UNIVERSITÉ PARIS 13
INSTITUT GALILÉE

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

Discipline : INFORMATIQUE

présentée et soutenue publiquement

par

Tayeb NEDJARI

le 17/09/98

Titre :

Réseaux de Neurones Artificiels et Connaissances Symboliques : Insertion, Raffinement et Extraction

Thèse dirigée par : Daniel KAYSER
& suivie par : Younès BENNANI

JURY

M. Fouad BADRAN
M. Younès BENNANI
M. Dusan CAKMAKOV
M. Thierry DENOEUX
M. Daniel KAYSER
M. Gérard PLATEAU, Président

A la mémoire

des regrettés

Ahmed SAOUDI

et

Abdellah SELLAM

Remerciements

Cette thèse a été réalisée au sein du Laboratoire d'Informatique de Paris Nord (LIPN), dans l'équipe Algorithmique et Combinatoire.

Tout d'abord, je remercie Monsieur Younès BENNANI, maître de conférence à l'université Paris 13, qui m'a permis de découvrir les réseaux connexionnistes et qui a encadré mes travaux de recherche. Sa compétence, ses conseils, ses remarques constructives et ses critiques m'ont été d'une aide précieuse.

Je tiens à remercier Monsieur Daniel KAYSER, Professeur à l'université Paris 13, d'avoir accepté de diriger mes recherches. Il m'a fait profiter de sa grande expérience en lisant ce document. Ses conseils avisés ont notamment permis d'améliorer de nombreux points de ce rapport.

Je tiens aussi à remercier Monsieur Dusan CAKMAKOV, Professeur à l'université de Skopje (République de Macedonia), de l'enthousiasme avec lequel il a accepté de rapporter mon travail.

Je remercie Monsieur Fouad BADRAN, maître de conférence au Cnam, pour avoir accepté d'être rapporteur de ma thèse. Sa lecture méticuleuse m'a permis d'améliorer la présentation de ce travail.

Je voudrais également remercier Monsieur Thierry DENOEUX, enseignant-chercheur à l'université de Compiegne, d'avoir accepté de rapporter ma thèse. Ses remarques pertinentes m'ont permis de corriger plusieurs point de ce manuscrit. Qu'il trouve ici l'expression de mon profond respect.

Je tiens à remercier Monsieur Gérard PLATEAU, Professeur à l'université Paris 13, d'avoir accepté si gentiment de participer au jury de ma soutenance.

Je voudrais remercier mes collègues du LIPN qui par leur aide et amitié ont

contribué à l'aboutissement de cette thèse (*Si on peut parler d'aboutissement quand on fait de la recherche !*). Tout particulièrement, je tiens à remercier Houcine SENOUSSI pour l'aide qu'il m'a apportée dans la rédaction de ce document, je lui exprime ici toute mon amitié et ma reconnaissance. Une mention particulière à nos trois secrétaires : Brigitte, Jacqueline et Maryse pour leur gentillesse et leur dévouement.

Je remercie tout particulièrement Ahmed, Brigitte, Claudine, Djamel, Sohbi, Evelyne et leur famille. Grâce à leur hospitalité, j'ai pu supporter l'absence de ma famille et de tous ces êtres chers que j'ai laissé dans mon pays.

Je remercie mes anciens collègues de SECOIA avec qui j'ai passé deux années durant lesquelles j'ai pu apprécier leur amabilité et leur grandeur d'esprit. Je remercie aussi les collègues avec lesquels j'ai enseigné à l'université de Paris-Val de Marne et aux IUT de Marne la Vallée, Saint-Denis et Bobigny pour l'ambiance amicale et pour le travail d'équipe. Une pensée à mes élèves des deux associations MathHelp et Bondy-Nord pour leur estime et leur grand respect.

Enfin et c'est par là que j'aurais dû commencer, je ne saurais comment remercier mes parents (*si loin !*) dont le dévouement n'a d'égal que leur affection, mes soeurs pour les encouragements qu'elles m'ont prodigués, ma femme pour sa confiance et sa patience, et mes ami(e)s pour leur sympathie, aide et conseils.

Résumé

La prolifération et le grand succès des applications incorporant des Réseaux de Neurones Artificiels (RNA) dans plusieurs domaines montrent l'utilité du paradigme RNA. Néanmoins, ce paradigme a une limite : son incapacité inhérente à fournir une explication des résultats obtenus. C'est essentiellement pour vaincre cette limite que plusieurs chercheurs se sont intéressés à combiner les RNA et les systèmes symboliques de manière à profiter de leurs avantages et éviter leurs faiblesses. Dans cette thèse, nous décrivons notre contribution à ce domaine de recherche. Nos travaux s'articulent autour de trois axes : l'extraction de règles à partir d'un RNA, l'insertion de connaissances dans un RNA, et l'utilisation d'un RNA pour raffiner une base de règles existante.

Dans le premier axe : après une étude critique des principales techniques développées, nous avons proposé deux techniques d'extraction de règles. La première, $MITER$, associe à chaque neurone un calibre de poids. Ce dernier est traduit sous forme d'une règle symbolique de la forme *M-parmi-N*. La seconde, $EMIRE$, extrait des règles à partir d'un RNA sans tenir compte de sa structure interne et en utilisant uniquement ses entrées pertinentes. Dans le deuxième axe : après une présentation des principales techniques existantes, nous avons proposé deux techniques d'insertion de règles symboliques dans un RNA. La première, $RuleNeur$, associe à chaque règle écrite sous une forme normale disjonctive un neurone. La seconde, $OpNeur$, associe à chaque opérateur logique (ET ou OU) un neurone. Dans le troisième axe, après une présentation des différents systèmes hybrides, nous avons proposé le système $RANNI$ qui combine les deux axes précédents en utilisant un module probabiliste pour leur mise-à-jour.

Mots-clés : réseaux de neurones artificiels, systèmes symboliques, information mutuelle, extraction de règles, insertion des connaissances dans un réseau de neurones, raffinement de règles, systèmes hybrides.

Abstract

It is becoming increasingly apparent that without some form of explanation capability, the full potential of trained Artificial Neural Networks (ANN) may not be realized. It is particularly for this reason that hybrid connectionist-symbolic systems are developed to combine ANN with symbolic systems in order to take an advantage of their strength and avoid their weakness. In this thesis, we deal with three topics: extraction of rules from trained ANN, insertion of knowledge into ANN and the use of ANN to refine existing rules base.

In the first topic: after a survey and critiques of the principal existing techniques, we present two methods for rules extraction. The first, $MITER$, associates to each neuron a *template weight* which is interpreted in *M-of-N* symbolic rule. The second, $EMIRE$, extracts rules from ANN independently of its internal structure and using only relevant network inputs. The relevance of an input is evaluated by the use of mutual information.

In the second topic: after a survey on the principal existing techniques, we present two methods to insert symbolic rules in ANN. The first, $RuleNeur$, associates to each rule with normal disjunctive format a neuron. The second, $OpNeur$, associates to logical operator (AND or OR) a neuron.

In the last topic: after a survey on the principal existing hybrid systems, we propose the system $RANNI$ that combines the two precedent topics and uses a probabilistic module to update them.

keywords : artificial neural networks, symbolic systems, mutual information, rule extraction from neural networks, knowledge insertion into neural networks, rule refinement using neural networks, hybrid systems.

Table des matières

Introduction et notations

Introduction

Lorsque plusieurs systèmes entrent en compétition pour résoudre un certain type de problèmes, il est parfois utile de les combiner de manière à profiter des points forts de chacun d'entre eux. Cette combinaison permet en général d'obtenir de meilleures performances et d'élargir le champ d'application de chacun des systèmes.

Deux catégories de systèmes sont en compétition car d'une part, ils servent à accomplir le même type de tâches (exemple : classement) et d'autre part ils interviennent presque dans les mêmes domaines d'application (exemple : biologie). Ces deux types de systèmes sont :

1. *Les systèmes symboliques :* ils cherchent à modéliser explicitement des processus de résolution en utilisant des bases de connaissances.

2. *Les systèmes connexionnistes :* ils utilisent des réseaux d'automates simples capables d'extraire une relation pertinente liant des évènements d'un processus connu par des exemples.

Depuis quelques années, plusieurs chercheurs s'intéressent à combiner les deux types de systèmes pour obtenir une meilleure modélisation possédant une bonne capacité d'inférence sur de nouvelles données. Dans ce document, nous décrivons notre contribution à ce domaine de recherche. Cette contribution consiste en deux méthodes d'extraction de règles à partir d'un système connexionniste, deux méthodes d'initialisation d'un système connexionniste à

partir d'une base de règles et un système hybride combinant les deux systèmes symbolique et connexionniste. Ce document est organisé de la manière suivante :

- Le chapitre 1 contient une description générale des deux approches symbolique et connexionniste (Réseau de Neurones Artificiel, RNA). Nous décrivons aussi les avantages et les faiblesses de chacune des deux approches, ainsi que la complémentarité existant entre elles. Ce chapitre présente aussi l'utilité de tout modèle capable de combiner ces deux approches en profitant de leurs avantages et en évitant leurs faiblesses.

- Le chapitre 2 est un aperçu des techniques développées pour munir les réseaux de neurones artificiels de la capacité d'explication. Plus précisément, nous décrivons dans ce chapitre les mécanismes, les procédures et les algorithmes destinés à extraire des règles à partir d'un RNA. Ce chapitre introduit aussi une taxonomie pour classer les différentes techniques et évaluer leur efficacité. Ce chapitre contient une critique de l'algorithme *SubSet* [120] et une description d'une heuristique que nous proposons pour améliorer ses performances. Ce chapitre présente aussi une adaptation de l'algorithme *M-of-N* [121] pour l'extraction de règles de la forme *M-parmi-N* et *non(M)-parmi-N*. Nous présentons à la fin de ce chapitre, une nouvelle direction pour l'extraction de règles floues.

- Le chapitre 3 introduit la notion de l'information mutuelle et décrit son utilisation pour évaluer la pertinence de l'ensemble des connexions du réseau. Nous montrons dans ce chapitre comment utiliser cette évaluation pour définir des calibres de poids. Ces derniers sont traduits en règles symboliques de la forme *M-parmi-N*. Cette traduction est le principe de notre algorithme $MITER$ d'extraction de règles à partir d'un RNA qui sera présenté dans ce chapitre. À la fin de ce troisième chapitre, nous décrivons notre second algorithme, nommé $EMIRE$, pour l'extraction de règles à partir d'un RNA. $EMIRE$ ne prend en considération que les entrées pertinentes du réseau sans tenir compte de sa structure interne.

La pertinence des entrées est mesurée par l'utilisation de l'information mutuelle.

- Le chapitre 4 commence par un aperçu des différentes techniques développées pour initialiser un RNA à partir d'une base de règles. En suite, nous proposons deux nouvelles méthodes ayant le même objectif. Dans la première, nommée $OpNeur$, un réseau à une ou deux couches cachées, selon la nature de l'application, est initialisé à partir de la définition d'un concept. La seconde, nommée $RuleNeur$, initialise un réseau multicouches à partir d'une base de règles. Le nombre de conclusions intermédiaires dans cette dernière est inférieur au nombre de couches cachées dans le réseau.

- Le chapitre 5 commence par un aperçu des différents systèmes hybrides connexionnistes-symboliques. En suite, nous proposons notre système hybride $RANNI$ ayant trois composantes : un système à base de connaissances, un réseau connexionniste et un module probabiliste. Ce système permet de réviser la théorie initiale d'un domaine et d'améliorer ses caractéristiques (Raffinement de règles). Ce chapitre décrit aussi les coopérations possibles des systèmes symbolique et connexionniste dans le but d'améliorer l'apprentissage, l'optimisation d'une base de règles et la réduction de la topologie d'un RNA.

En conclusion, nous décrivons les principaux résultats de cette thèse. Nous proposons aussi quelques perspectives de recherche.

Notations

IA	Intelligence Artificielle
RNA	Réseau de Neurones Artificiels
SBC	Système à Base de Connaissances
SS	Système Symbolique
MITER	Mutual Information and Template for Extracting Rules
EMIRE	Examples and Mutual Information for Rules Extraction
OpNeur	from logical Operator to Neuron
RuleNeur	from symbolic Rule to Neuron
RANNI	Rules and Artificial Neural Networks Interaction
IM	Information Mutuelle
H(.)	entropie
H(.|.)	entropie conditionnelle
P(.)	Probabilité
P(.|.)	Probabilité conditionnelle
va	variable aléatoire
A_i	la fonction d'activation du neurone i
f	la fonction de transition
θ_i	le seuil du neurone i
W_i	le vecteur des poids d'entrée du neurone i
ω_{ij}	le poids de la connexion entre les deux neurones j et i
T_i	le calibre de poids du neurone i
t_{ij}	la j^e composante du calibre T_i
$F(.)$	la distance euclidienne entre T_i et W_i
V_i	la valeur minimisant $F(.)$

n_i	le nombre de connexions d'entrée du neurone i
m_i	le nombre de connexions d'entrée pertinentes du neurone i
m_i^P	le nombre de connexions d'entrée pertinentes et de poids positif du neurone i
m_i^N	le nombre de connexions d'entrée pertinentes et de poids négatif du neurone i
M_i	le nombre de connexions d'entrée nécessaire à l'activation du neurone i
x_{ij}	l'entrée j du neurone i
y_i	la sortie du neurone i
X_i	l'ensemble des connexions d'entrée du neurone i
X_i^*	l'ensemble des connexions d'entrée pertinentes du neurone i
$\mathbb{X}$	le domaine des valeurs d'entrée (x_{ij})
$\mathbb{Y}$	le domaine des valeurs de sortie (y_i)
x	une variable aléatoire
y	une variable aléatoire
$AvgIM_i$	l'information mutuelle moyenne des connexions d'entrée du neurone i
k	le nombre d'intervalles utilisé dans la quantification
β	un facteur d'adaptation
IV_{ij}	j^e intervalle des poids du neurone i
FND	Forme Normale Disjonctive
FNC	Forme Normale Conjonctive
r_i	la règle i
ca_i	le cardinal de l'ensemble d'antécédents de r_i
na_i	le nombre d'antécédents de la règle r_i
nc_i	le nombre de conjonctions de la règle r_i
nac_{ij}	le nombre d'antécédents de la j^e conjonction de la règle r_i
a_{ij}	le j^e antécédent de la règle r_i
I_i	la présentation symbolique du i^e neurone d'entrée
H_i	la présentation symbolique du i^e neurone caché
O_i	la présentation symbolique du i^e neurone de sortie

Liste des tableaux

Table des figures

Chapitre 1

Interaction connexionniste-symbolique : généralités

1.1 Introduction

Les systèmes hybrides essayent de tirer parti des avantages des différents systèmes qui les composent pour élargir leur champ d'application, apporter quelques solutions à des problèmes jusqu'alors insolubles et atteindre de meilleures performances. Dans ce chapitre, nous nous intéressons à l'approche hybride connexionniste-symbolique. Nous commençons par définir ses deux composantes : le Système Symbolique (SS) et le Réseau de Neurones Artificiel (RNA), et nous montrons ce qu'apporte chacun de ces deux systèmes à l'autre. Nous insistons beaucoup plus sur le RNA sur lequel est basé la plus grande partie de notre travail. En suite, nous présentons les différentes tâches qui peuvent être accomplies par un RNA et les raisons qui nous font choisir un RNA plutôt que d'autres paradigmes classiques pour la résolution d'un problème donné.

Avant de réaliser un système hybride, dans notre cas un système connexionniste-symbolique, nous devons :

1. répartir le travail entre les deux composantes du système ;
2. établir des liens entre eux ;
3. définir les coopérations possibles et les mécanismes nécessaires à leur combinaison.

Les objectifs étant :

1. garder les avantages et éliminer les faiblesses de chacune des deux composantes ;
2. minimiser la complexité effective du système hybride par rapport à ses composantes ;
3. maximiser le gain et la portée du système hybride par rapport à ses composantes.

Ce chapitre est organisé de la manière suivante : dans la deuxième section, nous décrivons les divers systèmes symboliques intervenant dans des systèmes hybrides et leur aspect numérique. Nous décrivons aussi les avantages et les faiblesses de ces systèmes symboliques. Dans la troisième section, nous décrivons la structure des RNA, leurs modes d'apprentissage, leurs différentes architectures, leurs domaines d'utilisation, leurs avantages et leurs faiblesses. Dans la quatrième section, nous présentons le modèle connexionniste-symbolique en décrivant la complémentarité existant entre ses deux composantes. Nous décrivons aussi l'utilité de ce modèle qui combine les deux systèmes précédents. Nous terminons cette section par une description des différents types de connaissance pris en compte dans un système connexionniste-symbolique et les raisons du choix d'un RNA dans la construction d'un système hybride numérique-symbolique.

1.2 Systèmes symboliques

1.2.1 Généralités

Les systèmes symboliques sont des systèmes basés sur la manipulation d'entités nommées symboles et qui peuvent représenter une action sous forme d'une structure symbolique [102]. Cette dernière est constituée d'un ensemble de symboles liés par des opérateurs logiques (ET, OU, NON, ...).

L'apprentissage symbolique consiste à appliquer des opérations d'induction logique à un échantillon d'occurrences d'un phénomène (une base d'exemples et de contre exemples) pour induire des relations pertinentes permettant d'expliquer le phénomène. Les exemples utilisés pour cette induction possèdent des ensembles de caractéristiques qui les décrivent et permettent de faire des comparaisons entre les différents exemples. Certaines de ces caractéristiques seront utilisées dans les règles induites [26].

Plusieurs types de SS ont été proposés. Parmi eux nous pouvons citer les systèmes experts à base de règles de production, les réseaux sémantiques et les systèmes à base d'objets [19, 122].

Ne manipulant que des symboles à l'origine, les SS ont présenté l'inconvénient de ne pas être utilisables dans les raisonnements approximatifs qui sont utiles en IA. Certains auteurs ont essayé de pallier cette faiblesse en intégrant l'aspect numérique [60] ou en définissant une logique sémantiquement bien fondée pour le raisonnement approximatif dans différents contextes et pour des problèmes à plusieurs valeurs [115].

Parmi les SS dotés de l'aspect numérique, nous pouvons citer le système expert $MYCIN$ qui utilise des coefficients numériques associés aux prémisses [80], certains systèmes de déduction automatique, comme $SETHEO$ qui utilise des heuristiques numériques dans leurs résolutions [55], d'autres systèmes de représentation de connaissances, comme $SHIRKA$ [110], permettaient dès leur création d'appeler des procédures externes, provenant par exemple d'une bibliothèque d'analyse numérique, la logique floue fait aussi partie des systèmes symboliques et elle utilise des valeurs numériques pour représenter les degrés d'appartenance à des ensembles.

1.2.2 Avantages

Il y a plusieurs raisons pour lesquelles les systèmes symboliques rencontrent un grand succès :

1. la grande capacité d'explication ;
2. la séparation des connaissances et du système qui les exploite. Cette séparation facilite le développement et l'exploitation de ces deux composantes ;
3. le caractère structuré des connaissances qui facilite la représentation ;

4. la nature déclarative facilitant l'expression des connaissances ;
5. la grande capacité de représentation et de traitement des propositions ;
6. la composition des représentations explicitement liées par des règles facilite l'exploitation de ces représentations ;
7. la possibilité d'utiliser différents niveaux d'abstraction selon les besoins ;
8. l'existence de nombreux résultats mathématiques portant sur les systèmes symboliques.

1.2.3 Faiblesses

Dans le cas d'un système symbolique en général et d'un système expert en particulier, nous constatons les faiblesses suivantes :

1. la dégradation brutale des performances : le système ne donne pas de réponses devant des situations non-prédéfinies ou des données incomplètes ou bruitées [57] ;
2. le système ne s'adapte pas au changement de son environnement : une modification dans une partie du domaine étudié demande une modification dans la base de connaissances, sauf si on le couple avec un système d'apprentissage symbolique ;
3. le coût élevé des opérations symboliques et la difficulté de la construction et de la maintenance des applications de grande taille ;
4. l'incapacité d'un SS à réaliser certaines tâches : par exemple la reconnaissance des formes où la faiblesse des SS provient de leur incapacité à prendre en compte certains types de données ;
5. la rigidité : les systèmes symboliques de représentation de connaissances sont limités à des inférences très rigides [19] ;

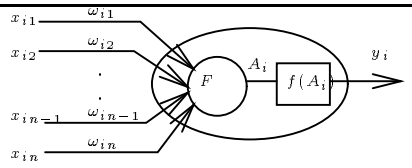
6. la fragilité des systèmes symboliques : elle se traduit par l'incapacité de traiter l'un des aspects suivants [122] :
   - absence de règles applicables [78],
   - interaction entre règles : manque de consistance ou de complétude dans une base de règles partielle,
   - ajout et modification de règles ;
7. la lenteur des méthodes séquentielles utilisées (explosion combinatoire) ;
8. l'ancrage des symboles : les symboles utilisés dans un système symbolique n'ont pas toujours une signification évidente pour l'utilisateur [59] ;
9. la simplification de l'apprentissage se fait au prix de la construction des attributs complexes à partir des attributs initiaux [111]. Cependant, la construction d'un attribut complexe est difficile.

Après cette brève description des SS, nous allons décrire dans la section suivante les réseaux de neurones artificiels.

1.3 Réseaux connexionnistes

1.3.1 Généralités

Une autre approche différente de la précédente basée sur les systèmes à base de règles trouve son origine dans les travaux de McCulloch et Pitts en 1943 [88]. Ils ont proposé un modèle mathématique très simplifié du neurone biologique : le neurone formel. Ils décrivent le comportement de ces neurones en structure de réseau. Chaque neurone est un processeur élémentaire, ses caractéristiques sont décrites par FIG. 1.1.



ω_{ij} : le poids de la connexion entre le neurone j et le neurone i,
x_{ij} : l'entrée j du neurone i,
$A_i = F(\omega_{ij}, x_{ij})$: activation,
f : la fonction de transition,
θ_i : le seuil du neurone i,
$y_i = f(A_i + \theta_i)$, la sortie du neurone i.

FIG. 1.1 – *Neurone formel*

Un réseau connexionniste est un ensemble de neurones interconnectés. Il se caractérise par son architecture et les fonctions de ses éléments [84]. Le premier réseau est le perceptron de Rosenblatt [112]. Ce dernier est une machine adaptative basée sur un classifieur linéaire permettant de calculer la plupart des fonctions logiques. Un rapport de Minsky et Papert [92] révèle les limites de ces systèmes. Ces derniers ne peuvent pas séparer des classes non linéairement séparables. En 1970, l'explosion des modèles basés sur la logique et le codage symbolique en IA conduit à l'abandon relatif des méthodes statistiques connexionnistes.[1]

Ces dernières années, on assiste à un renouveau des méthodes connexionnistes. Des nouveaux modèles ont été mis au point comme le perceptron multicouches [113] qui dépasse les limitations des perceptrons.

1. Bennani, Y. Introduction aux Réseaux Connexionnistes, Tutorial, LIPN, Université Paris 13, 1997.

1.3.2 Structure du réseau

a- Différents types d'unité

Dans un réseau connexionniste, chaque unité effectue un traitement local de l'information. On distingue deux classes principales d'unités suivant le calcul de leur entrée :

- *Les unités produit scalaire :* ce sont des unités basées sur des automates à seuil où l'activité propagée est un produit scalaire. Un exemple de ce type d'unité est décrit par FIG. 1.2.

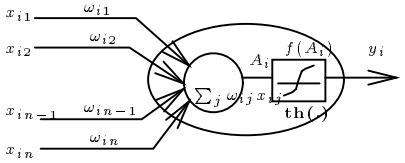


FIG. 1.2 – *Unité produit scalaire*

- *Les unités distances :* ce sont des unités basées sur un calcul de distance ou de similarité. Un exemple de ce type d'unité est décrit par FIG. 1.3.

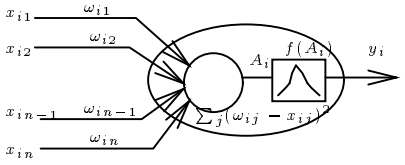


FIG. 1.3 – *Unité distance*

b- Fonctions de transfert

La sortie y_i d'une unité est calculée en fonction des entrées en utilisant une fonction de transfert. On peut donc écrire :

$$y_i = f(A_i).$$

Dans FIG. 1.4, on trouve les fonctions de transfert les plus utilisées.

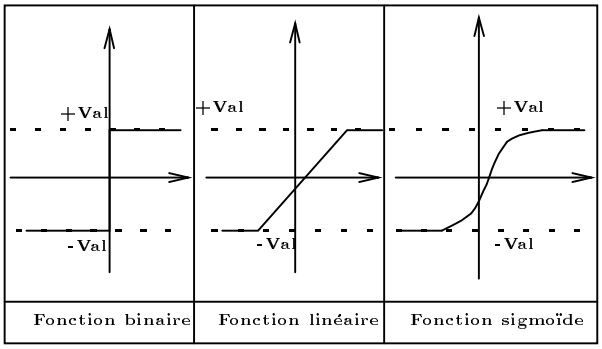


FIG. 1.4 – *Les principales fonctions de transfert*

c- Fonctionnement d'un réseau connexionniste

L'un des principes des modèles connexionnistes est la simplicité des éléments utilisés, comparée à la complexité des opérations réalisées. Le fonctionnement global des réseaux est régi localement par des lois de propagation d'activité. On distingue deux modes de fonctionnement :

- *Mode apprentissage :* le réseau est modifié en fonction d'exemples. On peut modifier les poids des connexions, l'architecture et les fonctions de transition des neurones.

- *Mode reconnaissance :* le réseau est utilisé pour traiter des données. Les poids de l'architecture du réseau sont fixés. Des exemples sont présentés à l'entrée du réseau qui calcule les sorties correspondantes en fonction de l'architecture et des poids appris.

1.3.3 Apprentissage connexionniste

L'apprentissage n'est pas utilisé pour apprendre une représentation exacte des données mais pour extraire une relation liant ces données. Donc, il essaie

d'estimer le modèle statistique qui a généré ces données. Dans les trois paragraphes suivants, nous décrivons les deux modes d'apprentissage, l'évaluation de l'apprentissage et l'algorithme d'apprentissage le plus utilisé.

a- Modes d'apprentissage

L'apprentissage est l'une des caractéristiques les plus importantes des réseaux connexionnistes. Dans ces derniers, on distingue deux types d'apprentissage:

1. *Apprentissage supervisé:* ce mode consiste à présenter aux systèmes des couples entrée-sortie (caractéristiques de la forme et étiquette de la forme) qui sont associés par les réseaux suivants des règles locales. Les mécanismes d'apprentissage des réseaux sont simples, ils consistent en des opérations sur les neurones et des modifications des poids des connexions suivant des règles d'apprentissage [27]. Hebb énonce un principe d'adaptation des neurones à leurs environnement en modifiant l'efficacité des connexions entre les neurones [64].

2. *Apprentissage non supervisé:* ce mode d'apprentissage consiste à ne pas utiliser l'étiquette des formes présentées. Les réseaux, suivant des règles locales d'apprentissage, s'auto-organisent pour représenter l'espace d'entrée. Les données qui se ressemblent vont, à l'issue de l'apprentissage, se trouver réunies dans des groupes qui seront donc des ensembles de formes [76].

b- Évaluation de l'apprentissage

L'évaluation d'un apprentissage porte sur la capacité de généralisation à des exemples non vus durant l'apprentissage. Elle comporte les trois étapes suivantes:

1. *L'évaluation statistique:* pour évaluer la fonction apprise par le réseau, on considère une base d'exemples répartie en trois bases. La première

base sert à l'apprentissage afin d'ajuster les différents paramètres du réseau pour obtenir une bonne approximation de la fonction étudiée. La deuxième base de validation est utilisée pour arrêter l'apprentissage et pour éviter la mémorisation par cœur des exemples, comme montre FIG. 1.5. La troisième base est utilisée après l'apprentissage pour évaluer la qualité des décisions prises par le réseau. Les tailles des différentes bases peuvent varier selon l'application et selon le nombre d'exemples disponibles.

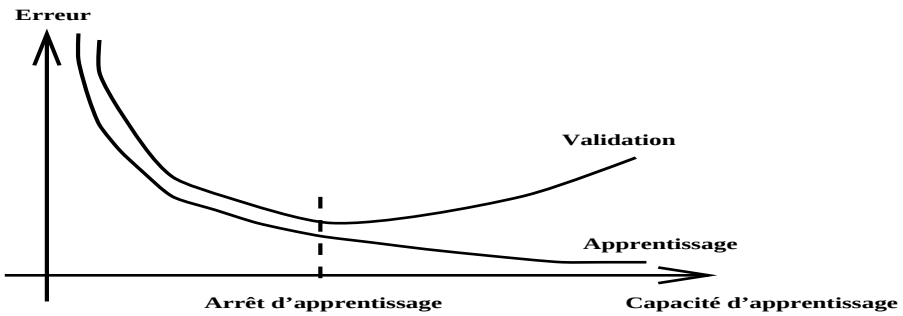


FIG. 1.5 – *L'évolution des erreurs d'apprentissage et de validation en fonction de la capacité d'apprentissage*

2. *La capacité d'un réseau :* la capacité d'un RNA pour l'approximation d'une fonction dépend de ses paramètres et de la base d'exemples. Il y a un compromis à trouver entre le nombre de paramètres et la taille de la base d'exemples pour éviter la *"sur-généralisation"* (la non-capacité d'apprendre) et la *"sur-spécialisation"* (apprentissage par cœur). L'évaluation d'un réseau sur un problème donné peut se faire de façon empirique en minimisant une erreur sur l'ensemble d'apprentissage et de validation.

3. *L'amélioration de la généralisation :* à partir d'une base d'exemples incomplète ou incertaine, un RNA ne peut pas définir la fonction qui couvre la base d'exemples. Il a donc besoin de connaissances supplémentaires qui peuvent être exprimées sous forme d'hypothèses. Comme il existe en

général une infinité de fonctions solutions, ces hypothèses sont utilisées pour réduire l'espace de recherche de la solution. Par exemple et comme le montre FIG. 1.6, on peut éliminer la fonction $f_2(x)$ en supposant que la fonction qu'on cherche est continue et dérivable, autrement dit une fonction qui forme une surface régulière. Ces méthodes, permettant de réduire l'espace de recherche, sont basées sur des techniques de régularisation.

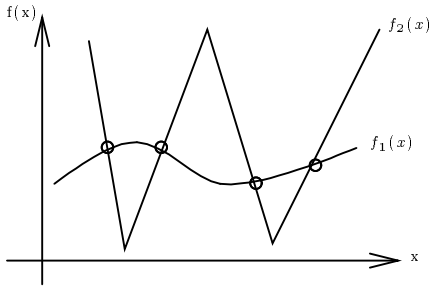


FIG. 1.6 – *Exemple de deux fonctions d'apprentissage*

Comme nous l'avons déjà précisé, la capacité de généralisation d'un RNA dépend du nombre des paramètres de ce réseau et aussi de la quantité et la qualité des informations contenu dans les exemples. Si le nombre de paramètres est très élevé, le RNA risque d'apprendre par cœur. Pour éviter ce problème, une technique dite *OBD (Optimal Brain Damage : décroissance des poids)* [82] a été développée permettant la suppression des connexions inutiles durant l'apprentissage. Pour ce faire, cette technique commence par faire tendre les poids des connexions à supprimer vers zéro.

c- Algorithme de rétro-propagation du gradient

Cet algorithme permettant l'apprentissage d'un réseau avec des unités cachées a été proposé pratiquement en même temps par Rumelhart [113], Parker [105] et Le Cun [81].

C'est un algorithme d'apprentissage supervisé. Le principe consiste à réduire une distance quadratique entre réponses désirées et calculées au moyen d'une descente de gradient dans l'espace des poids. Il repose sur le calcul de dérivées partielles. Cet algorithme utilise deux mécanismes simples : une opération locale sur les neurones du réseau et une modification des paramètres du réseau afin de minimiser la fonction coût. Un tel modèle est donc capable d'apprendre à reconnaître des formes déduites les unes des autres par des transformations simples de couche en couche.

Le but de l'apprentissage est de trouver la meilleure configuration de poids pour séparer les classes d'échantillons d'étiquettes différentes. La méthode d'apprentissage et l'architecture du réseau permettent de déterminer automatiquement une partition de l'espace de représentations, ce qui revient à trouver les poids des connexions qui minimisent une fonction de coût : l'erreur quadratique entre les sorties désirées et les sorties obtenues.

L'algorithme de rétro-propagation comprend quatre étapes :

1. *Propagation de l'activité :* l'activité des cellules de la couche d'entrée est propagée aux cellules des couches suivantes jusqu'aux cellules de sortie.
2. *Calcul de l'erreur par rapport à une fonction coût :* la fonction de coût la plus souvent utilisée est la somme des moindres carrés.
3. *Rétro-propagation de l'erreur :* l'algorithme de rétro-propagation modifie les poids en fonction du gradient de l'erreur.
4. *Modification des poids :* les poids sont modifiés en utilisant le gradient de l'erreur par rapport aux poids avec un coefficient de modification.

1.3.4 Différents modèles

Il existe de nombreux réseaux de neurones ayant des fonctionnements et des architectures différentes [36]. Nous pouvons citer les modèles suivants :

a- Adaline

L'Adaline est un système adaptatif simple qui cherche à trouver des relations entre des entrées et des sorties désirées. Pour trouver une relation parmi une infinité de relations qui passent par les entrées et les sorties désirées, Adaline suppose que cette relation est linéaire. On peut donc définir Adaline comme étant une fonction linéaire qui permet d'attribuer à chaque entrée une réponse désirée, le résultat obtenu étant optimal au sens des moindres carrés [133].

b- Perceptron

Le perceptron est un réseau basé sur des neurones formels (FIG. 1.7), réalisé pour modéliser la perception visuelle [112]. Il est composé de trois modules :

- la rétine R qui reçoit des informations de l'extérieur ;
- les cellules d'associations qui réalisent un pré-traitement des données présentées sur la rétine ;
- la cellule de décision est composée de simples automates à seuil avec des coefficients linéaires déterminés par apprentissage.

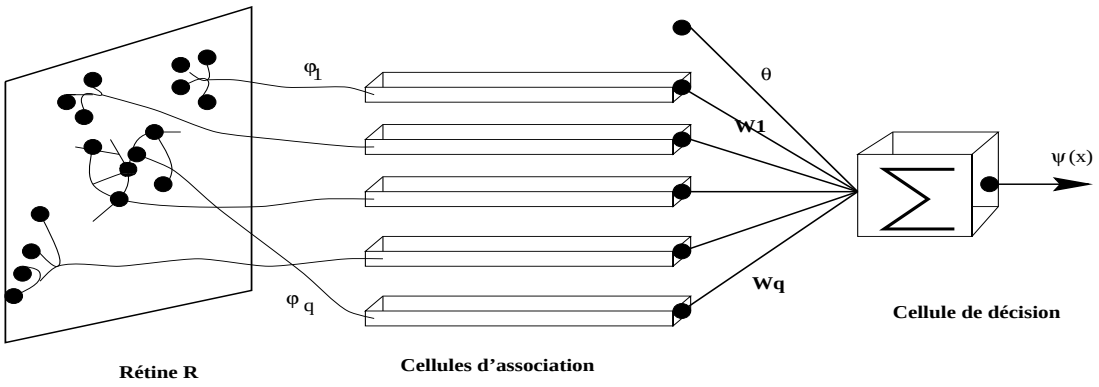


FIG. 1.7 – *Le perceptron de Rosenblatt*

Les fonctions linéairement séparables (par exemple : la fonction logique *et* présentée dans FIG. 1.8) peuvent être calculées par un perceptron.

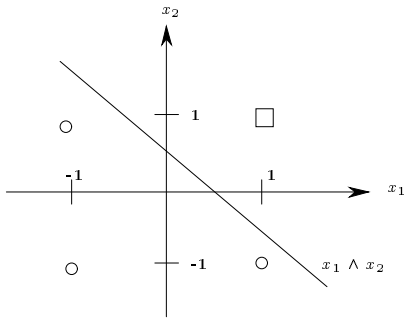


FIG. 1.8 – *Exemple : la fonction logique "et"*

Par contre, les fonctions non linéairement séparables (par exemple : la fonction logique *Xor* présentée par FIG. 1.9) ne peuvent pas être calculées par un perceptron.

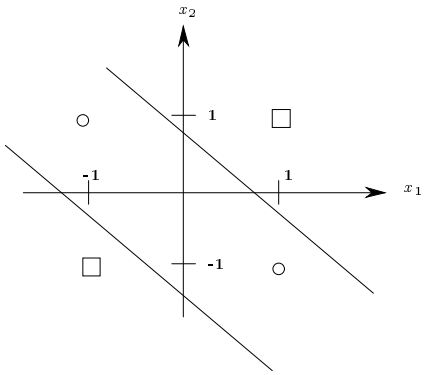


FIG. 1.9 – *Exemple : la fonction logique "Xor"*

c- Perceptron multi-couches : *MLP*

Le *MLP(Multi Layers Perceptron)* repousse les limites du perceptron de Rosenblatt. Il est basé sur le calcul de frontières non linéaires séparant les

classes dans l'espace des formes. La fonction logique "*Xor*" est un problème non linéairement séparable qui peut être résolu par un MLP, comme le montre FIG. 1.10.

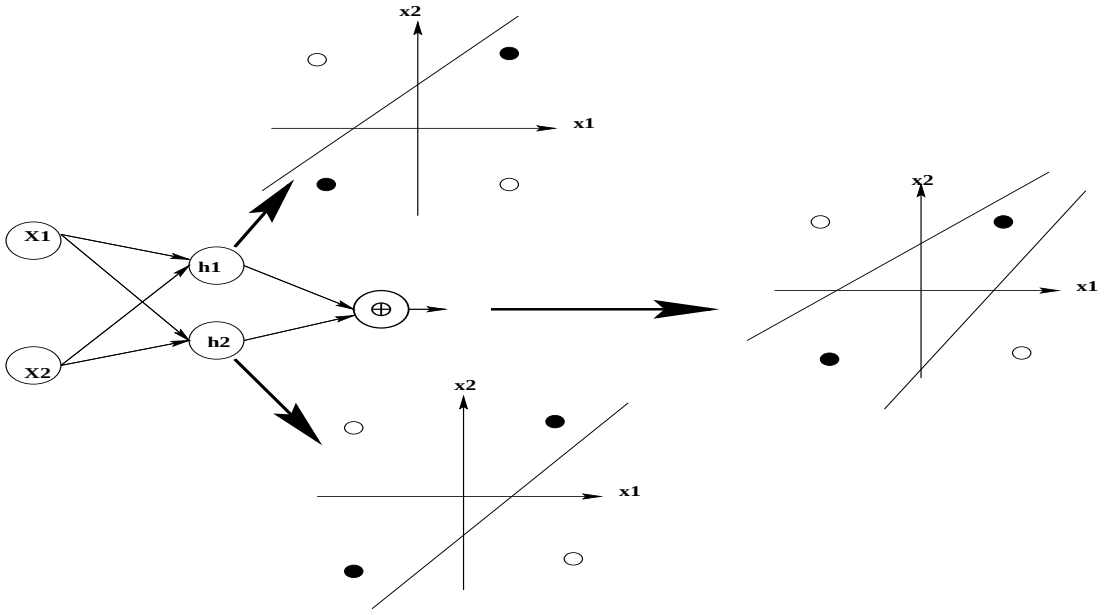


FIG. 1.10 – *Approximation de la fonction "Xor" par un MLP*

La structure du MLP comporte plusieurs couches ce qui permet de séparer des formes non linéairement séparables. Il a été démontré que toute fonction continue de [-1,+1] dans $\mathbb{R}$, pouvait être approchée uniformément par un réseau possédant une couche cachée d'unités sigmoïdes. La structure multi-couches est un filtre de convolution non linéaire qui à partir d'une projection des activités sur plusieurs couches peut séparer des formes non linéairement séparables.

Le perceptron multi-couches, montré dans FIG. 3.3, est un système associatif qui permet d'extraire une relation pertinente entre des entrées et des

sorties présentées sous forme de couples. L'algorithme d'apprentissage supervisé le plus communément utilisé est la rétro-propagation du gradient.

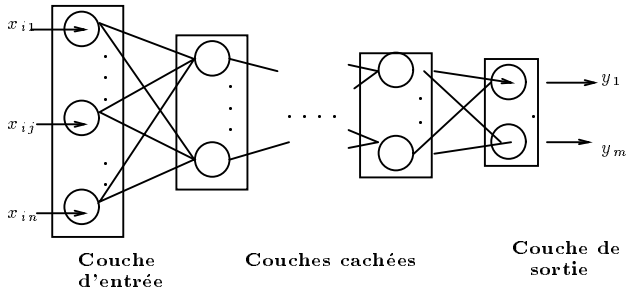


FIG. 1.11 – *Perceptron multi-couches*

Le traitement de l'information réalisé par un perceptron multi-couches repose sur l'interaction d'activités d'unités élémentaires à l'intérieur d'un réseau. Parmi ces unités, on peut distinguer :

- les unités d'entrée, stimulées par des vecteurs d'activités qui correspondent aux formes à classifier. Les vecteurs d'activités peuvent être binaires ou réels, les composantes sont en général normalisées dans l'un des deux intervalles [-1, +1] ou [0, +1] ;

- les unités cachées, inaccessibles à l'utilisateur dont l'activité dépend des relations avec les autres cellules. Ces relations se font par l'intermédiaire des connexions dont les valeurs associées sont établies par apprentissage. Ces unités cachées peuvent être sur une ou plusieurs couches ;

- les unités de sortie, dont chaque configuration d'activité correspond à la reconnaissance d'une forme d'entrée particulière. Les composantes du vecteur de sortie varient dans l'un des deux intervalles [-1, +1] ou [0, +1]. Le vecteur de sortie désiré comparé avec le vecteur de sortie contient un coefficient à 1 pour désigner la classe de la forme, les autres coefficients étant tous à -1 ou à 0.

d- Architecture à poids partagés : $TDNN$

Le *TDNN (Time Delay Neural Network)* est un MLP qui prend en compte la notion du temps [79]. Ce type de réseau est un cas particulier des MLP avec des connexions locales et des contraintes d'égalité entre les poids du réseau. Ce sont des architectures adaptées à la nature dynamique de la parole [12].

e- Carte topologique : TM

La *TM (Topological Map)* est un réseau qui regroupe les données selon un critère de voisinage en formant une topologie dans l'espace d'entrée. Ce dernier se partitionne en utilisant un apprentissage non-supervisé [76].

f- Quantification vectorielle avec apprentissage : LVQ

Le *LVQ (Learning Vector Quantization)* est un réseau qui a été proposé par Kohonen [77]. Il s'agit d'une adaptation de méthodes de quantification vectorielle au problème de la classification. LVQ a été conçu comme un cas particulier et une variation des cartes topologiques [76].

g- Réseau d'unités gaussiennes : RBF

Le *RBF (Radial Basic Function)* est un réseau dont la fonction de transition suit une loi gaussienne [108]. La fonction d'activation est un ellipsoïde dont le centre et l'étendue sont donnés par les deux paramètres : la moyenne et la variance. Les cellules d'un RBF réagissent localement sur l'espace d'entrée.

1.3.5 Utilisation

Il est très important de savoir de quoi les RNA sont capables afin de déterminer avec précision leur rôle dans un système hybride [13]. Les différentes tâches qui peuvent être accomplies par un RNA sont :

1. *La modélisation :* il s'agit de modéliser les comportements dynamiques d'un système donné.

2. *La discrimination :* ou le classement ou encore la classification supervisée, il s'agit de classer des objets en respectant au mieux des classes *a priori*. Ces objets sont décrits par des vecteurs de caractéristiques. Ces derniers représentent les différents attributs qui caractérisent un objet ;

3. *La classification :* ou la classification non supervisée, il s'agit de regrouper des individus dans des classes homogènes en fonction de l'ensemble de variables sélectionnées. Il n'y a donc pas de classement *a priori*.

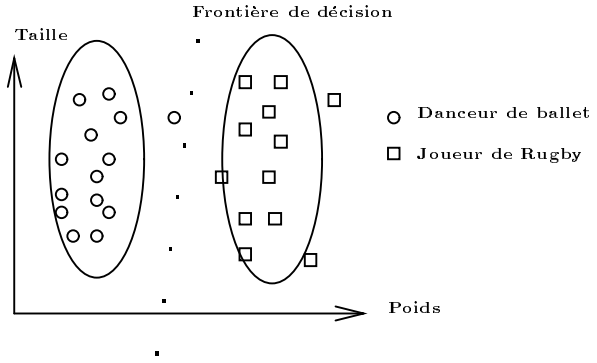


FIG. 1.12 – *Approximation des probabilités a posteriori des classes* $P(C_i|x)$

Ces différentes tâches peuvent aussi être réalisées en utilisant d'autres approches. Parmi ces approches, nous pouvons citer :

1. *L'analyse des données :* les différentes méthodes utilisées dans cette approche peuvent être regroupées en deux grandes classes. La première classe contient des méthodes d'analyse factorielle qui cherchent les directions d'allongement maximal dans un nuage de points. La seconde classe est composée des méthodes de classification automatique dont le principe est de regrouper les données (les individus) en classes homogènes selon un critère donné [47].

2. *Les méthodes de régression :* elles permettent d'extraire de l'information sous forme de règles liant une donnée de sortie nommée *"variable*

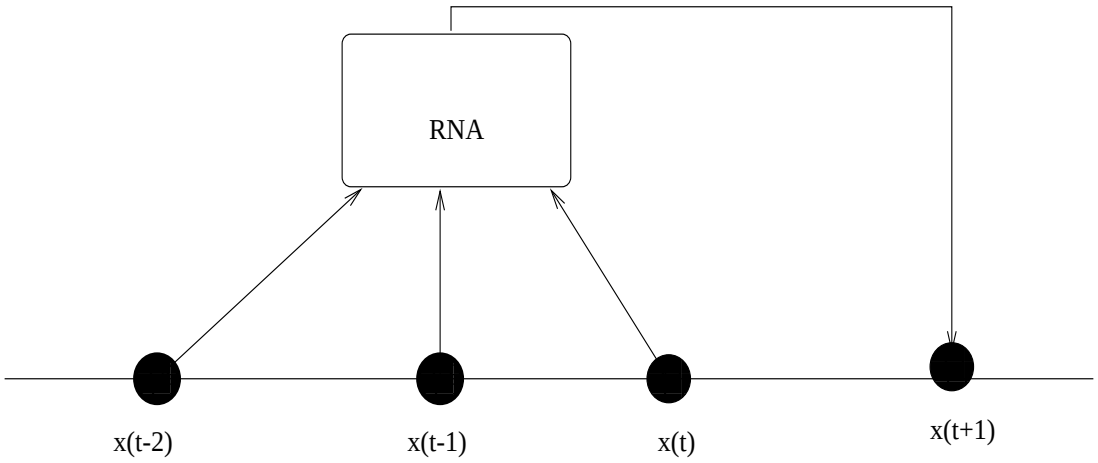


FIG. 1.13 – *Approximation des probabilités a posteriori d'émission* $P(x|C_i)$

réponse" à un ensemble de données d'entrée nommées *"variables de prédiction"* [56]. Le résultat de ces méthodes dépend du choix des variables de prédiction qui sont évaluées par des méthodes d'estimation de paramètres.

3. *Les méthodes de corrélation :* contrairement aux méthodes précédentes, ces méthodes étudient directement les relations entre variables en produisant un coefficient de corrélation. Ce qui permet la mesure de la dépendance linéaire entre variables [107].

4. *Les arbres de décision :* ils sont utilisés pour représenter aussi bien des fonctions booléennes que d'autres fonctions plus générales. Comme nous montre la figure FIG. 1.14, l'arbre de décision est caractérisé par une racine représentant la fonction étudiée, des nœuds intermédiaires permettant de réaliser une classification et des feuilles représentant les valeurs de sortie [20].

Les RNA diffèrent, de ces approches, par leur forme fonctionnelle, les classes de fonctions approximées, les critères optimisés et les algorithmes d'apprentis-

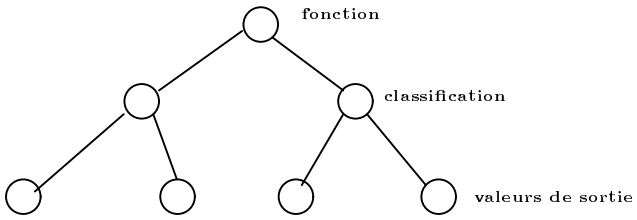


FIG. 1.14 – *Exemple d'un arbre de décision*

sage. L'apprentissage consiste à déterminer à partir d'un ensemble d'exemples une fonctionnelle caractérisant le modèle [69].

1.3.6 Avantages

Les RNA ont plusieurs avantages dont les principaux sont :

1. la résistance au bruit (plusieurs RNA donnent de bons résultats avec des données d'entrée bruitées) ;

2. la prise en compte des dépendances non-linéaires entre les données et aussi de la notion du temps (dans le cas des réseaux récurrents qui se distinguent par le fait de donner des réponses en ne se basant pas uniquement sur les entrées mais aussi sur les réponses précédentes [53]) ;

3. la facilité d'implantation sur une machine parallèle grâce au mode de fonctionnement massivement parallèle des RNA (une telle implantation permet une exécution très rapide, ce qui est intéressant en particulier dans les applications en temps réel) ;

4. la résistance aux pannes (la redondance d'information dans un réseau distribué lui permet de fonctionner même en cas de panne de quelques neurones) ;

5. la représentation distribuée des connaissances ;

6. la génération des surfaces de séparation entre des classes arbitrairement complexes ;

7. la capacité de l'architecture à prendre en compte des connaissances *a priori* sur le problème ;

8. des bons résultats pour les traitements de bas niveau.

1.3.7 Faiblesses

Malgré tous leurs avantages, les RNA présentent quelques points faibles qui sont les suivants :

1. l'opacité des connaissances générées sous forme de poids synaptiques (un RNA est considéré comme une boîte noire où les poids n'ont pas une signification explicite pour l'utilisateur) ;

2. la détermination d'une manière empirique des nombres de neurones et de couches cachées. Néanmoins il y a des travaux comme OBD [82] et HVS [134] qui apportent quelques solutions ;

3. l'influence de l'état initial d'un réseau et l'ordre de présentation des exemples sur le comportement du réseau ;

4. la construction des RNA pour la manipulation du numérique les rend moins adaptés à la manipulation des symboles (surtout pour des applications de haut niveau comme les démonstrations mathématiques).

Après une description séparée des deux approches symbolique et connexionniste, nous allons décrire l'approche connexionniste-symbolique combinant les concepts et techniques des deux approches.

1.4 Approche connexionniste-symbolique

1.4.1 Généralités

Nous commençons cette section par une citation de Kodratoff qui a prévu l'intérêt d'un système connexionniste-symbolique [75] :

"les méthodes numériques ont fait leur preuves depuis longtemps, les méthodes symboliques sont en cours de test dans la communauté scientifique. Il est clair qu'une avance technologique importante attend ceux qui sauront combiner les deux approches afin de corriger leurs points faibles respectifs".

L'approche connexionniste-symbolique regroupe des méthodes issues des approches RNA et IA. La première, proche du numérique, peut avoir des vertus explicatives en mettant en oeuvre des mécanismes d'extraction de règles. La seconde, symbolique à l'origine, quantifie des fonctions de classement ayant de bonnes performances en utilisant des critères numériques. Plus généralement, la proximité des problèmes rend naturelle la coopération entre les approches numériques en particulier neuronales, et les approches issues de l'IA. L'approche connexionniste-symbolique constitue donc un domaine de recherche qui combine différents outils des deux approches (connexionniste et symbolique) dont les objectifs principaux sont : les performances (temps de calcul, taux d'erreur de classement, ...), le caractère explicatif de la fonction apprise et la capacité de traiter des données complexes [49].

Avant de décrire cette approche[2], nous présentons dans le schéma du FIG. 1.15 les différentes interactions existant entre les systèmes symboliques et les systèmes connexionnistes.

2. Nedjari, T. Base de Connaissances et Réseau de Neurones Artificiel, *Poster présenté dans les journées du LIPN pour le CNRS, Université Paris 13, 1995.*

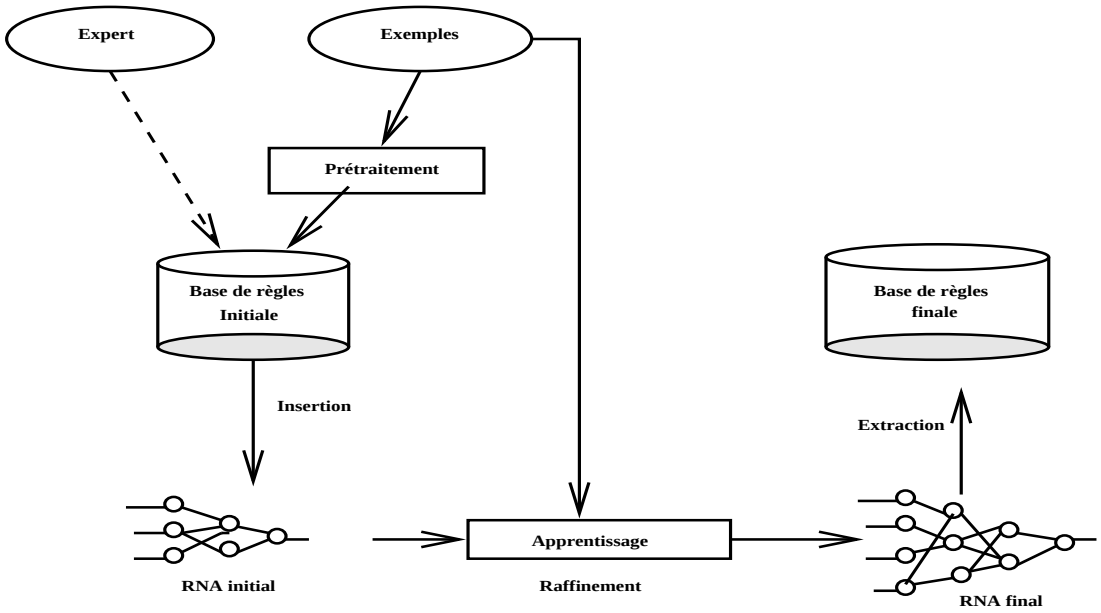


FIG. 1.15 – *Schéma général d'interaction connexionniste-symbolique*

1.4.2 Complémentarité symbolique-connexionniste

La conception d'un système connexionniste-symbolique est justifiée par la complémentarité entre les deux approches. Cette complémentarité concerne aussi bien l'aspect cognitif que l'aspect psychométrique. Le tableau TAB. 1.1 contient une comparaison globale des deux systèmes.[3]

Les références [57, 66] présentent un tableau (TAB. 1.2) comparatif des deux approches connexionniste et symbolique pour montrer le peu de chance pour que l'une des deux options *"tout-symbolique"* ou *"tout-numérique"* aboutisse à la résolution des problèmes posés [51].

3. Bien que les termes utilisés dans ce tableau ainsi que dans ceux qui suivent ne possèdent pas de définition formelle, nous avons cependant reproduit intégralement ces tableaux, en considérant que ces termes possédaient par eux-mêmes une signification intuitive suffisante pour éclairer le lecteur.

TAB. 1.1 – *Comparaison des deux systèmes symbolique et connexionniste*

Modèle Symbolique	Modèle Connexionniste
haut niveau	bas niveau
logique	analogique
séquentiel	parallèle
discret	continu
programmation	adaptation
connaissance experte	connaissance fortuite
apprentissage symbolique	apprentissage numérique
explicite	implicite
modulaire	distribué

Le tableau TAB. 1.3 représente la complémentarité des deux systèmes symbolique et connexionniste du point de vue psychométrique [3].

Étant donné la difficulté de proposer un nouveau paradigme qui tire parti des avantages et évite les faiblesses de chacun de ces deux systèmes, la solution est de proposer un système hybride [65, 51]. Notons pour finir ce paragraphe que la conception de systèmes connexionniste-symbolique n'est pas la seule façon d'envisager la relation entre les systèmes connexionnistes et les systèmes symboliques. Une autre méthode est celle dite *"élimination connexionniste"* où il faut éliminer tout système symbolique non nécessaire à la description d'une cognition. Une autre encore est *"implantation connexionniste"*, qui inversement à la première voit que les systèmes connexionnistes ne sont rien d'autres qu'une présentation différente des systèmes symboliques [38].

1.4.3 Utilisation du système connexionniste-symbolique

Dans plusieurs domaines, il est indispensable de fournir une explication à

TAB. 1.2 – *Complémentarité des systèmes symbolique et connexionniste du point de vue cognitif (- : point faible et + : point fort)*

Critères cognitifs	Symbolique	Connexionniste
Modélisation de *"Savoir-faire"*	-	+
Modélisation de *"Savoir-que"*	+	-
Traitement des données continues	-	+
Traitement des structures symboliques	+	-
Prise en compte d'informations statistiques	-	+
Structuration des connaissances	+	-
Capacité d'apprentissage par exemple	-	+
Capacité de généralisation	-	+
Capacité d'explication	+	-
Temps de développement	-	+
Modularité	+	-

l'utilisateur pour lui montrer comment le système a abouti à un résultat donné. Cette solution donnée sous forme d'information explicite peut permettre à l'utilisateur de comprendre les mécanismes du fonctionnement d'un tel système.

L'incapacité d'un système symbolique ou connexionniste à résoudre seul tous les types de problèmes (par exemple, la reconnaissance de la parole est un problème difficile pour un système symbolique et la planification est un problème difficile pour les RNA), a conduit à combiner ces deux approches.

1.4.4 Différents types de connaissance

Les différentes connaissances rencontrées dans un système connexionniste-

TAB. 1.3 – *Complémentarité des systèmes symbolique et connexionniste du point de vue psychométrique (- : point faible et + : point fort)*

Critères psychométriques	Symbolique	Connexionniste
"Savoir-que"	+	-
"Savoir-faire"	-	+
Composante inductive de la pensée	-	+
Composante déductive de la pensée	+	-
Composante intuitive de la pensée	-	+
Composante rationnelle de la pensée	+	-
Composante associative de la pensée	-	+
Composante logique de la pensée	+	-
Composante holistique de la pensée	-	+
Composante analytique de la pensée	+	-
Composante analogique de la pensée	-	+
fonctions d'identification d'un concept	-	+
Mode d'apprentissage explicite ou sélectif	+	-
Modèle d'apprentissage implicite ou non-sélectif	-	+

symbolique peuvent être regroupées en trois classes [73] :

1. *Les connaissances logiques :* ce sont des connaissances représentées par la logique propositionnelle. Les différents symboles représentant ces connaissances sont liés par des opérateurs logiques : ET, OU et NON. Ce type de connaissance peut être exprimé sous la forme :

$$\text{"}Si\ x_1\ ET\ NON(x_2)\ alors\ y\text{"}.$$

2. *Les connaissances numériques :* les différents symboles représentant les connaissances sont pondérés par des valeurs numériques et liés aux autres

caractéristiques par des opérateurs arithmétiques : *l'addition* et *la soustraction.* Ce type de connaissances peut avoir une représentation finale sous la forme :

$$\text{“}y = a\,x_1 + b\,x_2 + c\text{”}.$$

3. *Les connaissances logico-numériques :* il s'agit de combiner les deux types de connaissances précédents afin d'obtenir des expressions de la forme suivante :

$$\text{“}Si\ (a\,x_1 + b > c)\ ET\ (d\,x_2 + e < 0)\ alors\ y\text{”}.$$

1.4.5 Choix d'un RNA dans un système numérique-symbolique

Contrairement aux méthodes statistiques, les RNA possèdent une puissance et une simplicité d'expression, ils apportent généralement de bonnes performances dans des applications complexes. Les RNA offrent aussi la possibilité de concevoir des systèmes hybrides, d'y incorporer aisément de la connaissance *a priori* sur le problème.[4]

Les RNA ont plusieurs caractéristiques dont les principales sont :

1. *L'apprentissage :* la première caractéristique des RNA par rapport aux autres méthodes numériques est qu'ils sont basés sur l'apprentissage à partir d'exemples.

2. *Le parallélisme :* les traitements sont parallèles au niveau des neurones, alors que les autres méthodes mathématiques ou statistiques utilisent une démarche séquentielle.

4. Bennani, Y. Introduction aux Réseaux Connexionnistes, Tutorial, LIPN, Université Paris 13, 1997.

3. *Le calcul local :* l'ajustement des poids se fait au niveau du neurone en utilisant les connexions et les activations de ce neurone.

4. *La non-linéarité :* la capacité d'un RNA à résoudre des problèmes non-linéairement séparables [43]. Cette non-linéarité est difficile à surmonter par d'autres méthodes mathématiques.

5. *Le stockage indirect des données :* contrairement aux autres méthodes, un RNA n'a pas besoin de stocker toutes les données d'une façon brute mais les codifie en gardant leur trace sous forme de poids associés aux différentes connexions.

6. *La transformation :* un RNA ne se limite pas à la classification mais il va jusqu'à la transformation des données brutes sous une forme adéquate qui facilite leur classification. Dans les autres méthodes mathématiques, cette transformation est faite durant la phase de pré-traitement .

Les réseaux multi-couches entraînés par l'algorithme de rétro-propagation du gradient sont les modèles connexionnistes les plus étudiés et les plus utilisés à ce jour. Cependant, une utilisation efficace demande une analyse préalable du problème, car il ne faut pas oublier l'arsenal puissant des méthodes statistiques.

1.5 Conclusion

Dans ce chapitre nous avons présenté les systèmes symboliques, les systèmes connexionnistes et leur coopération dans un modèle connexionniste-symbolique. De plus, nous avons détaillé les avantages et les faiblesses de chaque système et le besoin de construire un modèle qui combine les deux systèmes en profitant de leurs points forts et en évitant leurs faiblesses. Nous avons terminé ce chapitre par une description des raisons du choix d'un RNA dans la construction d'un système hybride numérique-symbolique.

Chapitre 2

Étude des techniques d'extraction de règles à partir d'un RNA

2.1 Introduction

La prolifération et le grand succès des applications incorporant la technologie des RNA dans différents domaines, tels que le commerce, l'industrie et la médecine, montrent l'utilité du paradigme RNA. Les principales caractéristiques d'un RNA à l'origine de ce succès sont :

- La simplicité de l'acquisition de la solution : la méthode d'acquisition est plus simple que dans d'autres systèmes (exemple : système symbolique).
- La forme compacte sous laquelle l'information est stockée : un ensemble d'exemples est stocké dans le réseau connexionniste en modifiant les poids des connexions. La représentation de chaque exemple est répartie entre plusieurs connexions où chacune est impliquée dans le stockage d'autres exemples.
- La résistance au bruit : des données d'entrée bruitées n'empêchent pas le réseau de trouver une solution acceptable.

Néanmoins, le succès des RNA a un prix : l'incapacité inhérente à expliquer sous une forme compréhensible le processus par lequel une décision donnée ou une sortie a été atteinte. C'est cette faiblesse qui limite le champ d'utilisation des RNA en les empêchant d'atteindre des domaines où l'explication est primordiale (par exemple, la médecine et la sécurité). Dans ce cas, le système utilisé doit fournir un certain degré de transparence des solutions. Pour ce faire, il doit être capable de valider la sortie du RNA sous toutes les conditions d'entrée possibles et de déterminer l'ensemble des conditions sous lesquelles un neurone de sortie d'un RNA est activé [4, 17].

En plus de la contribution directe à l'amélioration de l'utilité globale d'un RNA, l'ajout de la capacité *"d'explication"* contribue à la compréhension de *"comment"* les approches symboliques et connexionnistes peuvent être intégrées avec profit dans l'intelligence artificielle. Cet ajout fournit aussi un moyen

pour établir des liens entre les deux approches connexionniste et symbolique. Outre la capacité à expliquer, les systèmes symboliques disposent d'une notion très contraignante de cohérence : vérifier que quelles que soient les entrées, les règles activées ne pourront conclure p et $\neg p$. La cohérence d'un système connexionniste est beaucoup moins forte [7].

Pour pallier cette faiblesse, plusieurs travaux ont été consacrés à la recherche de techniques pouvant doter les RNA de la capacité d'explication, c.-à-d. l'extraction des règles à partir des RNA [4]. Dans ce chapitre, nous donnons un aperçu de ces différents travaux. Nous nous limitons à l'extraction de règles à partir d'un RNA multi-couches direct en utilisant l'apprentissage supervisé.

Ce chapitre est organisé comme suit : la section suivante définit les deux types d'apprentissage utilisés dans un RNA pour l'extraction de règles. Dans la troisième section, nous détaillons les raisons qui font l'intérêt de l'extraction de règles à partir d'un RNA. La quatrième donne un aperçu général des problèmes d'extraction et de raffinement de règles. La cinquième propose des critères de classification des différentes techniques d'extraction. Les sixième et septième sections décrivent les algorithmes les plus importants pour l'extraction de règles logiques ou floues à partir d'un RNA. Dans la huitième section, nous donnons une critique du premier algorithme développé par Shavlik et Towell pour l'extraction de règles à partir d'un RNA [120]. En suite, nous proposons une heuristique pour améliorer la performance de cet algorithme. Dans la neuvième section, nous proposons une adaptation de l'algorithme *M-of-N* [121] avec l'extraction de règles de type *M-parmi-N* et *non(M)-parmi-N*. Dans la dernière section, nous proposons une méthode d'utilisation de l'information mutuelle pour l'extraction de règles floues à partir d'un RNA.

2.2 Différents types d'apprentissage pour l'extraction de règles

On dispose d'un ensemble d'exemples appelé ensemble d'apprentissage. Cet ensemble ne constitue généralement qu'une partie de l'ensemble d'exemples décrivant le phénomène à étudier. Dans le cas de l'apprentissage supervisé, chaque exemple est un couple (description, classe) où la description, généralement du type attribut-valeur, est associée à une classe donnée. Dans le cas particulier où il n'y a que deux classes, on parle d'apprentissage de concept à partir des exemples (exemples positifs) et des contre-exemples (exemples négatifs). L'objectif de l'apprentissage consiste à construire à partir des exemples une fonction de classement permettant d'une part de bien reclasser les exemples d'apprentissage et d'autre part d'avoir de bonnes performances sur de nouveaux exemples. Pour estimer celles-ci, un ensemble test, indépendant de l'ensemble d'apprentissage, est employé. Nous distinguons deux types d'apprentissage, au niveau des RNA, pour l'extraction de règles :

2.2.1 Apprentissage par contraintes

Dans ce type d'apprentissage, le principe est de contraindre les poids à prendre l'une des trois valeurs -1, 0 ou $+1$ afin de maintenir l'interprétation symbolique du réseau. Il s'agit de minimiser une fonction objectif composée de deux fonctions : la fonction de l'erreur quadratique et la fonction de contraintes. La fonction objectif est de la forme :

$$O = E + \lambda \sum_{j=1}^{n} g(\omega_{ij}),$$

où λ est un paramètre expérimental compris entre 0 et 1.

2.2.2 Apprentissage libre

Dans ce deuxième type d'apprentissage, les poids du réseau sont initialisés

de façon aléatoire. En minimisant une fonction d'énergie, les poids changent de valeur jusqu'à une stabilisation du réseau. Ces valeurs ne sont pas directement interprétables et elles nécessitent le développement de méthodes spécialisées pour leur interprétation. L'algorithme de rétro-propagation du gradient est le plus utilisé dans ce type d'apprentissage [21, 83, 113].

2.3 Pourquoi extraire des règles à partir d'un RNA ?

Doter les RNA de la capacité d'explication, c.-à-d. rendre leur fonctionnement compréhensible par l'utilisateur permet d'étendre leur utilisation à des applications où ils étaient jugés inadaptés, d'en induire des théories scientifiques, d'acquérir à partir d'eux des connaissances pour les systèmes d'IA symbolique, d'optimiser l'architecture du réseau et d'y améliorer la généralisation [4, 99]. Dans la suite de ce paragraphe, nous détaillons ces différents points.

2.3.1 Aptitude à expliquer

L'explication a été marquée par deux directions de recherche : la première s'intéresse à la représentation des connaissances à expliquer [32] ; la seconde étudie la modélisation du raisonnement explicatif [93]. Néanmoins, ces deux directions adoptent le point de vue : le rôle du système est de générer des explications sur la solution et de les transmettre à l'utilisateur [8, 37].

Dans le domaine de l'intelligence artificielle, le terme *"explication"* est lié à une structure explicite qui peut être utilisée intérieurement pour le raisonnement et l'apprentissage, et extérieurement pour l'explication des résultats à l'utilisateur. Les utilisateurs des systèmes d'intelligence artificielle symbolique bénéficient d'une représentation déclarative explicite des connaissances. L'aptitude d'explication en intelligence artificielle symbolique inclut les étapes in-

termédiaires du processus de traitement, comme la trace d'exécution de règles, qui peuvent être utilisées pour répondre à la question *"Comment"*. Gallant observe que l'aptitude d'explication permet une vérification de la logique interne du système, donnant ainsi à un utilisateur novice la possibilité d'avoir un aperçu sur le problème étudié [46].

L'expérience a montré que l'aptitude d'explication, et en particulier, la capacité à générer une explication régulière (significative et cohérente), est l'une des fonctions les plus importantes fournies par les systèmes d'intelligence artificielle symbolique. Nous pouvons citer deux groupes travaillant sur les Systèmes à Base de Connaissances (SBC) et qui montrent le rôle de l'explication dans ces systèmes. Le premier groupe est EVA (Évaluation, Validation et Acquisition), il a comme objectif l'aide à la construction d'un SBC en fournissant une explication. Cela consiste à présenter à un interlocuteur les connaissances dont ce système dispose. Cette présentation est faite par différentes méthodes : présenter la fonction d'un concept, donner un exemple, faire une analogie entre un mécanisme et un autre [2]. Le deuxième groupe est $GENE$ (Génération d'Explication NEgociée), il a comme objectif de définir une fonction d'un SBC permettant à l'usager d'affiner sa requête. Cette fonction a nécessité la spécification d'un système d'aide à la formulation du problème de l'usager d'un SBC en langue naturelle et intégrant la génération d'explication [8].

Pour certaines catégories d'utilisateurs, l'aptitude d'explication est indispensable et un système, même ayant de très bonnes performances par ailleurs, qui n'en est pas doté est donc insuffisant. À l'inverse des systèmes symboliques d'intelligence artificielle, les RNA n'ont pas de représentation explicite de la connaissance et par conséquent sont incapables de fournir une explication de la connaissance acquise. Il devient de plus en plus apparent que l'absence de la capacité *"d'explication"* dans les RNA limite leur utilité. C'est cette insuffisance que les systèmes d'extraction de règles cherchent à pallier.

Ainsi, les développements futurs dans les RNA doivent intégrer l'aptitude d'explication. Cette tâche d'extraction de règles à partir d'un réseau de neurones peut être réalisée en s'inspirant de l'intelligence artificielle symbolique, où les praticiens ont fait des expériences avec des formes variées d'explication. Avant d'aboutir à l'explication, le système passe par plusieurs états intermédiaires. La description de cette évolution forme ce qu'on appelle les *"règles de traçage"*. L'utilisation de ces *"règles de traçage"* a un avantage considérable : rendre l'explication compréhensible dans le cas général ; toutefois elle a ses limites : d'abord, dans le cas d'une base de règles mal organisée et contenant une centaine de prémisses par règle, cette transparence ne peut être maintenue ; ensuite, l'explication basée sur les *"règles de traçage"* est rigide [52, 93] : en effet, l'une des critiques majeures qui leur est faite est qu'elles reflètent toujours la structure courante de la base de connaissances ; ce qui rend difficile leur adaptation aux modifications de cette dernière. En plus, les *"règles de traçage"* peuvent faire référence à des procédures internes, comme le calcul, de même qu'elles peuvent introduire des répétitions, comme dans le cas d'une inférence créée plus d'une fois ; une dernière critique de l'utilisation des *"règles de traçage"* est que la granularité de l'explication est souvent inappropriée [52].

Une autre limite de l'aptitude d'explication dans les systèmes symboliques de l'IA, pouvant être évitée dans l'extraction de règles à partir d'un RNA, est fournie par Moore et Swartout [93]. Cette limite est que, en réponse a certaines situations, un SS peut se contenter d'une simple réponse *"pas de réponse"*. Cette limite provient de la rigidité des SS : là où d'autres systèmes effectuent des rapprochements entre l'exemple présenté et les cas prévus, ces SS exigent pour traiter cet exemple qu'il entre dans le cadre précis d'un de ces cas. Pour dépasser cette limite, certains travaux se sont inspirés des techniques du langage naturel et d'artifices comme les initiatives mixtes, le modèle utilisateur et les stratégies d'explication explicite.

L'intégration de l'aptitude d'explication à travers l'extraction de règles dans

les RNA doit donc, tout en s'inspirant des méthodes utilisées dans les systèmes symboliques, éviter leurs faiblesses dans la mesure du possible.

2.3.2 Extension des RNA à d'autres domaines

Dans les domaines où la sécurité est primordiale, l'utilisateur attend du système qu'il utilise non seulement la réponse à la question *"quelle décision prendre ?"*, mais aussi et surtout une justification de cette réponse, ou encore une description de la manière dont il a abouti à cette décision. Il en résulte que seuls les systèmes capables de fournir une explication sont utilisables dans ces domaines. Doter les RNA de la capacité à extraire des règles élargit donc leur domaine d'utilisation.

2.3.3 Étude des données et induction des théories scientifiques

Par l'étude des données, nous entendons la recherche des relations existantes entre elles et non connues à l'avance. Les RNA ont déjà prouvé leur efficacité pour ce genre de tâches, puisqu'ils permettent de l'accomplir même dans le cas des problèmes non linéaires. Grâce à l'extraction de règles, l'utilité de ces relations ne se limite plus à la résolution d'un problème donné mais elle recouvre la possibilité d'engendrer des théories scientifiques.

2.3.4 Acquisition des connaissances pour des systèmes d'IA

L'une des raisons principales de l'introduction des algorithmes d'apprentissage symbolique est la résolution du problème appelé *"Acquisition des connaissances"* [114]. Les connaissances ainsi acquises le sont sous forme explicite, mais sont obtenues difficilement et à un coût élevé [118]. Dans les RNA, les connaissances sont exprimées sous forme implicite et leur acquisition est plus facile et moins coûteuse. Les RNA ont été utilisés pour assister les systèmes

symboliques dans la tâche d'acquisition de connaissances. Pour ce faire, il a fallu avoir recours à l'extraction de règles. Deux travaux récents ont été consacrés à cet aspect de l'utilisation des RNA. Dans le premier [127], Thrun et al. présentent des résultats qui suggèrent que les règles extraites à partir d'un RNA sont plus performantes que les méthodes d'apprentissage symbolique. Le deuxième travail que nous devons à [29], contient un développement des techniques d'extraction de règles symboliques à partir d'un RNA qui peuvent ajouter des règles à la base de connaissances.

2.3.5 Optimisation de l'architecture du réseau

Un autre avantage de l'extraction de règles à partir des RNA est d'aider l'utilisateur à décider si le réseau a une structure et une taille optimales. En effet, dans un RNA doté de l'explication, non seulement les solutions fournies sont transparentes mais en plus elles donnent la possibilité d'accéder, pour les interpréter sans ambiguïté, aux états internes du système. Ce faisant, elles permettent de supprimer les neurones inutiles ou sources d'erreur de manière à optimiser le réseau.

2.3.6 Amélioration de la généralisation dans un RNA

Lorsqu'un ensemble de données limité ou non-représentatif du problème traité est utilisé pour l'apprentissage dans un RNA, il est difficile de déterminer avec les méthodes classiques d'évaluation, telle que la *"validation-croisée"*, à partir de quel point le réseau se trompe en généralisation. Avec la capacité d'exprimer la connaissance existante dans un RNA par un ensemble de règles symboliques, le processus d'extraction de règles fournit à l'utilisateur un système capable d'anticiper ou de prédire l'ensemble des circonstances dans lesquelles un échec de généralisation peut se produire. En plus, ce système utilise les règles extraites pour identifier les régions de l'espace de données non suffisamment représentées dans l'espace d'entrée, permettant ainsi à l'utilisateur de compléter ce dernier.

Dans cette section, nous avons souligné l'importance de l'extraction des règles. Avant de passer aux algorithmes utilisés pour cette tâche, nous allons présenter un processus étroitement lié à l'extraction de règles : il s'agit du raffinement d'une base de règles.

2.4 Extraction de règles et raffinement

La connaissance acquise durant la phase d'apprentissage est codée par :

a- l'architecture du réseau (exemple : le nombre de neurones cachés) ;

b- la fonction d'activation associée à chaque neurone caché ou de sortie ;

c- l'ensemble des paramètres numériques (valeurs réelles des poids).

L'objectif de l'extraction de règles à partir d'un RNA est de donner une interprétation compréhensible des effets de (a), (b) et (c).

Un autre processus a le même objectif : c'est le raffinement des règles d'une base de connaissances symbolique. Tandis que le processus d'extraction de règles commence par une base de règles vide, le point de départ du processus de raffinement de règles est un ensemble initial non-vide de connaissances exprimées sous forme de règles symboliques. Cette base de règles initiale peut être incomplète ou incertaine [53]. Le but du raffinement des règles est alors de combiner l'apprentissage dans les RNA et les techniques d'extraction de règles pour produire un meilleur ensemble de règles symboliques. Cet ensemble peut être appliqué de nouveau au problème initial. Dans le processus de raffinement de règles, la base de règles initiale (qui peut être nommée *"connaissances antérieures"*) est insérée dans le RNA par l'initialisation d'un ensemble de poids. Dans ce contexte, il s'agit des connaissances antérieures servant à démarrer la phase d'apprentissage dans les RNA.

Différentes techniques peuvent être utilisées pour résoudre les problèmes d'extraction et de raffinement de règles à partir d'un RNA. Dans la section suivante, nous décrivons les critères de classification de ces algorithmes d'extraction et de raffinement de règles.

2.5 Critères de classification des algorithmes d'extraction et de raffinement

Un premier schéma de classification des algorithmes d'extraction et de raffinement de règles a été présenté par Craven et Shavlik [29]. Ce schéma est basé essentiellement sur la transparence et la fidélité des règles par rapport au réseau duquel elles sont extraites. Avec le développement de nouvelles techniques, ce schéma s'est avéré insuffisant. C'est alors qu'un nouveau schéma a été présenté par Andrews et al. [4]. Ce schéma est basé sur les critères suivants :

1. la puissance expressive des règles extraites ;
2. la *"transparence"* d'une vue prise dans une technique d'extraction de règles à partir des unités sous-jacentes d'un RNA ;
3. la complexité algorithmique de la technique d'extraction de règles ;
4. la qualité des règles extraites ;
5. la possibilité d'élargir le champ d'application d'une technique d'extraction de règles.

La puissance expressive des règles concerne la sortie présentée à l'utilisateur. Certaines techniques présentent les sorties comme un ensemble de règles exprimées sous forme de symboles logiques conventionnels (c.-à-d. de la logique propositionnelle). Ces règles sont de la forme *"Si ... Alors ... Sinon ..."*. D'autres travaux expriment les connaissances existantes dans un RNA en utilisant des concepts probabilistes ou de la logique floue. Cela permet aux règles

d'utiliser le concept de fonction d'appartenance pour exprimer ce qui est *"partiellement"* vrai. Par exemple, "si x est *bas* et y est *haut* alors z est *moyen*", où *bas*, *haut* et *moyen* sont des ensembles flous qui correspondent à des fonctions d'appartenance.

Le deuxième critère reprend en le généralisant le schéma de classification utilisé par Craven et Shavlik [29]. Ce critère porte sur la relation entre les règles extraites et l'architecture interne d'un RNA [4]. Il permet de regrouper les techniques d'extraction/raffinement en deux catégories : *"décompositionnelle"* et *"pédagogique"*. Une troisième catégorie, appelée *"éclectique"*, combine les éléments des deux catégories précédentes. Dans certains travaux les deux premières approches sont appelées respectivement *"liaison"* et *"boîte-noire"* [123].

Dans l'approche *"décompositionnelle"*, l'extraction de règles se fait au niveau de chaque unité individuelle (cachée ou de sortie) du RNA. Les sorties calculées à partir de chaque unité cachée ou de sortie prennent la forme d'un résultat binaire (oui/non) qui correspond à la forme des règles résultantes. Donc chaque unité cachée ou de sortie peut être interprétée par une règle logique. Cela réduit le problème d'extraction de règles à un problème de détermination des situations dans lesquelles la règle est *"vraie"*, c.-à-d. l'ensemble des liens d'entrée dont la combinaison des poids dépasse le seuil du neurone. Les règles extraites au niveau de chaque unité individuelle sont rassemblées pour former une base de règles pour tout le réseau.

Dans l'approche *"pédagogique"*, le RNA est traité comme une boîte noire. Donc, extraire des règles d'une manière pédagogique, c'est chercher la fonction calculée par le RNA, sous forme d'un concept, sans regarder sa structure interne. Dans les règles extraites, les prémisses sont les entrées du réseau, les conclusions ses sorties [29]. Cette approche est utilisée en conjonction avec un algorithme d'apprentissage symbolique qui se sert du RNA pour la génération d'exemples.

L'approche *"éclectique"* incorpore des éléments des deux approches précédentes. Une technique de cette approche utilise des connaissances sur l'architecture interne et les entrées/sorties du réseau [128].

Le troisième critère porte sur la complexité des algorithmes utilisés en temps (nombre d'opérations) et en espace (taille de la base de règles). Le paramètre principal intervenant dans cette complexité est la taille de l'espace de résolution, c.-à-d., les différentes combinaisons possibles des poids au niveau de chaque neurone pour son activation.

Dans le quatrième critère [129], il s'agit d'évaluer la qualité des règles extraites. Cette évaluation se fait en utilisant les critères suivants :

1. L'exactitude : l'ensemble des règles est dit exact s'il peut classer correctement des exemples non vus précédemment.
2. La fidélité : l'ensemble de règles présente un haut niveau de fidélité s'il peut imiter le comportement du RNA dont il est extrait.
3. La consistance : si les différents apprentissages aboutissent au même ensemble de règles, ce dernier est dit consistant.
4. La compréhensibilité : l'ensemble de règles est d'autant plus compréhensible que le nombre de règles et celui de leurs prémisses sont faibles. Weiss et al. [132] signalent que le résultat d'application de ce critère doit être pondéré par des paramètres concernant la taille du réseau.
5. L'amélioration : ce critère concerne exclusivement les techniques de raffinement. Il s'agit de la capacité de ces techniques à corriger et à compléter la base de règles initiale [53].
6. La complexité du réseau : pour un problème donné, la complexité effective de l'architecture du réseau (nombres de paramètres) ne doit pas influencer la base de règles extraite [132].

Une évaluation de la qualité des règles produites par une technique d'extraction ou de raffinement de règles est très significative pour un utilisateur. Malheureusement, il s'est avéré qu'une évaluation rigoureuse de la qualité des règles produites par une technique d'extraction/raffinement de règles est un phénomène relativement récent. Cela limite la possibilité de comparer ces différentes techniques en utilisant la qualité des règles.

Le dernier critère de classification porte sur la capacité des techniques d'extraction/raffinement à réaliser des tâches autre que l'extraction de règles, par exemple participer à l'apprentissage (apprentissage à contraintes), modifier la topologie du réseau ou être utiliser dans le contrôle.

Parmi les six critères cités, seuls les deux premiers sont unanimement utilisés car ils concernent directement la performance des techniques. Dans la suite, tout en insistant sur ces deux critères, nous évoquons, selon le contexte, l'un ou l'autre des quatre derniers.

2.6 Algorithmes d'extraction de règles

Dans cette section, nous présentons les algorithmes d'extraction de règles les plus importants. Ces algorithmes sont regroupés selon l'approche qu'ils utilisent (décompositionnelle, pédagogique ou éclectique) [4, 99].

2.6.1 Approche décompositionnelle

L'approche décompositionnelle regroupe les techniques qui extraient des règles logiques conventionnelles au niveau de chaque unité cachée ou de sortie du RNA. C'est cette approche qui a été utilisée par les premiers travaux sur l'extraction de règles.

Historiquement, le premier travail utilisant cette approche est celui proposé par Fu [42]. Dans ce travail, l'auteur présente l'algorithme *KT (Know-*

ledge Translator). Le principe de *KT* a été repris et approfondi par Towell et Shavlik qui ont proposé l'algorithme *SubSet* [120].

Le principe des deux algorithmes présentés dans [42, 120] est de trouver, pour chaque unité, un ensemble de sous-ensembles de poids positifs dont la somme des poids de chaque sous-ensemble dépasse le seuil de cette unité. Puis pour chaque sous-ensemble, trouver l'ensemble des sous-ensembles de poids négatifs où la somme des poids positifs des sous-ensembles et des poids négatifs de chaque sous-ensemble ne dépasse pas le seuil de l'unité.

La particularité de *Subset* est d'utiliser un RNA où la valeur calculée par la fonction d'activation pour chaque unité (cachée ou de sortie) est proche de *un* (neurone actif) ou proche de *zéro* (neurone inactif). *SubSet* a été appliqué au domaine biologique où il a démontré ses très bonnes performances et sa capacité à produire un ensemble de règles plus petit que d'autres bases de règles fournies par des systèmes experts

Cependant, ces deux algorithmes ont une complexité exponentielle. Cette complexité élevée est due au temps nécessaire pour trouver tous les sous-ensembles de poids négatifs ou positifs [121]. Pour remédier à ce problème, Fu propose de limiter la taille de l'espace de résolution en fixant un seuil sur le nombre d'antécédents de chaque règle extraite [44]. Malheureusement, l'utilisation de cette heuristique a un inconvénient : le risque de perdre des règles pertinentes. La simplicité des deux algorithmes *KT* et *SubSet* fait d'eux un moyen très utile pour l'explication des mécanismes de l'extraction de règles.

RuleNet de MacMillan et al. [89] est une autre technique de cette classe d'algorithmes. Dans cette technique, le RNA incorpore l'approche décompositionnelle dans l'apprentissage pour l'extraction de règles logiques. L'idée de base de *RuleNet* est de répéter les trois étapes suivantes :

1. lancer l'apprentissage dans le RNA ;

2. extraire des règles symboliques (en utilisant les connexions fortes du réseau) ;

3. injecter ces relations dans le RNA.

Le processus se termine quand la base de règles résultante caractérise de manière adéquate le problème étudié. Le RNA utilisé est composé de trois couches, le rôle de la couche cachée étant de coder les *"unités conditionnelles"*. Chacune de ces unités correspond à une règle. L'inconvénient majeur de cette technique est son manque de généralisation dû à la spécificité de l'architecture du réseau et à la dépendance entre l'adaptation des poids et le problème étudié [118, 129].

Une autre technique est celle proposé par Towell et Shavlik dans [120], il s'agit de l'algorithme *M-of-N*. Cet algorithme est une composante du système hybride *KBANN (Knowledge Based Artificial Neural Networks)* capable d'initialiser, raffiner et extraire des règles. Une règle *M-of-N* est de la forme :

"Si *M parmi les N antécédents sont vrais* alors *conclusion*".

D'après Towell et Shavlik, l'un des atouts principaux de l'approche *M-of-N* est l'affinité naturelle entre les règles de cette forme et le seuil inductif des RNA.

Les auteurs donnent une comparaison détaillée de leur nouvelle méthode avec l'algorithme *Subset* et les techniques d'induction symbolique. Il ressort de cette comparaison que l'algorithme *M-of-N* représente un progrès significatif par rapport aux autres méthodes. En effet, il améliore la capacité des règles extraites à généraliser sur les exemples non vus durant l'apprentissage et la qualité de ces règles. Une autre caractéristique intéressante de cet algorithme est que les règles qu'il produit ont parfois des performances supérieures au RNA dont il les extrait. Cette dernière caractéristique peut être attribuée à la réduction des ajustements sur les exemples d'apprentissage.

L'apport de l'algorithme *M-of-N* à *KBANN*, qui est un système hybride d'interaction connexionniste-symbolique, peut être résumé par les points suivants [29] :

1. un ensemble de règles pour initialiser le RNA [120] ;
2. un régime spécial d'adaptation des poids du RNA par groupement ;
3. l'approximation des unités cachées par des unités à seuil binaire (cela est fait en donnant au paramètre a de la fonction d'activation $\frac{1}{1+e^{-ax}}$ une valeur plus grande que 0,5) ;
4. l'utilisation des termes intermédiaires pour présenter les unités cachées sous forme de règles. Cela implique que l'approche n'est pas suffisamment capable de donner une description exacte du réseau dont les règles seront extraites.

Étant essentiellement un système de raffinement de règles, *M-of-N* présente une autre caractéristique : la signification des unités cachées du RNA générées par le processus d'initialisation, ne risque pas d'être perdue. Cette caractéristique est intéressante car en son absence la compréhension des règles extraites peut être dégradée.

La méthode *M-of-N* a été testée avec succès dans plusieurs domaines et particulièrement sur deux problèmes de biologie moléculaire [120]. En effet, il ressort de la comparaison entreprise par Towell et Shavlik, que les performances de cette approche sont meilleures que celles des autres techniques de l'apprentissage symbolique.

Une autre technique décompositionnelle pour le raffinement des règles est celle présentée par Tresp et al. [131]. Le principe de cette méthode est de coder la base de règles initiale sous forme de fonctions gaussiennes à plusieurs variables. Les auteurs proposent quatre stratégies différentes pour préserver la connaissance initiale et raffiner la base de règles durant l'apprentissage, ainsi

que deux méthodes pour améliorer la compréhension des règles. Cette méthode a été appliqué à la prédiction des prix des logements dans la banlieue de Boston (USA) en fonction de trente entrées pertinentes (par exemple : le nombre de chambres...). La caractéristique la plus importante de cette technique est l'interprétation probabiliste de l'architecture du RNA qui permet aux fonctions gaussiennes d'agir comme des classificateurs et de se manifester dans les règles extraites. Ces règles sont de la forme suivante :

"Si *le nombre de chambres est approximativement 5,4*
Alors *le prix de la maison est approximativement 14000$*".

Une autre technique de l'approche décompositionnelle a été proposée par Denoeux dans [33]. Cette méthode nommée "la rétro-propagation contraignant les poids" a la particularité de contraindre les poids à s'approcher de -1, 0 ou $+1$ pendant l'apprentissage. Pour ce faire, elle ajoute un terme complémentaire à la fonction d'erreur classique. Le point fort de cette méthode est l'interprétation symbolique directe des poids après la stabilisation du réseau où le $+1$ est traduit par une confirmation, le -1 par une négation et le 0 est négligé. En se basant sur ce travail, Kane et al. ont proposé une autre technique décompositionnelle qui utilise le même principe (contraindre les poids d'être dans l'ensemble $\{-1, 0, +1\}$). La particularité de cette méthode est de contraindre les cellules cachées d'être des cellules numériques et les cellules de sortie d'être des cellules logiques. Les règles extraites sont de la forme logico-numérique. Les auteurs ont testé leur méthode avec succès sur un problème de biologie moléculaire [74].

Dans [124, 125], Taha et Ghosh proposent deux techniques décompositionnelles : *Partial-RE* et *Full-RE*. La première s'inspire des techniques KT [42] et $SubSet$ [129] et diffère de ces deux dernières par l'association d'un coefficient à chaque règle extraite et la réécriture des règles en cas d'une binarisation. Cette dernière est utilisée dans le cas où les entrées du réseau ne sont pas binaires et qui transforme en 1 les entrées supérieures à la moyenne des entrées et en 0 les autres. La deuxième technique est une généralisation de la première aux règles

logico-numériques. *Full-RE* utilise l'algorithme $Chi2$ pour la discrétisation des entrées et elle passe par des règles intermédiaires avant d'aboutir aux règles finales. Ces deux techniques ont été testées avec succès sur le problème Iris [1] et elles ont donné des bons résultats sur un autre problème réel : le contrôle d'un réservoir d'eau.

Anouar et al. [6] ont aussi proposé une méthode d'extraction de règles basée sur une recherche arborescente. Cette méthode utilise une énumération de tous les cas possibles au niveau de chaque neurone, ce qui la rend très coûteuse en temps et en espace même pour des architectures peu complexes. Les auteurs ont testé leur méthode avec succès sur un problème de classification en biologie.

Dans [5], Andrews et Geva présentent la technique $RULEX$ dont l'objectif est d'exploiter le comportement d'un perceptron multi-couches et la manière dont il a été construit. Contrairement à la plupart des autres techniques, $RULEX$ ne s'applique pas seulement aux données discrètes mais aussi aux données continues ou mixtes. Cette méthode pose des contraintes sur l'erreur de rétro-propagation du gradient. Les unités cachées sont des unités sigmoïdes et sont utilisées pour le partage des données d'entrée en un ensemble de régions disjointes. Chaque région est donc représentée par une unité cachée composée d'un ensemble d'arêtes. Chaque arête correspond à une dimension de l'entrée. La sortie d'une unité cachée est la somme des seuils des activations de ses arêtes. Une sortie est significative si la valeur présentée à l'entrée appartient au rang actif de l'arête. Pour classifier un vecteur par les unités cachées, chaque composante de ce vecteur doit appartenir au rang actif de l'arête correspondante. Cela donne lieu à des règles propositionnelles de la forme suivante :

"Si *(Arête-1 est active) ET ... ET (Arête-N est active)*

1. Iris est un problème de classification qui décrit trois espèces d'Iris : *Setosa*, *Versicolor* et *Virginia*. La base Iris contient 150 exemples. Chaque espèce est décrit par 50 exemples. La base des exemples est partagé en deux bases : une base d'apprentissage de 89 exemples et une base de test de 61 exemples. Chaque exemple est caractérisé par quatre attributs : longueur-sépale, largeur-sépale, longueur-pépale et largeur-sépale [41, 95].

alors *le motif appartient à la classe i".*

Dans le cadre de cette thèse, nous avons proposé un autre développement appartenant à l'approche décompositionnelle [96]. Il s'agit de la technique *MITER (Mutual Information and Template for Extracting Rules)*[2] dont la caractéristique est d'utiliser le critère de l'information mutuelle pour évaluer l'information contenue dans chaque connexion d'entrée par rapport à la sortie du même neurone. Cette évaluation est utilisée pour définir un calibre de poids, puis extrait une règle symbolique de la forme *M-parmi-N* à partir de ce dernier, et cela pour chaque neurone caché ou de sortie.

Les deux techniques *RULEX* et *MITER* présentent l'avantage d'éliminer les antécédents redondants/contradictoires et les règles redondantes. Contrairement aux autres méthodes décompositionnelles, comme *KT* et *Subset*, qui emploient des variantes des techniques de *"recherche et test"*, *RULEX* et *MITER* améliorent la performance de l'extraction de règles par l'interprétation directe du vecteur des poids en règles. Par conséquent, *RULEX* et *MITER* évitent les problèmes de calcul rencontrés par les autres techniques décompositionnelles. Elles n'ont pas recours aux heuristiques pour contrôler la recherche dans l'espace des solutions.

2.6.2 Approche pédagogique

L'un des premiers travaux utilisant l'approche pédagogique pour l'extraction de règles est celui de Saito et Nakano [114]. Dans leur méthode, le RNA est traité comme une boîte noire. Des règles décrivant un diagnostic médical sont extraites à partir des changements au niveau des unités d'entrée et de sortie. Pour éviter les combinaisons non significatives des entrées (c.-à-d. les symptômes médicaux) et pour éliminer les informations redondantes (c.-à-d. la répétition des symptômes), les auteurs ont été obligés de contraindre la taille de l'espace de résolution. Malgré l'utilisation de cette heuristique, le nombre

2. Cette technique est présentée en détail dans le chapitre 3.

de règles extraites, même sur un problème simple, reste très important. C'est l'une des faiblesses majeures de cette technique.

La technique *VI-Analysis* développée par Thrun [127], associe à chaque unité d'entrée un intervalle de validité, où chaque valeur d'activation appartient à cet intervalle, et utilise une procédure de génération et de test pour extraire des règles symboliques à partir d'un RNA. Ce dernier n'est pas construit spécialement pour faciliter l'extraction de règles. Les étapes principales de *VI-Analysis* sont les suivantes :

1. associer des intervalles arbitraires aux unités du RNA qui vont intervenir dans la résolution du problème. Ces intervalles constituent des contraintes sur les valeurs d'entrée et sur les activations des sorties ;
2. raffiner les intervalles par la détection et l'exclusion de certaines valeurs d'activation. Ces valeurs exclues sont probablement incompatibles avec les poids et les seuils du réseau ;
3. tester la consistance des intervalles de l'étape (2) : si le résultat du test est négatif retourner à l'étape (2). Un intervalle est inconsistant s'il n'y a pas d'activation qui peut satisfaire les contraintes imposées par les intervalles de validité.

Cette méthode fournit un outil générique pour la vérification de la cohérence des règles à l'intérieur du RNA. Elle a été appliquée avec succès à plusieurs problèmes (Xor et Monk[3] où les données sont discrètes et le bras d'un robot cinématique où les données sont continues). Bien que le champ d'application de *VI-Analysis* ne semble pas se limiter à une classe de problèmes dans un domaine particulier, Thrun signale que sa méthode n'arrive pas à générer un ensemble complet de règles dans des domaines relativement complexes (par

3. Le problème Monk est défini dans la section expérimentation (3.4.4) du chapitre 3.

exemple, *NETtalk* : un réseau de neurones qui apprend à "prononcer" l'Anglais) [127].

RULENEG est une autre technique pédagogique développée par Pop et al. [109, 63]. Dans cette technique, les règles extraites sont exprimées sous forme d'une disjonction de conjonctions. Une description informelle de l'algorithme *RULENEG* est la suivante :

1. initialiser la base de règles à l'ensemble vide ;
2. pour chaque exemple s de l'ensemble d'exemples d'apprentissage :
   - trouver la classe C de s en utilisant le RNA,
   - si s n'est pas classé par les règles existantes :
     * initialiser la classe C par une nouvelle règle r,
     * pour chaque entrée i du réseau :
       faire une copie s' de s,
       entrer une négation sur la i^e entrée de s', pour lui choisir une valeur dans r,
       trouver la classe C' de s' en utilisant le RNA,
       si $C \neq C'$ alors ajouter la i^e entrée et sa vraie valeur à r,
     * ajouter r à la base de règles.

RULENEG extrait une règle conjonctive à partir de chaque exemple. Il en résulte un avantage : les règles produites traduisent tous les exemples appris et un inconvénient : le nombre de règles est très élevé. Un prototype initial de *RULENEG* a donné de bons résultats sur deux problèmes, une formule logique et le problème des trois moines (Monk).

Dans [114], Sestito et Dillon présentent une autre technique : il s'agit de la méthode *BRAINNE* qui est considérée comme appartenant à l'approche pédagogique car son principe est de mesurer la proximité entre les entrées et les

sorties du réseau. Ces mesures sont utilisées pour la génération de l'ensemble des règles. Cette méthode suppose un régime d'apprentissage spécial basé sur :

1. un réseau avec m entrées et n sorties ;
2. transformation du réseau en un autre réseau avec $(m+n)$ entrées et n sorties ;
3. lancement de l'apprentissage au niveau de ce nouveau réseau ;
4. amélioration de la comparaison entre les poids de chacune des m unités initiales d'entrée et l'ensemble des unités cachées d'une part, et entre les poids liant chacune des n entrées additionnelles et les unités cachées correspondantes d'autre part ;
5. si la différence entre deux valeurs est grande aller à 3.

L'innovation majeure de la technique $BRAINNE$ est sa capacité à discrétiser les données continues à l'entrée sans passer par une phase de description. $BRAINNE$ segmente automatiquement les données continues en données discrètes et extrait directement les règles correspondantes de la forme

"Si ... alors ... sinon ...".

Les auteurs appliquent leur technique avec succès au problème de l'interprétation des données sous-marines sonores.

BIO-RE est une autre technique pédagogique proposée par Taha et Ghosh [123, 125]. Cette technique permet d'extraire des règles binaires à partir de tout RNA ayant des entrées binaires. Dans le cas où les entrées du réseau ne sont pas binaires, une procédure de binarisation est utilisée qui transforme en 1 les entrées supérieures à la moyenne des entrées et en 0 les autres. Les règles extraites sont de la forme *"si ... alors ..."* où les prémisses sont liées par les deux opérateurs logiques (ET et NON). Dans le cas d'un RNA à entrées non-binaires, les règles sont réécrites en utilisant les valeurs d'entrée initiales, c.-à-d. avant binarisation. *BIO-RE* a été testé avec succès sur deux problèmes réels : Iris et

le contrôle d'un réservoir d'eau.

Dans le cadre de cette thèse, nous avons proposé un autre développement appartenant à l'approche pédagogique [97]. Il s'agit de la technique *EMIRE* (*Examples and Mutual Information for Rules Extraction*) [4] dont la caractéristique est de chercher pour chaque sortie du réseau, le sous-ensemble de neurones d'entrée les plus pertinents. Pour ce faire, elle utilise l'information mutuelle pour mesurer les dépendances entre les sorties et les entrées du réseau. Ces dépendances traduisent l'information apportée par chaque neurone d'entrée à une sortie du réseau. Après la sélection des neurones d'entrée les plus pertinents, une règle est extraite à partir de chaque exemple présenté à l'entrée du réseau. Ensuite, la base de règles est optimisée en utilisant l'une des techniques classiques.

Dans [29], Craven et Shavlik présentent un système hybride d'interaction symbolique-connexionniste qui à la particularité de pouvoir intégrer aussi bien l'approche décompositionnelle que l'approche pédagogique. En effet, dans ce système la procédure d'extraction de règles utilise, selon la nature du problème, soit une méthode décompositionnelle (par exemple, l'algorithme KT [44]), soit une méthode pédagogique (par exemple, l'algorithme *VI-Analysis* [127]). Une fois les règles extraites, chacune d'elles est testée pour voir si elle décrit la sortie du réseau correspondante. Ces deux opérations (extraction de règles et test) sont répétées tant que l'un des critères d'arrêt suivants n'est pas satisfait :

1. les règles extraites reflètent le comportement du réseau. Autrement dit, on aboutit aux mêmes réponses en utilisant le réseau ou les règles extraites ;
2. le nombre d'itérations a atteint un seuil fixé à l'avance ;
3. les dernières itérations n'ont pas produit de nouvelles règles (critère de patience).

Les auteurs montrent que les performances du système, c.-à-d. la qualité des règles extraites dépendent de la complexité de la technique utilisée pour l'extraction de règles. Ainsi, les résultats obtenus avec une technique aussi simple que *KT*

4. Cette technique est présentée en détail dans le troisième chapitre.

sont moins bons que ceux obtenus avec *VI-Analysis*, qui est une méthode bien plus complexe.

2.6.3 Approche éclectique

La première technique d'extraction de règles utilisant l'approche éclectique est la méthode *DEDEC (Decision Detection by Rule Extraction from Neural Networks)* développée par Tickle et al. [128]. La particularité de *DEDEC* est d'extraire des règles symboliques à partir d'un ensemble de cas individuels. Dans cette technique, l'extraction de règles est traitée comme un processus lié à l'identification. Il s'agit d'identifier l'ensemble minimal d'informations requises pour distinguer un objet particulier des autres objets. Les cas individuels à partir desquels les règles sont extraites sont créés en présentant les entrées (c.-à-d. les attributs) au RNA et en observant les sorties résultantes. Le choix des entrées est fait de manière à explorer l'espace de résolution d'une façon optimale. *DEDEC* examine les cas et les classes par ordre d'importance. Cela est fait par l'utilisation des vecteurs des poids pour ordonner les unités d'entrée (c.-à-d. les attributs du problème étudié) selon leurs contributions aux sorties du RNA. Les règles sont extraites à partir de ces cas considérés comme étant les unités d'entrée les plus pertinentes du RNA. C'est le classement des unités d'entrée (c.-à-d. les antécédents d'une règle), en se basant sur l'analyse des vecteurs des poids du RNA, qui fait de *DEDEC* une technique éclectique. Cette technique emploie aussi une heuristique pour arrêter le traitement une fois que l'ensemble des règles est devenu stable (le critère de patience). Le choix de l'heuristique dépend de la signification relative des unités d'entrée sélectionnées. Un premier prototype de *DEDEC* a été testé avec succès sur des problèmes réels (Monk et la prédiction des niveaux de production de lait d'un troupeau de vaches).

Une deuxième méthode, nommée *IIIE*, a été proposée par Bennani et Bossaert [11]. Pour extraire des règles, cette technique commence par calculer l'influence de chaque neurone d'entrée sur tous les neurones de sortie. Ce calcul se déroule en deux étapes : d'abord l'influence des neurones d'entrée sur la couche cachée, en suite ceux de cette dernière sur les neurones de sortie. Les neurones d'entrée avec une faible influence n'interviennent pas dans les règles extraites. Pour chaque exemple présenté en entrée, une règle est extraite à la sortie. La base de règles finale interpré-

tant le comportement du RNA est fournie à l'utilisateur après optimisation, c.-à-d. après l'élimination de toutes les règles redondantes ou contradictoires.

Toutes les techniques que nous avons décrites jusqu'à présent produisent des règles logiques conventionnelles. Dans la section suivante, nous présentons une autre catégorie de méthodes, il s'agit des systèmes *NeuroFlou* dont la particularité est d'extraire des règles floues.

2.7 Extraction de règles floues

Comme les techniques discutées précédemment (les systèmes logiques conventionnels), les systèmes *NeuroFlou* incluent trois phases. La première est la phase d'initialisation : les règles floues sont insérées dans le RNA et les fonctions d'appartenance sont générées. La deuxième phase celle de l'apprentissage : les fonctions d'appartenance sont accordées aux exemples de la base d'apprentissage. Durant la troisième phase, le système analyse les fonctions d'appartenance puis extrait des règles floues à partir de ces dernières.

L'un des premiers travaux appartenant à cette catégorie est celui de Masuoka et al. [87]. Ces derniers utilisent l'approche décompositionnelle pour raffiner un ensemble de règles floues données par un expert. Dans leur technique la première phase consiste à associer à chaque règle une unité d'entrée, une ou deux unités cachées et une unité de sortie. Cette dernière est utilisée pour représenter la fonction d'appartenance de chaque antécédent de règle (c.-à-d. les variables d'entrée). Les opérations floues sur les variables d'entrée (par exemple : ET, OU, etc.) sont représentées par une deuxième phase distincte appelée la phase *règle-réseau*. Les fonctions d'appartenance qui constituent les règles résultantes sont représentées dans une troisième phase appelée la phase de sortie. Le motif de la phase d'entrée est utilisé dans cette phase de sortie. Pour obtenir un ensemble compact de règles à la sortie, cette technique est dotée d'une opération d'élagage durant la phase *règle-réseau*. Il s'agit d'éliminer toutes les connexions qui ne dépassent pas une valeur seuil.

Dans [14], Berenji démontre l'utilité d'un RNA spécialisé pour raffiner une base

de connaissances approximativement correcte. Cette base est constituée de règles floues données par un contrôleur. Le problème traité dans ce travail est l'équilibrage de *"charette-perche"*. La caractéristique de cette technique est la connaissance préalable des règles décrivant les opérations faites par le contrôleur. Le RNA est utilisé pour modifier les fonctions d'appartenance des règles de pré-condition et de conclusion.

Horikawa et al. [71] ont développé trois types de RNA qui peuvent identifier d'une façon automatique les règles floues. Les auteurs accordent les fonctions d'appartenance correspondantes aux règles par la modification des poids en utilisant l'algorithme de rétro-propagation du gradient. Dans cette technique, la base de règles initiale est donnée par un expert. Elle peut être aussi créée par des sélections itératives à travers les combinaisons possibles des variables d'entrée et quelques fonctions d'appartenance.

Un autre modèle de RNA flou est FuNe-I développé par Halgamuge et al. [58]. Il utilise un processus basé sur l'identification des règles pertinentes. Pour chaque sortie, les règles ont une forme disjonctive ou conjonctive. Les auteurs appliquent avec succès leur technique sur plusieurs problèmes (Iris, classification des images, reconnaissance des images sous-marines et reconnaissance des chiffres manuscrits).

Les deux techniques $FNES$ (Fuzzy Neural Expert System) de Hayashi [61] et *Fuzzy-MLP* de Mitra [91] ont pour objectif de fournir à l'utilisateur une explication ou une justification de *comment* le RNA est arrivé à une conclusion donnée. Dans les deux techniques, les antécédents d'une règle sont déterminés par l'analyse et l'ordonnancement des vecteurs de poids du RNA. Ceci permet de déterminer l'influence relative de ces vecteurs sur une sortie donnée (classe). $FNES$ suppose la participation d'un expert durant la phase d'entrée pour assurer la convergence des données d'entrée dans le format demandé. Dans la technique *Fuzzy-MLP*, ce traitement se fait d'une façon automatique. Les deux modèles ont été appliqués dans un domaine médical de diagnostic. *Fuzzy-MLP* a donné un ensemble de résultats meilleurs (en terme d'exactitude) que ceux de FNS. Cette amélioration est due à l'architecture complexe du RNA utilisée dans *Fuzzy-MLP* (trois couches cachées

contre une seule couche cachée dans $FNES$). L'architecture du réseau de $FNES$ utilise des connexions directes entre la couche d'entrée et la couche de sortie.

Okada et al. [103] insèrent des éléments d'initialisation de connaissances, de raffinement de règles (c.-à-d. l'accord des fonctions d'appartenance) et d'extraction de règles dans un système d'inférence floue utilisant un RNA à sept couches. Dans cette technique, deux couches du modèle sont utilisées pour fournir une représentation des fonctions d'appartenance pour les variables d'entrée (représentées par une couche d'entrée différente) et une autre couche est utilisée pour présenter les fonctions d'appartenance pour les règles résultantes. D'autres couches sont aussi utilisées pour construire des antécédents des règles et les règles conséquentes. Les auteurs apportent une amélioration significative à l'exactitude de la prédiction du modèle par rapport aux RNA à trois couches. Ces améliorations ont été validées sur des applications dans le domaine de la finance.

2.8 Critique de la méthode *"SubSet"*

Dans la section précédente, nous avons présenté une étude détaillée des principales techniques d'extraction de règles. Cette étude contenait une critique sommaire de certaines techniques. Dans cette section, nous illustrons une critique détaillée de la méthode $SubSet$ [120] et nous proposons une amélioration.

2.8.1 Présentation

Le principe de $Subset$ [120] est de trouver, pour chaque unité cachée ou de sortie, l'ensemble des sous-ensembles de poids positifs dont la somme des poids dépasse le seuil de cette unité. Ensuite, pour chacun de ces sous-ensembles, trouver l'ensemble des sous-ensembles de poids négatifs vérifiant la propriété : la somme des éléments du sous ensemble des poids positifs et de ceux du sous-ensemble des poids négatifs ne dépasse pas le seuil. Une règle est formée à partir de chaque couple de sous-ensemble de poids sélectionnés.

Une description informelle de cet algorithme est la suivante :

Pour chaque unité cachée ou de sortie :

- Soient :
  * $\mathcal{P}$ l'ensemble des poids des connexions de l'unité,
  * $\mathcal{P}_p$ l'ensemble des poids positifs de l'unité,
  * $\mathcal{P}_n$ l'ensemble des poids négatifs de l'unité,
  * $\mathcal{P} = \mathcal{P}_p \cup \mathcal{P}_n$.

- Trouver les sous-ensembles de $\mathcal{P}_p$ dont la somme des éléments est supérieure au seuil. Soit $\mathcal{S}_p$ l'ensemble de ces sous-ensembles.

- Pour chaque élément P de $\mathcal{S}_p$:
  * trouver les sous-ensembles Q de $\mathcal{P}_n$ qui vérifient la propriété : la somme des éléments de P et de ceux de Q est inférieure au seuil. Notons $\mathcal{S}_p(P)$ l'ensemble de ces sous-ensembles.
  * Pour chaque élément Q de $\mathcal{S}_p(P)$, former la règle suivante :
    `` Si P *et* $\bar{Q}$ alors *le concept désigné par l'unité* ".

Outre l'explosion combinatoire, cette méthode présente une autre faiblesse : elle ne permet pas dans tous les cas de trouver les bonnes règles comme le montre l'exemple suivant :

Soient θ le seuil d'un neurone, P est un sous-ensemble de poids positifs dont la somme est supérieure au seuil et Q un sous-ensemble de poids négatifs. Soient Q_1 et Q_2 deux sous-ensembles, dont l'union est égale à l'ensemble Q et tel que la somme des éléments de P avec les éléments de Q_1 ou de Q_2 est inférieure au seuil θ.

En appliquant l'algorithme $SubSet$ à cet exemple, les règles extraites sont de la forme :

$$\text{"Si } P \textit{ et } \bar{Q_1} \text{ alors } Conclusion\text{"}, \tag{2.1}$$

$$\text{"Si } P \textit{ et } \bar{Q_2} \text{ alors } Conclusion\text{"}. \tag{2.2}$$

Q_1 et Q_2 étant indépendants, de la règle 2.1, nous pouvons obtenir la règle suivante :

$$\text{"Si } P \textit{ et } \bar{Q_1} \textit{ et } Q_2 \text{ alors } Conclusion\text{"}. \tag{2.3}$$

D'autre part, la somme des éléments de P et de ceux de Q_2 ne dépassant pas le seuil, nous avons la règle :

$$\text{"Si } P \textit{ et } Q_2 \text{ alors } non(Conclusion)\text{"}. \tag{2.4}$$

En utilisant une nouvelle fois l'indépendance de Q_1 et Q_2, nous avons

$$\text{"Si } P \textit{ et } Q_2 \textit{ et } \bar{Q_1} \text{ alors } non(Conclusion)\text{"}. \tag{2.5}$$

Les règles 2.3 et 2.5 sont contradictoires.

Dans la section suivante, nous proposons une amélioration de cet algorithme qui permet de réduire l'explosion combinatoire tout en garantissant que les règles extraites ne sont pas contradictoires.

2.8.2 Amélioration

Pour améliorer la méthode $SubSet$, nous cherchons pour chaque sous-ensemble positif P, le sous-ensemble négatif Q tel que d'une part, la somme de leurs éléments est inférieure au seuil, et d'autre part la somme des éléments de P et de ceux du complément de Q dans l'ensemble des poids négatifs ($C(Q) = \mathcal{P}_n - Q$) est supérieure

au seuil. Une description de cette nouvelle version $SubSet^*$ est la suivante :

Pour chaque unité cachée ou de sortie :

- Soient :
  - * $\mathcal{P}$ l'ensemble des poids des connexions de l'unité,
  - * $\mathcal{P}_p$ l'ensemble des poids positifs de l'unité,
  - * $\mathcal{P}_n$ l'ensemble des poids négatifs de l'unité,
  - * $\mathcal{P} = \mathcal{P}_p \cup \mathcal{P}_n$.

- Trouver les sous-ensembles de $\mathcal{P}_p$ dont la somme des éléments est supérieure au seuil. Soit $\mathcal{S}_p$ l'ensemble de ces sous-ensembles.

- Pour chaque élément P de $\mathcal{S}_p$:
  - * trouver les sous-ensembles Q de $\mathcal{P}_n$ qui vérifient les deux propriétés suivantes :
    - + la somme des éléments de P et de ceux de Q est inférieure au seuil,
    - + la somme des éléments de P et de ceux du complément de Q $(C(Q) = \mathcal{P}_n - Q)$ est supérieure au seuil.
    - + Notons $\mathcal{S}_p(P)$ l'ensemble de ces sous-ensembles.
  - * Pour chaque élément Q de $\mathcal{S}_p(P)$, former la règle : ''Si P *et* $\bar{Q}$ alors *le concept désigné par l'unité*".

Cette amélioration, tout en réduisant l'explosion combinatoire, ne parvient pas à l'éviter complètement.

2.9 Nouvelle version de la méthode *"M-of-N"*

Dans cette section, nous présentons une adaptation de la méthode décompositionnelle *M-of-N* permettant d'extraire des règles du type *"M-parmi-N"* pour des

entrées $x_{ij} \in \{0, 1\}$. Les règles ayant cette forme présentent le double avantage de refléter le fonctionnement logique de neurones, et donc d'être plus compréhensibles et plus performantes, et d'éviter l'explosion combinatoire des autres méthodes tel que KT [42] et $SubSet$ [120].

2.9.1 Présentation

La méthode *M-of-N* a été proposée par Towell et Shavlik dans [120]. Une règle extraite par *M-of-N* est de la forme :

"Si *M parmi les N antécédents sont vrais* alors *conclusion*".

D'après Towell et Shavlik, l'un des atouts principaux de l'approche *M-of-N* est l'affinité naturelle entre les règles de cette forme et le seuil inductif des RNA. L'exemple FIG. 2.1 montre la façon dont les règles sont extraites à partir d'un neurone caché ou de sortie en appliquant la méthode *M-of-N*. L'algorithme *M-of-N* est décrit comme suit :

1. générer un RNA en utilisant le système $KBANN$;
2. lancer l'apprentissage en utilisant l'algorithme de rétro-propagation du gradient ;
3. pour chaque unité cachée ou de sortie, former des groupes de poids similaires ;
4. mettre les poids de chaque groupe à sa moyenne ;
5. éliminer tout groupe sans influence significative sur l'activation du neurone ;
6. garder tous les poids constants et optimiser le seuil de chaque unité cachée ou de sortie par l'algorithme de rétro-propagation du gradient ;
7. former une seule règle pour chaque unité cachée ou de sortie. Cette règle dépend du seuil et des poids des connexions restantes ;
8. simplifier l'ensemble des règles.

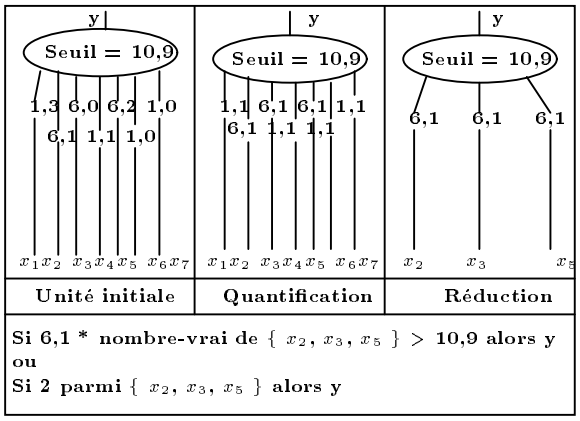


FIG. 2.1 – *L'extraction de règles par la méthode M-of-N*

La méthode *M-of-N* a été testée avec succès dans le domaine de la biologie moléculaire : la reconnaissance du *"prometer"* et la détermination du *"Splite-junction"* [120].

2.9.2 Adaptation

Dans cette adaptation de la méthode *"M-of-N"*, la forme *"M-parmi-N"* est présentée de deux manières différentes. Dans la première, la conclusion est vraie si et seulement si au moins M antécédents sont vrais. Dans la seconde, la conclusion est vraie si et seulement si au plus N-M antécédents sont vrais. Lorsque c'est la première présentation qui est utilisée, nous gardons la notation "M-parmi-N". Par contre, si c'est la deuxième présentation qui est utilisée, nous parlerons de règles de type *"non(M)-parmi-N"*. Autrement dit, selon qu'on utilise la notation *"M-parmi-N"* ou *"non(M)-parmi-N"*, M désigne soit le nombre minimal d'antécédents qui doivent être vrais pour que la conclusion soit vraie, soit le nombre maximal d'antécédents qui peuvent être faux sans empêcher la conclusion d'être vérifiée.

Soit un neurone caché i ayant n_i connexions d'entrée. Notons ω_{ij} le poids de sa j^e connexion d'entrée et θ_i le seuil de ce neurone que nous supposons positif. Pour extraire des règles, nous commençons par découper l'ensemble des poids positifs ω_{ij} en $(n_i + 1)$ intervalles (IV_{ij}, avec $0 \leq j \leq n_i$) définis de la manière suivante :

$$\begin{cases} IV_{i0} & = &]-\infty, \frac{\theta_i}{n_i}] \\ : & & \\ IV_{ij} & = &]\frac{\theta_i}{n_i+1-j}, \frac{\theta_i}{n_i-j}], \quad 1 < j < n_i \\ : & & \\ IV_{in_i} & = &]\theta_i, +\infty[\end{cases}$$

Soit $M_i = n_i + 1 - j$, pour chaque intervalle IV_{ij} nous extrayons la règle suivante :

"Si M_i parmi $\{I_j / \omega_{ij} \in IV_{ij}\}$ alors H_i".

D'après FIG. 2.2, nous remarquons en particulier que l'intervalle IV_{i0} ne donne pas de règles, ce qui signifie que les connexions dont le poids est inférieur à $\frac{\theta_i}{n_i}$ n'ont pas une influence significative sur l'activation du neurone. Nous éliminons donc les poids dont la valeur est inférieure à $\frac{\theta_i}{n_i}$ car nous jugeons qu'ils n'ont pas une grande influence sur l'activation du neurone.

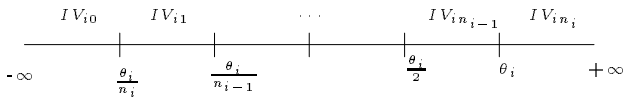


FIG. 2.2 – *Le regroupement des poids d'un neurone*

Ensuite, nous associons à chaque intervalle IV_{ij} les valeurs absolues des poids négatifs correspondants. Les poids négatifs de l'intervalle IV_{ij} ont une influence sur les poids positifs de cet intervalle. Autrement dit, l'effet d'un poids positif ω_{ij} appartenant à un intervalle IV_{ij} est annulé par un autre poids négatif du même intervalle. En tenant compte des poids positifs et négatifs de chaque intervalle IV_{ij}, la règle extraite d'un neurone caché et à partir de l'intervalle IV_{ij} est de la forme :

"Si $(M_j + M_j^N)$ *parmi* $\{I_j \ / \ \omega_{ij} \in IV_{ij}$ *et* $\bar{I}_j \ / \ -\omega_{ij} \in IV_{ij}\}$ alors H_i",

avec M_j (resp. M_j^N) le nombre de poids positifs (resp. négatifs) de l'intervalle IV_{ij}.

Après quelques expérimentations, nous avons pu constater que cette classification basée uniquement sur le seuil θ_i est insuffisante, mais seulement dans le cas où le seuil θ_i est strictement inférieur à la plupart des poids positifs et négatifs (en valeur absolue). Afin d'améliorer cette classification, nous avons introduit un deuxième critère pour segmenter les $(n_i + 1)$ intervalles en d'autres sous-intervalles. Il s'agit d'introduire un pas qui dépend de la valeur moyenne des poids positifs ou négatifs selon le signe du seuil.

Les règles extraites sont de la forme *"M-parmi-N"* avec $M \leq N$, mais si $M > N$, la règle ne peut pas être vérifiée. Pour éviter l'extraction de ce type de règles pour un intervalle donné IV_{ij} où le nombre de poids positifs appartenant à cette intervalle est inférieur à M_j, les poids de cette intervalle IV_{ij} sont ajoutés à l'intervalle IV_{ij+1}. Par conséquent, les poids d'un intervalle qui ne peuvent pas activer le neurone sont fusionnés avec les poids de valeurs inférieurs de l'intervalle suivant et ainsi de suite.

Nous avons aussi distingué un cas particulier : les neurones "stables". Ceux sont des neurones qui sont toujours actifs ou toujours inactifs indépendamment des valeurs de x_{ij}. Les seuils des neurones recevant la sortie de neurones "stables-actifs" sont diminués par les valeurs des entrées de ces neurones. Les neurones "stables-actifs" sont ceux dont la somme des poids négatifs est supérieure à leur seuil. Les neurones "stables-inactifs" sont ceux dont la somme des poids positifs est inférieure au seuil. Il résulte donc qu'une règle de la forme *"M-parmi-N"* où l'un des N antécédents est toujours non-actifs est équivalente à une règle de la forme *"M-parmi-(N-1)"*.

Dans le cas d'un seuil négatif θ_i d'un neurone i, le regroupement en $(n_i + 1)$ intervalles revient à trouver un ensemble de sous-ensembles de poids négatifs dont la somme des poids de chaque sous-ensemble ne dépasse pas le seuil de ce neurone. Puis pour chaque sous-ensemble, trouver l'ensemble des sous-ensembles de poids positifs du même intervalle où la somme des poids négatifs des sous-ensembles et des poids positifs de chaque sous-ensemble un à un ne dépasse pas le seuil du neurone. Autrement dit, nous cherchons les connexions qui peuvent inhiber le neurone. Donc, les règles extraites sont de la forme suivante :

"Si *non(M) parmi N antécédents* alors *Conclusion*",

où M est le nombre de connexions des deux sous-ensembles de poids positifs et négatifs.

2.9.3 Algorithme

Une description informelle de cette adaptation de l'algorithme *M-of-N* est la suivante :

1. apprendre le problème avec un RNA en utilisant l'algorithme de rétro-propagation du gradient ;
2. calculer la sortie de chaque neurone et cela pour chaque vecteur d'entrée ;
3. quantifier les sorties par regroupement ;
4. éliminer toute connexion sans influence significative sur l'activation du neurone ;
5. extraire des règles de la forme *M-parmi-N* pour les neurones du seuil positif et de la forme *non(M)-parmi-N* pour les neurones du seuil négatif ;
6. simplifier l'ensemble des règles.

Dans la quatrième étape de cet algorithme, nous avons jugé suffisant l'élimination des poids de faibles valeurs et qui ne dépassent pas une certaine valeur seuil. Dans cette étape, nous pouvons aussi utiliser le critère de l'information mutuelle pour mesurer la pertinence de chaque connexion d'entrée afin d'éliminer les connexions les moins pertinentes.

Une autre adaptation possible de cet algorithme est de remplacer des poids après l'apprentissage par les valeurs de regroupement de différents intervalles. Par la suite, relancer l'apprentissage en gardant les poids fixes afin d'optimiser les seuils des différents neurones.

L'algorithme résultant peut être réécrit comme suit :

1. apprendre le problème avec un RNA en utilisant l'algorithme de rétro-propagation du gradient ;
2. calculer la sortie de chaque neurone et cela pour chaque vecteur d'entrée ; 3. quantifier les sorties par regroupement en un ensemble fixe d'intervalles ;
4. sélectionner les connexions pertinentes en utilisant l'information mutuelle ;
5. garder les poids constants et optimiser les seuils en utilisant l'algorithme de rétro-propagation du gradient ;
6. extraire des règles de la forme *M-parmi-N* et *non(M)-parmi-N* ;
7. simplifier l'ensemble des règles extraites.

L'algorithme *M-of-N* a été testé avec succès sur des problèmes de biologie moléculaire. Par manque de compétence dans ce domaine et en raison de la complexité des problèmes testés par *M-of-N*, notre adaptation n'a pas pu obtenir des résultats meilleurs. Dans la section suivante, nous proposons une nouvelle direction de l'utilisation de l'information mutuelle[5] pour l'extraction de règles floues.

2.10 Extraction de règles floues : nouvelle direction

En logique propositionnelle, les entités manipulées sont des propositions. Ces dernières peuvent prendre l'une des deux valeurs : "vrai" ou "faux". Une proposition est composée de variables logiques reliées par des opérateurs logiques (ET, OU et NON). Il est bien établi que la logique propositionnelle est isomorphe à la théorie des ensembles en faisant correspondre, de manière appropriée, les opérateurs logiques aux opérateurs ensemblistes. De la même manière, à partir d'une théorie des ensembles flous, paramétrisés par le choix des opérateurs ensemblistes, une logique floue paramétrisée par des opérateurs logiques peut être construite. Dans cette logique, les antécédents de chaque règle sont entachés d'incertitude représentée par un

5. La notion de l'information mutuelle (IM) et son application aux RNA sont décrites dans le troisième chapitre.

degré de croyance [31]. Dans notre cas, ce degré de croyance est une valeur numérique (information mutuelle) [98].

2.10.1 Principe

Soit un neurone i, nous sélectionnons les connexions d'entrée les plus pertinentes. Le degré de pertinence de chaque connexion est défini par son information mutuelle. La règle floue extraite à partir du neurone i est de la forme "logique-numérique" suivante : "Si $\sum_j IM(x_{ij}; y_i) * I_j > \theta_i^*$ alors H_i", où x_{ij} une entrée pertinente, y_i la sortie du neurone et θ_i^* est la valeur du nouveau seuil calculée en fonction des poids, du biais et de l'information mutuelle moyenne. I_j est la représentation symbolique du j^e neurone d'entrée et H_i est la représentation symbolique du i^e neurone caché. Le *"Si ... alors ..."* formalise le raisonnement logique et l'expression $\sum_j IM(x_{ij}; y_i) * I_j > \theta_i^*$ formalise une information numérique.

2.10.2 Algorithme

Une description informelle de l'algorithme de cette méthode floue est la suivante :

1. apprendre le problème avec un RNA en utilisant l'algorithme de rétro-propagation du gradient ;
2. calculer la sortie de chaque neurone et cela pour chaque vecteur d'entrée ;
3. quantifier les valeurs de sortie des neurones par regroupement ;
4. calculer la pertinence des entrées en utilisant l'information mutuelle ;
5. calculer le seuil à l'aide de la formule suivante :
   $$\theta_i^* = \frac{\sum_j IM(x_{ij};y_i) * |\theta_i|}{\sum_j |\omega_{ij}|};$$
6. extraire des règles de la forme suivante :
   - à partir d'un neurone caché :
     ''Si $\sum_j IM(x_{ij}; y_i) I_j > \theta_i^*$ alors H_i" ;
   - à partir d'un neurone de sortie :
     ''Si $\sum_j IM(x_{ij}; y_i) H_j > \theta_i^*$ alors O_i".

Cette méthode n'ayant pas encore été testée, faute d'une application adaptée, il s'agit pour l'instant d'une simple "direction" de recherche.

2.11 Conclusion

Dans ce chapitre, nous avons décrit les principales techniques pour l'extraction de règles à partir d'un RNA. Ces techniques ont été classées selon l'approche qu'elles utilisent. Nous avons distingué trois approches : l'approche *"décompositionnelle"* dans laquelle les règles sont extraites au niveau de chaque neurone caché ou de sortie, l'approche *"pédagogique"* dans laquelle les règles sont extraites à partir du réseau en mettant en relation ses entrées avec ses sorties et l'approche *"éclectique"* qui combine les éléments des deux précédentes. Nous avons aussi présenté d'autres directions pour l'extraction d'autres types de règles. L'utilité de chacune de ces directions dépend du type de réseau, de la nature des entrées, de la complexité, de la nature de l'application et du niveau de transparence souhaité.

Compte tenu de la nature des problèmes où les RNA sont utilisés, il est clair que les techniques d'extraction de règles sont très utiles pour fournir une explication compréhensible du *comment* et *pourquoi* un RNA est arrivé à tel résultat ou telle conclusion. L'utilisateur est aussi intéressé d'une part par l'identification et si possible l'exploitation de la richesse potentielle de l'information existant dans un RNA et d'autre part par la découverte des relations existant entre un ensemble de données d'entrée et les sorties du RNA. Pour répondre à ces besoins, les techniques d'extraction de règles sont indispensables.

Dans le chapitre suivant, nous décrivons avec plus de détails nos deux méthodes d'extraction de règles nommées $MITER$ et $EMIRE$.

Chapitre 3

Extraction de règles symboliques à partir d'un RNA

3.1 Introduction

Comme le montre le chapitre précédent, beaucoup de travaux de recherches ont été consacrés à l'extraction de règles symboliques à partir d'un RNA ces dernières années. KT [42], SubSet [129], RuleNet [89], M-of-N [120], Full-RE [123], RULEX [5] et MITER [96] sont des exemples de techniques d'extraction de règles à partir d'un RNA. Toutes ces approches font partie des techniques décompositionnelles car l'extraction est faite au niveau de chaque neurone caché ou de sortie. Des heuristiques sont utilisées dans la plupart des méthodes de cette catégorie afin de limiter l'espace de recherche pour l'extraction de règles et pour améliorer la compréhension des règles extraites. VI-Analysis [126], RULENEG [109, 63], BRAINNE [118], BIO-RE [125] et EMIRE [97] sont d'autres techniques d'extraction de règles à partir des RNA sans tenir compte de sa structure interne (qui ne prennent en compte que les entrées et les sorties du RNA). Ces techniques sont nommées les techniques pédagogiques. DEDEC [128] et IIIE [11] font partie d'une troisième catégorie nommée éclectique qui combine les principes des deux catégories précédentes.

Le développement d'un module efficace d'extraction de règles est une tâche très difficile. Cette difficulté réside dans le passage d'un système numérique (un RNA) à un système symbolique dont la base de règles doit prendre les mêmes décisions que le RNA dont elles sont extraites. Pour développer une méthode d'extraction de règles, il faut donc prendre en considération la qualité des règles extraites liée principalement à l'amélioration de la compréhensibilité, la fidélité et l'exactitude de ces règles extraites.

Les réseaux de neurones artificiels ont prouvé leurs bonnes performances dans certains domaines (biologie, contrôle, traitement de signal, ...) [121]. Certains autres domaines, en particulier ceux où la sécurité est primordiale, comme la chirurgie médicale ou le nucléaire, leur sont restés fermés. Pour étendre le champ d'application des RNA à ces domaines, il faut les rendre capable de justifier leurs décisions [5]. Dans ce chapitre, nous proposons deux nouvelles méthodes pour extraire des règles d'un RNA, c.-à-d. pour le doter de la capacité d'explication.

Dans le première méthode, nous utilisons l'approche *"décompositionnelle"* décrite

dans le chapitre précédent. Notre méthode, nommée *MITER* (*Mutual Information and Template for Extracting Rules*), cherche pour chaque unité, le sous-ensemble de connexions d'entrée les plus pertinentes. Pour ce faire, elle utilise l'Information Mutuelle (IM) pour mesurer les dépendances entre les sorties et les entrées. Ces dépendances traduisent l'information contenue dans chaque connexion d'entrée durant l'apprentissage. Autrement dit, l'évaluation de l'IM de chaque connexion est une opération d'élagage qui sélectionne un sous-ensemble de connexions pertinentes à partir de toutes les connexions d'entrée [9]. Par la suite, nous justifions notre choix de l'IM parmi toutes les autres fonctions de la théorie de l'information (entropie, énergie, ...).

Les étapes de notre méthode *MITER* sont les suivantes : on commence par sélectionner, pour chaque unité cachée ou de sortie, les connexions d'entrée les plus pertinentes, puis on utilise cette sélection pour définir des *calibres des poids*. Chaque calibre est interprété par une règle symbolique de la forme *M-parmi-N*.

Notre seconde méthode, nommée *EMIRE* (*Examples and Mutual Information for Rules Extraction*), utilise l'approche *"pédagogique"* décrite dans le deuxième chapitre. *EMIRE* cherche pour chaque sortie du réseau, le sous-ensemble de neurones d'entrée les plus pertinents. Pour ce faire, elle utilise l'information mutuelle pour mesurer les dépendances entre les sorties et les entrées du réseau. Ces dépendances traduisent l'information apportée par chaque neurone d'entrée à une sortie du réseau. Autrement dit, l'évaluation de l'IM de chaque neurone d'entrée est une opération d'élagage qui sélectionne un sous-ensemble de neurones d'entrée pertinents à partir de l'ensemble des neurones d'entrée [9].

Les étapes de notre méthode *EMIRE* sont les suivantes : nous commençons par sélectionner, pour chaque unité de sortie, les neurones d'entrée les plus pertinents, puis nous utilisons cette sélection pour extraire une règle à partir de chaque exemple présenté à l'entrée du réseau. Ensuite, la base de règles est optimisée en utilisant l'une des techniques classiques [90, 117].

Ce chapitre est organisé de la manière suivante : nous commençons par définir la

notion d'information dans le contexte de la théorie de l'information.[1] Dans la troisième section, nous justifions le choix de l'IM comme fonction traduisant la quantité d'information contenue dans les connexions d'un RNA. La quatrième section est consacrée à notre méthode décompositionnelle $MITER$: la manière dont l'IM est calculée, l'utilisation de l'IM pour sélectionner les connexions pertinentes en vue de préparer l'extraction des règles, une description informelle de l'algorithme et des résultats expérimentaux. Dans la cinquième section, nous décrivons notre méthode pédagogique $EMIRE$ et nous donnons une description informelle de son algorithme, ainsi que les résultats de simulation.

3.2 Généralités

Nous allons commencer par définir quelques notions élémentaires pour le traitement de l'information : variable aléatoire, espace de probabilité et densité de probabilité. Ensuite, nous définissons l'entropie, l'information mutuelle et leur relation dans les deux contextes continu et discret.

3.2.1 Notions élémentaires

Une variable aléatoire (va) x pouvant prendre deux valeurs 0 ou 1 est appelée va de Bernoulli ou encore une va binaire. Soit a la probabilité que x prenne la valeur 1. On note : $P(x = 1) = a$, ce qui implique $P(x = 0) = 1 - a$.

Nous désignons par E_1 l'événement $x = 1$, la quantité d'information reçue par la réalisation de cet événement peut être évaluée par $-\log_2(a)$. De la même manière, la quantité d'information reçue par la réalisation de E_0 est égale à $-\log_2(1 - a)$. La quantité d'information moyenne reçue par la réalisation d'une va x est égale à la somme pondérée des quantités d'information qu'apporte chacune de ses réalisations. Dans le cas d'une va binaire, elle est donc égale à $-a \log_2(a) - (1 - a) \log_2(1 - a)$. Dans la suite, nous utiliserons toujours le logarithme à base deux, l'indice 2 sera

1. Godlewski, P. Introduction à la Théorie de l'Information et du Codage. *Notes de cours et exercices, ENST, 117 p., 1986.*

donc supprimé.

La probabilité d'une *va* x est désignée par $P(x)$. Soient deux *va* x et y, nous rappelons les formules suivantes :

- $P(x, y) = P(x|y)P(y) = P(y|x)P(x)$, où $P(x, y)$ est l'intersection des deux probabilités et $P(x|y)$ est la probabilité conditionnelle ;
- $P(y) = \sum_x P(x, y)$ dans le cas discret ;
- $P(y) = \int_x P(x, y)\, dx$ dans le cas de densités continues ;
- $P(x_1, x_2, .., x_n) = \prod_n P(x_i)$ dans le cas où les x_i sont indépendantes.

Soient Ω un ensemble dénombrable, généralement fini et $T = \mathcal{P}(\Omega)$ l'ensemble des événements probabilisables par rapport à Ω. Le couple (Ω, T) est appelé espace mesurable. La probabilité $P(E)$ est une application de (Ω, T) dans $[0, 1]$. (Ω, T, P) est un espace de probabilité avec les propriétés suivantes :

1. T contient $\emptyset$ et Ω ;
2. T est stable pour les opérations : complément, réunion et intersection ;
3. $P(.)$ est positive, $P(E) \geq 0$;
4. $P(.)$ est normée, $P(\Omega) = 1$;
5. $P(.)$ est σ-additive : la probabilité de l'union d'événements disjoints est égale à la somme des probabilités de ces évènements.

3.2.2 Définition de l'entropie

Soit x une *va* à valeurs dans un ensemble $\mathcal{A}$ fini ou dénombrable, c.-à-d. une application d'un espace probabilisable (Ω, T, P) dans $\mathcal{A}$. L'entropie de la *va* x est la quantité d'information définie par la formule suivante :

$$H(x) = -\sum_{a \in A} P(x = a) \log P(x = a), \tag{3.1}$$

où $P(x = a)$ est la probabilité de l'événement $x = a$ *(une valeur de* $\mathcal{A}$*)*. FIG. 3.1 représente la relation entre probabilité et entropie binaire donné par

$$H_2(x) = -a\ \log(a) - (1-a)\ \log(1-a),\ \text{où}\ \begin{cases} a = P(x=1) \\ \text{et} \\ 1-a = P(x=0). \end{cases}$$

D'après cette figure, on remarque que l'entropie est nulle pour une probabilité égale à 0 ou à 1, elle prend sa valeur maximale qui est égale à 1 pour une probabilité égale à $0,5$. L'entropie $H(x)$ est la quantité d'information qu'apporte, en moyenne, la réalisation de x, autrement dit l'entropie mesure l'incertitude attachée au phénomène aléatoire modélisé par x.

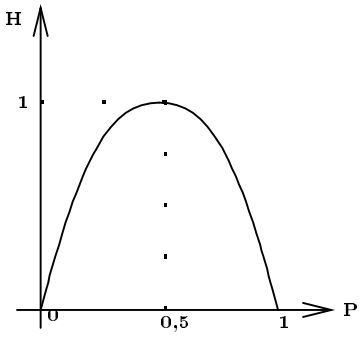


FIG. 3.1 – *Relation entre entropie binaire et probabilité*

Soient $a_1, a_2, ..., a_n$ les n éléments de $\mathcal{A}$ et soit $P_i = P(x = x_i)$. On peut écrire $H(x) = H(P_1, P_2, .., P_n) = -\sum_{i=1}^{n} P_i \log P_i$. L'entropie $H(x)$ a les deux propriétés suivantes :

1. $H(x)$ est positive ou nulle ;
2. $H(P_1, P_2, .., P_n)$ est maximale pour une distribution uniforme ($P_i = \frac{1}{n}$). On remarque que la plus grande incertitude correspond à l'équiprobabilité : $H(x) \leq \log(n)$.

Soient deux *va* x et y définies de l'espace (Ω, T, P) dans $\mathcal{A}$ et $\mathcal{B}$ respectivement. L'entropie $H(x, y)$ du couple (x, y) appelée l'information conjointe, est calculée par la formule suivante :

$$H(x,\, y) = \sum_{A,\, B} P(x,\, y) \log\, P(x,\, y). \tag{3.2}$$

L'entropie conditionnelle (information conditionnelle) $H(x|y)$ est définie par

$$H(x|y) = -\sum_{A,\, B} P(x,\, y) \log\, P(x|y) = \sum_{B} P(y) \sum_{A} -P(x|y) \log\, P(x|y). \tag{3.3}$$

3.2.3 Information mutuelle

La quantité d'information associée à la réalisation d'un événement $E \in T$ est définie par :

$$H(E) = -\log(P(E)). \tag{3.4}$$

Dans le cas de deux événements indépendants E et F, nous pouvons écrire :

$$P(E \cap F) = P(E).P(F), \tag{3.5}$$

ce qui implique

$$H(E \cap F) = H(E) + H(F). \tag{3.6}$$

D'une façon générale et pour $P(E) \neq 0$, nous pouvons écrire

$$H(E \cap F) = H(E) + H(E|F) \tag{3.7}$$

où $H(E|F) = -\log(P(E|F))$ est l'information associée à F après le calcul de la probabilité de E, ce qui donne $P(F|E) = P(E \cap F)/P(E)$.

L'information mutuelle mesure l'information apportée sur la réalisation d'un événement par la réalisation d'un autre événement. Cette mesure peut être négative. L'information mutuelle entre un événement E et lui-même n'est rien d'autre que l'entropie de cet événement.

L'information mutuelle de deux événements E et F, que nous noterons $IM(E\,;\,F)$, est donnée par la formule suivante :

$$IM(E\,;\,F) = H(E) + H(F) - H(E \cap F). \tag{3.8}$$

L'information mutuelle est symétrique entre deux événements. Elle est nulle si les deux événements sont indépendants. Nous pouvons réécrire l'équation 3.8 sous la forme suivante :

$$IM(E\,;F) = \log \frac{P(E \cap F)}{P(E)P(F)}, \tag{3.9}$$

ou encore sous la forme

$$IM(E\,;F) = \log \frac{P(F|E)}{P(F)}. \tag{3.10}$$

L'information mutuelle moyenne entre deux *va* x et y est définie par

$$IM(x\,;y) = \sum_x \sum_y P(x,\,y) \log \frac{P(x,\,y)}{P(x)P(y)}. \tag{3.11}$$

Cette information mutuelle moyenne est positive, mais elle peut être nulle dans le cas de deux variables indépendantes : $P(x,\,y) = P(x)\,.\,P(y)$.

Il est intéressant d'étendre la notion d'information mutuelle moyenne au cas continu où les variables aléatoires sont à valeurs dans un ensemble non dénombrable. L'IM est donnée par la formule suivante :

$$IM(x\,;\,y) = \int\int p(x,\,y) \log \frac{p(x,\,y)}{p(x)p(y)}\,dx\,dy. \tag{3.12}$$

Cette quantité d'information apportée par l'IM est positive et elle peut être infinie. La relation entre entropie et information mutuelle dans le cas continu reste la même que dans le cas discret Eq. 3.8. L'entropie dans le cas continu est appelée entropie différentielle.

3.2.4 Liens entre information mutuelle moyenne et entropies

FIG. 3.2 montre le lien existant entre entropies et information mutuelle. Les deux entropies $H(x)$ et $H(y)$ sont représentées par les aires des deux ellipses. L'IM est représentée par l'intersection des deux ellipses. La formule mettant en relation les entropies et l'information mutuelle moyenne est la suivante :

$$IM(x\,;\,y) = H(x) + H(y) - H(x,\,y) = H(y) - H(y|x). \tag{3.13}$$

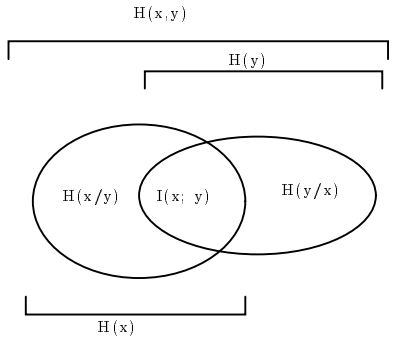


FIG. 3.2 – *Lien entre information mutuelle et entropies*

3.3 Pourquoi utiliser l'information mutuelle ?

En même temps qu'elle fournit une explication, une technique d'extraction de règles utilise l'information contenue dans les données d'entrée pour réduire l'incertitude sur les sorties. Cette réduction se heurte à deux difficultés : l'insuffisance de l'information d'entrée et l'opération d'apprentissage sous-optimale [24]. Si la première difficulté peut être surmontée aisément en complétant la base d'exemples, il n'en est pas de même pour la seconde. En effet, pour rendre optimale l'opération d'apprentissage, il faut sélectionner les caractéristiques et les connexions pertinentes. Pour ce faire, il est nécessaire d'utiliser un outil adapté et peu coûteux.

Dans ce travail, nous proposons l'information mutuelle comme outil pour réaliser cette double sélection. La raison de ce choix est que l'IM est le chemin naturel pour exprimer la pertinence des variables d'entrée pour la prédiction. Pour le prouver, nous allons réécrire l'équation Eq. 3.13 de la manière suivante :

$$\begin{aligned} IM(X\ ;Y) &= H(Y) - H(Y|X) \\ &= -\sum_y P(y)\log P(y) + \sum_x P(x)\sum_y P(y|x)\log P(y|x), \end{aligned} \tag{3.14}$$

où X est un ensemble de variables d'entrée et Y est une variable de sortie. La première sommation est constante et elle ne dépend pas des variables d'entrée. Par contre, la deuxième sommation en dépend et elle est fonction d'entropies conditionnelles. Par conséquence, plus la valeur de ces dernières est faible, plus l'IM est

élevée. Cela, implique une légère incertitude sur la valeur de sortie Y après avoir connu les variables d'entrée. Autrement dit, $IM(X\ ;Y)$ mesure combien X et Y sont dépendantes. Donc, $IM(X\ ;Y)$ est grande quand des entrées X de valeurs similaires produisent des sorties Y similaires, elle est petite quand des entrées X de valeurs similaires produisent des sorties Y différentes [16].

3.4 $MITER$: méthode décompositionnelle

Dans notre travail, nous utilisons l'information mutuelle dans un RNA entre deux variables x_{ij} (la j^e entrée du i^e neurone) et y_i (la sortie du i^e neurone), c.-à-d. la fonction qui mesure l'incertitude sur y_i après la connaissance de x_{ij}

$$IM(x_{ij}\ ;y_i) = H(y_i) - H(y_i|x_{ij}), \tag{3.15}$$

où $H(y_i)$ est l'entropie qui mesure l'incertitude initiale sur la sortie. Cette entropie est définie par :

$$H(y_i) = -\sum_{y\in\mathbb{Y}} P(y)\log P(y), \tag{3.16}$$

où $P(y)$ est la probabilité de la variable y_i pour une valeur donnée et $\mathbb{Y}$ représente le domaine de y_i. $H(y_i|x_{ij})$ est l'entropie conditionnelle qui mesure l'incertitude moyenne sachant la valeur du vecteur d'entrée. Elle est définie par :

$$H(y_i|x_{ij}) = -\sum_{x\in\mathbb{X}}\sum_{y\in\mathbb{Y}} P(y,x)\log P(y|x), \tag{3.17}$$

$$\text{où}\quad P(y,x) = P(y|x)P(x). \tag{3.18}$$

$P(y|x)$ est la probabilité conditionnelle de la sortie y_i par rapport à l'entrée x_{ij} pour une valeur donnée et $\mathbb{X}$ est le domaine de x_{ij}.

En utilisant les équations précédentes, nous pouvons réécrire la formule de l'information mutuelle comme suit :

$$IM(x_{ij}\ ;y_i) = \sum_{x\in\mathbb{X}}\sum_{y\in\mathbb{Y}} P(x,y)\log\frac{P(x,y)}{P(x)P(y)} \tag{3.19}$$

L'information mutuelle est la quantité par laquelle la connaissance fournie par le vecteur d'entrée réduit l'incertitude sur la sortie. Dans la section suivante, nous

décrivons comment utiliser l'IM pour mesurer la pertinence des connexions d'entrée au niveau de chaque neurone caché ou de sortie.

3.4.1 Sélection de connexions par information mutuelle

Dans les techniques d'extraction de règles à partir d'un RNA, nous sommes souvent confrontés à des contraintes pratiques sur le nombre de neurones et sur le nombre de connexions de chaque neurone. En réalité, l'information nécessaire à la détermination des sorties correctes du RNA avec une ambiguïté minimale est contenue dans un nombre limité de connexions. Nous pouvons donc, par une sélection adéquate, réduire le nombre de connexions.

Nous formulons ce problème de *réduction de Connexions* de la manière suivante : pour un neurone i donné ayant n_i connexions d'entrée, nous cherchons à trouver un sous-ensemble de m_i ($m_i \leq n_i$) connexions qui est *très informatif* sur l'activation du neurone. Un sous-ensemble de connexions d'entrée est informatif s'il contient des connexions capable de donner la bonne réponse. Autrement dit, notre problème peut être reformulé comme suit : soit un neurone i de sortie y_i et X_i l'ensemble des n_i connexions d'entrée, trouver le sous-ensemble $X_i^* \subset X_i$ avec m_i connexions d'entrée qui minimise $H(y_i/X_i^*)$, c.-à-d. qui maximise l'information mutuelle $IM(X_i^*\,;y_i)$.

Nous utilisons l'IM pour identifier les connexions pertinentes [15]. Comme chaque connexion contribue à l'activation du neurone, un sous-ensemble de connexions ne donne qu'une solution approximative. Cette solution est acceptable, car même si toutes les connexions qu'il contient sont pertinentes, l'ensemble de toutes les connexions n'est, en général, pas traité d'une façon optimale durant l'apprentissage.

Nous présentons tous les exemples de la base à l'entrée du RNA, et pour chaque exemple, nous calculons la sortie de chaque neurone caché ou de sortie. Si les entrées sont dans $\{0, +1\}$, les sorties des neurones seront dans l'intervalle $[0, 1]$. Cela est dû à l'utilisation dans l'apprentissage d'une fonction sigmoïde qui projette $I\!R$ dans $[0, 1]$. Avant d'utiliser l'information mutuelle pour sélectionner les connexions pertinentes de chaque neurone, nous devons quantifier les données par regroupement

afin d'avoir un nombre fixe de valeurs. Nous regroupons les valeurs de l'intervalle $[0, +1]$ de chaque neurone en k sous-intervalles. La valeur de k est définie de manière empirique. De la même façon, nous regroupons les valeurs entre -1 et $+1$ en k intervalles si les valeurs d'entrée sont dans $\{-1, +1\}$. Généralement, $k = 3$ et la taille du deuxième intervalle est trois fois plus grande que les deux autres, cela est due à la nature de la fonction sigmoïde utilisée dans l'apprentissage.

Pour chaque neurone, nous calculons l'information mutuelle $IM(x_{ij}; y_i)$, entre chaque connexion d'entrée et la sortie, puis nous choisissons les connexions qui maximisent l'information sur l'activation du neurone. La procédure de sélection des connexions d'entrée peut être décrite de la manière suivante :

Pour chaque neurone i avec n_i connexions d'entrée :

1. initialisation :
   - $X_i =$ ensemble de n_i connexions,
   - $X_i^* = \emptyset$;
2. quantifier l'ensemble des valeurs (les entrées et les sorties du neurone) en k intervalles, où k est une valeur expérimentale ;
3. pour chaque entrée x_{ij}, calculer l'IM avec la sortie $IM(x_{ij}\ ; y_i)$;
4. mettre dans X_i^* les entrées pour lesquelles $IM(x_{ij}\ ; y_i) \geq \frac{AvgIM_i}{k}$:
   $X_i^* = X_i^* \cup \{x_{ij}\}$, avec $AvgIM_i = \frac{\sum_j (IM(x_{ij}; y_i)}{n_i}$ l'IM moyenne entre les entrées et la sortie du neurone et k le nombre d'intervalles utilisé dans la quantification ;
5. restituer l'ensemble X_i^* des connexions sélectionnées.

Pour sélectionner les entrées les plus pertinentes d'un neurone, nous avons remarqué qu'il ne suffit pas de calculer l'information mutuelle entre chaque connexion d'entrée de ce neurone et sa sortie, mais il faut tenir compte en plus de l'information

mutuelle entre les différentes entrées de ce neurone [96].

Après avoir calculé l'IM entre chaque entrée et la sortie $IM(x_{ij};y_i)$ et entre chaque deux connexions d'entrée $IM(x_{ij};x_{ij'})$, nous choisissons les connexions pertinentes de la manière suivante : la première connexion sélectionnée est celle qui maximise l'information sur l'activation du neurone, ensuite parmi les connexions restantes, nous sélectionnons celles qui maximisent l'information sur l'activation du neurone et qui sont les moins liées avec les connexions déjà sélectionnées. Par exemple, si deux connexions x_{ij} et $x_{ij'}$ sont très liées, $IM(x_{ij};x_{ij'})$ sera très grande et après le choix de la meilleur connexion, la sélection de la deuxième sera pénalisée. Un paramètre β est utilisé pour régulariser l'importance relative de l'information mutuelle entre les connexions et il est nécessaire pour éliminer l'information redondante.

Une description informelle de cette procédure est la suivante :

Pour chaque neurone i avec n_i connexions d'entrée :

1. initialisation :
   - $X_i =$ ensemble de n_i connexions,
   - $X_i^* = \emptyset$;
2. quantifier l'ensemble des valeurs en k intervalles, où k est une valeur expérimentale ;
3. pour chaque connexion x_{ij}, calculer l'IM avec la sortie $IM(x_{ij}\ ;y_i)$;
4. choisir la connexion pour laquelle $IM(x_{ij};y_i)$ est maximum :
   - $X_i = X_i - \{x_{ij}\}$,
   - $X_i^* = \{x_{ij}\}$;
5. calculer l'information mutuelle moyenne $AvgIM_i = \frac{\sum_j IM(x_{ij};y_i)}{n_i}$;
6. calculer l'information mutuelle $IM(x_{ij};x_{ij'})$ pour chaque paire de connexions $(x_{ij},x_{ij'})$ avec $x_{ij} \in X_i$ et $x_{ij'} \in X_i^*$;

7. choisir la connexion qui maximise
$IM(x_{ij'}\ ;y_i) - \beta \sum_{x \in X_i^*} IM(x_{ij'}\ ;x)$
où β est une valeur expérimentale appartenant
à $]0, 1]$, généralement égale à 0,3.
8. si $(IM(x_{ij'}\ ;y_i) - \beta \sum_{x \in X_i^*} IM(x_{ij'};x)) > \frac{AvgIM_i}{k}$ alors
   - $X_i = X_i - \{x_{ij'}\}$,
   - $X_i^* = X_i^* \cup \{x_{ij'}\}$,
   - aller à 6 ;
9. restituer l'ensemble X_i^* des connexions sélectionnées.

3.4.2 Extraction d'une règle à partir d'un calibre de poids

Une fois les connexions pertinentes sélectionnées, nous remplaçons le vecteur de poids de chaque neurone caché ou de sortie par un autre vecteur plus facilement interprétable que nous appelons le *calibre de poids*. En plus, nous avons besoin d'associer à chaque neurone i un autre paramètre que nous notons M_i et qui représente le nombre de connexions pertinentes *nécessaires* à activer le neurone. Le *calibre de poids* permet de paramétrer une région dans l'espace de poids. Cette région correspond à une fonction symbolique spécifique [1]. Le calibre de poids T_i associé au neurone i est défini de la manière suivante :

- il a autant d'éléments que de connexions dans le neurone ;
- pour la connexion j, $t_{ij} = +V_i$ (resp. $-V_i$) si la connexion est pertinente et son poids est positif (resp. négatif) et $t_{ij} = 0$ si la connexion n'est pas pertinente.

La valeur de V_i qui minimise la distance euclidienne entre T_i et le vecteur de poids W_i ($||T_i - W_i||^2$) est donnée par :

$$V_i^* = \frac{\sum_j |\omega_{ij}|}{m_i} \tag{3.20}$$

où m_i est le nombre de connexions pertinentes et ω_{ij} est le poids de la j^e connexion. Dans la section suivante, nous démontrons comment nous avons abouti à cette valeur de V_i^*.

Démonstration

Soient deux vecteurs $T_i = (t_{i1}, t_{i2}, ... t_{in})$ et $W_i = (\omega_{i1}, \omega_{i2}, ... \omega_{in})$, nous cherchons à minimiser la distance euclidienne $||T_i - W_i||^2$, c.-à-d. la fonction

$$F(V_i) = \sum_J (t_{ij} - \omega_{ij})^2,\ J = \{t_{ij} \,/\, t_{ij} \in \{-V_i, 0, +V_i\}\}.$$

La valeur de V_i^* qui minimise cette fonction vérifie $\frac{dF(V_i)}{dV_i} = 0$.

$$F(V_i) = \sum_J (t_{ij} - \omega_{ij})^2,$$

$$F(V_i) = \sum_{J_0} (t_{ij} - \omega_{ij})^2 \ + \ \sum_{J_1} (t_{ij} - \omega_{ij})^2 \ + \ \sum_{J_{-1}} (t_{ij} - \omega_{ij})^2,$$

$$\text{avec } J = J_0 \cup J_1 \cup J_{-1}, \text{où } \left\{ \begin{array}{l} J_0 \ = \ \{t_{ij}/t_{ij} = 0\}, \\ J_1 \ = \ \{t_{ij}/t_{ij} = +V_i\}, \\ J_{-1} \ = \ \{t_{ij}/t_{ij} = -V_i\}. \end{array} \right.$$

$$F(V_i) = \sum_{J_0} \omega_{ij}^2 \ + \ \sum_{J_1} (V_i - \omega_{ij})^2 \ + \ \sum_{J_{-1}} (-V_i - \omega_{ij})^2.$$

$$\text{Nous avons sur } \left\{ \begin{array}{l} J_0 \ : \ t_{ij} = 0 \ \Rightarrow \ \frac{dt_{ij}}{dV_i} = 0, \\ J_1 \ : \ t_{ij} = V_i \ \Rightarrow \ \frac{dt_{ij}}{dV_i} = 1, \\ J_{-1} \ : \ t_{ij} = -V_i \ \Rightarrow \ \frac{dt_{ij}}{dV_i} = -1 \ ; \end{array} \right.$$

d'où

$$\frac{dF(V_i)}{dV_i} = 0 \ + \ 2 * \sum_{J_1} (V_i - \omega_{ij}) \ + \ 2 * \sum_{J_{-1}} (V_i + \omega_{ij}),$$

$$\frac{dF(V_i)}{dV_i} = 0,$$

$$\sum_{J_1} V_i - \sum_{J_1} \omega_{ij} + \sum_{J_{-1}} V_i + \sum_{J_{-1}} \omega_{ij} = 0.$$

Les valeurs de ω_{ij} sont positives sur J_1 et négatives sur J_{-1}, donc

$$\sum_{J_1} V_i + \sum_{J_{-1}} V_i - \sum_{J_1} |\omega_{ij}| - \sum_{J_{-1}} |\omega_i| = 0$$

$$m_i * V_i - \sum_{J_1 \cup J_{-1}} |\omega_{ij}| = 0.$$

Ce qui donne

$$V_i^* = \frac{\sum_j |\omega_{ij}|}{m_i}. \quad \square$$

Le paramètre M_i est calculé de la manière suivante : nous remplaçons dans l'équation de l'activation du neurone ω_{ij} par t_{ij} et θ_i par un nouveau seuil θ_i^* fonction de M_i. Nous obtenons l'équation suivante :

$$\sum_j t_{ij} x_{ij} + \theta_i^* > 0$$

Pour trouver la valeur de θ_i^*, nous distinguons deux cas :

Premier cas : les valeurs d'entrée x_{ij} sont dans $\{0, +1\}$.

Pour donner une valeur à θ_i^*, nous considérons le pire des cas, $x_{ij} = 1$ pour tout $t_{ij} = -V_i$. Dans ces conditions, nous pouvons écrire l'équation précédente comme suit :

$$-m_i^N V_i + M_i V_i + \theta_i^* > 0, \text{ c.-à-d. } \theta_i^* > (m_i^N - M_i) V_i,$$

où m_i^N représente le nombre de valeurs $-V_i$ dans le calibre et M_i est le nombre de valeur $+V_i$ nécessaires à l'activation du neurone. Cela signifie en particulier que le neurone ne peut être activé avec $(M_i - 1)$ connexions de valeur $+V_i$, c.-à-d. que

$$-m_i^N V_i + (M_i - 1) V_i + \theta_i^* \leq 0, \text{ c.-à-d. } \theta_i^* \leq (m_i^N - M_i + 1) V_i.$$

Nous choisissons donc pour θ_i^* une valeur comprise entre les deux valeurs $(m_i^N - M_i) V_i$ et $(m_i^N - M_i + 1) V_i$, soit

$$\theta_i^* = (m_i^N - M_i + \frac{1}{2}) V_i$$

Second cas : les valeurs d'entrée x_{ij} sont dans $\{-1, +1\}$.

Pour donner une valeur à θ_i^*, nous considérons le pire des cas, les entrées x_{ij} négatives (resp. positives) correspondent à des poids positifs (resp. négatifs). Dans ces conditions, nous procédons de la même façon que dans le premier cas et nous obtenons :

$$\theta_i^* = (m_i^N - M_i + \frac{1}{2})V_i + (m_i^P - M_i + \frac{1}{2})V_i, \text{ c.-à-d. } \theta_i^* = (m_i^N + m_i^P - 2M_i + 1)V_i$$

où m_i^P (resp. m_i^N) représente le nombre de connexions avec la valeur $+V_i$ (resp. $-V_i$), M_i est le nombre de fois où $x_{ij} * t_{ij}$ est égale à $+V_i$. m_i est le nombre de connexions pertinentes qui n'est rien d'autre que le nombre de composantes non-nulles du vecteur T_i, autrement dit $m_i = m_i^N + m_i^P$. Nous pouvons réécrire θ_i^* sous la forme :

$$\theta_i^* = (m_i - 2M_i + 1)V_i.$$

Pour trouver la valeur de M_i, nous cherchons la valeur $V_i^{'*}$ qui minimise la distance euclidienne $||T_i' - W_i'||^2$, où T_i' et W_i' s'obtiennent à partir de T_i et W_i en y ajoutant les seuils θ_i et θ_i^*. La valeur $V_i^{'*}$ est donnée par :

$$V_i^{'*} = \begin{cases} \frac{\sum_j |\omega_{ij}| + (m_i^N - M_i + \frac{1}{2})\theta_i}{m_i + (m_i^N - M_i + \frac{1}{2})^2} & \text{si} \quad x_{ij} \in \{0, +1\} \\ \frac{\sum_j |\omega_{ij}| + (m_i - 2M_i + 1)\theta_i}{m_i + (m_i - 2M_i + 1)^2} & \text{si} \quad x_{ij} \in \{-1, +1\}. \end{cases}$$

Dans la section suivante, nous démontrons comment nous avons abouti à cette valeur de $V_i^{'*}$.

Démonstration

Soient deux vecteurs $T_i' = (t_{i1}, t_{i2}, \ldots t_{in}, \theta_i^*)$ et $W_i' = (\omega_{i1}, \omega_{i2}, \ldots \omega_{in}, \theta_i)$. Nous pouvons noter $\theta_i^* = \lambda_i V_i$ et nous cherchons à minimiser la distance euclidienne $||T_i' - W_i'||^2$, c.-à-d. la fonction

$$F^*(V_i) = \sum_J (t_{ij} - \omega_{ij})^2 + (\lambda_i V_i - \theta_i)^2, \quad \text{où} \quad J = \{t_{ij} \,/\, t_{ij} \in \{-V_i, 0, +V_i\}\}.$$

$$F^*(V_i) = \sum_J (t_{ij} - \omega_{ij})^2 + (\lambda_i V_i - \theta_i)^2,$$

$$F^*(V_i) = \sum_{J_0}(t_{ij} - \omega_{ij})^2 \; + \; \sum_{J_1}(t_{ij} - \omega_{ij})^2 \; + \; \sum_{J_{-1}}(t_{ij} - \omega_{ij})^2 + (\lambda_i V_i - \theta_i)^2,$$

$$\text{avec } J = J_0 \cup J_1 \cup J_{-1}, \text{où } \left\{ \begin{array}{l} J_0 \; = \; \{t_{ij}/t_{ij} = 0\}, \\ J_1 \; = \; \{t_{ij}/t_{ij} = +V_i\}, \\ J_{-1} \; = \; \{t_{ij}/t_{ij} = -V_i\}. \end{array} \right.$$

$$F^*(V_i) = \sum_{J_0}\omega_{ij}^2 \; + \; \sum_{J_1}(V_i - \omega_{ij})^2 \; + \; \sum_{J_{-1}}(V_i + \omega_{ij})^2 + (\lambda_i V_i - \theta_i)^2.$$

$V_i'^*$ qui minimise cette fonction vérifie $\frac{dF^*(V_i)}{dV_i} = 0$.

$$\text{Nous avons donc sur } \left\{ \begin{array}{l} J_0 \; : \; t_{ij} = 0 \; \Rightarrow \; \frac{dt_{ij}}{dV_i} = 0, \\ J_1 \; : \; t_{ij} = V_i \; \Rightarrow \; \frac{dt_{ij}}{dV_i} = 1, \\ J_{-1} \; : \; t_{ij} = -V_i \; \Rightarrow \; \frac{dt_{ij}}{dV_i} = -1 \; ; \end{array} \right.$$

d'où

$$\frac{dF^*(V_i)}{dV_i} = 0 \; + \; 2 * \sum_{J_1}(V_i - \omega_{ij}) \; + \; 2 * \sum_{J_{-1}}(V_i + \omega_{ij}) + 2\lambda_i(\lambda_i V_i - \theta_i),$$

$$\frac{dF^*(V_i)}{dV_i} = 0,$$

$$\sum_{J_1} V_i - \sum_{J_1}\omega_{ij} + \sum_{J_{-1}} V_i + \sum_{J_{-1}}\omega_{ij} + \lambda_i^2 V_i - \lambda_i\theta_i = 0.$$

Les valeurs de ω_{ij} sont positives sur J_1 et négatives sur J_{-1}, donc

$$\sum_{J_1} V_i + \sum_{J_{-1}} V_i - \sum_{J_1}|\omega_{ij}| - \sum_{J_{-1}}|\omega_i| + \lambda_i^2 V_i - \lambda_i\theta_i = 0$$

$$\lambda_i^2 V_i + m_i * V_i - \sum_{J_{-1}\cup J_1}|\omega_{ij}| - \lambda_i\theta_i = 0.$$

Ce qui donne

$$V_i'^{*} = \frac{\sum_j |\omega_{ij}| + \lambda_i \theta_i}{m_i + \lambda_i^2}. \quad \square$$

La valeur de V'^{*} dépend du paramètre M_i. Nous choisissons ce dernier de telle sorte que $V_i'^{*} = V_i^{*}$, c.-à-d. que les distances euclidiennes entre T_i et W_i d'une part et entre T_i' et W_i' d'autre part atteignent leur minimum pour la même valeur. La valeur de M_i est donc la partie entière de la fraction suivante :

$$M_i = \begin{cases} \left[\frac{(m_i^N + \frac{1}{2}) \sum_j |\omega_{ij}| - m_i\, \theta_i}{\sum_j |\omega_{ij}|}\right] & \text{si} \quad x_{ij} \in \{0, +1\} \\ \left[\frac{(m_i + 1) \sum_j |\omega_{ij}| - m_i\, \theta_i}{2 \sum_j |\omega_{ij}|}\right] & \text{si} \quad x_{ij} \in \{-1, +1\}. \end{cases}$$

Une fois que nous associons un calibre T_i et une valeur M_i à chaque neurone caché ou de sortie, nous pouvons extraire, d'un neurone caché, une règle de la forme :

$$\text{Si } M_i \text{ parmi } \{\mathbb{I}_j,\ \bar{\mathbb{I}_j}\} \text{ alors } H_i,$$

$$\text{où } \mathbb{I}_j = \{I_j / \, t_{ij} = +V_i\} \text{ et } \bar{\mathbb{I}_j} = \{\bar{I_j} / \, t_{ij} = -V_i\},$$

et à partir d'un neurone de sortie, une règle de la forme :

$$\text{Si } M_i \text{ parmi } \{\mathbb{H}_j,\ \bar{\mathbb{H}_j}\} \text{ alors } O_i,$$

$$\text{où } \mathbb{H}_j = \{H_j / \, t_{ij} = +V_i\} \text{ et } \bar{\mathbb{H}_j} = \{\bar{H_j} / \, t_{ij} = -V_i\}.$$

Les règles extraites sont de la forme *M-parmi-N*. Le choix de cette forme de règles est justifié par des expériences indiquant que les RNA sont bien adaptés à l'apprentissage du concept *M-parmi-N* [40] et d'autres expériences utilisant *ID3* montrent l'utilité du concept *M-parmi-N* [94]. On peut aussi dire que c'est une forme intermédiaire entre le numérique et le symbolique. Les règles conjonctives sont obtenues si $M = N$ et les règles disjonctives si $M = 1$ [129].

Les règles extraites au niveau de chaque neurone caché ou de sortie sont réécrites sous forme de règles propositionnelles. L'ensemble de ces règles est optimisé (élimination de règles redondantes et des conclusions intermédiaires) pour former une base de règles reflétant le mieux possible le comportement de tout le réseau. Les opérations que nous venons de décrire (la détermination du calibre et de la valeur

de M_i, et l'extraction d'une règle de la forme *M-parmi-N*) peuvent être récapitulées par la procédure suivante :

Pour chaque unité sigmoïde de n_i connexions d'entrée et de seuil θ_i :

1. pour chaque connexion d'entrée x_{ij} :
   - calculer l'information mutuelle avec la sortie : $IM(x_{ij}\ ; y_i)$,
   - si $IM(x_{ij}\ ; y_i) < \frac{AvgIM_i}{k}$ alors $t_{ij} = 0$ sinon si $\omega_{ij} > 0$ alors $t_{ij} = +V_i$ sinon $t_{ij} = -V_i$. où $AvgIM_i = \frac{\sum_j IM(x_{ij}\ ; y_i)}{n_i}$ et k est la valeur expérimentale utilisée dans la quantification ;

2. si $x_{ij} \in \{0, +1\}$ alors $M_i = [\frac{(m_i^N + \frac{1}{2}) \sum_j |\omega_{ij}| - m_i\, \theta_i}{\sum_j |\omega_{ij}|}]$ sinon si $x_{ij} \in \{-1, +1\}$ alors $M_i = [\frac{(m_i + 1) \sum_j |\omega_{ij}| - m_i\, \theta_i}{2 \sum_j |\omega_{ij}|}]$;

3. pour un neurone caché, la règle extraite est de la forme : ''Si M_i parmi $\{\mathbb{I}_j, \bar{\mathbb{I}_j}\}$ alors H_i", où $\mathbb{I}_j = \{I_j / t_{ij} = +V_i\}$ et $\bar{\mathbb{I}_j} = \{\bar{I_j} / t_{ij} = -V_i\}$.
   où I_i représente le i^e neurone d'entrée et H_i représente le i^e neurone caché.

4. pour un neurone de sortie, la règle extraite est de la forme : ''Si M_i parmi $\{\mathbb{H}_j, \bar{\mathbb{H}_j}\}$ alors O_i" où $\mathbb{H}_j = \{H_j / t_{ij} = +V_i\}$ et $\bar{\mathbb{H}_j} = \{\bar{H_j} / t_{ij} = -V_i\}$.
   H_i représente le i^e neurone caché et O_i représente le i^e neurone de sortie.

L'exemple suivant illustre la relation entre le calibre des poids et la fonction symbolique : soit un neurone caché i avec trois connexions d'entrée pondérées par les poids ω_{i1}, ω_{i2} et ω_{i3} et un seuil θ_i. Pour $x_{ij} \in \{0, +1\}$, soient $T_i = (+V_i\ ,\ 0\ ,\ -V_i\ ,\ \frac{V_i}{2})$,

et $M_i = 1$. Nous extrayons donc la règle suivante : si 1 parmi $\{I_1, \bar{I}_3\}$ alors H_i, où I_1, I_2 et I_3 représentent les trois neurones d'entrée liés au neurone caché i représenté par H_i.

3.4.3 Algorithme

Une description informelle de l'algorithme utilisé pour l'extraction de règles est la suivante :

1. apprendre le problème avec un RNA multi-couches ;
2. pour chaque vecteur d'entrée, calculer la sortie de chaque neurone ;
3. normaliser les sorties par regroupement ;
4. sélectionner les connexions d'entrée pertinentes par l'utilisation de l'IM ;
5. définir le calibre des poids et le nombre de connexions nécessaires à l'activation du neurone ;
6. extraire des règles de la forme *M-parmi-N* ;
7. optimiser la base de règles extraites.

3.4.4 Expérimentations

Dans cette section, nous présentons les résultats de simulation [2] de notre méthode *MITER* appliquée à des problèmes artificiels.

2. Toutes les expérimentations réalisées au cours de cette thèse ont été faites sous le simulateur SN commercialisé par Neuristique [18].

a- Problème de Règle-plus-exception

Comme exemple d'une expression logique qui peut être traité par notre algorithme, nous décrivons le problème *Règle-plus-exception*. Ce problème est défini par l'expression suivante :

$$y = x_1 x_2 \vee \bar{x_1}\bar{x_2}\bar{x_3}\bar{x_4}$$

Pour le résoudre, nous définissons un RNA avec l'architecture <4-2-1> représenté par FIG. 3.3 après apprentissage. L'apprentissage consiste à minimiser une fonction de coût par ajustement des poids du réseau. Pour ce faire, nous avons employé l'une des mesures les plus utilisées, il s'agit de l'erreur quadratique, c.-à-d. le carré de la distance euclidienne entre sorties désirées et calculées. Ce qui présente l'adéquation entre le réseau et les données. L'apprentissage de cette expression logique se fait à partir de sa table de vérité en utilisant l'algorithme de la rétro-propagation du gradient. Les valeurs d'entrée sont dans $\{-1, +1\}$, $\beta = 0,3$ et $k = 5$.

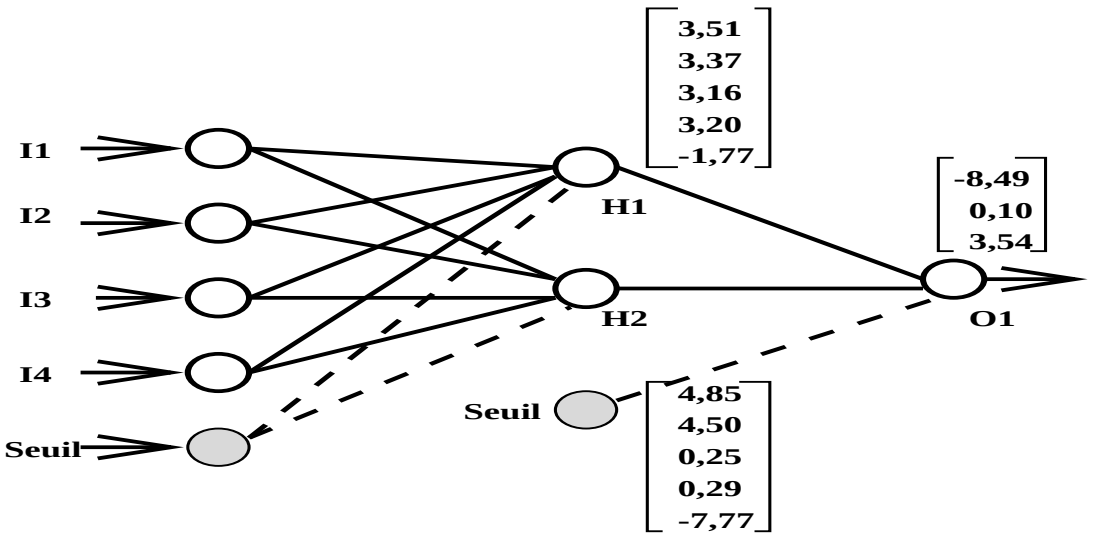


FIG. 3.3 – *Réseau obtenu après apprentissage*

Les résultats intermédiaires sont récapitulés dans les tableaux suivants :

TAB. 3.1 – *Règle-plus-exception : neurones cachés*

premier neurone caché				
poids	3,506	3,370	3,161	3,197
seuil	-1,772			
information mutuelle	0,066	0,066	0,066	0,066
$\frac{AvgIM_i}{k}$	0,013			
règle extraite	$Si\ 1\ parmi\ \{I_1,\ I_2,\ I_3,\ I_4\}\ alors\ H_1$			

second neurone caché				
poids	4,847	4,495	0,251	0,290
seuil	-7,775			
information mutuelle	0,311	0,311	0,000	0,000
$\frac{AvgIM_i}{k}$	0,031			
règle extraite	$Si\ 2\ parmi\ \{I_1,\ I_2\}\ alors\ H_2$			

TAB. 3.2 – *Règle-plus-exception : neurone de sortie*

neurone de sortie		
poids	-8,491	10,288
seuil	3,544	
information mutuelle	0,586	0,104
$\frac{AvgIM_i}{k}$	0,069	
règle extraite	$Si\ 1\ parmi\ \{\bar{H}_1,\ H_2\}\ alors\ O_1$	

Nous pouvons réécrire les règles extraites sous la forme :

$$\begin{cases} \text{Si} \quad I_1 \vee I_2 \vee I_3 \vee I_4 \quad \text{alors} \quad H_1, \\ \text{Si} \quad I_1 \wedge I_2 \quad \text{alors} \quad H_2, \\ \text{Si} \quad \bar{H}_1 \vee H_2 \quad \text{alors} \quad O_1. \end{cases}$$

En combinant ces trois règles, nous obtenons la règle finale suivante :

$$\text{Si } (I_1 \wedge I_2) \vee (\bar{I}_1 \wedge \bar{I}_2 \wedge \bar{I}_3 \wedge \bar{I}_4) \text{ alors } O_1,$$

En remplaçant les I_i par des x_i et O_i par y_i, nous obtenons la règle initiale apprise par le réseau.

$$y = x_1 x_2 \vee \bar{x_1}\bar{x_2}\bar{x_3}\bar{x_4}.$$

b- Problème de parité impaire

Dans ce problème, les objets sont représentés par des vecteurs de dix composantes à valeurs binaires. Un objet est un exemple si le nombre de 1 dans les trois premières composantes est impaire. Nous avons utilisé l'architecture <10-4-2> pour apprendre ce problème à partir de la table de vérité. Les valeurs d'entrée sont dans $\{-1, +1\}$, $\beta = 0,3$ et $k = 5$.

Les résultats pour les quatre neurones cachés sont :

TAB. 3.3 – *Parité impaire : neurones cachés*

premier neurone caché										
poids	-2,059	-2,433	-2,071	0,003	-0,031	0,070	0,036	-0,056	0,019	0,063
seuil	-0,014									
IM	0,205	0,181	0,165	0,003	0,000	-0,000	-0,000	-0,000	-0,000	-0,001
$\frac{AvgIM_i}{k}$	0,011									
règle	*Si 2 parmi* $\{\bar{I}_1,\ \bar{I}_2,\ \bar{I}_3\}$ *alors* H_1									

deuxième neurone caché										
poids	-3,988	-3,772	4,455	0,007	0,023	-0,027	-0,071	-0,057	0,047	-0,001
seuil	0,018									
IM	0,205	0,182	0,164	0,001	0,000	0,000	0,000	-0,000	-0,000	-0,001
$\frac{AvgIM_i}{k}$	0,012									
règle	*Si 2 parmi* $\{\bar{I}_1,\ \bar{I}_2,\ I_3\}$ *alors* H_2									

troisième neurone caché										
poids	4,608	-4,021	-4,213	0,031	0,041	-0,009	-0,025	-0,080	-0,033	-0,040
seuil	-0,067									
IM	0,205	0,204	0,182	0,001	0,000	0,000	0,000	-0,000	-0,000	-0,001
$\frac{AvgIM_i}{k}$	0,012									
règle	$Si\ 2\ parmi\ \{I_1,\ \bar{I}_2,\ \bar{I}_3\}\ alors\ H_3$									

quatrième neurone caché										
poids	2,287	-3,720	2,281	-0,044	-0,057	-0,054	0,013	0,022	0,002	0,018
seuil	0,042									
IM	0,595	0,257	0,227	0,001	0,001	0,000	0,000	0,000	-0,000	-0,001
$\frac{AvgIM_i}{k}$	0,022									
règle	$Si\ 2\ parmi\ \{I_1,\ \bar{I}_2,\ I_3\}\ alors\ H_4$									

Pour les deux neurones de sortie, nous avons obtenu les résultats suivants :

TAB. 3.4 – *Parité impaire : neurones de sortie*

premier neurone de sortie				
poids	-4,587	7,491	7,549	-6,924
seuil	-3,536			
information mutuelle	0,239	0,205	0,205	0,032
$\frac{AvgIM_i}{k}$	0,051			
règle extraite	$Si\ 2\ parmi\ \{\bar{H}_1,\ H_2,\ H_3\}\ alors\ O_1$			

deuxième neurone de sortie				
poids	3,108	-8,921	-8,970	8,234
seuil	1,555			
information mutuelle	0,239	0,205	0,205	0,032
$\frac{AvgIM_i}{k}$	0,051			
règle extraite	$Si\ 2\ parmi\ \{H_1,\ \bar{H}_2,\ \bar{H}_3\}\ alors\ O_2$			

Donc, les règles extraites à partir des quatre neurones cachés sont :

$$\begin{cases} \text{Si 2 parmi} \quad \{\bar{I}_1,\ \bar{I}_2,\ \bar{I}_3\} \quad alors \quad H_1 \\ \text{Si 2 parmi} \quad \{\bar{I}_1,\ \bar{I}_2,\ I_3\} \quad alors \quad H_2 \\ \text{Si 2 parmi} \quad \{I_1,\ \bar{I}_2,\ \bar{I}_3\} \quad alors \quad H_3 \\ \text{Si 2 parmi} \quad \{I_1,\ \bar{I}_2,\ I_3\} \quad alors \quad H_4 \end{cases}$$

et celles à partir des deux neurones de sortie sont :

$$\begin{cases} \text{Si 2 parmi} \quad \{\bar{H}_1,\ H_2,\ H_3\} \quad alors \quad O_1 \\ \text{Si 2 parmi} \quad \{H_1,\ \bar{H}_2,\ \bar{H}_3\} \quad alors \quad O_2 \end{cases}$$

La première sortie représente la classe des exemples. Nous pouvons réécrire la règle finale extraite de cette unité sous la forme suivante :

$$\text{Si } \bar{I}_1 I_2 I_3 \vee I_1 \bar{I}_2 I_3 \vee I_1 I_2 \bar{I}_3 \vee \bar{I}_1 \bar{I}_2 \bar{I}_3 \text{ alors } O_1.$$

Ce qui représente bien la classe des exemples.

c- Problème "Xor" bruité

Le problème "Xor" est identifié par Minsky et Papert comme une fonction logique non linéairement séparable [92]. Pour apprendre cette fonction, nous avons utilisé un RNA avec une architecture <4-2-1> et la table de vérité de cette fonction. Nous avons quatre variables d'entrée I_1, I_2, I_3 et I_4 et une variable de sortie O_1 représentant la fonction logique "Xor". Cette dernière est définie par

$$O_1 = (I_2 \wedge \bar{I}_4) \vee (\bar{I}_2 \wedge I_4).$$

Les deux variables I_1 et I_3 jouent le rôle de bruit.

Les règles extraites à partir des deux neurones cachés sont :

$$\begin{cases} \text{Si 1 parmi} \quad \{I_2,\ \bar{I}_4\} \quad alors \quad H_1 \\ \text{Si 2 parmi} \quad \{I_2,\ \bar{I}_4\} \quad alors \quad H_2 \end{cases}$$

et la règle extraite à partir du neurone de sortie est :

$$\begin{cases} \text{Si 1 parmi} \quad \{\bar{H}_1,\ H_2\} \quad alors \quad O_1 \end{cases}$$

Après une réécriture de ces trois règles, nous obtenons la règle "Xor" apprise par le réseau.

d- Les trois problèmes "Monk"

Définition

Le domaine du problème "Monk" est défini par une base[3] qui a permet une première comparaison internationale des différents algorithmes d'apprentissage [127]. "Monk" est composé de trois problèmes : "Monk 1", "Monk 2" et "Monk 3". Dans ces trois problèmes, l'objet nommé "Monk" est décrit par six attributs. Le tableau TAB. 3.5, décrit ces différents attributs et les valeurs qui peuvent leurs être attribuées. La base "Monk" est composée de 432 exemples.

TAB. 3.5 – *Les attributs et leurs valeurs du problème "Monk"*

Attribut	**Valeurs**
forme-tête	ronde, carré, octogone
forme-corps	ronde, carré, octogone
est-souriant	oui, non
tenir	épée, ballon, drapeau
couleur-veste	rouge, jaune, verte, bleue
a-une-cravate	oui, non

Les trois problèmes "Monk" sont définis comme suit :

1. **Monk 1 :** ce premier problème est défini par :
   Si (forme-tête = forme-corps) ou (couleur-veste = rouge)
   Alors "Monk".
   En utilisant la notation du tableau TAB. 3.5, cette règle peut être réécrite sous la forme :
   Si ((forme-tête = ronde) et (forme-corps = ronde)) ou
   ((forme-tête = carré) et (forme-corps = carré)) ou
   ((forme-tête = octogone) et (forme-corps = octogone)) ou
   (couleur-veste = rouge) Alors "Monk".

3. Cette base et un rapport technique sont disponibles par ftp à l'adresse archive.cis.ohio-state.edu dans le répertoire pub/neuroprose.

2. **Monk 2 :** ce deuxième problème est défini par :
   Si exactement deux parmi les six attributs ont leurs premières valeurs
   Alors "Monk".
   Cette règle peut être détaillée en un ensemble de quinze règles de la forme suivante :
   Si (forme-tête = ronde) et (forme-corps = ronde) et
   non (est-souriant = oui) et non (tenir = épée) et
   non (couleur-veste = rouge) et non (a-une-cravate = oui)
   Alors "Monk".
   Si on n'utilise pas de négation dans ces règles et on énumère tous les cas possibles, on peut décrire ce deuxième problème par 142 règles possibles de la forme suivante :
   Si (forme-tête = ronde) et (forme-corps = ronde) et
   (est-souriant = non) et (tenir = ballon) et
   (couleur-veste = jaune) et (a-une-cravate = non)
   Alors "Monk".

3. **Monk 3 :** ce troisième problème est défini par :
   Si ((forme-corps $\neq$ octogone) et (couleur-veste $\neq$ bleue)) ou
   ((tenir = épée) et (couleur-veste = verte))
   Alors "Monk".
   Cette dernière règle peut être détaillée en six règles. L'une de ces règles est la suivante :
   Si ((forme-corps = ronde) et (couleur-veste = rouge)) ou
   ((tenir = épée) et (couleur-veste = verte)) Alors "Monk".

Notation

Chacun des trois problèmes "Monk" utilise six attributs dont chacun prend plusieurs valeurs (de 2 à 4). D'après TAB. 3.5, 17 valeurs sont possibles pour les six attributs. Soient a_i le i^e attribut ($1 \leq i \leq 6$) et I_j la j^e valeur ($1 \leq j \leq 17$).[4]

4. Notant qu'avec cette nouvelle codification, on perd l'information de monovaluation des attributs ce qui rend le problème plus difficile.

Les trois problèmes "Monk" peuvent être redéfinis comme suit :

1. **Monk 1 :** ce premier problème est défini par la règle suivante :

   "Si $(a_1 = a_2)$ ou $(a_5 = 1)$ Alors Monk",

   où $a_5 = 1$ est la première valeur pour le 5^e attribut (rouge). En remplaçant les attributs par leurs valeurs, on obtient la règle suivante :

   "Si $I_1 I_4 \vee I_2 I_5 \vee I_3 I_6 \vee I_{12}$ Alors Monk".

2. **Monk 2 :** ce deuxième problème est défini par la règle suivante :

   "Si exactement 2 parmi $\{a_1 = 1,\ a_2 = 1,\ a_2 = 1,\ a_2 = 1,\ a_2 = 1,\ a_2 = 1\}$ Alors Monk",

   où 1 est la première valeur de l'attribut concerné (TAB. 3.5). En remplaçant les attributs par leurs valeurs, on obtient la règle suivante :

   "Si exactement 2 parmi $\{I_1,\ I_4,\ I_7,\ I_9,\ I_{12},\ I_{16}\}$ Alors Monk".

3. **Monk 3 :** ce troisième problème est défini par la règle suivante :

   "Si $(a_4 = 1$ et $a_5 = 3)$ ou $(a_2 \neq 3$ et $a_5 \neq 4)$ Alors Monk".

   En remplaçant dans cette dernière règle les attributs par leurs valeurs, on obtient la règle suivante :

   "Si $I_9 I_{14} \vee \bar{I_6} \bar{I_{15}}$ Alors Monk".

Expérimentation

Les résultats de simulation obtenus par *MITER* sur les trois problèmes "Monk" sont les suivants :

1. **Monk 1 :** nous avons utilisé l'architecture <17-4-1> pour apprendre ce premier problème défini par 432 exemples. Comme il a été recommandé pour

l'utilisation de cette base, nous avons utilisé 124 exemples pour l'apprentissage et le reste pour le test.

Les règles extraites à partir des quatre neurones cachés sont les suivantes :

$$\begin{cases} \text{Si 3 parmi} & \{I_1,\ I_3,\ I_4\} & alors & H_1 \\ \text{Si 3 parmi} & \{I_2,\ \bar{I}_3,\ I_5\} & \text{alors} & H_2 \\ \text{Si 4 parmi} & \{\bar{I}_1,\ \bar{I}_2,\ I_3,\ \bar{I}_5,\ I_6\} & \text{alors} & H_3 \\ \text{Si 4 parmi} & \{\bar{I}_1,\ \bar{I}_2,\ I_3,\ I_4,\ \bar{I}_5\} & \text{alors} & H_4 \end{cases}$$

La règle extraite à partir du neurone de sortie est la suivante :

$$\text{Si 1 parmi} \quad \{H_1,\ H_2,\ H_3\} \quad alors \quad O_1$$

Pour chaque attribut a_i, si l'une de ses composantes I_j est égale à 1, les autres composantes sont égales à 0. Après la réécriture des règles précédentes, nous obtenons la règle finale suivante :

"Si $I_1I_4 \vee I_2I_5 \vee I_3I_6 \vee I_3\bar{I}5$ Alors Monk".

Nous remarquons que cette règle n'est pas identique à la solution.

2. **Monk 2 :** nous avons utilisé l'architecture <17-2-1> pour apprendre ce deuxième problème défini par 432 exemples. Nous avons utilisé 169 exemples pour l'apprentissage et le reste pour le test.

Les règles extraites à partir des deux neurones cachés sont les suivantes :

$$\begin{cases} \text{Si 2 parmi} & \{I_1,\ I_4,\ I_7,\ I_9,\ I_{12}\} & \text{alors} & H_1 \\ \text{Si 3 parmi} & \{I_1,\ I_4,\ \bar{I}_8,\ I_9,\ I_{12}\} & \text{alors} & H_2 \end{cases}$$

La règle extraite à partir du neurone de sortie est la suivante :

$$\text{Si 2 parmi} \quad \{H_1,\ \bar{H}_2\} \quad alors \quad O_1$$

Pour chaque attribut a_i, si l'une de ses composantes I_j est égale à 1, les autres composantes sont égales à 0. Après la réécriture des règles précédentes, nous

obtenons la règle finale suivante :

"Si $I_1 I_4 \bar{I7} \bar{I9} \bar{I_{12}} \vee I_1 I_7 \bar{I4} \bar{I9} \bar{I_{12}} \vee I_1 I_9 \bar{I4} \bar{I7} \bar{I_{12}} \vee I_1 I_{12} \bar{I4} \bar{I7} \bar{I9} \vee I_4 I_7 \bar{I1} \bar{I9} \bar{I_{12}} \vee I_4 I_9 \bar{I1} \bar{I7} \bar{I_{12}} \vee I_4 I_{12} \bar{I1} \bar{I7} \bar{I9} \vee I_7 I_9 \bar{I1} \bar{I4} \bar{I_{12}} \vee I_7 I_{12} \bar{I1} \bar{I4} \bar{I9} \vee I_9 I_{12} \bar{I1} \bar{I4} \bar{I7}$ Alors Monk".

Nous remarquons aussi que cette règle n'est pas identique à la solution.

3. **Monk 3 :** nous avons utilisé l'architecture <17-2-1> pour apprendre ce troisième problème défini par 432 exemples. Nous avons utilisé 122 exemples pour l'apprentissage et le reste pour le test.

Les règles extraites à partir des deux neurones cachés sont les suivantes :

$$\begin{cases} \text{Si 2 parmi} \quad \{I_9,\ I_{10},\ I_{14}\} & \text{alors} \quad H_1 \\ \text{Si 1 parmi} \quad \{I_6,\ I_{15}\} & \text{alors} \quad H_2 \end{cases}$$

La règle extraite à partir du neurone de sortie est la suivante :

$$\text{Si 1 parmi} \quad \{H_1,\ \bar{H_2}\} \quad alors \quad O_1$$

Pour chaque attribut a_i, si l'une de ses composantes I_j est égale à 1, les autres composantes sont égales à 0. Après la réécriture des règles précédentes, nous obtenons la règle finale suivante :

"Si $I_9 \vee I_{14} \vee I\bar{10} \vee \bar{I_6} \bar{I_{15}}$ Alors Monk".

Nous remarquons, encore une fois, que cette règle n'est pas identique à la solution.

Étude comparative

Dans cette section, nous présentons les résultats obtenus par différentes méthodes et techniques d'extraction de règles à partir d'un RNA appliquées aux trois problèmes "Monk".[5]

TAB. 3.6 – *Les performances des techniques d'extraction sur "Monk"*

Technique connexionniste-symbolique	Monk 1	Monk 2	Monk 3
TBA [1]	100%	100%	97,2%
RULEX [5]	100%	100%	100%
LAP [62]	100%	100%	95%
RULENEG [62]	91%	56%	97%
DEDEC [128]	100%	100%	100%
TREX [50]	100%	85%	95,4%
MITER [96]	83,3%	84,3%	66,7%
REAL [29]	77%	62%	97%

TAB. 3.7 – *Les performances des RNA sur "Monk"*

Technique connexionniste	Monk 1	Monk 2	Monk 3
RNA	100%	100%	93,1%
RNA+Decay	100%	100%	97,2%

Les résultats obtenus sur le problème Monk sont comparables avec ceux obtenus par REAL [29] et ils sont moins bons que les autres méthodes d'extraction de règles à partir d'un RNA. Nous pensons que cette infériorité de performance de *MITER* par rapport aux autres méthodes sur le problème Monk est due à :

- **le nombre de règles extraites :** *MITER* extrait à partir de chaque neurone caché ou de sortie une seule règle de la forme *M-parmi-N* (cette règle peut être réécrite sous une forme normale disjonctive). Par contre, la plupart des autres méthodes extraient plusieurs règles à partir de chaque neurone ;
- **la pertinence des entrées :** *MITER* partage les connexions d'entrée de chaque neurone caché ou de sortie en deux classes seulement : pertinentes et

5. Toutes les méthodes présentées dans les tableaux TAB. 3.6, TAB. 3.7 et TAB. 3.8 ont été testées sur la même base et avec le même nombre d'échantillons en apprentissage et en test.

TAB. 3.8 – *Les performances des techniques symboliques sur "Monk"*

Technique symbolique	Monk 1	Monk 2	Monk 3
Cascade [127]	100%	100%	97,2%
AQ17-DCI [127]	100%	100%	94,2%
AQ17-HCI [127]	100%	93,1%	100%
AQ15-GA [127]	100%	86,8%	100%
AP [127]	100%	81,3%	100%
FOIL [127]	100%	69,2%	100%
Imind [67]	95,8%	82,9%	93,3%
ID3 [127]	98,6%	67,9%	94,4%
ID5R [127]	79,7%	69,1%	95,6%
AQR [127]	95,9%	79,7%	87,0%
CN2 [127]	100%	69%	89,1%
CLASSUEB [127]	71,8%	64,8%	80,8%
PRISM [127]	86,3%	72,7%	90,3%
ECOBWEB [127]	71,8%	67,4%	68,2%

non pertinentes. Cette division en deux classes a une influence directe sur les règles extraites : les entrées pertinentes interviennent dans les règles avec la même importance, par contre les entrées non pertinentes n'interviennent pas dans ces règles extraites. Dans la plupart des autres méthodes d'extraction de règles, les entrées interviennent dans les règles selon leur degré de pertinence. Autrement dit, dans les règles extraites à partir d'un neurone, l'entrée la plus pertinente intervient plus que les autres et ainsi de suite ;

– **le type de règle :** les règles extraites par *MITER* sont de la forme *M-parmi-N*. Elles peuvent être réécrites sous une forme normale disjonctive, où chaque conjonction comporte le même nombre d'antécédents. Or, la plupart des méthodes d'extraction de règles arrivent à surmonter cette dernière contrainte au prix d'un nombre élevé de règles extraites. Ce dernier conduit à un problème d'optimisation de règles qui nécessite l'utilisation d'heuristiques pour limiter

le nombre de règles et le nombre de prémisses par règles ;

- **le domaine dépendant :** à part *MITER* et *REAL*, les autres méthodes de TAB. 3.6 ont été testées et développées par rapport au problème "Monk". Cela permet en quelque sorte d'adapter et d'améliorer ces techniques en se basant sur les performances obtenues sur ce problème. Par ailleurs, *REAL*, qui n'a pas de bonnes performances sur "Monk", est la méthode la plus performante dans le domaine de la biologie moléculaire [29].

Le TAB. 3.9 résume les résultats obtenus par les RNA et par *MITER* sur les différents problèmes présentés précédemment. A chaque performance est associée un intervalle de confiance.[6]

TAB. 3.9 – *Les performances de MITER*

Problème	**Architecture**	**RNA**	***MITER***
Règle-plus-exception	<4-2-1>	100% [80,64%-100%]	100% [80,64%-100%]
Xor	<4-2-1>	100% [80,64%-100%]	100% [80,64%-100%]
Parité impaire	<10-4-1>	100% [99,63%-100%]	100% [99,63%-100%]
Monk 1	<17-4-1>	100% [99,12%-100%]	83,3% [79,49%-86,52%]
Monk 2	<17-2-1>	100% [99,12%-100%]	84,3% [80,57%-87,43%]
Monk 3	<17-2-1>	95,3% [92,87%-96,93%]	66,7% [62,13%-70,98%]

6. Les intervalles de confiance sont calculés à l'aide d'une formule adéquate [12].

3.5 *EMIRE* : méthode pédagogique

Dans cette section, nous allons présenter notre méthode d'extraction de règles *EMIRE* qui utilise l'approche pédagogique. Elle s'applique uniquement aux problèmes de classification. La première étape de la méthode consiste à déterminer les entrées pertinentes pour chaque sortie du réseau. Le calcul de la pertinence utilise l'information mutuelle [97]. Ensuite, pour un ensemble d'exemples présenté à l'entrée du réseau, une règle est extraite pour chaque exemple. Les règles extraites ne mettent en jeu que les sorties et les entrées pertinentes, et sont de la forme suivante :

"Si *liste-conjonctive* Alors *conclusion*".

Cette technique ne s'applique efficacement qu'à des réseaux de petite taille ayant des entrées binaires, mais elle présente en contrepartie les avantages suivants :

1. elle peut être utilisée pour l'extraction de règles à partir de n'importe quel type de réseau (RBF, MLP ou réseau récurrent) ;
2. elle ne demande pas un apprentissage spécifique (libre ou à contraintes) ;
3. elle extrait des règles indépendamment de l'architecture interne du réseau (nombre de neurones et de couches cachés) ;
4. elle permet l'utilisation des outils de minimisation logique déjà existants ;
5. elle ne demande pas une réécriture des règles (par exemple, de la forme M-parmi-N à une FND).

Une description détaillée d'*EMIRE*, ainsi que des résultats de simulation sont donnés dans les deux paragraphes suivants :

3.5.1 Algorithme

Une description informelle de l'algorithme de cette méthode est la suivante :

```
1. apprendre le problème avec un RNA en utilisant l'algorithme
   de rétro-propagation du gradient à partir d'une base
   d'exemples ;
```

2. détermination des entrées pertinentes :
   pour chaque neurone de sortie :
   - calculer l'IM de chaque entrée par rapport à cette sortie : $IM(x_{ij};y_i) = \sum_{x\in\mathbb{X}}\sum_{y\in\mathbb{Y}} P(x,y)\log\frac{P(x,y)}{P(x)P(y)}$,
   - déduire le seuil de cette sortie : $\theta_i^* = \frac{\sum_j IM(x_{ij};y_i)}{n_i * k}$, où n_i est le nombre de neurones d'entrée du réseau et k est la valeur de quantification utilisée dans le calcul de l'IM,
   - pour chaque entrée, comparer son IM avec le seuil θ_i^* et décider si elle est pertinente ;
3. pour chaque exemple de la base :
   a. le présenter à l'entrée du réseau,
   b. calculer la sortie correspondante,
   c. former une règle à partir des entrées pertinentes et de la sortie active. Cette règle est de la forme : "$Si\ [Non]\ I_i\ [ET\ [Non]\ I_j]^*\ alors\ O_i$", où I_i est la i^e entrée du réseau, O_i est la i^e sortie du réseau, [ ] est un terme optionnel et [ ]* signifie que [ ] peut être répété n fois où $n \geq 0$.
   Une entrée I_i intervient dans la prémisse sous une forme négative si la valeur correspondant dans l'exemple est égale à 0 ou -1, elle intervient sous une forme positive dans le cas contraire (pour la valeur $+1$).
   d. s'il s'agit d'une nouvelle règle, l'ajouter à la base de règles ;
4. optimiser la base de règles : supprimer toutes les règles contradictoires ou redondantes ;
5. combiner les règles ayant la même conclusion. La règle ainsi obtenue est donc en forme normale disjonctive.

3.5.2 Expérimentations

Dans cette section, nous présentons les résultats expérimentaux de la technique *EMIRE* sur des problèmes déjà présentés dans la partie expérimentation de la méthode *MITER* et dont le nombre de neurones d'entrée ne dépasse pas dix.

a- Problème de Règle-plus-exception

En utilisant l'information mutuelle (voir TAB. 3.10), seul les deux premières entrées sont sélectionnées. Nous présentons les différents exemples à l'entrée du réseau pour en obtenir des règles. Après l'élimination des deux dernières règles contradictoires, nous gardons la règle suivante : "Si $I_1 \wedge I_2$ Alors O_1." Nous remarquons

TAB. 3.10 – *Règle-plus-exception : neurones d'entrée*

information mutuelle	0,124	0,124	0,013	0,013
$\frac{AvgIM_i}{k}$	0,02			
règles extraites	$Si\ I_1 \wedge I_2\ alors\ O_1$ $Si\ \bar{I}_1 \wedge \bar{I}_2\ alors\ O_1$ $Si\ \bar{I}_1 \wedge \bar{I}_2\ alors\ \bar{O}_1$			

qu'*EMIRE* n'a pas pris compte la deuxième conjonction de la règle initiale. D'après la table de vérité de cette fonction logique, la deuxième conjonction n'influence pas fréquemment le résultat de cette fonction (elle n'intervient qu'une fois sur seize).

b- Problème de parité impaire

Les trois premières entrées ont été sélectionnées en utilisant l'information mutuelle. En suite, nous présentons les exemples à l'entrée du réseau. Une règle est extraite pour chaque exemple activant la sortie. Dans les règles extraites n'interviennent que les trois entrées pertinentes : I_1, I_2 et I_3. En éliminant les règles redon-

dantes, nous obtenons les quatre règles suivantes :

$$\begin{cases} \text{“Si } \bar{I}_1 I_2 I_3 \text{ alors } O_1\text{”}, \\ \text{“Si } I_1 \bar{I}_2 I_3 \text{ alors } O_1\text{”}, \\ \text{“Si } I_1 I_2 \bar{I}_3 \text{ alors } O_1\text{”}, \\ \text{“Si } \bar{I}_1 \bar{I}_2 \bar{I}_3 \text{ alors } O_1\text{”}. \end{cases}$$

Nous pouvons réécrire ces quatre règles sous la forme normale disjonctive suivante : “Si $\bar{I}_1 I_2 I_3 \vee I_1 \bar{I}_2 I_3 \vee I_1 I_2 \bar{I}_3 \vee \bar{I}_1 \bar{I}_2 \bar{I}_3$ alors O_1”, c'est la règle recherchée.

c- Problème “Xor” bruité

Nous utilisons l'IM pour mesurer la pertinence des quatre entrées par rapport à la sortie. Les deux entrées I_1 et I_3 ont une valeur (IM) proche de zéro. Nous allons donc prendre en compte seulement les deux entrées I_2 et I_4. En présentant à l'entrée du réseau les exemples de la table de vérité de “Xor”, nous obtenons directement la règle apprise par le réseau et représentant la fonction logique “Xor”.

Le TAB. 3.11 résume les résultats obtenus par les RNA et par *EMIRE* sur les problèmes présentés précédemment. A chaque performance est associée un intervalle de confiance.[7]

TAB. 3.11 – *Les performances d'EMIRE*

Problème	**Architecture**	**RNA**	***EMIRE***
Règle-plus-exception	<4-2-1>	100% [80,64%-100%]	93,75% [71,67%-98,89%]
Xor	<4-2-1>	100% [80,64%-100%]	100% [80,64%-100%]
Parité impaire	<10-4-1>	100% [99,63%-100%]	100% [99,63%-100%]

7. Les intervalles de confiance sont calculés à l'aide d'une formule adéquate [12].

3.6 Conclusion

Il y a un quasi-consensus sur le fait que les modèles hybrides connexionniste-symbolique constituent une voie prometteuse pour le développement de systèmes plus robustes et plus puissants que les systèmes symboliques et connexionnistes déjà existants. Dans ce chapitre nous avons proposé deux méthodes appartenant à ce modèle. Notre première méthode *MITER* offre une facilité pour l'extraction de règles à partir d'un RNA. En plus, elle est suffisamment souple pour être appliquée à des classes de problèmes très variées. Les résultats obtenus par cette méthode sont dûs essentiellement à l'utilisation de l'information mutuelle qui s'avère être un outil puissant pour mesurer les dépendances générales entre variables. Nous avons aussi montré l'utilité de notre seconde méthode *EMIRE* pour des applications d'entrées binaires et de petite taille.

Il est important de mentionner que l'extraction de toutes les règles est un problème NP-complet et l'alternative possible est d'extraire les principales règles qui couvrent la plupart des concepts de l'application étudiée.

Chapitre 4

Insertion de règles symboliques dans un RNA

4.1 Introduction

Bien que les RNA soient utilisés avec succès dans plusieurs domaines, il n'existe pas de méthode déterminante pour construire les réseaux adaptés à la résolution d'un problème donné. En effet, pour construire un tel réseau, nous sommes obligés de procéder de manière empirique. Cette construction présente l'inconvénient de la lenteur due d'une part au coût du choix de la topologie (nombre de couches et nombre de neurones par couche), et d'autre part, à celui de la convergence si elle a lieu. La lenteur de la convergence résulte d'une part, de l'initialisation empirique du réseau conduisant à une valeur élevée de la fonction de coût et d'autre part, à la présence de minima locaux.

Pour choisir une bonne topologie en évitant une construction empirique, nous pouvons partir d'une base de règles initiale qui permettra de construire directement le RNA. Cette base de règles initiale qui représente la théorie initiale du domaine peut être incertaine. Elle est aussi incomplète car dans les domaines mal maîtrisés, les experts arrivent à fournir quelques règles mais n'arrivent pas à cerner la totalité du problème.

La première difficulté à vaincre est donc de rendre les RNA capables d'utiliser les connaissances symboliques. Ensuite, le RNA construit doit pouvoir compléter la base de règles et la rendre plus certaine. Les travaux qui ont été faits, notamment par Shavlik et Towell, permettent d'obtenir un réseau de neurones à plusieurs couches. Le nombre de couches cachées est égal au nombre de conclusions intermédiaires. Or, tout problème traité par un réseau à plusieurs couches cachées peut être traité par un réseau à deux couches cachées.[1] L'objectif de l'initialisation d'un RNA à partir d'une base de règles est de construire le réseau de façon non empirique. Notre première idée a été d'utiliser la méthode d'initialisation proposé par Towell [129] sans la modifier, puis transformer le réseau résultant en un réseau équivalent à deux couches cachées. Autrement dit, éliminer ou fusionner des couches cachées du premier réseau pour aboutir à un réseau équivalent à deux couches cachées seulement [28]. Cette direction a été abandonnée faute de résultats. Nous proposons donc, deux nouvelles

1. D'après le théorème de Kolmogorov [45] et le théorème de Stone-Weierstrass [30, 72].

méthodes ayant le même objectif. La première, nommée *OpNeur*, permet d'initialiser un RNA à partir de la définition d'un concept donnée sous une forme normale disjonctive. La seconde, nommée *RuleNeur*, permet d'initialiser un RNA à partir d'une base de règles où le nombre de couches cachées est inférieur au nombre de conclusions intermédiaires [100]. En effet, la méthode *OpNeur* se base sur le choix d'une architecture réduite qui assure de bonnes performances, alors que la méthode *RuleNeur* s'intéresse au choix d'une architecture minimale du réseau.

Le chapitre est organisé comme suit : dans la section suivante, nous détaillons les raisons qui font l'intérêt de l'initialisation d'un RNA à partir d'une base de règles. La troisième section contient un aperçu général des différentes techniques développées pour l'initialisation d'un RNA. La quatrième section définit les types d'information symbolique et de règles utilisées dans l'initialisation. Les cinquième et sixième sections décrivent nos deux méthodes d'initialisation d'un RNA à partir d'une base de règles : *OpNeur* qui construit un neurone qui reflète l'un des deux opérateurs logiques ET ou OU et *RuleNeur* qui construit un neurone à partir d'une règle écrite sous une forme normale disjonctive.

4.2 Pourquoi initialiser un RNA à partir d'une base de règles ?

Plusieurs approches théoriques [30, 45, 72] ont montré que les RNA à trois couches sont capables d'effectuer l'approximation de fonctions continues de $I\!R^n$ dans $[0, 1]$. Ces approches sont fondées sur le théorème de Kolmogorov [45] et le théorème de Stone-Weierstrass [30, 72]. Dans le premier théorème, l'auteur précise le nombre d'unités du réseau mais il ne prédit pas la nature des neurones, c.-à-d. les fonctions d'activation qui permettent cette approximation. Dans le deuxième théorème, l'auteur montre qu'avec un réseau à trois couches et avec un nombre *"suffisant"* de neurones, on peut approximer toute fonction continue. Si ces deux théorèmes montrent l'existence d'un réseau de neurones à trois couches capable d'approximer n'importe quelle fonction continue, ils ne précisent ni comment construire le réseau, ni combien de neurones cachés sont nécessaires, ni comment calculer les paramètres de ce réseau. C'est l'utilisation d'une base de règles pour initialiser un RNA qui per-

mettra d'offrir une première solution aux deux premiers problèmes. Pour résoudre le troisième problème, l'algorithme de rétro-propagation sera appliqué au réseau ainsi construit pour adapter les paramètres du réseau (poids et seuils). Sans une bonne phase d'initialisation, l'algorithme de rétro-propagation risque de rencontrer des minima locaux et ainsi ne pas converger. L'initialisation d'un RNA à partir d'une base de règle apparaît ainsi comme une opération doublement utile : pour construire le réseau et pour assurer la convergence de l'algorithme calculant ses paramètres.

D'une façon théorique, il est possible d'approximer n'importe quelle fonction en utilisant un RNA multi-couches. En pratique, la détermination du réseau *ad hoc* et l'apprentissage des poids ne sont pas toujours faciles à obtenir. Lorsque un réseau multi-couches apprend à partir d'exemples une relation entre un ensemble d'entrées et un ensemble de sorties, il se heurte à trois difficultés essentielles :

1. la lenteur de la convergence de l'algorithme de rétro-propagation ;
2. l'existence des minima locaux ;
3. le choix de l'architecture initiale.

En initialisant le réseau à partir d'une base de règles, nous aurions plus de chance à vaincre les difficultés ci-dessus et à profiter d'autres avantages des RNA. Ainsi un réseau initialisé permet

1. une convergence plus rapide : en effet, l'initialisation du réseau se fait près d'un minimum *"intéressant"* de la fonction de coût et la probabilité de rencontrer un minimum local est faible ;
2. l'architecture du réseau est définie d'une manière non empirique ;
3. l'initialisation étant faite à partir d'une base de règles représentant la théorie initiale du domaine, ainsi l'apprentissage se fait non seulement à partir des exemples mais prendra en compte les connaissances du domaine ;
4. l'utilisation d'une base de règles permet de profiter de la puissance de certaines techniques symboliques (élagage des arbres de décision, construction des prototypes, ...).

4.3 Techniques d'initialisation d'un RNA

Depuis quelques années, plusieurs travaux ont été consacrés à une nouvelle manière de construire les RNA. Leur nouveauté réside dans l'utilisation de connaissances symboliques. Dans cette section, nous décrivons rapidement ces travaux avant de détailler notre contribution à ce domaine.

L'un des premiers travaux menés pour la construction d'un RNA guidée par une base de règles a été proposé par Towell et Shavlik [130]. Ces auteurs ont proposé un système hybride nommé *KBANN (Knowledge Base Artificial Neural Networks)* qui prend en compte des règles de la forme

"Si *liste-conjonctive-de-faits* Alors *Conclusion*",
ou
"Si *liste-disjonctive-de-faits* Alors *Conclusion*".

Dans [85], Mahoney et Mooney initialisent un réseau de neurones à partir de règles dont chacune est munie d'autant de coefficients de croyance qu'elle a de prémisses. A chaque règle, ils associent un neurone dont les poids des connexions d'entrée sont égaux aux coefficients de croyance.

Dans [86], Martin propose la construction d'une architecture d'un RNA à partir d'un arbre binaire. Les aspects pratiques de cette théorie ont été approfondis par Paugam-Moisy [106]. Cette méthode d'initialisation passe par trois étapes principales :

1. créer un RNA à partir d'un arbre binaire ;
2. ajouter d'une façon heuristique des connexions et des neurones en utilisant la connaissance *a priori* du problème étudié ;
3. entraîner le réseau en utilisant l'algorithme de rétro-propagation du gradient.

Dans [46], Gallant construit un système expert neuronal (automate linéaire à valeurs discrètes dans {-1, 0, +1}) dont la tâche est de concevoir la base de connaissances d'un système expert.

Dans [119], Sethi présente une méthode pour construire un RNA à deux couches cachées à partir d'un arbre de décision. Dans ce réseau, la première couche cachée est une couche de partitionnement pour représenter une règle de décision quelconque. Chaque neurone de la deuxième couche cachée réalise une conjonction et chaque neurone de la couche de sortie réalise une disjonction.

Dans [135], Yau et Manry présentent une méthode pour construire un RNA à partir des règles de Bayes sous hypothèse gaussienne.

Dans [34], Denoeux et al. présentent une méthode pour construire un RNA à partir de règles de plus proche voisin parmi un ensembles de prototypes.

Dans [94], Murphy et Pazzani présentent une méthode pour construire un RNA en utilisant l'algorithme ID3 de l'apprentissage symbolique.

Dans [54], Glorennec présente une méthode pour construire un réseau de neurones à partir d'un ensemble de règles de production en logique floue de Lukasiewicz ou de Zadeh.

4.4 Différents types d'information symbolique

Les règles d'initialisation d'un réseau de neurones peuvent être divisées en deux classes :

1. *Règles définitionnelles :* comme leur nom l'indique, ces règles donnent la définition d'une partie du domaine à étudier. Elles ne peuvent donc être modifiées. Il en résulte que les poids de connexions issues de ces règles restent invariants durant l'apprentissage. Pour assurer cette invariance, il est recommandé d'ajouter au terme d'erreur un terme régulateur R défini par

$$R = \lambda \sum_j \frac{(\omega_{ij}^{init} - \omega_{ij})^2}{1 + (\omega_{ij}^{init} - \omega_{ij})^2},$$

où ω_{ij}^{init} est le poids initial de la connexion entre les deux neurones i et j, ω_{ij} est le nouveau poids de cette même connexion et qui doit être très proche de

ω_{ij}^{init} et λ est un coefficient d'adaptation.

L'utilisation de ce type de règles présente l'inconvénient de rendre l'apprentissage lent. Cette lenteur trouve son origine dans l'ajout, à la fonction de coût, d'un terme régulateur R qui dépend de plusieurs poids.

2. *Règles variables :* ce sont des règles incertaines données par une méthode symbolique ou par un expert dans un domaine mal maîtrisé. Ces règles aident à initialiser le réseau, mais les poids correspondants peuvent être modifiés au cours de l'apprentissage.

C'est le deuxième type de règles que nous utilisons dans nos deux techniques d'initialisation. En plus, des contraintes sont imposées au type de règles à insérer et qui sont les suivantes :

- les règles sont de la Forme Normale Disjonctive :

  "Si *disjonction de plusieurs listes conjonctives de faits* alors *Conclusion*" ;

- l'opérateur de négation est accepté dans la partie prémisse d'une règle, mais non dans sa conclusion ;

- la base de règles est hiérarchique ;

- la base de règles est non récurrente.

Dans les deux sections suivantes, nous présentons nos deux techniques d'initialisation *OpNeur* (*from logical Operator to Neuron*) et *RuleNeur* (*from symbolic Rule to Neuron*) et nous montrons leur efficacité à travers des exemples. Dans la première méthode, à chaque règle nous associons plusieurs neurones, où chaque neurone réalise soit une conjonction ou une disjonction. L'objectif étant d'éviter d'avoir plus de deux couches cachées, cette méthode n'est utilisée que si la base de règles ne comporte pas de conclusions intermédiaires. Ce dernier cas est traité par la deuxième méthode qui à une règle associe un seul neurone.

4.5 *OpNeur* : insertion d'un opérateur logique dans un neurone

Dans cette section, notre but est d'initialiser un réseau de neurones à partir de la définition d'un concept. Ce dernier peut être défini par une ou plusieurs règles présentées sous une forme normale disjonctive. Ce concept peut être obtenu par une méthode symbolique (par exemple : *Imind* [67]) ou par un expert (dans le cas d'un problème artificiel : une ou plusieurs règles FND peuvent remplacer l'expert). Nous proposons donc une méthode d'initialisation d'un réseau à partir de la définition d'un concept. Cette méthode, nommée *OpNeur*, permet la construction d'un réseau à une couche cachée dans le cas de problèmes simples et la construction d'un réseau à deux couches cachées dans le cas de problèmes complexes. Si le problème est de nature inconnue, il est recommandé de commencer par une architecture simple (à une couche cachée).

Soit un neurone i de fonction d'activation

$$A_i = f(\sum_{i=1}^{n} w_{ij}x_{ij} - \theta_i).$$

La fonction f a deux états : si le neurone est actif, A_i tend vers 1, sinon elle tend vers 0. Selon la valeur du seuil, nous pouvons déterminer le type d'opération effectuée par le neurone. Pour des entrées binaires $x_{ij} \in \{0, +1\}$, nous pouvons distinguer deux types d'opérations :

1. **une conjonction :** le seuil θ_i est défini par $\theta_i = m_i^P * \omega - \mu$;
2. **une disjonction :** le seuil θ_i est défini par $\theta_i = \omega - \mu$.

Tel que, m_i^P le nombre de connexions pertinentes d'entrée (nous supposons que tout poids positif est pertinent) avec des poids positifs, ω le poids d'initialisation (une valeur arbitraire) et μ une valeur expérimentale comprise entre 0 et ω.

Les valeurs initiales des poids (ω) doivent être différentes de zéro, pour que les incréments d'adaptation des poids seront non nuls durant l'apprentissage qui utilise l'algorithme de rétro-propagation du gradient. Pour conserver une vitesse de

convergence acceptable, il est conseillé de choisir les poids initiaux en fonction des entrées afin de se situer dans la zone linéaire de la sigmoïde [68].

4.5.1 Première initialisation : cas simple

D'un point de vue théorique, pour toute fonction continue il existe un réseau à trois couches capable de l'approximer. Cependant, on ne connaît pas l'architecture exacte du réseau, le nombre optimum de neurones, le nombre de connexions et le nombre de couches du réseau de complexité minimale. Un nombre de neurones très petit induira une modélisation insuffisante, mais un très grand nombre entraîne une surparamétrisation du modèle qui nuira aux performances en généralisation. L'utilisation d'information *a priori* est très précieuse pour choisir une architecture initiale adéquate [68].

En se basant sur les travaux de Sethi [119] et sur la théorie de Denoeux [35] : *"Toute expression logique, mise sous forme normale disjonctive, peut être représentée par un réseau possédant une couche cachée d'unités ET, et une couche de sortie composée d'unités OU"*, nous construisons un réseau de neurones à trois couches :

1. une couche d'entrée : elle correspond aux antécédents de la règle ;
2. une couche cachée : elle réalise les conjonctions de la règle ;
3. une couche de sortie : elle réalise les disjonctions des différentes conjonctions de la couche cachée.

Une description informelle de la première phase de l'algorithme *OpNeur* qui nous permet la construction d'un RNA à trois couches est la suivante :

```
1. associer à chaque antécédent un neurone dans la couche
   d'entrée ;
2. associer à chaque conjonction du concept un neurone dans la
   couche cachée ;
3. associer à chaque disjonction du concept un neurone dans la
   couche de sortie ;
```

4. initialiser les poids : associer la valeur ω aux antécédents positifs et la valeur $-\omega$ aux antécédents négatifs (les négations) ;
5. initialiser les seuils :
   - les neurones de la couche cachée ont un seuil θ_i défini par : $\theta_i = m_i^P * \omega - \mu$
     où m_i^P est le nombre de poids d'entrée positifs de ce neurone et μ est une valeur expérimentale (généralement égale à $0,3$).
   - les neurones de la couche de sortie ont un seuil défini par : $\theta_i = \omega - \mu$;
6. ajouter à la couche d'entrée des neurones qui correspondent aux antécédents non présents dans la base de règles initiale ;
7. ajouter aux couches cachée et de sortie des neurones (le nombre de ces neurones ajoutés dépend de l'architecture du réseau) ;
8. ajouter des connexions pondérées par des poids proches de 0 afin d'obtenir un réseau totalement connecté ;
9. lancer l'apprentissage au niveau du réseau jusqu'à sa stabilisation.

Dans le cas d'un échec de la première phase de *OpNeur*, la deuxième phase présentée dans la section suivante est utilisée.

4.5.2 Deuxième initialisation : cas complexe

Certains problèmes peuvent être résolus par un petit réseau à quatre couches (deux couches cachées) alors qu'il faut une infinité de neurones avec un réseau à trois couches [23]. Pour cette raison, il ne faut pas se limiter sur un réseau à trois couches dont une cachée. Dans la deuxième phase de *OpNeur*, nous procédons d'une manière presque analogue à la première phase. Chaque conjonction de la règle est scindée en plusieurs conjonctions (selon le nombre d'antécédents par conjonction).

Les neurones de la première couche cachée réalisent les différentes sous-conjonctions. Les conjonctions de ces différentes sous-conjonctions sont réalisées par les neurones de la deuxième couche cachée. La couche de sortie réalise les différentes disjonctions. Le réseau ainsi obtenu comporte deux couches cachées.

Afin d'adapter notre méthode d'initialisation *OpNeur* à l'extraction de règles, les étapes suivantes peuvent être ajoutées à l'algorithme précédent.

10. si les neurones cachés réalisent des conjonctions et les neurones de sortie réalisent des disjonctions aller à 12, sinon modifier les seuils pour que les neurones cachés (resp. de sortie) réalisent des conjonctions (resp. disjonctions) ;
11. si les seuils viennent d'être modifiés pour la j^e itération, avec $j \leq Max$ aller à 9 (critère de la patience, Max est une valeur expérimentale) ;
12. si $j \leq Max$ alors extraire des règles FND (des conjonctions au niveau des neurones cachés et des disjonctions au niveau des neurones de sortie), sinon extraire des règles en utilisant l'une de nos deux méthodes $MITER$ ou $EMIRE$;
13. fin.

4.5.3 Expérimentations

TAB. 4.1, montre les différentes architectures obtenues par la méthode *OpNeur* sur les différents problèmes étudiés dans le chapitre 3.

Chacun de ces problèmes est défini par une règle de forme normale disjonctive. C'est pour cette raison que toutes les architectures, obtenues par *OpNeur*, comportent une seule couche cachée et une couche de sortie d'un seul neurone. La couche cachée réalise les conjonctions de la règle et la couche de sortie réalise les disjonctions de ces différentes conjonctions. Comme *OpNeur* ne prend en compte que

TAB. 4.1 – *Différentes architectures obtenues par "OpNeur"*

Problème	**Architecture initiale**	**Architecture complétée**	**Performance**
Règle-plus-exception	<4-2-1>	<4-2-1>	100% [80,64%-100%]
Xor	<2-2-1>	<4-2-1>	100% [80,64%-100%]
Parité impaire	<3-4-1>	<10-4-1>	100% [99,63%-100%]
Monk 1	<7-4-1>	<17-4-1>	100% [99,12%-100%]
Monk 2	<6-15-1>	–	–
Monk 3	<4-2-1>	<17-2-1>	95,3% [92,87%-96,93%]

les entrées présentes dans la règle, nous nous sommes conduit à compléter la couche d'entrée par les autres entrées non-présentes dans la règle initiale. Les neurones de ces dernières sont connectés aux neurones de la couche cachée. Ces connexions sont pondérées avec des poids proches de zéro.

Comme le montre TAB. 4.1, les différentes architectures ont atteint les performances souhaitées pour les différents problèmes à l'exception de "Monk 2". Dans ce dernier, nous avons abouti à une architecture avec un nombre élevé de neurones cachés. Vu la complexité de celle-ci, nous avons préféré de ne pas poursuive les tests.

4.6 *RuleNeur* : insertion d'une règle dans un neurone

Dans les algorithmes d'initialisation d'un réseau de neurones à partir d'une base de règles, un neurone est associé à chaque opérateur logique (ET ou OU). Par contre,

dans les algorithmes d'extraction de règles à partir d'un RNA (voir chapitre 2), une ou plusieurs règles sont extraites de chaque neurone caché ou de sortie. Chacune de ces règles utilise les différents opérateurs logiques (OU, ET et NON). Dans cette seconde méthode, notre objectif est de décrire comment l'initialisation d'un réseau de neurones à partir d'une base de règles n'est rien d'autre qu'une *opération réciproque* de l'extraction de règles. Autrement dit, dans les algorithmes d'extraction au moins une règle FND est extraite à partir de chaque neurone, nous proposons donc une méthode d'initialisation où une règle FND comportant plusieurs opérateurs logiques (ET, OU ou NON) est transformée en un seul neurone. Cette insertion peut être vue comme une fusion indirecte des neurones et elle permet d'obtenir un réseau avec une topologie beaucoup plus simple : moins de neurones et moins de couches cachées que d'autres techniques comme $KBANN$ [130]. L'initialisation et l'extraction apparaissent ainsi comme deux opérations *"réciproques"*.

4.6.1 Principe

Pour construire un RNA à partir d'une base de règles, la plupart des techniques d'initialisation utilisent les correspondances de TAB. 4.2 entre la base de règles et le RNA.

TAB. 4.2 – *Correspondances entre base de règles et RNA*

Base de règles	**RNA**
règle de décision	neurone formel
faits primaires	cellules d'entrée
conclusions intermédiaires	cellules cachées
conclusions finales	cellules de sorties
dépendances	connexions

Dans le réseau de neurones construit, chaque neurone réalise une conjonction ou une disjonction selon la valeur du seuil θ_i et des poids ω_{ij} [120].

La figure FIG. 4.1 montre une construction d'un RNA à partir d'une base de règles en utilisant $KBANN$ [130]. Dans cet exemple, les trois règles de la base

Base initiale	Base modifiée
si I_1 alors O_1	si I_1 alors H_1
si $I_2 \wedge I_3$ alors O_1	si $I_2 \wedge I_3$ alors H_2
si $I_4 \wedge I_5 \wedge I_6$ alors O_1	si $I_4 \wedge I_5 \wedge I_6$ alors H_3
	si $H_1 \vee H_2 \vee H_3$ alors O_1

Réseau de neurones initialisé ($KBANN$)

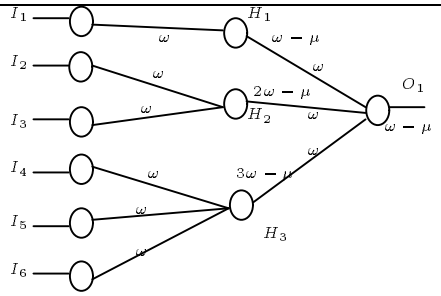


FIG. 4.1 – *Insertion d'une règle dans un neurone*

initiale ont la même conclusion. Pour construire le RNA, cette base de règles a été modifiée de manière à utiliser les conclusions intermédiaires H_1, H_2 et H_3. Le réseau construit a une seule couche cachée. Si les conclusions intermédiaires dépendaient à leur tour d'autres conclusions, nous obtenons un RNA à plusieurs couches cachées (le nombre de ces dernières est égale au nombre de conclusions intermédiaires). Pour des grandes bases, le nombre de couches cachées peut donc être très élevé. Or, nous savons qu'à tout réseau à plusieurs couches utilisé pour résoudre un problème donné, il existe un réseau équivalent ne comportant que deux couches cachées au plus [30, 45, 72]. C'est pour obtenir de tels réseaux que nous avons développé la méthode d'initialisation *RuleNeur* présentée dans cette section.

4.6.2 Démarche

Pour initialiser un RNA à partir d'une base de règles, toute en limitant le nombre de couches cachées à deux, nous associons un seul neurone caché ou de sortie à chaque

règle ayant la forme normale disjonctive.[2] La création d'un neurone à partir d'une règle ayant plusieurs opérateurs logiques, se fait à l'aide de la réciproque de la méthode *M-of-N* [129].

Notre méthode *RuleNeur* est basée sur le bon choix du seuil θ_i et des poids ω_{ij}. Pour toute règle présentée sous une FND, nous choisissons un seuil $\theta_i = \omega - \mu$, où ω est une valeur initiale arbitraire et $0 < \mu < \frac{\omega}{na_i}$ avec na_i le nombre d'antécédents de la règle r_i. Nous associons à chaque connexion un poids $\frac{\omega}{nac_{ij}}$ où nac_{ij} est le nombre d'antécédents de la j^e conjonction de la règle r_i.

En appliquant cette nouvelle initialisation à la base de règles initiale de FIG. 4.1, nous obtenons le RNA simplifié de la FIG. 4.2.

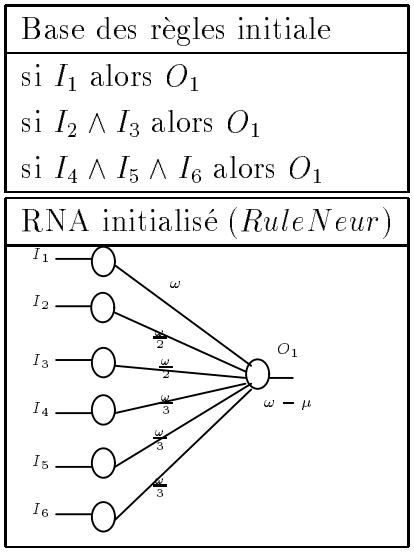


FIG. 4.2 – *Exemple 1 : initialisation d'un réseau de neurones*

L'extraction de règles à partir du neurone de sortie par la méthode *M-of-N* donne

2. Dans certaines cas complexes, l'association une règle/un neurone minimise le nombre de couches sans le limiter à deux. Toute fois, même dans ces cas, il est possible de réécrire les règles de manière à obtenir un réseau à deux couches cachées.

les mêmes règles que celles utilisées dans son initialisation. Dans ce premier exemple, la base de règles extraites est la même que la base d'initialisation. Cela est dû d'une part, aux prémisses qui ne se répètent pas dans les différentes conjonctions et d'autre part, au nombre d'antécédents qui diffère d'une conjonction à une autre. Dans la suite de ce paragraphe, nous étudions les cas particuliers où les règles extraites diffèrent des règles d'initialisation.

Nous utilisons la notation suivante pour décrire les quatre cas possibles d'initialisation d'un neurone i à partir d'une règle r_i :

r_i : la i^e règle écrite sous une forme normale disjonctive ;

ca_i : le cardinal de l'ensemble d'antécédents utilisés dans la règle ;

na_i : le nombre d'antécédents utilisés dans la règle, où $na_i \geq ca_i$ (un antécédent est compté autant de fois qu'il se répète dans les différentes conjonctions de la règle) ;

a_{ij} : j^e antécédent de la règle r_i ;

nac_{ij} : nombre d'antécédents de la j^e conjonction de la règle r_i ;

nc_i : nombre de conjonctions de la règle r_i.

Nous décrivons ces quatre cas possibles comme suit :

Cas 1 : dans ce premier cas, un antécédent ne peut être présent dans plus d'une conjonction et le nombre d'antécédents diffère d'une conjonction à une autre.

$$\text{Si } ca_i = na_i \; et \; nac_{ij} \neq nac_{il}, \quad \text{avec} \quad j, l \in \{1, ..., nc_i\} \quad \text{alors}$$

$$\begin{cases} \omega_{it} = \frac{\omega}{nac_{ij}}, & \text{avec} \quad t \in \{1, ..., na_i\} \quad et \quad j \in \{1, ..., nc_i\} \\ & et \\ \theta_i = \omega - \mu, & \text{avec} \quad 0 < \mu < \frac{\omega}{na_i}. \end{cases}$$

Dans ce premier cas, les règles extraites sont les mêmes que les règles d'initialisation (FIG. 4.2).

Cas 2 : dans ce deuxième cas, un antécédent ne peut être présent que dans une conjonction et le nombre d'antécédents peut être différent d'une conjonction à une autre.

$$\text{Si } ca_i = na_i \; et \; \exists \; j,l \in \{1,...,nc_i\} \; / \; j \neq l \; et \; nac_{ij} = nac_{il} \text{ alors}$$

$$\begin{cases} \omega_{it} = \frac{\omega}{nac_{ij}}, & \text{avec} \quad t \in \{1,...,ca_i\} \quad et \quad j \in \{1,...,nc_i\} \\ & et \\ \theta_i = \omega - \mu, & \text{avec} \quad 0 < \mu < \frac{\omega}{na_i}. \end{cases}$$

Dans ce deuxième cas, les règles extraites sont plus générales que les règles d'initialisation (FIG. 4.3).

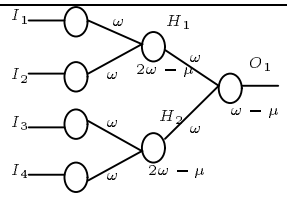
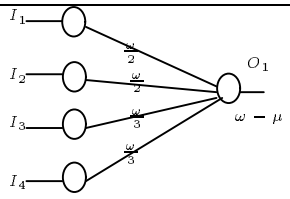
Base initiale	Base modifiée
si $I_1 \wedge I_2$ alors O_1 si $I_3 \wedge I_4$ alors O_1	si $I_1 \wedge I_2$ alors H_1 si $I_3 \wedge I_4$ alors H_2 si $H_1 \vee H_2$ alors O_1
Réseau initialisé ($KBANN$)	Réseau initialisé ($RuleNeur$)
	
Règle extraite (M-of-N) : Si 2 parmi $\{I_1, I_2, I_3, I_4\}$ alors O_1	

FIG. 4.3 – *Exemple 2 : initialisation d'un réseau de neurones*

Cas 3 : dans ce troisième cas, un antécédent peut être présent dans une ou plusieurs conjonctions et le nombre d'antécédents par conjonction peut être le même pour deux conjonctions différentes.

$$\text{Si } ca_i \neq na_i \; et \; \exists \; j,l \in \{1,...,nc_i\} \; / \; j \neq l \; et \; nac_{ij} = nac_{il} \text{ alors}$$

$$\begin{cases} \omega_{it} = \frac{\omega}{nac_{ij}}, & \text{avec} \quad t \in \{1,...,ca_i\} \quad et \quad j \in \{1,...,nc_i\} \\ & et \\ \theta_i = \omega - \mu, & \text{avec} \quad 0 < \mu < \frac{\omega}{na_i}, \end{cases}$$

Dans ce troisième cas, les règles extraites sont plus générales que les règles d'initialisation (FIG. 4.3).

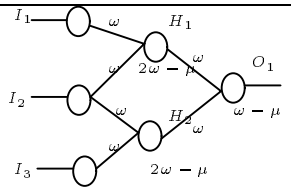
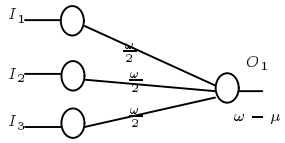
Base initiale	Base modifiée
si $I_1 \wedge I_2$ alors O_1 si $I_2 \wedge I_3$ alors O_1	si $I_1 \wedge I_2$ alors H_1 si $I_2 \wedge I_3$ alors H_2 si $H_1 \vee H_2$ alors O_1
Réseau initialisé ($KBANN$)	Réseau initialisé ($RuleNeur$)
	
Règle extraite (M-of-N) : Si 2 parmi $\{I_1, I_2, I_3\}$ alors O_1	

FIG. 4.4 – *Exemple 3 : initialisation d'un réseau de neurones*

Cas 4 : dans ce dernier cas, un antécédent peut être présent dans une ou plusieurs conjonctions et le nombre d'antécédents diffère d'une conjonction à une autre.

$$\text{Si } ca_i \neq na_i \; et \; \exists\, a_{ij},\; a_{il} \;/\; j \neq l \;,\; a_{ij} = a_{il} \; et \; nac_{ij} \neq nac_{il} \;\text{ alors}$$

$$\begin{cases} \omega_{ij} \;=\; \omega_{il} \;=\; \frac{\omega}{min(nac_{ij}, nac_{il})}, & j, l \in \{1, ..., ca_i\} \\ et & \\ \theta_i = \omega - \mu, & \text{avec } 0 < \mu < \frac{\omega}{na_i} \end{cases} .$$

Nous pouvons aussi extraire des règles qui ne peuvent pas être vérifies : des règles de la forme *M-parmi-N* avec $M > N$ (FIG. 4.4).

Après avoir étudié les différents cas d'initialisation d'un neurone à partir d'une

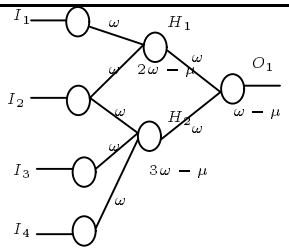
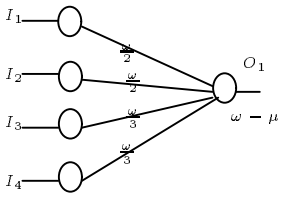
Base initiale	Base modifiée
si $I_1 \wedge I_2$ alors O_1 si $I_2 \wedge I_3 \wedge I_4$ alors O_1	si $I_1 \wedge I_2$ alors H_1 si $I_2 \wedge I_3 \wedge I_4$ alors H_2 si $H_1 \vee H_2$ alors O_1
Réseau initialisé ($KBANN$)	Réseau initialisé ($RuleNeur$)
	
Règles extraites (M-*of*-N) : Si 2 parmi $\{I_1,\ I_2\}$ alors O_1, Si 3 parmi $\{I_3,\ I_4\}$ alors O_1	

FIG. 4.5 – *Exemple 4 : initialisation d'un réseau de neurones*

règle FND, nous présentons une description informelle de l'algorithme *RuleNeur* :

1. écrire les règles de la base initiale sous la forme normale disjonctive ;
2. associer un neurone caché à chaque conclusion intermédiaire ;
3. associer un neurone de sortie à chaque conclusion finale ;
4. choisir le poids de chaque connexion en fonction du nombre d'occurrences de l'antécédent ;
5. choisir le seuil de chaque neurone en fonction de ses connexions d'entrée et de l'opération à réaliser ;
6. lancer l'apprentissage jusqu'à la stabilisation du réseau.

4.6.3 Expérimentations

TAB. 4.3, montre les différentes architectures obtenues par la méthode *RuleNeur* sur les différents problèmes étudiés dans le chapitre 3.

TAB. 4.3 – *Différentes architectures obtenues par "RuleNeur"*

Problème	**Architecture initiale**	**Architecture complétée**	**Performance**
Règle-plus-exception	<4-1>	<4-1>	93,7% [71,61%-98,87%]
Xor	<2-1>	–	–
Parité impaire	<3-1>	<10-1>	77,5% [74,84%-79,95%]
Monk 1	<7-1>	<17-1>	89,5% [86,25%-92,05%]
Monk 2	<6-1>	<17-1>	86,5% [82,95%-89,40%]
Monk 3	<4-1>	<17-1>	93,1% [90,31%-95,13%]

Chacun de ces problèmes est défini par une règle FND. C'est pour cette raison que toutes les architectures, obtenues par *RuleNeur*, ne comportent pas de couches cachées. Ces différentes architectures sont composées donc de deux couches seulement : une couche d'entrée et une couche de sortie. Cette dernière comporte un seul neurone réalisant une disjonction de plusieurs conjonctions de la règle FND. Pour chaque architecture, la couche d'entrée est complétée par les neurones non-présents dans la règle définitionnelle du problème correspondant. Les neurones de la couche d'entrée sont connectés aux neurones de la couche suivante (cachée ou de sortie). Ces connexions sont pondérées par des poids proches de zéro.

Le problème "Xor" est un problème non-linéairement séparable et qui nécessite

donc l'utilisation d'un RNA avec une couche cachée. Comme l'architecture obtenue par *RuleNeur* ne comporte pas de couche cachée, nous l'avons systématiquement rejeté.

Comme le montre TAB. 4.3, à l'exception du problème "Monk 3", les performances obtenues par les différentes architectures sont inférieures aux performances attendues. Nous pensons donc que *RuleNeur* est une méthode plus adaptée à des problèmes complexes définis par des règles qui contiennent des conclusions intermédiaires. Ces dernières permettent la création d'architectures avec des conclusions intermédiaires.

4.7 Conclusion

La façon triviale d'utiliser un réseau multi-couches est de choisir un réseau à trois couches avec suffisamment de neurones et totalement connecté entre couches voisines. De cette façon, le réseau obtenu est très complexe et dont l'apprentissage sera non seulement long mais incertain car il ne tient pas compte des spécificités du problème. Or, ces spécificités permettent, en général, de simplifier l'architecture du réseau et par la suite l'apprentissage au niveau de cette architecture [68]. L'intégration de connaissances explicites avec l'apprentissage à partir d'exemples semble être un moyen naturel de faire évoluer les modèles connexionnistes. D'une manière générale, l'utilisation des spécificités du domaine pour guider l'incorporation d'information symbolique permet de réduire le nombre de paramètres à apprendre par le réseau et facilite l'apprentissage.

Dans ce chapitre, nous avons proposé deux méthodes d'initialisation d'un RNA à partir d'une base de règles (*OpNeur* et *RuleNeur*). A travers ces deux méthodes, nous avons pu montrer que les connaissances symboliques peuvent être insérées dans un réseau de neurones afin de faciliter la tâche de sa construction et de l'apprentissage. Le RNA construit peut être utilisé pour le raffinement et l'amélioration d'une base de règles incertaine ou incomplète.

Chapitre 5

Système hybride connexionniste-symbolique

5.1 Introduction

Dans plusieurs domaines d'application, il est difficile d'acquérir un domaine complet de connaissances et de le représenter par des règles. De plus, les connaissances acquises peuvent être incertaines ou inconsistantes. Les systèmes experts peuvent aussi souffrir de la fragilité des règles qui demandent la satisfaction de tous les antécédents pour que la conclusion soit vérifiée. Des progrès substantiels ont marqué la dernière décennie pour résoudre ces problèmes. La capacité d'une représentation donnée par un traitement symbolique reste limitée quand les données d'entrée sont bruitées. Les modèles connexionnistes peuvent apprendre à partir d'un domaine bruité en donnant des bonnes performances. Par contre, ces modèles ne peuvent pas incorporer des connaissances initiales du domaine et il ne peuvent pas fournir une explication symbolique. De telles observations ont motivé plusieurs études de systèmes hybrides qui exploitent la complémentarité entre les caractéristiques du système symbolique et le paradigme RNA [42, 124, 129].

Ce chapitre est organisé comme suit : la section suivante donne un aperçu général des plus importants systèmes hybrides qui combinent les systèmes symboliques et les RNA. Dans la troisième section, nous présentons notre système hybride *RANNI* (*Rules and Artificial Neural Networks Interaction*) qui exploite les caractéristiques complémentaires de systèmes symboliques (apprentissage symbolique ou système expert) et de réseaux connexionnistes pour réviser le domaine de connaissances initial et améliorer ses caractéristiques. Nous décrivons aussi dans cette troisième section les trois composantes de ce système hybride et qui sont : le module basé sur les connaissances, le module connexionniste et le module probabiliste. La dernière section présente un mécanisme de collaboration entre deux méthodes d'apprentissage supervisé (symbolique et connexionniste) en vue d'obtenir un résultat satisfaisant. Nous étudions aussi la possibilité d'optimiser une base de règles en utilisant un réseau connexionniste, ainsi que la possibilité de réduire la topologie d'un réseau à l'aide d'une base de règles.

5.2 Différents systèmes hybrides

Plusieurs recherches ont été menées pour la conception de systèmes hybrides qui combinent les systèmes symbolique et connexionniste. La motivation principale de tel système hybride et d'incorporer la complémentarité entre système symbolique et RNA. Ces systèmes passent en général par quatre phases :

1. la phase de représentation de la base de règles, où le domaine de connaissances initial est extrait et représenté sous un format symbolique (exemple : système à base de règles) ;

2. la phase d'initialisation, où le domaine de connaissances présenté sous une forme symbolique est transformé sous forme d'une architecture connexionniste initiale ;

3. la phase d'apprentissage, où il s'agit de l'apprentissage au niveau de cette nouvelle architecture à partir de la base d'exemples en utilisant en général l'algorithme de rétro-propagation du gradient ;

4. la phase d'extraction de règles, où l'architecture connexionniste modifiée est transformée de nouveau sous forme d'une base de règles pour fournir une explication et justifier les réponses prises par le RNA.

KBANN (Knowledge Based Artificial Neural Network) est le premier système hybride qui a été développé par Towell et Shavlik [130] pour établir une complémentarité entre les caractéristiques de systèmes à base de connaissances et celles du RNA. *KBANN* est un système qui transforme un domaine de connaissances représenté en logique propositionnelle, sous forme d'une architecture neuronale. Une phase d'apprentissage, utilisant l'algorithme de rétro-propagation du gradient, est nécessaire pour le raffinement du domaine de connaissances initial. *KBANN* a été appliqué à deux problèmes de la biologie moléculaire et les résultats obtenus montrent qu'il généralise mieux que d'autres systèmes symboliques (système expert [57], ID3 [40], ...). Cependant, *KBANN* transforme des règles binaires en un RNA à plusieurs couches où le nombre de couches cachées est égal au nombre de conclusions intermédiaires. Les règles avec un facteur de certitude ne sont pas traitées dans ce système. De plus, le nombre de neurones cachés est ajouté, durant la phase d'apprentissage,

d'une façon empirique.

RAPTURE est un autre système hybride qui combine les méthodes d'apprentissage connexionniste et symbolique. Il a été développé pour la révision de bases de connaissances probabilistes [85]. *RAPTURE* est capable de traiter des règles ayant des facteurs de certitudes. Cependant, la structure du réseau obtenu par *RAPTURE* dépend d'une part du domaine d'application et d'autre part de la hiérarchie de la base de règles.

Un autre exemple de systèmes hybrides est le modèle *KBCNN (Knowledge Based Conceptual Neural Network)* proposé par Fu [44]. Ce système révise et apprend des connaissances en se basant sur un RNA transformé à partir d'une base de règles. Cette dernière représente la théorie initiale du domaine. Les trois modèles *KBANN*, *RAPTURE* et *KBCNN* ont quelques similarités dans leurs conception et fonctionnement.

Dans [51], Giacometti a proposé le système *SYNHESYS (SYmbolic-Neural Hybrid Expert SYstem Shell)*. Ce système est composé de deux modules : un système expert et un module connexionniste. Ces deux modules peuvent interagir à travers un gestionnaire d'interactions ou à travers des processus de transfert de connaissances. Les connaissances initiales utilisées dans *SYNHESYS* sont exprimées sous une forme propositionnelle et elles sont étendues à la logique floue.

Dans [124], Taha et Ghosh ont proposé le système hybride *HIA (Hybrid Intelligent Architecture)*. *HIA* est capable de réviser un domaine initial de connaissances et extraire de nouvelles règles basées sur les exemples d'apprentissage.[1] Ce système est superficiellement similaire au système hybride *KBANN* [120], cependant il

1. génère une architecture à trois couches indépendamment de la base initiale ;
2. révise les paramètres d'entrée en utilisant un codage spécifique ;
3. combine trois approches : symbolique, connexionniste et statistique pour extraire de nouvelles connaissances.

1. *HIA* a été testé avec succès sur un problème de contrôle d'un réservoir d'eau.

Dans [22], Challo et al. proposent le système hybride *FLNN (Fuzzy Logic Neural Networks)* qui combine un système connexionniste avec la logique floue. Dans *FLNN*, le RNA est utilisé d'une part, pour adapter les fonctions d'appartenance de variables floues et d'autre part, pour raffiner et extraire des règles floues. En général, un système standard de la logique floue a trois composantes :

1. le degré d'appartenance des entrées à un ensemble flou ;
2. une base de règles floues, qui représente les relations floues entre les variables d'entrée-sortie. La sortie de cette base est déterminée en se basant sur le degré d'appartenance ;
3. le moteur d'inférence, qui contrôle la base de règles.

Un tel système de la logique floue souffre de deux problèmes principaux : le premier est que la désignation correcte de la fonction d'appartenance qui représente chaque variable d'entrée et de sortie n'est pas triviale. L'approche habituelle est de designer une fonction initiale d'appartenance, généralement sous forme d'un triangle ou d'un trapèze, et elle est raffinée en utilisant des heuristiques. Le second problème est la difficulté de la fonction d'apprentissage à s'adapter à l'environnement du problème étudié. *FLNN* a pu surmonter ces deux problèmes en combinant la logique floue et les RNA. Dans ce système hybride flou, un RNA est utilisé pour remplacer le module de la base de règles floues. Les fonctions Max et Min sont généralement utilisées comme fonction d'activation dans ce réseau. Un algorithme d'apprentissage supervisé ou non-supervisé est utilisé à la place du moteur d'inférence pour adapter les paramètres de l'architecture du réseau. Après l'apprentissage, et à partir des exemples disponibles du domaine, le RNA est utilisé pour raffiner les fonctions initiales d'appartenance et la base de règles floues. En plus, il peut être utilisé pour extraire de nouvelles règles floues.

5.3 *RANNI* : système hybride

Dans cette section, nous présentons notre système hybride *RANNI* (*Rules and Artificial Neural Networks Interaction*) qui combine les systèmes à base de connaissances et les RNA [101]. *RANNI*, comme tout autre système hybride d'interac-

tion connexionniste-symbolique, passe par quatre phases : représentation de connaissances du domaine, transformation de ces connaissances sous forme d'un RNA, apprentissage au niveau du réseau et l'extraction de règles à partir de ce dernier. La dernière phase est la plus importante (mais aussi la plus difficile) car elle permet de munir un RNA de la capacité d'explication et de lui valider ses sorties (extraire des règles qui reflètent exactement le comportement du réseau). De plus, elle peut être utilisée pour raffiner et maintenir des connaissances initiales acquises à partir du domaine de l'expert.

RANNI, présenté par FIG. 5.1, est composé des trois modules suivants :

1. le module basé sur les connaissances, dans ce module, les connaissances sont définies dans un format basé sur des règles qui permettent l'établissement d'une architecture connexionniste. Ce premier module représente la théorie initiale du domaine donnée par un expert ou par un système d'apprentissage symbolique sous forme de règles ;

2. le module de l'architecture connexionniste, dans ce module : les règles obtenues par le premier module sont transformées sous forme d'une architecture connexionniste initiale. Cette nouvelle architecture (un RNA multi-couches) est entraînée en utilisant l'algorithme de rétro-propagation du gradient afin d'améliorer la théorie initiale du domaine. Après cette phase d'apprentissage, l'architecture connexionniste finale (avec des poids mis à jour) peut être vue comme une théorie révisée du domaine. Elle peut être utilisée pour mettre à jour le système symbolique initial en ajoutant de nouvelles règles et en mettant à jour d'autres règles existant déjà. En plus, elle peut être formulée, si besoin est, sous forme de règles pour accomplir la tâche d'explication ;

3. le module probabiliste, met à jour les deux modules précédents par le calcul de l'information mutuelle existant entre entrée-entrée et entrée-sortie. Ce troisième module probabiliste analyse l'ensemble des données et calcule l'information mutuelle entre les différents paramètres d'entrée et aussi entre les paramètres d'entrée et de sortie. L'information mutuelle est utilisée pour sélectionner les entrées pertinentes utilisées pour la construction de l'architecture initiale du réseau. Elle est aussi utilisée pour fournir des règles supplémen-

taires au premier module. Dans le cas où les deux systèmes symbolique et connexionniste ont des performances proches, il y a possibilité de fournir à l'utilisateur une décision basée sur la combinaison des deux décisions prises par les deux systèmes.

Dans les paragraphes suivants, nous représentons les différentes tâches accomplies par le système hybride *RANNI*.

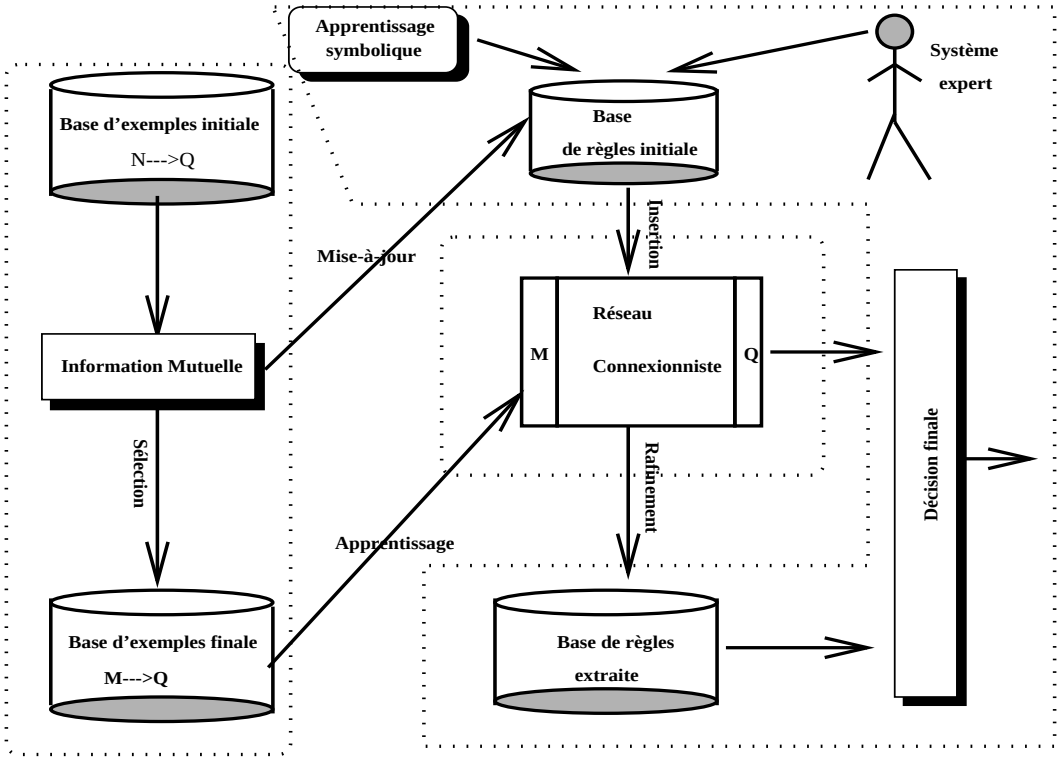


FIG. 5.1 – *RANNI : architecture hybride symbolique-connexionniste*

5.3.1 Représentation de connaissances initiales

La théorie initiale du domaine peut être divisée en deux parties : les connaissances qui représentent des règles opérationnelles du domaine et l'ensemble des données qui

représente un historique du domaine d'application. Dans plusieurs applications, l'acquisition de connaissances souffre de la difficulté de l'extraction d'un domaine complet de règles opérationnelles à partir d'un domaine représentatif. Donc, nous avons besoin de chercher une autre source d'information pour acquérir d'autres connaissances à partir du domaine d'application. Une source d'information est l'ensemble de données utilisé pour l'apprentissage et la validation au niveau du réseau. Le module probabiliste utilise l'information mutuelle pour d'une part, analyser l'ensemble des données disponibles afin d'extraire une information sous forme de règles symboliques, d'autre part sélectionner les entrées les plus pertinentes.

Une approche pour l'extraction d'information à partir de l'ensemble de données disponibles est d'étudier l'information mutuelle reliant ces données. Nous avons étudié deux approches différentes pour l'extraction du domaine de connaissances à partir de l'ensemble de données. La première est de calculer l'information mutuelle entre chaque couple entrée-entrée et entrée-sortie. La seconde est de sélectionner les entrées les plus pertinentes pour l'apprentissage [9].

Ces deux approches peuvent être appliquées indépendamment et leur résultats peuvent être combinés avec la base de règles initiale. Par la suite, nous présentons d'une part, comment des règles basées sur l'information mutuelle peuvent être extraites à partir de l'ensemble de données disponibles? et d'autre part, comment l'espace des caractéristiques d'entrée peut être réduit?

Dans la sélection des entrées, nous gardons toute entrée qui peut avoir une influence significative sur l'activation des sorties. Donc, pour toute entrée d'une information mutuelle supérieure à $\frac{AvgIM_i}{k}$, où $AvgIM_i$ est l'information mutuelle moyenne apportée par toutes les entrées et k est la valeur de quantification utilisée dans le calcul de l'IM. Par contre, pour la production de nouvelles règles nous ne tenons compte que des entrées fortement liées et des entrées d'une grande influence sur l'activation des sorties. Cette restriction est due à la possibilité de ces nouvelles règles de modifier la théorie initiale du domaine. Pour cette raison, une entrée ne peut intervenir dans une règle sauf si son information mutuelle dépasse l'information mutuelle moyenne ($AvgIM_i$) des entrées.

a- Réduction de la dimension d'entrée

Dans la plupart des applications, les données d'entrée sont bruitées et peuvent être redondantes. Pour obtenir des bonnes performances au niveau du réseau, il est très important de garder les données pertinentes et de réduire la complexité du réseau. Pour ces raisons, nous utilisons l'IM pour sélectionner les entrées pertinentes.

Une description informelle de cette étape est la suivante :

1. quantifier l'ensemble des valeurs par regroupement ;
2. calculer la pertinence de chaque entrée par rapport à toutes les sorties en utilisant l'information mutuelle ;
3. calculer le seuil à l'aide de la formule suivante :
$$\theta_i^* = \frac{\sum_i \sum_j IM(x_{ij};y_i)}{\|x_{ij}\|*\|y_i\|*k};$$
où $\|x_{ij}\|$ est le nombre des neurones d'entrée, $\|y_i\|$ est le nombre des neurones de sortie et k est la valeur de quantification ;
4. pour chaque sortie, garder les entrées dont l'IM dépasse le seuil θ_i^*.

Les entrées sélectionnées sont utilisées d'une part, pour la construction de l'architecture initiale du RNA et d'autre part, pour la révision de la base de règles initiale.

Par la suite, si le nombre des entrées sélectionnées est important, nous pouvons aussi utiliser un réseau diabolo[2] pour compresser cette dimension d'entrée. Les neurones de la couche cachée correspondent aux entrées compressées.

2. Un réseau diabolo est un perceptron multi-couches avec une seule couche cachée. Le nombre de neurones est le même pour la couche d'entrée et la couche de sortie, mais il est supérieur au nombre de neurones de la couche cachée. Il est entraîné pour reproduire les exemples reçus à son entrée sur sa sortie en minimisant une distance quadratique [10].

b- Production de règles

Les règles produites à partir d'exemples par le module probabiliste sont utilisées comme des contraintes pour la révision de la base de règles initiale ou comme des règles supplémentaires à cette base initiale. Nous distinguons donc deux types de règles :

1. **Nouvelles règles :** une description informelle de cette phase de production de règles est la suivante :

   - quantifier l'ensemble des valeurs par regroupement ;
   - pour chaque sortie :
     - * calculer l'information mutuelle entre chaque entrée et cette sortie ;
     - * calculer le seuil à l'aide de la formule suivante : $\theta_i^+ = \frac{\sum_j IM(x_{ij};y_i)}{\|x_{ij}\|}$; où $\|x_{ij}\|$ est le nombre des neurones d'entrée ;
     - * pour les entrées dont l'IM dépasse le seuil θ_i^+, ajouter à la base de règles initiale des règles de la forme suivante : ``Si I_j alors O_i".

2. **Règles de mise-à-jour :** une description informelle de cette phase de production de règles de mise-à-jour est la suivante :

   - quantifier l'ensemble des valeurs par regroupement ;
   - calculer l'information mutuelle entre chaque deux entrées ;
   - calculer le seuil à l'aide de la formule suivante : $\theta_i^{+*} = \frac{\sum_{j(j\neq k)} IM(x_{ik};x_{ij})}{\|x_{ij}\|-1}$; où $\|x_{ij}\|$ est le nombre des neurones d'entrée ;
   - pour les entrées dont l'IM dépasse le seuil θ_i^{+*}, produire des règles de la forme suivante : ``Si I_j alors I_k" ;
   - mettre à jour la base de règles initiale.

Pour la mise-à-jour de la base de règles initiale, nous distinguons les deux cas suivants :

2.a- Modification d'une règle de la base initiale : nous montrons par un exemple, le cas d'une mise-à-jour dans la base de règle initiale. Soient les deux règles suivantes :

$$\text{“Si } I_1 \textit{ et } I_2 \text{ alors } O_1\text{”}, \tag{5.1}$$

$$\text{“Si } I_1 \text{ alors } I_2\text{”}. \tag{5.2}$$

La règle 5.1 appartient à la base de règles initiale et la règle 5.2 est produite par le module probabiliste. Cette deuxième règle mis-à-jour la première et on obtient la règle suivante :

$$\text{“Si } I_1 \text{ alors } O_1\text{”}. \tag{5.3}$$

Les règles 5.1 et 5.2 sont remplacées par la règle 5.3.

2.b- Suppression d'une règle de la base initiale : nous montrons par un exemple, le cas de suppression d'une règle de la base de règles initiale. Soient les deux règles suivantes :

$$\text{“Si } \bar{I_1} \textit{ et } I_2 \text{ alors } O_1\text{”}, \tag{5.4}$$

$$\text{“Si } I_1 \text{ alors } I_2\text{”}. \tag{5.5}$$

La règle 5.4 appartient à la base de règles initiale et la règle 5.5 est produite par le module probabiliste. Cette deuxième règle mis-à-jour la première et on obtient la règle suivante :

$$\text{“Si } \bar{I_1} \textit{ et } I_1 \text{ alors } O_1\text{”}. \tag{5.6}$$

La règle 5.6 ne peut pas être vérifiée. Par la suite, les trois règles 5.4, 5.5 et 5.6 ne font pas partie de la base de règles révisée.

Après la mise-à-jour de la base de règles initiale (ajout, modification et suppression), nous obtenons une base de règles révisée. C'est cette dernière base qui va nous servir à la création de l'architecture connexionniste initiale.

5.3.2 Construction de l'architecture connexionniste

Dans cette phase, nous utilisons l'une de nos deux techniques (*OpNeur* ou *RuleNeur*)[3] pour la construction d'une architecture connexionniste à partir de la théorie initiale du domaine (base de règles révisée). Si le problème étudié est défini par des règles FND qui ne comportent pas de conclusions intermédiaires, il est recommandé d'utiliser la méthode *OpNeur*, sinon il faut utiliser la méthode *RuleNeur*. Cette nouvelle architecture multi-couches peut apprendre de nouveaux concepts durant la phase d'apprentissage à partir de l'ensemble de données et en utilisant l'algorithme de rétro-propagation du gradient [100].

5.3.3 Extraction de règles à partir d'un RNA

L'extraction de règles à partir d'un RNA est la composante principale[4] du système hybride *RANNI* et elle aide principalement à

1. fournir un raisonnement et une explication ;

2. alléger le problème d'acquisition de connaissances ;

3. raffiner le domaine de connaissances initial ;

4. fournir une aptitude de vérification.

Dû à ces aptitudes, l'extraction de règles à partir d'un RNA est très essentielle pour le développement d'un système hybride connexionniste-symbolique puissant, robuste et explicite.

Le module d'extraction de règles transforme l'architecture connexionniste de nouveau sous forme d'une base de règles [96, 97]. Cette transformation est beaucoup plus difficile que l'initialisation d'un RNA à partir d'une base de règles car un module

3. Ces deux techniques sont décrites dans le chapitre 4.

4. Nedjari, T. Extraction de l'Explication à partir d'un Réseau de Neurones Artificiel, *Poster présenté dans les journées de l'école doctorale de l'Institut Galilée, Université Paris 13, 1997.*

d'extraction de règles efficace doit être capable de respecter les trois conditions suivantes :

1. une garantie que les règles extraites ne sont pas en contradiction avec ce qui reconnu vrai dans le domaine ;
2. un raffinement du domaine incertain ;
3. une extraction de nouvelles règles.

Dans ce module, nous utilisons l'une de nos deux méthodes d'extraction de règles *MITER* ou *EMIRE*.[5] L'utilisation de l'une des deux méthodes dépend de la nature des entrées, de la complexité du réseau et du niveau de transparence souhaité. En général, s'il s'agit d'une architecture simple avec peu d'exemples, il est recommandé d'utiliser *EMIRE*, sinon c'est *MITER* qui est utilisée.

5.4 Coopération symbolique-connexionniste

Dans plusieurs domaines la coopération entre des méthodes symboliques et numériques est nécessaire. Un point focal dans cette collaboration est le passage du numérique au symbolique et vice-versa. Dans les trois sous-sections suivantes, nous présentons un mécanisme de collaboration entre deux méthodes d'apprentissage symbolique et connexionniste, nous étudions la possibilité de réduire la topologie d'un réseau à l'aide d'une base de règles, ainsi que la possibilité d'optimiser une base de règles en utilisant un RNA.

5.4.1 Apprentissage parallèle connexionniste-symbolique

Les méthodes d'apprentissage (connexionniste ou symbolique) employées pour résoudre un problème donné fournissent des solutions différentes pour un même jeu de données. Pour cette raison, nous allons donc étudier les concepts donnés par un classifieur hybride capable de confronter les résultats de deux méthodes de classification de nature différente afin d'obtenir un résultat plus pertinent. Pour ce faire, nous

5. Ces deux méthodes sont décrites dans le chapitre 3.

avons donc étudié l'intégration directe de deux méthodes symbolique et connexionniste (voir la possibilité d'intégrer d'autres méthodes, par exemple statistique, pour obtenir un système de type multi-agents [39]) dans un système hybride capable de les faire coopérer afin qu'il améliore leurs performances.

Les méthodes de classification automatique de données peuvent être regroupées en trois catégories: connexionniste, symbolique (formation de concepts) et statistiques [48]. Dans cette coopération, nous avons retenu l'une de nos deux méthodes d'extraction de règles à partir d'un RNA multi-couches [98] selon la convenance de l'application et la méthode symbolique *Imind* [67]. Cette dernière est une méthode symbolique développée par Henniche et qui fait une classification de façon incrémentale. Elle permet d'obtenir des hiérarchies de concepts de façon efficace.

Notre proposition consiste donc à faire collaborer ces deux méthodes de façon à atténuer pour chacune d'elle les inconvénients cités dans le chapitre 1. Nous étudions aussi la possibilité d'une coopération connexionniste-symbolique qui consiste à la réalisation d'un apprentissage parallèle à partir des mêmes données.

Pour développer un système mettant en œuvre la collaboration entre ces deux méthodes (extraction de règles à partir d'un RNA et une méthode symbolique incrémentale), il faut définir l'ensemble des opérations internes, les mécanismes de communication entre les deux méthodes et le choix de classes à modifier.

Nous nous intéressons donc à l'apprentissage symbolique non pas pour initialiser un réseau de neurones, mais pour qu'il apprenne en parallèle avec ce dernier. Cette parallélisation d'apprentissage semble être intéressante dans le cas où les deux systèmes n'arrivent pas à aboutir séparément à des bonnes performances pour un problème donné.

Comme le montre FIG. 5.2, l'apprentissage est fait à partir de la même base d'exemples par les deux approches connexionniste et symbolique. Après apprentissage, l'approche symbolique [67] fournit des règles sous forme normale disjonctive, ce qui définit le concept appris. Pour obtenir des règles au niveau de l'approche

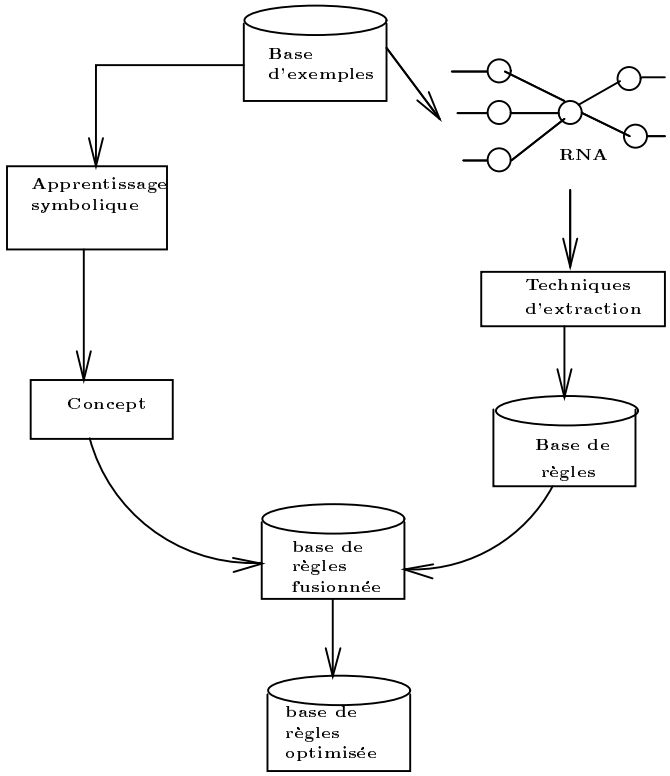


FIG. 5.2 – *Apprentissage parallèle connexionniste-symbolique*

connexionniste, nous utilisons l'une de nos deux techniques d'extraction de règles que nous avons développés [96, 97]. Les règles obtenues par l'une de nos deux méthodes sont aussi réécrites sous une forme normale disjonctive. Les deux bases de règles obtenues sont fusionnées pour en obtenir une nouvelle. Au niveau de la nouvelle base de règles, les redondances et les contradictions sont éliminées par des techniques classiques [90, 117]. L'importance de cette parallélisation d'apprentissage est de résoudre un problème donné par deux techniques différentes et d'une manière complémentaire.

5.4.2 Optimisation d'une base de règles à l'aide d'un RNA

Pour la simplification du réseau, nous utilisons l'information mutuelle pour la sélection des connexions pertinentes et par la suite optimiser l'architecture du réseau [9, 16]. On peut aussi utiliser l'une des méthodes d'élagage (OBD [82], OCD [25], HVS [134], ..). Dans ces dernières méthodes, il s'agit de simplifier un réseau surdimensionné au cours de l'apprentissage. Les critères de sélection de neurones ou de connexions à supprimer sont faciles à élaborer dans la mesure où l'élagage intervient une fois que le réseau a convergé. On veut dire par élagage, une tendance à l'élimination progressive des poids inutiles. Les poids éliminés ne sont pas simplement les poids les plus petits mais aussi d'autres poids qui sont sélectionnés en se basant sur les deux principes suivants :

1. alterner la phase d'élagage avec la phase d'apprentissage en supprimant d'une façon brutale les poids et les neurones contrôlés par une mesure de sensibilité ;

2. utiliser une fonction de coût composée de l'erreur quadratique et d'un terme de complexité.

Dans [68], plusieurs mesures de sensibilité ont été proposées pour la suppression d'une connexion ou d'un neurone dans un RNA multi-couches. Parmi ces méthodes, *OBD* (*Optimal Brain Damage*) de Le Cun et al. [82] est la plus utilisée. Cette méthode est caractérisée d'une part, par la simplicité de son implantation et d'autre part, par ses bons résultats. Le Cun et al. travaillent sur un réseau qui a convergé et pour faciliter les calculs, ils supposent que la matrice Hessienne[6] est diagonale. Ce qui réduit d'une façon considérable la formule qui calcule la variation de l'erreur quadratique. Les poids qui entraînent une petite sensibilité sont supprimés. Cependant, le coût de calcul est élevé, car la sensibilité est un indice moyen calculé sur les exemples de la base d'apprentissage et elle exige en outre le calcul de la matrice Hessienne sur l'ensemble des poids du réseau. De plus, il est nécessaire d'introduire un test d'arrêt de l'élagage.

6. C'est une matrice basée sur les poids du réseau et calculée en utilisant la dérivée et la dérivée seconde de la fonction de coût.

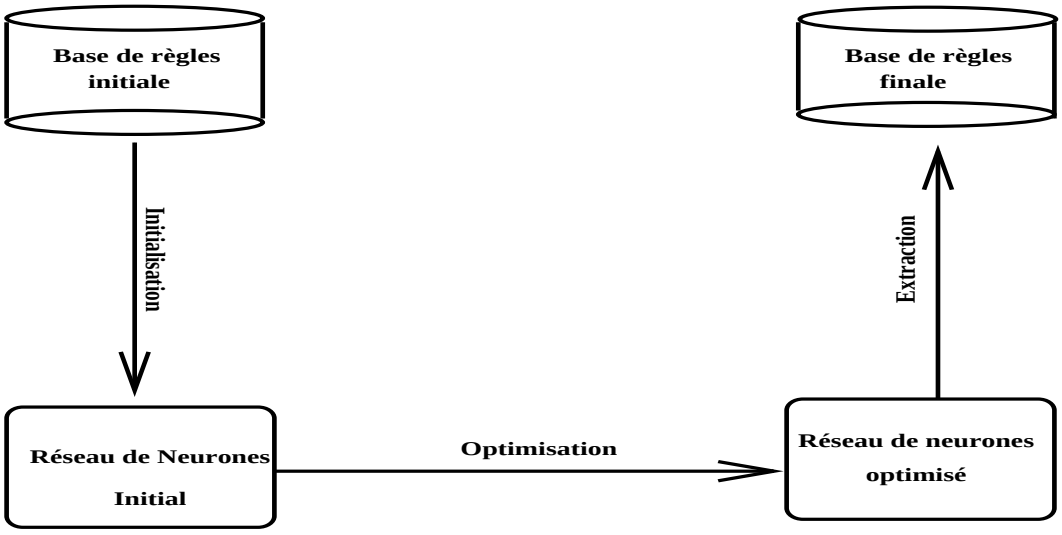


FIG. 5.3 – *Amélioration d'une base de règles à l'aide d'un RNA*

Comme le montre la FIG. 5.3, nous initialisons un réseau de neurones à partir d'une base de règles non optimale. Cette base de règles est obtenue par une méthode symbolique [67] ou à l'aide d'un expert. Nous utilisons l'une de nos deux méthodes $OpNeur$ et $RuleNeur$ (développées dans le chapitre 4) pour l'initialisation du réseau. Afin d'aboutir à un réseau avec une topologie réduite, nous lançons l'apprentissage au niveau du réseau en utilisant la méthode OBD afin d'éliminer le maximum de connexions non utiles. Une fois que le réseau se stabilise, nous appliquons l'une de nos deux méthodes d'extraction de règles $MITER$ et $EMIRE$ (développées dans le chapitre 3) afin d'obtenir une base de règles plus compréhensible (un nombre minimum de règles et d'antécédents par règles) et plus performante que la base de règles initiale.

5.4.3 Réduction de la topologie d'un RNA à l'aide d'une base de règles

Il existe plusieurs techniques d'optimisation d'un réseau de neurones, autrement dit, l'élimination des poids et des neurones non significatifs. Le but de cette optimisation est d'obtenir un réseau avec une architecture plus simple que le réseau initial

en gardant les mêmes performances, voire des performances meilleurs en généralisation. Comme le montre FIG. 5.4, nous proposons une méthodologie d'optimisation de la topologie du réseau à l'aide d'une base de règles en trois étapes :

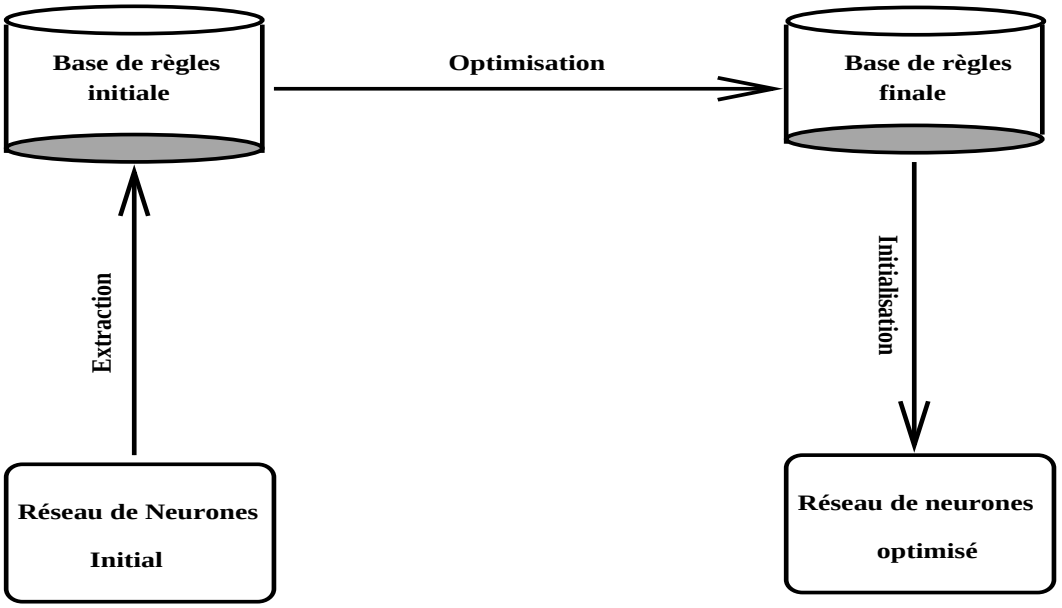


FIG. 5.4 – *Réduction de la topologie d'un RNA à l'aide d'une base de règles*

1. *Extraction de règles à partir d'un réseau de neurones :* pour accomplir cette tâche, nous utilisons l'une de nos deux méthodes d'extraction de règles [98] ;

2. *Optimisation des règles :* nous optimisons la base de règles extraites par des techniques classiques d'optimisation. Ces dernières, nous aident à éliminer les redondances et les contradictions au niveau de cette base de règles [90, 117] ;

3. *Initialisation d'un réseau de neurones :* afin d'obtenir une topologie réduite du réseau, nous initialisons, de nouveau à partir de cette base de règles optimisée, un RNA en utilisant l'une de nos deux méthodes d'initialisation du réseau à partir d'une base de règles [100].

5.5 Conclusion

Dans ce chapitre, nous avons présenté certains mécanismes de coopération entre les méthodes connexionniste, symbolique et probabiliste. L'objectif de cette coopération est de construire un système hybride capable de compenser les faiblesses de chacune de ces méthodes par l'apport d'informations supplémentaires provenant des autres. Une étude approfondie de ces mécanismes reste à faire, toutefois les résultats présentés dans les chapitres 3 et 4 en font une piste sérieuse de recherche en vue d'obtenir des performances meilleurs que celles de chacune de ces méthodes prises séparément.

Conclusion et perspectives

Conclusion

Dans ce document, nous avons présenté notre travail de recherche qui s'inscrit dans le cadre de l'interaction des réseaux de neurones artificiels et des systèmes symboliques. La motivation de ce travail est double : d'une part, munir les RNA de l'aptitude d'explication afin qu'ils justifient leurs résultats et améliorent leurs performances et d'autre part, insérer dans un RNA des connaissances symboliques provenant d'un système expert ou d'un système d'apprentissage symbolique.

Dans cette optique, notre travail a porté sur trois niveaux différents mais complémentaires :

- l'extraction d'un ensemble de règles symboliques à partir d'un RNA afin de mieux répondre à la question : comment le RNA a abouti à un tel résultat ? ;
- la construction de l'architecture d'un RNA d'une façon non empirique en se basant sur des règles symboliques ;
- la proposition d'un système hybride d'interaction symbolique-connexionniste qui permet de raffiner une base de règles incertaine ou incomplète donnée par un expert où une méthode symbolique de l'IA en utilisant un RNA.

Dans le premier travail, nous avons commencé par une étude critique de la plupart des méthodes développées dans ce domaine récent d'interaction symbolique-connexionniste, réputé difficile (car il s'agit d'accéder à une information implicite, c.-à-d. à une information qui *n'est pas immédiatement disponible*). Nous avons proposé deux techniques qui ont pour but de solliciter la mise en règles descriptives de la manière dont un RNA a réalisé une tâche. La première ($MITER$), analyse les poids de chaque neurone caché ou de sortie afin d'interpréter leur fonctionnement

par des règles symboliques de la forme *M-parmi-N*. Une règle est extraite de chaque neurone. Essentiellement pour cette dernière raison, $MITER$ peut être appliquée même à des réseaux avec une architecture complexe. Dans la seconde ($EMIRE$), les règles sont extraites à partir des exemples sans regarder la structure interne du réseau. $EMIRE$ nécessite l'utilisation des techniques classiques d'optimisation de règles. C'est pour cette raison que $EMIRE$ est appliquée à des réseaux avec une architecture simple. Dans ces deux techniques, l'information mutuelle est utilisée pour sélectionner les entrées pertinentes. $MITER$ et $EMIRE$ ont été appliquées avec succès sur la plupart des problèmes sur lesquels ont été testés. Nous avons aussi proposé une autre direction pour l'extraction de règles de la logique floue à partir d'un RNA.

Dans le deuxième travail, après une étude sommaire de la plupart des techniques développées pour insérer des connaissances symboliques dans un RNA, nous avons pu proposé deux techniques de construction d'une architecture connexionniste à partir d'une information symbolique. Dans la première ($OpNeur$), le RNA est construit à partir de la définition d'un concept donnée par des règles de la forme normale disjonctive. Le réseau ainsi construit a une ou deux couches cachées. Dans la seconde ($RuleNeur$), à partir d'un ensemble de règles de la logique des propositions, un réseau est construit où le nombre de ses couches cachées est inférieur au nombre de conclusions intermédiaires dans les règles d'initialisation.

Dans le troisième travail, les résultats que nous avons obtenus des différentes composantes de notre système hybride $RANNI$, doivent permettre d'ores et déjà une validation sur des problèmes réels (biologie, médecine, ...). Parallèlement à l'étude de ce système hybride, nous avons commencé à travailler sur la possibilité de coopération des systèmes connexionnistes avec les systèmes d'apprentissage symbolique.

Perspectives

À l'issue de ce travail, cinq perspectives peuvent être envisagées :

- Dans nos deux méthodes d'extraction de règles, nous utilisons des paramètres prenant des valeurs tirées de nos différentes simulations. Le choix de ces valeurs permet de faire le compromis entre plusieurs critères : l'exactitude, la fidélité et la compréhensibilité des règles extraites. Il serait intéressant de traiter ces paramètres comme étant des hyperparamètres du modèle, les valeurs cherchées étant celles qui réalisent le meilleur compromis entre ces différents critères.

- L'affirmation selon laquelle "*RuleNeur*" est une méthode plus adaptée à des problèmes définis par des règles qui contiennent des conclusions intermédiaires reste à valider sur un jeu de données complexes.

- Les techniques d'insertion de règles dans des RNA utilisent des règles de la logique des propositions. Par conséquent, un premier problème de recherche que nous pouvons approfondir dans la suite du travail présenté dans ce document est d'insérer et d'extraire des règles de la logique des prédicats, voire d'ordre supérieur.

- Les différentes composantes de *RANNI* ont été validées sur plusieurs applications. Par contre, pour prouver le bon fonctionnement du système il faut donc le valider sur des applications réelles.

- Dans des applications où chacune des deux approches connexionniste et symbolique ne peut pas obtenir des bons résultats, il est intéressant de développer des mécanismes nécessaires à un apprentissage parallèle symbolique-connexionniste.

Il reste à évoquer les perspectives de l'approche d'interaction connexionniste-symbolique. Nous pensons qu'en son état actuel, les techniques de cette approche permettent de résoudre certaines des difficultés rencontrées par les systèmes symboliques et les réseaux connexionnistes pris séparément. Ces techniques sont tout juste à leur début, il leur reste donc plusieurs problèmes à résoudre, mais aussi de nombreuses directions à explorer dans ce domaine.

Les principaux problèmes de recherche sont les suivants :

- L'initialisation d'un RNA à partir d'une base de règles, conduit souvent à des réseaux ayant une grande architecture. Or, tout problème résolu par cette grande architecture peut être résolu par un réseau avec une ou deux couches cachées seulement. Par conséquent, un premier problème de recherche est de combiner l'approche *"initialisation d'un réseau de neurones à partir d'une base de règles"* et *"la recherche heuristique d'une architecture minimale"*.

- Dans plusieurs applications, il a été constaté que l'ensemble de règles permettait de généraliser mieux que le RNA dont il extrait [129]. La même remarque a été faite à propos des règles symboliques grammaticales extraites à partir d'un RNA récurrent [53]. Expliquer cette différence de performance, et identifier les conditions dans lesquelles elle est obtenue est un deuxième problème de recherche pour améliorer le fonctionnement des RNA.

- Un troisième problème vient du fait que les règles sont extraites après l'apprentissage [129, 53]. Deux questions se posent : peut-on extraire des règles durant l'apprentissage? peut-on en insérer?

- Un quatrième problème est l'étude de la complexité des algorithmes d'extraction de règles. Cette complexité est due d'une part à la nature du problème, car il s'agit de réaliser des techniques qui profitent des avantages de l'IA symbolique et des RNA, et évitent leurs faiblesses. D'autre part, le problème de recherche d'un ensemble minimal de règles imitant le réseau avec une grande fidélité est généralement un problème difficile, voire NP-Complet.

- La plupart des techniques d'extraction de règles demandent quelques heuristiques pour contraindre la taille de l'espace de recherche. Par exemple, l'utili-

sation des seuils pour filtrer les entrées qui n'ont pas d'effet significatif sur la décision finale produite par le réseau. Par conséquent, un cinquième problème de recherche peut être décomposé en deux parties : la première est l'évaluation de l'impact d'une telle heuristique sur la qualité et l'efficacité des règles produites, la seconde est l'étude de la possibilité de déterminer ces heuristiques par apprentissage et non pas par expérience.

- Dans la plupart des techniques d'extraction, les règles extraites sont de la logique de proposition. Un sixième problème de recherche serait d'étudier la possibilité d'extraire des règles logiques d'ordre supérieur (la logique des prédicats, la logique modale, ...).

- Un autre problème d'extraction de règles est de formuler un ensemble de critères pour adapter l'ensemble des techniques à l'exigence du problème étudié. Par exemple, trouver un moyen pour déterminer la technique d'extraction de règles optimale pour une application donnée.

- Le dernier problème soulevé, est probablement le plus important mais aussi le plus difficile de ceux qui restent à résoudre. Il s'agit de la possibilité de faire en sorte que l'explication (extraction de règles) ne soit plus une couche ajoutée aux RNA mais qu'elle rentre dans leur conception. Si cet objectif était atteint, il serait inutile d'étudier la fidélité des règles par rapport au réseau, car c'est ce dernier qui donnerait la signification de ses connexions.

Bibliographie

[1] Alexander, J.A., and Mozer, M.C. Template-Based Algorithms for Connectionist Rule Extraction. *NIPS-7, Neural Information Processing Systems, Denver, CO, pp. 609-616, 1995.*

[2] Alexis, K., Bouali, F., Hors, P., et al. EVA : Modélisation et Représentation de Connaissances Hétérogènes, Guidées par les Besoins en Explication, Validation et Acquisition. *Actes des Cinquièmes Journées Nationales de l'IA, Nancy, pp. 263-282, 1995.*

[3] Amy, B., Giacometti, A. et Gut, A. Modèles Connexionnistes de l'Expertise. *Actes des Journées Internationales Neuro-Nîmes'90, EC2, Nanterre, pp. 99-119, 1990.*

[4] Andrews, A., Diederich, J., and Tickle, A.B. A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Technical Report, Neurocomputing Research Center, Queensland University of Technology, Australia, 66 p., 1995.*

[5] Andrews, R., and Geva, S. Rule Extraction from a Contrained Error Back Propagation MLP. *Proceeding of 5th Australian Conference on Neural Networks, Brisbane, Queensland, pp. 9-12, 1994.*

[6] Anouar, F., Badran, F., Sorror, C. et Thiria, S. Interprétation Symbolique des Réseaux de Neurones : Extraction de Règles de Décision. *Rapport interne, Conservatoire National des Arts et Métiers, 12 p., 1994.*

[7] Ayel, M. et Rousset, M.-C. La Cohérence dans les Bases de Connaissances. *Cepadues Editions, Collection Intelligence Artificielle, 106 p., 1990.*

[8] Baker, M., Dessalles, J.-L., Joab et al. La Génération d'Explication Négociées dans un Système à Base de Connaissances. *Actes des Cinquièmes Journées Nationales de l'IA, Nancy, pp. 297-316, 1995.*

[9] Battiti, R. Using Mutual Information for Selecting Features in Supervised Neural Net Learning. *IEEE Transactions on Neural Networks, Vol. 5, N. 4, pp. 537-550, 1994.*

[10] Benjamin, D. Régularisation Appliquée au Traitement d'Images, *Thèse de doctorat, Université Paris 13, 165 p., 1997.*

[11] Bennani, Y., and Bossaert, F. From Implicit to Explicit Information in Multilayer Neural Networks. *ICSC'96, UK, 1996.*

[12] Bennani, Y. Approches Connexionnistes pour la Reconnaissance Automatique du Locuteur : Modélisation et Identification. *Thèse de Doctorat, Université de Paris 11, N. 1948, 255 p., 1992.*

[13] Bennani, Y. et Gallinari, P. Approches Connexionnistes pour la Détection et l'Identification de Perturbations dans un Réseau Téléphonique : Modélisation et Discrimination. *Actes des VII^{es} journnées du Laboratoire d'Informatique de Paris Nord, pp. 141-142, 1997.*

[14] Berenji, H.R. Refinement of Approximate Reasoning-Based Controllers by Reinforcement Learning. *Proceeding of the 8th International Machine Learning Workshop, Evanston IL, pp. 475-479, 1991.*

[15] Bonnlander, B.V., and Weigned, A.S. Selecting Input Variables Using Mutual Information and Nonparametric Density Estimation. *Technical Report, 9 p., 1996.*

[16] Bonnlander, B.V. Nonparametric Selection of Input Variables for Connectionist Learning. *PhD Thesis, Depatement of Computer Science, University of Colorado, 90 p., 1996.*

[17] Bouchereau, L. and Bourgine, P. Extraction of Semantic Features and Logical Rules from a Multilayer Neural Network. *International Joint Conference on Neural Networks Washington DC, Vol. 2, pp. 579-582, 1990.*

[18] Bouttou, L. et Le Cun, Y. SN : un Simulateur pour Réseaux Connexionnistes, *Proceeding Neuro-Nîmes, EC2 Edition, pp. 371-382, 1988.*

[19] Brachman, R.J. and McGuinness, D.L. Knowledge Representation, Connectionism, and Conceptual Retrieval. *SIGIR Proceding, 11th International Conference on Representation and Decision in Information Retrieval, Grenoble, pp. 161-174, 1988.*

[20] Breiman, L., Friedman, J.H., Olsen R.A., and Stone, C.J. Classification and Regression Trees. *Wadsworth & Brooks, Cole Advanced Books & Software, Monterey, Ca, 1984.*

[21] Bryson, A., Denham, W., and Drefuss, S. Optimal Programming Problem with Ineduality Contraints: Necessary Conditions for External Solutions. *AAAI Journal, Vol. 1, pp. 25-44, 1963.*

[22] Challo, R., Mclauchlan, R.A., Clark, D.A., and Omar, S.I. A Fuzzy Neural Hybrid System. *In IEEE International Conference on Neural Networks, Vol. 3, pp. 1654-1657, Orlando, 1994.*

[23] Chester, D.L. Why Two Hidden Layers are Better than One. *Proceedings of the International Joint Conference on Neural Networks, Erlbaum, Vol. 1, pp. 265-268, 1990.*

[24] Cherkauer, K.J., and Shavlik, J.W. Rapid Quality Estimation of Neural Network Input Representations. *Neural Information Processing Systems, Vol. 8, 1996.*

[25] Cibas, T., Fogelman Soulie, F., Gallinari, and P. Raudys, S. Variable Selection with Optimal Cell Damage. *International Conference on Artificail Neural Networks, ICANN'94, 1994.*

[26] Cohen, P.R., and Feigenbaum, E.A. The Handbook of Artificial Intelligence, *Pitman Edition, 1982.*

[27] Comon, P. Classification Supervisée par des Réseaux Multicouches. *Traitement du Signal, Vol. 6, N. 6, pp. 387-407, 1991.*

[28] Cottrell, M., Girard, B., Girad, Y., and Mangeas, M. Times Series and Neural Network: a Statistical Method for Weight Elimination. *1st European Symposium on Artificial Neural Networks, Brussels, Belgique, D. Facto Editeur, pp. 157-164, 1993.*

[29] Craven, M. W., and Shavlik, J.W. Using Sampling and Queries to Extract Rules from Trained Neural Networks. *Machine Learning: Procedings of the 11th International Conference, San Francisco, CA, 8 p., 1994.*

[30] Cybenko, G. Approximation by Superposition of Sygmoïdal Functions. *Mathematics of Control, Signals and Systems, Vol. 2, N. 4, pp. 303-314, 1989.*

[31] D'Alché-Buc, F. Modèles Neuronaux et Algorithmes Constructifs pour l'Apprentissage de Règles de Décision. *Thèse de doctorat, Université Paris 11, 208 p., 1993.*

[32] David, J., and Krivine, J. Explaining Reasoning from Knowledge-Level Models. *ECAI-90, pp. 186-188, 1990.*

[33] Denoeux, T. Generation of Symbolique Rules in Back-propagation Networks. *Alexender I., Taylor J. (eds): Artificial Neural Network, North-Holland, Amsterdam, pp. 711-714, 1992.*

[34] Denoeux, T., Lengellé, R., and Canu, S. Initialization of Weights in a Feedforward Neural Network Using Prototypes. *Kohonen T., Makisara K., Simula O., Kangas J. (eds): Artificial Neural Networks, North-Holland, pp. 623-628, 1991.*

[35] Denoeux, T. Génération Automatique de Règles de Classification par l'Algorithme de Rétropropagation de Gradient. *Actes des 3^{es} journées "Symbolique-Numérique", Université de Dauphine, Paris, pp. 273-284, 1992.*

[36] Devillers, L. Reconnaissance de Parole Continue avec un Système Hybride Neuronal et Markovien. *Thèse de Doctorat, Université Paris 11, N. 94-05, 266 p., 1992.*

[37] Diederich, J. Explanation and Artificial Neural Networks. *International Journal of Man-Machine Studies, Vol. 37, pp. 335-357, 1992.*

[38] Dinsmore, J. The Symbolic and Connectionist Paradigms: Closing the Gap. *Lawrence Erlbaum Associates, Chapter 1, 1992.*

[39] El Fallah-Seghrouchni, A. Multi-Agent Systems as a Paradigm for Intelligent System Design. *International Journal of Computing and Informatics, N. 21, pp. 173-184, 1997.*

[40] Fisher D.H., and McKusick, K.B. An Empirical Comparison of ID3 and Back-Propagation. *Proceeding of the 11th International Joint Conference on Artificial Intelligence, pp. 788-793, 1989.*

[41] Fisher D.H. The Use of Multiple Measurements in Taxonomie Problems. *Annal of Fugenics, Vol. 7, N. 2, pp. 179-188, 1986.*

[42] Fu, L.M. Rule Generation from Neural Networks. *IEEE Transactions on Systems, Man and Cybernetics, Vol. 28, pp. 1114-1124, 1994.*

[43] Fu, L.M. Neural Networks in Computer Intelligence. *Mc Graw-Hill Edition, 460 p., 1994.*

[44] Fu, L.M. Rule Learning by Searching on Adapted Nets. *Proceeding of the 9th National Conference on Artificial Intelligence, Anaheim, CA, pp. 590-595, 1991.*

[45] Funahashi, K.I. On the Approximate Realization of Continuous Mappings by Neural Networks. *Neural Networks, Vol. 2, pp. 183-192, 1989.*

[46] Gallant, S.I. Connectionist Expert Systems. *Communication of the ACM, Vol. 31, N. 2, pp. 235-247, 1988.*

[47] Gallinari, P., Thiria, S., and Soulie, F.F. Multilayer Perceptrons and Data Analysis. *IEEE International Conference on Neural Networks, San Diego, California, 1988.*

[48] Gançarski, P. et Wemmert, C. Collaboration entre Méthodes de Classification, *Cinquième Rencontre de la Société Francophone de Classification, Lyon, France, pp. 279-282, 1997.*

[49] Gascuel, O. et Gallinari, P. Méthodes Symboliques-Numériques de Discrimination. *Actes des Cinquièmes Journées Nationales de l'IA, Nancy, pp. 29-76, 1995.*

[50] Geva, S. Direct Production Rule Extraction. *Technical Report, QUT NRC, Brisbane, Australia, 1995.*

[51] Giacometti, A. Modèles Hybrides de l'Expertise. *Thèse de doctorat, École Nationale Supérieure des Télécommunications, Paris, 283 p., 1992.*

[52] Gilbert, N. Explanation and Dialogue. *The Knowledge Engineering Review, Vol. 4, N. 3, pp. 235-247, 1989.*

[53] Giles, C.L., and Omlin, C.W. Extraction, Insertion, and Refinement of Symbolic Rules in Dynamically Driven Recurrent Networks. *Connection Science, Vol. 5, N. 3, pp. 307-328, 1993.*

[54] Glorennec, P.Y. Un Réseau NeuroFlou Evolutif. *Proceeding Neuro-Nîmes, pp. 301-314, 1991.*

[55] Goller, C. A Connectionist Control Component for the Theorem Prover SETHEO. *ECAI-94, Combining Symbolic and Connectionist Processing, Amsterdam, pp. 88-93, 1994.*

[56] Gunst, R. F., and Mason, R. L. Regression Analysis and its Application: a Data Oriented Approch. *Statistics text books and Monographs, Vol. 34, 1980.*

[57] Gutknecht, M., and Pfeifer, R. Experiments with a Hybrid Architecture: Integrating Expert Systems with Connectionist Networks. *Proceeding of the 10th International Conference on Artificial Intelligence, Expert Systems and Natural Language, Avignon, 1990.*

[58] Halgamuge, S. K., and Glesner, M. Neural Networks in Designing Fuzzy Systems for Real World Applications. *Fuzzy Sets and Systems, Vol. 65, N. 1, pp. 1-12, 1994.*

[59] Harnad, S. The Symbolic Grounding Problem. *Physica, N. 42, pp. 335-346, 1990.*

[60] Haton, J.-P., Bouzid, N. et al. Le Raisonnement en Intelligence Artificielle : Modèles, Techniques et Architectures pour les Systèmes à Base de Connaissances. *Inter Edition, 1991.*

[61] Hayashi, Y. A Neural Expert System with Automated Extraction of Fuzzy If-Then Rules and its Application to Medical Diagnostics. *Advances in Neural Information Processing Systems, Denver, CO, Vol. 3, pp. 578-584, 1990.*

[62] Hayward, R., Pop, E., and Diederich, J. Extracting Rules for Grammar Recognition from Cascade-2 Networks. *Proceedings IJCAI-95, Workshop on Machine Learning and Natural Language Processing, 1995.*

[63] Hayward, R., Ho-Stuart, C. Diederich, J., and Pop, E. RULENEG: Extracting Rules from a Trained ANN by Stepwise Negation. *Technical Report, QUT NRC, Brisbane, Australia, 1996.*

[64] Hebb, D.O. The Organization of Behavior. *John Wiley and Sons, New York, 1949.*

[65] Hendler, J. Problem Solving and Reasoning: a Connectionist Perspective. *Connectionist in Perspective, Elsevier Science Publishers, North Holland, pp. 229-243, 1989.*

[66] Hendler, J. On the Need for Hybrid Systems. *Connection Science, Special Issue on Hybrid Connectionnist/Symbolic Systems, 1989.*

[67] Henniche, M. Apprendre Incrémentalement dans l'Espace des Généralisation Maximalement Spécifiques d'Instances. *Actes des Journées Ingénierie des Connaissances et Apprentissage Automatique, Roscoff, France, pp. 125-136, 1997.*

[68] Hérault, J. et Jutten, C. Réseaux Neuronaux et Traitement du Signal. *Edition HERMES, 313 p., 1993.*

[69] Hertz, J., Krogh, A., and Palmer, R.G. Introduction to the Theory of Neural Communication. *Lecture Notes Volumes in the Santa Fe Institute Studies in the Sciences of Complexity, Addson Wesley, 1991.*

[70] Holte, R.C., Acker, L.E., and Porter, B.W. Concept Learning and the Problem of Smal Disjuncts. *In Proceeding IJCAI-89, Detroit, MI, pp. 813-819, 1989.*

[71] Horikawa, S., Furuhashi, T., and Uchikawa, Y. On Fuzzy Modeling Using Fuzzy Neural Networks with the Back-Propagation Algorithm. *IEEE Transactions on Neural Networks, Vol. 3, N. 5, pp. 801-806, 1992.*

[72] Hornik, K., Stinchcombe, M., and White, H. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks, Vol. 2, pp. 359-366, 1989.*

[73] Kane, R. Extraction de Règles à l'Aide de Réseaux de Neurones. *Thèse de Doctorat, Université Paris 6, 171 p., 1994.*

[74] Kane, R., Tchoumatchenko, I., and Milgram, M. Knowledge Extraction Using Contrained Neural Networks. *Proceeding of European Conference on Machine Learning, Vienne, 1993.*

[75] Kodratoff, Y. Intelligence Artificielle et Approche Connexionniste. *Bulletin de Liaison de la Recherche en Informatique et en Automatique, N. 129, pp. 16-18, 1989.*

[76] Kohonen, T. Self-Organisation and Associative Memory. *Springer Series in Information Sciences, Springer-Verlag, Vol. 8, 1984.*

[77] Kohonen, T. The Neural Phonetic Typewriter. *IEEE Computer, pp. 11-22, 1988.*

[78] Kwasny, S.C., and Faisal, K.A. Symbolic Parsing Via Subsymbolic Rules. *Lawrence Erlbaum Associates, pp. 209-235, 1992.*

[79] Lang, K., and Hinton, G. The Development of the Time Delay Neural Network Architecture for Speech Recognition. *Canergie Mellon University, TR CMU-CS, N. 152, 1988.*

[80] Laurière, J.-L. Intelligence Artificielle : Résolution de Problèmes par l'Homme et la Machine. *Edition Eyrolles, 1987.*

[81] Le Cun, Y. Modèles Connexionnistes de l'Apprentissage. *Thèse de Doctorat, Université Paris 6, 1987.*

[82] Le Cun, Y., Denker, J. S., and Solla, S. A. Optimal Brain Damage. *Neural Information Processing Systems, NIPS-90, Denver, CO, Vol. 2, pp. 598-605, 1990.*

[83] Le Cun, Y. Learning Processes in Asymmetric Threshold Network. *Disordered Systems and Biological Organizations, Les Houches, France, Springer, pp. 223-240, 1986.*

[84] Lemberg, H. Les Réseaux Neuronaux Artificiels. *Langage et système, InfoPC, N. 2, pp. 20-28, 1991.*

[85] Mahoney, J.J., and Mooney, R.J. Combining Connectionist and Symbolic Learning to Refine Certainty Factor Rule Bases. *Connection Science, Vol. 5, N. 4, pp. 339-364, 1993.*

[86] Martin, P. Réseaux de Neurones Artificiels : Application à la Reconnaissance Optique de Partitions Musicales, *Thèse de Doctorat, Université Grenoble 1, 1992.*

[87] Masuoka, R. Neurofuzzy Systems: Fuzzy Inference Using a Structured Neural Network. *Proceeding of the International Conference on Fuzzy Logic and Neural Networks, Lizuka Japan, pp. 173-177, 1990.*

[88] Mc Culloch, W.S., and Pitts, W.A. A Logical Calculs of the Ideas Imminent in Nervous Activity. *Bulletin of mathematics biophysics, Vol. 5, pp. 115-133, 1943.*

[89] McMillan, C., Mozer, M. C., and Smolensky, P. The Connectionist Scientist Game: Rule Extraction and Refinement in a Neural Network. *Proceeding of the 13th Annual Conference of the Cognitive Science Society, Hillsdale NJ, 1991.*

[90] Miranker, D. TREAT: A Better Match Algorithm For AI Production Systems. *Proceeding of AAAI'87 Conference on Artificial Intelligence, 1987.*

[91] Mitra, S. Fuzzy MLP Based Expert System for Medical Diagnosis. *Fuzzy Sets and Systems, Vol. 65, N. 2, pp. 285-296, 1994.*

[92] Minsky, M., and Papert, S. Perceptrons. *MIT Press, Cambridge MA, 1969.*

[93] Moore, J. D., and Swartout, W. R. A Reactive Approach to Explanation. *IJCAI-89, pp. 1504-1510, 1989.*

[94] Murphy, P.M., and Pazzani, M.J. ID2-of-3: Constructive Induction of N-of-M Concepts for Discrimination in Decision Trees. *Proceeding of the 8th International Machine Learning Workshop, pp. 183-187, 1991.*

[95] Murphy, P.M., and Aha, D.W. UCI Repository of Machine Learning Database. *Technical repport, University of California, Departement of Computer Science, 1992.*

[96] Nedjari, T. MITER: Mutual Information and Template for Extracting Rules. *NIPS-96, Rule Extraction from Trained Artificial Neural Network Workshop, Snowmass, CO, USA, pp. 54-61, 1996.*

[97] Nedjari, T. Use of Mutual Information to Extract Rules from Artificial Neural Networks. *ICANNGA-97, International Conference on Artificial Neural Nets and Genetic Algorithms, pp. 565-569, Springer Wiem NewYork, Norwich, UK, 1997.*

[98] Nedjari, T. Trois Méthodes d'Extraction de Règles à partir d'un Réseau de Neurones Artificiel. *Cinquième Rencontre de la Société Francophone de Classification, Lyon, France, pp. 195-200, 1997.*

[99] Nedjari, T. et Bennani, Y. Extraction de Connaissances à partir d'un Réseau de Neurones Artificiel. *Rapport interne au Laboratoire d'Informatique de Paris 13, N. 98-03, 25 p., 1998.*

[100] Nedjari, T., and Bennani, Y. Symbolic-Connectionist Interaction. *Second International Conference on Cognitive and Neural Systems, CNS'98, Boston, MA, USA, 1998.*

[101] Nedjari, T., and Bennani, Y. A Hybrid System for Symbolic-Connectionist Interaction. *International Symposium on Neural Computation, NC'98, pp. 922-925, Publication by ICSC Academic Press, Vienna, Austria, 1998.*

[102] Newell, A. and Simon, H. A. Computer Science as Empirical Inquiry: Symbols and Search. *ACM Communications, pp. 113-126, 1976.*

[103] Okada, H., Masuoka, R., and Kawamura, A. Knowledge Based Neural Network Using Fuzzy Logic to Initialise a Multilayered Neural Network and Interpret Postlearning Results. *Fujitsu Scientific and Technical Journal FAL, Vol. 29, N. 3, pp. 217-226, 1993.*

[104] Orsier, B. Étude et Application de Systèmes Hybrides Neurosymboliques. *Thèse de doctorat, Université Joseph Fourier, Gronoble 1, 222 p., 1995.*

[105] Parker, D. B. A Comparison of Algorithms for Neuron-Like Cells. *Proceding of the Second Conference on Neural Networks for Computing, 1986.*

[106] Paugam-Moisy, H. Optimisation des Réseaux de Neurones Artificiels : Analyse et Mise en Oeuvre sur Ordinateurs Massivement Parallèles. *Thèse de Doctorat, ENS, Université Lyon 1, 1992.*

[107] Pearson, E. S., and Hartley, H. O. Biometrika Tables for Statificians. *Cambridge University Press, Cambridge, 1969.*

[108] Poggio, T. A Theory of Networks for Approximation and Learning. *MIT-AT Memo, N. 1140, 1989.*

[109] Pop, E., Hayward, R., and Diederiche, J. RULENEG: Extracting Rules from a Trained ANN by Stepwise Negation. *Technical Report, QUT NRC, Australia, 1994.*

[110] Rechenman, F., Uvietta, P. et Fontanille, P. SHIRKA : un Système à Base de Connaissances Orienté Objets. *Manuel de référence, INRIA/ARTEMIS, Grenoble, France, 1988.*

[111] Rendell, L., and Cho, H. Empirical Learning as a Function of Concept Character. *Machine Learning, N. 5, pp. 267-298, 1990.*

[112] Rosenblatt, F. The Perceptron: a Probabilistic Model for Information Storage in Brain. *Psychology Revue, Vol. 65, pp. 386-408, 1958.*

[113] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. Learning Internal Representation by Error Propagation. *Parallell Distributed Processing Exploration in the Microstructure of Cognition, MIT Press, Cambridge MA, 1987.*

[114] Saito, K., and Nakano, R. Medical Diagnostic Expert System Based on PDP Model. *Proceeding of IEEE International Conference on Neural Networks, San Diego CA, Vol. 1, pp. 255-262, 1988.*

[115] Schaerf, M., and Cadoli, M. Tractable Reasoning Via Approximation. *Artificial Intelligente, Elsevier Science B.V., Vol. 74, N. 2, pp. 249-310, 1995.*

[116] Schank, R., Collins, G.C., and Hunter, L.E. Transcending Inductive Category Formation in Learning. *Behavior Brain Sciences, N. 9, pp. 639-689, 1986.*

[117] Schmolze, J.G., and Snyder, W. Detecting Redundant Production Rules. *14th National Conference on Artificial Intelligence, AAAI-97, 1997.*

[118] Sestito, S., and Dillon, T. Automated Knowledge Acquisition. *Pertinence Hall, Australia, 1994.*

[119] Sethi, I.K. Entropynets: from Decisions Trees to Neural Networks. *IEEE proceeding, Vol. 78, N. 10, pp. 1605-1613, 1990.*

[120] Shavlik, J.W., and Towell, G.G. Extracting Refined Rules from Knowledge-Based Neural Networks. *Machine learning, Vol. 131, pp. 71-101, 1993.*

[121] Shavlik, J.W., Towell, G.G., and Mooney, R.J. Symbolic, and Neural Learning Algorithms: an Experimental Comparison. *Machine learning, N. 6, pp. 111-143, 1991.*

[122] Sun, R., and Bookman, L. How Do Symbols and Networks Fit Together. *Integrating Neural and Symbolic Process Workshop, pp. 20-23, 1993.*

[123] Taha, I., and Ghosh, J. Three Techniques for Extracting Rules from Feedforward Networks. *In Intelligent Engineering Systems Through Artificial Neural Networks, ASME Press, Vol. 6, 1996.*

[124] Taha, I., and Ghosh, J. A Hybrid Intelligent Architecture for Refining Input Characterization and Domain Knowledge. *In Proceeding of World Congress on Neural Networks, Vol. 2, pp. 284-287, 1995.*

[125] Taha, I., and Ghosh, J. Controlling Water Reservoirs Using a Hybrid Intelligent Architecture. *In Intelligent Engineering Systems Through Artificial Neural Networks, ASME Press, Vol. 5, pp. 63-68, 1995.*

[126] Thrun, S. Extracting Provably Correct Rules from Artificial Neural Networks. *Technical Report, IAI-TR-93-5, Institut für Informatik III, University of Bonn, 1993.*

[127] Thrun, S., Bala, J., Bloendron, E. et al. The MONK's problems: a Performance Comparison of Different Learning Algorithms. *Technical Report, CMU-CS-91-197, Carnegie Mellon University, 1991.*

[128] Tickel, A.B., Orlowski, M., and Diederich, J. DEDEC: Decision Detection by Rule Extraction from Neural Networks. *Technical Report, QUT NRC, Australia, 1994.*

[129] Towell, G. Symbolic Knowledge and Neural Networks: Insertion, Refinement and Extraction. *PhD Thesis, Departement of Computer Sciences, University of Wiscousin, Madison, 1991.*

[130] Towell, G., and Shavlik, J.W. Knowledge-Based Artificial Neural Networks. *Artificial Intelligence, N. 70, pp. 119-165, 1994.*

[131] Tersp, V., Hollatz, J., and Ahmad, S. Network Structuring and Training Using Rule-Based Knowledge. *Advanced in Neural Information Processing, Vol. 5, pp. 871-878, 1993.*

[132] Weiss, S., and Indurkhya, N. Optimised Rule Induction, *IEEE Expert, pp. 61-69, 1993.*

[133] Widrow, B., Hoff, M.E. Adaptive Switching Circuits, *IRE-WESCON Convention Record, N. 4, pp. 96-104, 1960.*

[134] Yacoub, M., and Bennani, Y. HVS: A Heuristic for Variable Selection in Multilayer Artificial Neural Network Classifier. *Intelligent Engineering Systems Through Artificial Neural Networks, Vol. 7, pp. 527-532, 1997.*

[135] Yau, H.C., and Maury, M.T. Iterative Improvement of a Gaussian Classifier. *Neural Networks, Vol. 3, N. 4, pp. 437-443, 1990.*

plusieurs domaines montrent l'utilité du paradigme RNA. Néanmoins, ce paradigme a une limite : son incapacité inhérente à fournir une explication des résultats obtenus. C'est essentiellement pour vaincre cette limite que plusieurs chercheurs se sont intéressés à combiner les RNA et les systèmes symboliques de manière à profiter de leurs avantages et éviter leurs faiblesses. Dans cette thèse, nous décrivons notre contribution à ce domaine de recherche. Nos travaux s'articulent autour de trois axes : l'extraction de règles à partir d'un RNA, l'insertion de connaissances dans un RNA, et l'utilisation d'un RNA pour raffiner une base de règles existante. Dans le premier axe : après une étude critique des principales techniques développées, nous avons proposé deux techniques d'extraction de règles. La première, $MITER$, associe à chaque neurone un calibre de poids. Ce dernier est traduit sous forme d'une règle symbolique de la forme *M-parmi-N*. La seconde, $EMIRE$, extrait des règles à partir d'un RNA sans tenir compte de sa structure interne et en utilisant uniquement ses entrées pertinentes. Dans le deuxième axe : après une présentation des principales techniques existantes, nous avons proposé deux techniques d'insertion de règles symboliques dans un RNA. La première, $RuleNeur$, associe à chaque règle écrite sous une forme normale disjonctive un neurone. La seconde, $OpNeur$, associe à chaque opérateur logique (ET ou OU) un neurone. Dans le troisième axe, après une présentation des différents systèmes hybrides, nous avons proposé le système $RANNI$ qui combine les deux axes précédents en utilisant un module probabiliste pour leur mise-à-jour.

TITRE en anglais : Symbolic Knowledges and Neural Networks: Insertion, Refinement and Extraction

---

RÉSUMÉ en anglais

It is becoming increasingly apparent that without some form of explanation capability, the full potential of trained Artificial Neural Networks (ANN) may not be realized. It is particularly for this reason that hybrid connectionist-symbolic systems are developed to combine ANN with symbolic systems in order to take an advantage of their strength and avoid their weakness. In this thesis, we deal with three topics: extraction of rules from trained ANN, insertion of knowledge into ANN and the use of ANN to refine existing rules base. In the first topic: after a survey and critiques of the principal existing techniques , we present two methods for rules extraction. The first, $MITER$, associates to each neuron a template weight which is interpreted in *M-of-N* symbolic rule. The second, $EMIRE$, extracts rules from ANN independently of its internal structure and using only relevant network inputs. The relevance of an input is evaluated by the use of mutual information. In the second topic: after a survey on the principal existing techniques, we present two methods to insert symbolic rules in ANN. The first, $RuleNeur$, associates to each rule with normal disjunctive format a neuron. The second, $OpNeur$, associates to logical operator (AND or OR) a neuron. In the last topic: after a survey on the principal existing hybrid systems, we propose the system $RANNI$ that combines the two precedent topics and uses a probabilistic module to update them.

DISCIPLINE : INFORMATIQUE

MOTS-CLÉS : réseaux de neurones artificiels, systèmes symboliques, information mutuelle, extraction de règles, insertion de connaissances dans un réseau de neurones, raffinement de règles, systèmes hybrides.

INTITULÉ ET ADRESSE DU LABORATOIRE :

Laboratoire d'Informatique de Paris Nord (UPRES-A 7030) Avenue J.-B. Clément 93430 Villetaneuse (France)