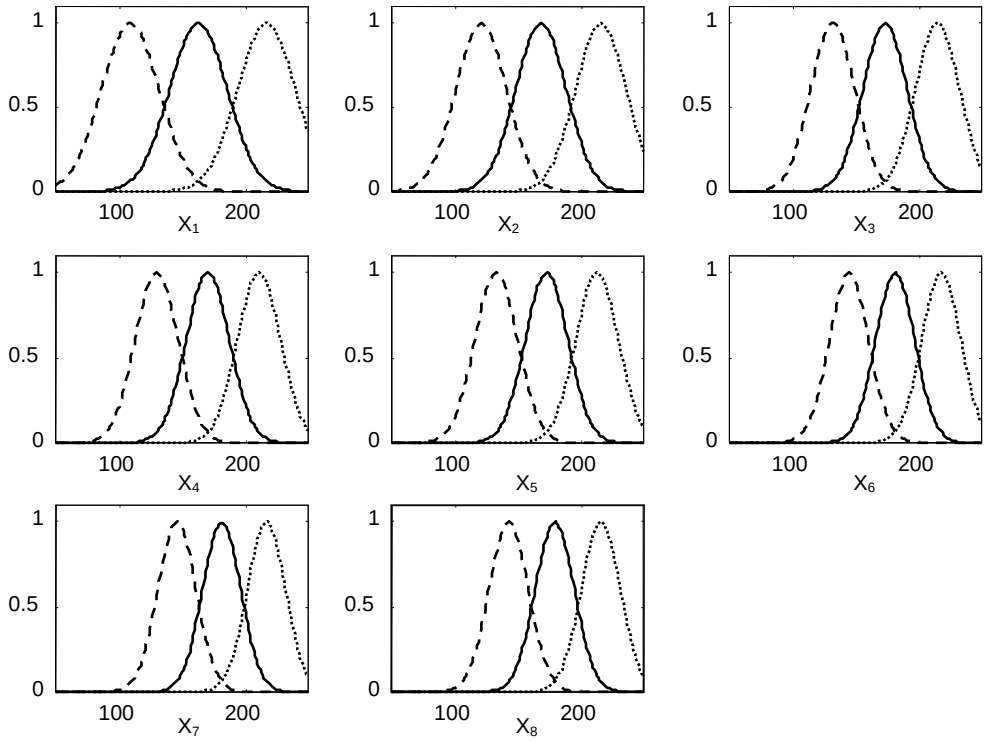


Fig. 7.18. Fonctions d'appartenance utilisées pour la fuzzification des caractéristiques de la base de données de textures.

7.5.3 Comparaison des résultats

Les résultats obtenus en utilisant nos modèles ainsi que ceux du système SUCRAGE sont illustrés dans le tableau (7.11). En permettant d'avoir un taux de classification égal à 99.125%, Le MLP et le NFC donne des résultats bien meilleurs que ceux du SUCRAGE. L'application de la classification étiquetée a permis d'améliorer davantage les performances de ces deux classificateurs. Ceci se caractérise notamment par le LMLP2 en donnant un taux de classification égal à 100%.

Tableau 7.11. Différents résultats de classification de la base de données texture

Nos résultats	MLP	99.125	LMLP1	99.5	LMLP2	100
	NFC	99.125	LNFC1	99.25	LNFC2	99.25
SUCRAGE	Numérique [93]	97.50	Symbolique [94]	97.50		

7.6 Conclusion

Les tests de classification effectués sur les bases de données précédentes montrent des résultats très variés : l'application de la classification étiquetée a permis d'avoir des améliorations acceptables dans la plupart des cas, voire très importantes dans certains cas, mais elle n'a accordé parfois que de minimes améliorations où même des dégradations des performances. Le tableau (7.12) récapitule les différents résultats obtenus : Acc. dénote accélération de l'apprentissage ; Amé. dénote amélioration du taux de classification et Deg. dénote dégradation des performances.

Tableau 7.11. Résultats d'application de la classification étiquetée sur toutes les bases de données

Base de données/ Classificateur	Iris	Vin	Cuisse humaine	Texture
MLP	Acc.	Simple Acc.	Deg	Acc et Amé
NFC	Acc. et Amé.	Acc.	Simple Amé.	Simple Acc
MutlNet OAA	Acc. et Amé.	Simple Acc.	Amé.	-
MutlNet OAO	Acc. et Amé.	Simple Acc.	Simple Amé.	-

Les résultats reportés sur le tableau (7.12) montrent que les performances de classification ont été généralement améliorées par la classification étiquetée, à l'exception du cas de la classification de la base de données de la cuisse humaine par le MLP. Toutefois, les performances de classification de cette base de données en utilisant le système multi-réseaux basé sur OAA sont améliorées. Cela peut mener à conclure que la méthode proposée est plus performante quand le nombre de classes est petit.

Par ailleurs, il est à noter que le temps d'apprentissage du LMLP1, pour un nombre donné d'itérations, est exactement le même que celui d'un MLP ordinaire, tandis que celui d'un LMLP2 égale K fois le temps du MLP (K est le nombre de classes). En revanche, le temps de classification du LMLP1 ou du LMLP2 est égal à K fois le temps nécessaire à un MLP. Cela ne pose pas de grand problème du fait que le MLP, après son apprentissage, effectue une classification rapide. D'ailleurs, ceci a constitué l'une des motivations qui nous ont menés à proposer cette méthode. Quant aux LNFC1 et LNFC2, les constatations précédentes sont aussi valables pour eux.

Conclusion générale et perspectives

Dans ce travail, nous avons proposé une nouvelle méthode de classification dont l'objectif est d'améliorer les performances des classificateurs entrainés par la RP. En nous basant sur cette méthode, nous avons introduit trois modèles de classification.

Afin de détailler nos apports, nous avons organisé cette thèse autour de trois parties dont l'objectif de la première est de situer le lecteur dans le contexte de notre travail et lui permettre de suivre les démarches faites dans cette étude. Dans la deuxième, nous avons analysé avec un peu de détail les performances de classification des systèmes étudiés. Finalement, la méthode proposée a été décrite et évaluée dans la troisième partie.

A l'issue de la deuxième partie, réservée à la classification par le perceptron multicouche et le classificateur neuro-flou, nous avons tiré les constats ayant motivé notre étude : nous avons noté que ces classificateurs permettent de bonnes capacités de classification, mais ils nécessitent une amélioration de leurs apprentissages. Nous avons également noté qu'ils admettent quelques propriétés communes. En premier lieu, les problèmes linéairement séparables sont plus faciles à résoudre par ces modèles. Deuxièmement, leurs sorties fournissent des estimés des probabilités a posteriori permettant leur utilisation dans d'autres processus de prise de décision. Troisièmement, après la phase d'apprentissage, ces classificateurs permettent une classification rapide. Nous avons ainsi proposé une méthode qui vise l'amélioration de l'apprentissage de ces classificateurs et qui met à profit leurs propres propriétés. L'idée de base consiste à simplifier l'entrainement en rendant les exemples d'apprentissage linéairement séparables par ajout des étiquettes, et d'effectuer des tests avec celles-ci pour classer les nouveaux exemples.

Nous avons d'abord introduit deux modèles de classification qui se basent sur l'application de la méthode proposée sur le perceptron multicouche et le classificateur neuro-flou. Nous avons ensuite étendu l'application de cette méthode avec les systèmes multi-réseaux : notre idée est de concevoir un système qui tient compte de complexité des problèmes attribués aux différents réseaux. Le troisième modèle proposé, le système étiqueté multi-réseaux,

comporte donc, dans une même structure, des réseaux simples et des réseaux étiquetés destinés à résoudre les parties difficiles du problème. En nous basant sur les deux stratégies les plus utilisées pour la décomposition des problèmes multi-classes (un-contre-tous et un-contre-un), nous avons introduit deux versions de systèmes étiquetés multi-réseaux de neurones.

Afin d'analyser les améliorations apportées par la classification étiquetée, nous avons traité en premier lieu des problèmes synthétiques bidimensionnels qui nous ont permis d'examiner les frontières de décision générées. Bien que l'objectif de la classification étiquetée consiste à corriger les problèmes de la RP, nous n'avons pas limité notre analyse à traiter les cas où le MLP échoue à classifier les exemples du problème traité. Nous avons également considéré les situations où le réseau parvient à le résoudre dans le but de nous assurer des performances de la classification étiquetée dans tous les cas de figure. Cette analyse a montré des améliorations importantes apportées par l'application de la classification étiquetée. Quant aux systèmes multi-réseaux, nous avons constaté des améliorations à la fois au niveau des réseaux et au niveau des systèmes complets.

Pour mettre en valeur la méthode proposée, nous avons ensuite effectué des tests de classification sur quatre bases de données : Iris, cuisse humaine, vin et texture. Les modèles introduits ont été alors évalués sur différents types de données. Les résultats obtenus montrent que ces modèles ont prouvé leur efficacité dans la plus part des problèmes traités.

Dans les systèmes multi-réseaux de neurones, la méthode OAO accorde généralement des résultats plus importants que ceux de la méthode OAA. L'application de la classification étiquetée avec ces derniers leur a permis d'avoir des résultats comparables avec ceux des systèmes basés sur OAO. Cela permet donc d'avoir les mêmes performances avec moins de réseaux.

En résumé, les travaux de recherche menés dans de cette étude montre que l'application de la méthode proposée présente les avantages suivants :

- L'implémentation de la classification étiquetée est simple parce qu'elle ne requiert aucune modification de l'algorithme d'apprentissage du classificateur utilisé.
- La classification étiquetée est une méthode générale pouvant être utilisée avec plusieurs classificateurs.
- L'apprentissage du CNF étiqueté s'effectue sans adaptation des fonctions d'appartenance, ce qui permet : une simplicité d'apprentissage, de garder le sens original des fonctions d'appartenance et la possibilité de changement du type des fonctions d'appartenance et des opérateurs OU-flous sans modification de l'algorithme d'apprentissage.

Néanmoins, nous avons constaté l'échec de cette méthode dans le cas de la classification de la base de données de la cuisse humaine par le MLP. Cela pourrait nous mener à conclure que la classification étiquetée est plus performante quand le nombre de classes est petit. En

outre, le premier mode d'apprentissage de cette méthode dépend considérablement des étiquettes, notamment dans le cas du MLP étiqueté, tandis que le deuxième mode est plus performant, mais son processus est plus compliqué et nécessite plus de temps.

L'aspect général de la classification étiquetée ouvre, à notre avis, la voie à de nombreuses applications et de divers développements. Nous voudrons ainsi conclure cette thèse en donnant quelques perspectives, que nous résumions dans les points suivants :

- D'abord, nous souhaitons vivement l'implémentation de cette méthode dans des applications réelles.
- Nous envisageons l'automatisation du processus de choix des étiquettes et sa combinaison avec les méthodes de sélection des caractéristiques. Nous pensons que ces méthodes pourront être exploitées pour l'introduction d'un mécanisme permettant la génération systématique des étiquettes, voire la génération de nouvelles représentations basées sur la combinaison des étiquettes avec les caractéristiques initiales des exemples traités.
- Nous envisageons également l'intégration d'un processus de sélection des règles dans le classificateur neuro-flou. Cela permettrait de réduire considérablement la taille du réseau et de simplifier son processus d'apprentissage.
- Comme il a été constaté dans les différents résultats, le premier mode d'apprentissage de la méthode proposé est plus simple que le deuxième, en revanche ce dernier accorde plus de souplesse dans le choix des étiquettes. Il serait donc intéressant de mener une étude permettant l'exploitation des propriétés de ces deux modes.

Références

[1] R. O. Duda, P. E. Hart and D. G. Stork. *Pattern Classification*, 2nd ed., John Wiley & Sons, New York, 2001.

[2] C. M. Bishop, *Neural networks for pattern recognition*. Clarendon press, Oxford, 1995.

[3] L. I. Kuncheva*, Combining pattern classifiers*, John Wiley & Sons, New Jersey, 2004.

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

[5] R. P. Lippmann. A critical overview of neural network pattern classifiers. *In Proc. IEEE Workshop on Neural Networks for Signal Processing*, pp. 266–275, 1991.

[6] L. Holmstrom, P. Koistinen, J. Laaksonen and E. Oja. Neural and statistical classifiers taxonomy and two case studies. *IEEE Transactions on Neural Networks*, vol. 8, pp.5-17, 1997.

[7] E. Parzen, On estimation of a probability density function and mode, *Annals of Mathematical Statistics*, vol. 33, pp. 1065–1076, 1962.

[8] Y. Lu. Knowledge integration in a multiple classifier system. *Applied Intelligence*, vol. 6, pp. 75–86, 1996.

[9] A. Al-Ani and M. Deriche, A New Technique for Combining Multiple Classifiers using The Dempster-Shafer Theory of Evidence. *Journal of Artificial Intelligence Research*, vol. 17, pp. 333-361, 2002.

[10] Z. Hongwei, O. Basir, A Scheme for Constructing Evidence Structures in Dempster-Shafer Evidence Theory for Data Fusion*, In Proc. IEEE International Symposium on Computational intelligence in robotics and Automation,* Kobe, Japan, pp.960-965, 2003.

[11] L. I. Kuncheva, J. C. Bezdek and R. P. W. Duin, Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, vol. 2, pp. 299-314, 2001.

[12] S.B. Cho and J. H. Kim. Combining multiple neural networks by fuzzy integral and robust classification. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, pp. 380-384, 1995.

[13] S. B. Cho and J. H. Kim. Multiple network fusion using fuzzy logic. *IEEE Transactions on Neural Networks*, vol. 6, pp. 497–501, 1995.

[14] W. S. Mcculloch, and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, vol. 5, 115-133, 1943.

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

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

[17] M. L. Minsky and S.A. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1969.

[18] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning internal representation by error propagation, *Parallel distributed processing: exploration in the microstructure of cognition*, D.E.Rumelhart and J.L.McClelland edition ,MIT press Cambridge, pp. 318-362, 1986.

[19] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan, New York, 1994.

[20] J. M. Zurada, *Introduction to Artificial Neural Systems*, PWS Publishing Company, Boston, 1992.

[21] A. Nigrin, *Neural Networks for Pattern Recognition*, MIT Press, Cambridge, 1993.

[22] L. Personnaz et I. Rivals, *Réseaux de neurones pour la modélisation, la commande et la classification*, CNRS Editions, Paris, 2003.

[23] B. Widrow and M. E. Hoff, Adaptive switching circuits. *1960 IRE WESCON Convention Record*, pp. 96-104, 1960.

[24] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington D.C., 1962.

[25] D.S. Broomhead and D. Lowe, Multivariable Functional Interpolation and adaptive Networks, *Complex Systems*, Vol.2 ,321-355, 1988.

[26] J. J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons. *In Proc. of the National Academy of Sciences*, USA, vol. 79, pp. 2554-2558, 1982.

[27] T. Kohonen, *Self Organization and Associative Memory*, Springer-Verlag, New York, 1984.

[28] T. Kohnen, The Self-Organizing Map, *in proc. IEEE*, Vol.78, 1464-1480, 1990.

[29] G. Cybenco, Approximation by Superposition of a sigmoidal Function, *Math, Control, Signals and systems*, vol. 2, pp. 303-314, 1989.

[30] K. Hornik, K. M. Stinchcombe and H. Wite, Universal approximation of an unknown mapping and its derivates using multilayer feedforward networks, *Neural Networks*, vol.3, pp. 551–560, 1990.

[31] K. Hornik, Approximation capabilities of multilayer feedforward networks, *Neural Networks*, vol.4, pp. 251–257, 1991.

[32] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley and B. W. Suter, The multilayer perceptron as an approximation to a Bayes optimal discriminant function*, IEEE Transactions on Neural Networks*, vol. 1, pp. 296–298, 1990.

[33] K. Funahashi, Multilayer neural networks and Bayes decision theory. *Neural Networks*, vol. 11, pp. 209-213, 1998.

[34] M. D. Richard and R.P. Lippmann, Neural networks classifiers estimate Bayesian a-posteriori probabilities, *Neural Computation*, vol. 3, pp. 461-483, 1991.

[35] A. P. Russo, Neural Networks for sonar signal processing, tutorial No. 8, *IEEE Conf. on Neural Networks for Ocean engineering*, Wachington, D.C., 1991.

[36] D. Nguyen and B. Widrow, Improving the learning speed two-layer neural networks by choosing initial values of the adaptive weights, *in Proc. 1990 IEEE Int. Joint Conf. Neural Networks*, San Diego, vol. 3, 21-26, 1990.

[37] J. F. Shepanski, Fast learning in artificial neural systems: multilayer perceptron training using optimal estimation*, IEEE International Conference on Neural Networks 1*, IEEE Press, New York, pp. 465-472, 1988.

[38] T. Masters, *Practical Neural Network Recipes in C++*, Academic Press, Boston, 1993.

[39] Y.F. Yam and T.W.S. Chow, Determining initial weights of feedforward neural networks based on least squares method, *Neural Processing letter*, vol. 2, pp. 13-17, 1995.

[40] Y.F. Yam and T.W.S. Chow, A new method in determining the initial weights of feedforward neural networks, *Neurocomputing*, vol.16, pp. 23-32, 1997.

[41] J. Y. F. Yam and T. W. S. Chow, A weight initialization method for improving training speed in feedforward neural network, *Neurocomputing*, vol. 30, pp. 219-232. 2000.

[42] R. A. Jacobs, Increased rates of convergence through learning rate adaptation, *Neural Networks*, vol. 1, pp. 295-307, 1988.

[43] M. Zurada, Lambda learning rule for feedforward neural networks, in *Proc, IEEE int*. *Conf. Neural Networks*, vol. 3, pp. 1808-1811. 1993.

[44] P. Chandra and Y. Singh, An activation function adapting training algorithm for sigmoidal feedforward networks, *Neurocomputing*, vol. 61, 2004, pp. 429– 437.

[45] K. Eom, K. Jung and H. Sirisena, Performance improvement of backpropagation algorithm by automatic activation function gain tuning using fuzzy logic, *Neurocomputing*, vol. 50, pp. 439 – 460, 2003.

[46] Y. H. Zweiri, J. F. Whidborne and L. D. Seneviratne, Three-term backpropagation algorithm, *Neurocomputing,* Vol. 50, pp. 305-318, 2003

[47] Y. H. Zweiri, Optimization of a Three-Term Backpropagation Algorithm Used for Neural Network Learning, *International Journal of Computational Intelligence*. vol. 3, pp. 322–327, 2006.

[48] X.G. Wang, Z. Tang, H. Tamura and M. Ishii, A modified error function for the backpropagation algorithm, *Neurocomputing*, vol. 57, pp. 477- 484, 2004

[49] L. A. Zadeh, Fuzzy Sets, Journal of Information and Control, Vol. 8, pp. 338-353, 1965.

[50] W. Siler and J. J. Buckley, *Fuzzy expert systems and fuzzy reasoning*, John Wiley & Sons, New Jersey, 2005.

[51] T. Takagi and M. Sugeno, Fuzzy Identification of Systems and its Applications to Modeling and Control, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 15, pp. 116-132, 1985.

[52] M. Sugeno and G. T. Kang, (1988). Structure Identification of Fuzzy Model, *Journal of Fuzzy Sets and Systems*, Vol. 28, pp. 15–33.

[53] P. Melin and O. Castillo, *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing*, Springer Berlin Heidelberg, New York, 2005.

[54] E. H. Mamdani and S. Assilian, An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller, *International Journal of Man-Machine Studies*, vol. 7, pp. 1-13, 1975.

[55] S. Mitra and Y. Hayashi, Neuro-Fuzzy rule generation: survey in soft computing framework, *IEEE Transactions on neural networks*, vol. 11, pp.748-768, 2000.

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

[57] J. S. R. Jang, ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, pp. 665, 1993.

[58] C.T. Lin and C.S.G. Lee, Neural network-based fuzzy logic control and decision system, *IEEE Transactions on Computers*, vol. 40, pp. 1320–1336, 1991.

[59] C. T. Sun and J. S. R. Jang, A neuro-fuzzy classifier and its applications, *In proc. of IEEE international conference on fuzzy systems*, San Francisco, pp. 94-98, 1993.

[60] F. Masulli and A. Sperduti, Learning Techniques for Supervised Fuzzy Classifiers, *Fuzzy Learning and Applications*. CRC press, ch. 4, pp. 147-169, 2001.

[61] A. Lotfi, Learning Fuzzy Systems, *Fuzzy Learning and Applications*, CRC press, ch. 6, pp.205-222, 2001.

[62] B. D. Chakraborty and N. R. Pal, A Neuro-Fuzzy Scheme for Simultaneous Feature Selection and Fuzzy Rule Based Classification, *IEEE Transactions on Neural Networks*. vol. 15, pp. 110-123, 2004.

[63] M. C. Su, C.H. Chou, E. Lai and J. Lee, A new approach to fuzzy classifier systems and its application in self-generating neuro-fuzzy systems, *Neurocomputing,* Vol. 69, pp. 586-614, 2006.

[64] A. V. Nandedkar, and P. K. Biswas, A Fuzzy Min-Max Neural Network Classifier With Compensatory Neuron Architecture, *IEEE Transactions on neural networks*, vol. 18, pp.42-54, 2007.

[65] D. Nauck and R. Kruse, A neuro-fuzzy method to learn fuzzy classification rules from data *Fuzzy Sets and Systems*, Vol. 89, pp. 277-288, 1997.

[66] L. I. Kuncheva, How Good Are Fuzzy If-Then Classifiers?, *IEEE Transactions on Systems, Man and Cybernetics-part B: Cybernetics*, vol. 30, pp. 501-509, 2000.

[67] H. Ishibuchi and T. Nakashima, Effect of Rule Weights in Fuzzy Rule-Based Classification Systems, *IEEE Transactions on fuzzy Systems*, vol. 9, pp. 506-515, 2001.

[68] N. K. Kasabov, Learning fuzzy rule and approximate reasoning in fuzzy neural networks and hybrid systems, *Fuzzy Sets and Systems*, Vol. 82, pp. 135-149, 1996.

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

[70] J. Vitela and J. Reifman, Premature Saturation in Backpropagation Networks: Mechanism and Necessary Conditions, *Neural Networks*, vol. 10, pp. 721–735, 1997.

[71] G. Auda, *Cooperative modular neural networks*, PhD Thesis, University of Waterloo, Canada, 1996.

[72] B. L. Lu and M. Ito, Task decomposition and module combination based on class relations: A modular neural network for pattern classification, *IEEE Transactions on neural networks*, vol. 10, pp.1244–1256, 1999.

[73] D. Price, S. Knerr, L. Personnaz and G. Dreyfus, Pairwise neural network classifier with probabilistic outputs, *Advances in Neural information processing systems*, vol. 7, 1994.

[74] T. F. Wu and R. C. Weng, Probability Estimates for Multi-class Classification by Pairwise Coupling, *Journal of Machine Learning Research*, vol. 55, pp. 975-1005, 2004.

[75] T. Hastie and R. Tibshirani, Classification by Pairwise Coupling, *Annals of statistics*, vol. 26, pp. 451-471, 1998.

[76] G. Ou and Y. L. Murphey, Multi-class pattern classification using neural networks, *Pattern recognition*, vol. 40, pp. 4-18, 2007.

[77] J. Fürnkranz, Round robin classification, *Journal of Machine Learning research*, vol. 2, pp. 721-747, 2002.

[78] R. Anand, K. Mehrotra, C. K. Mohan and S. Ranka, Efficient classification for multiclass problem using modular neural networks, *IEEE Transactions on neural networks*, vol. 6, pp. 117-124, 1995.

[79] J. Friedman, *Another approach to polychotomous classification*, Technical report, Department of Statistics, Stanford University, 1996.

[80] D. M. J. Taxa, M. van Breukelen, R. P. W. Duina and J. Kittler, Combining multiple classifiers by averaging or by multiplying?, *Pattern Recognition*, vol. 33, pp. 1475-1485, 2000.

[81] R.A. Fisher, The use of multiple measurements in taxonomic problems, *Annals of Eugenics*, vol. 7, pp. 179-188, 1936.

[82] S. Aeberhard, D. Coomans and O. de Vel, Comparative analysis of statistical pattern recognition methods in high dimensional settings, *Pattern Recognition*, Vol. 27, pp. 1065-1077, 1994.

[83] H. Brunzell and J. Eriksson, Feature reduction for classification of multidimensional data, *Pattern Recognition*, vol. 33, pp 1741-1748, 2000.

[84] H. Ishibuchi, K. Nozaki, N. Yamamoto and H. Tanaka, Selecting fuzzy if-then rules for classification problems using genetic algorithm, *IEEE Transactions on fuzzy Systems*, vol. 3, pp. 260-270, 1995.

[85] H. Ishibuchi, T. Nakashima and T. Morisawa, Voting in fuzzy rule-based systems for pattern classification problems, *Fuzzy Sets and Systems*, vol. 103, pp. 223-238, 1999.

[86] T. H. S. Li, N. R. Guo and C. L. Kuo, Design of adaptive model for classification problem, *Engineering Applications of Artificial Intelligence*, vol. 18, pp. 297-306, 2005.

[87] M. Russo, FuGeNeSys-a fuzzy genetic neural system for fuzzy modeling, *IEEE Transactions on fuzzy Systems*, vol. 6, pp. 373-387, 1998.

[88] UCI Repository of Machine Learning Databases. [Online]. Available: http://www.ics.uci.edu/pub/machine-learning-databases

[89] A.L. Corcoran and S. Sen, Using real-valued genetic algorithms to evolve rule sets for classification. In *Proc. of IEEE World Congress on Computational Intelligence*, pp. 120-124, 1994.

[90] S.Y. Ho, H.M. Chen, S.J. Ho, T.K. Chen, Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space, *IEEE Transactions on Systems, Man and Cybernetics-part B: Cybernetics*, vol. 34, pp 1031-1044, 2004.

[91] J.A. Roubos, M. Setenes and J Abonyi, Learning Fuzzy Classification Rules From Data, *Developments in Soft Computing*, R. John and R. Birkenhead, ch. 13 , Springer-Verlag, Berlin, pp. 108-115, 2001.

[92] H. Ishibuchi, T. Nakashima and T. Murata, Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. *IEEE Transactions on Systems, Man and Cybernetics-part B: Cybernetics*, vol. 29, pp. 601-618. 1999.

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

[94] H. Seridi, *Nouvelle approche qualitative du traitement des connaissances incertaines et contribution au développement de systèmes experts symboliques*, Thèse de l'Université de Reims, 1998.

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

Bibliographie de l'auteur

Journaux internationaux

[1] H. Seridi, H. Akdag, R. Mansouri, M. Nemissi, Approximate Reasoning in Supervised Classification Systems, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 10, pp 586-593, 2006.

[2] M. Nemissi, H. Seridi and H. Akdag, The Labeled Classification and its Applications, *International Journal of computational Intelligence*, Vol. 4, 2008.

[3] M. Nemissi, H. Seridi and H. Akdag, The Labeled Systems of Multiple Neural Networks, *International Journal of Neural Systems*, Vol. 18, pp. 321- 330, 2008.

Congrès internationaux

[4] M. Nemissi, H. Boudouda, H. Seridi and H. Akdag, Classification étiquetée neuro-floue, *in Proc. Colloque sur l'optimisation et les systèmes d'information (COSI'05)*, Bejaïa-Algérie, pp.252-262, 2005.

[5] M. Nemissi, H. Boudouda and H. Seridi, Comparing performances of the MLP, RVFLN and NFC for human tight image classification, *in Proc. International Workshop On Text, Image and Speech recognition*, Annaba-Algérie, pp. 125-131, 2005.

[6] M. Nemissi, H. Seridi and M. Z. Aissaoui, Performances Evaluation of the Labeled Neuro-Fuzzy Classifier, *in Proc. Colloque sur l'optimisation et les systèmes d'information (COSI'07)*, Oran- Algérie, pp. 421–439 , 2007.

[7] H. Boudouda, H. Seridi and M. Nemissi, Hybrid Algorithm of Unsupervised Automatic Classification, *International Conference on Modeling and Simulation (MS'07)*, Alger-Algérie, 2007.